UNIVERSIDAD DE LA REPÚBLICA

PEDECIBA

UPC

Resource Allocation and Management Techniques for Network Slicing in WiFi Networks

PhD Thesis Dissertation

by

Matías Richart

submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY.

Supervisors

Dr. Javier Baliosian	Universidad de la República
Dr. Joan Serrat	Universitat Politècnica de Catalunya
Dr. Juan-Luis Gorricho	Universitat Politècnica de Catalunya

Montevideo
November 2019

Resource Allocation and Management Techniques for Network Slicing in WiFi Networks,
Matías Richart.

ISSN not assigned yet

This thesis was written in LaTeX using the class PhDThesisPSnPDF (v2.3.1).
It contains a total of 198 pages.
Compiled with pdfLaTex on Monday 4th November, 2019.

A Gaby y Juanse.

Agradecimientos

Esta tesis es el resultado de un largo camino donde he recibido el apoyo de muchas personas e instituciones. En estas breves palabras quiero agradecer a todas ellas.

En primer lugar, quiero agradecer a Javier, que me ha acompañado desde los inicios de mi carrera académica y cuya guía durante todos estos años me ha permitido desarrollarme y crecer como investigador. Además, sus esfuerzos fueron fundamentales para que pudiese realizar la tesis en cotutela con la UPC.

A mis supervisores a la distancia, Joan y Juan-Luis, por su apoyo y esfuerzo para guiarme durante todos estos años. En particular quiero agradecerles la oportunidad de trabajar con ellos y el buen recibimiento durante mi estadía en Barcelona. Fue una experiencia muy linda y enriquecedora.

A Ramón Agüero, por sus sugerencias y contribuciones en los trabajos conjuntos en el contexto del proyecto ADVICE.

Quiero también agradecer a Eduardo y a todo el grupo MINA por el apoyo e impulso recibido en estos 10 años que estamos trabajando juntos. Ser parte de este grupo fue esencial para el desarrollo de la tesis, siempre atento a que contase con las mejores condiciones.

A Jorge, por las innumerables conversaciones en la oficina, cuyos aportes también se ven reflejados en esta tesis.

A Gaby y Juanse, que supieron lidiar con mucho amor las incontables horas de trabajo hasta tarde y los malhumores de los días difíciles. Esta tesis es parte de ellos también.

Por último, quiero reconocer el apoyo económico recibido durante el doctorado por parte de la Comisión Académica de Posgrado, Universidad de la República y del PEDECIBA Informática.

Abstract

Network slicing has recently been proposed as one of the main enablers for 5G networks; it is bound to cope with the increasing and heterogeneous performance requirements of these systems. To "slice" a network is to partition a shared physical network into several self-contained logical pieces (slices) that can be tailored to offer different functional or performance requirements. Moreover, a key characteristic of the slicing paradigm is to provide resource isolation as well as an efficient use of resources. In this context, a slice is envisioned as an end-to-end virtual network which permits that the infrastructure operators lease their resources to service providers (tenants) through the dynamic, and on-demand, deployment of slices. Tenants may have complete control over the slice functions and resources, and employ them to satisfy their client's demands.

This paradigm, which changes the way networks have been traditionally managed, was initially proposed for the wired realm (core networks). More recently, several 5G actions from the scientific community and the industry have proposed solutions to integrate slicing to the Radio Access Networks (RANs). However, these works focus on general architectures and frameworks for the management and instantiation of network slices avoiding details on how the slices are implemented and enforced in the wireless devices. Even more, while some techniques for slice enforcement already exist, most of them concentrate on cellular technologies, ignoring WiFi networks. Despite of their growing relevance and ubiquity, there are not many works addressing the challenges that appear when trying to apply slicing techniques over WiFi networks.

In this scenario, this thesis contributes to the problem of slicing WiFi networks by proposing a solution to enforce and control slices in WiFi Access Points. The focus of this work is on a particular and complex variant of network slicing called QoS Slicing, in which slices have specific performance requirements.

The main thesis contributions are divided in three: (1) a detailed analysis of the network slicing problem in RANs in general and in WiFi in particular, as well as a study and definition of the QoS Slicing problem, (2) a resource allocation model and mechanism for Wifi devices, and (3) a QoS Slicing solution to enforce and control slices with performance requirements in WiFi Access Points.

Given the novelty of the slicing concept and the complexity of the problem, a detailed study of the slicing problem was performed providing a comprehensive definition of the slicing concept, as well as a classification of the slicing variants. We also introduce the two main problems of slicing wireless resources: resource allocation and isolation. In the scope of those problems, this thesis contributes with a novel approach where the resource allocation problem is divided on two sub-tasks: Dynamic Resource Allocation, and Enforcement and Control.

As a previous step to the construction of a QoS Slicing solution, we propose a novel method of proportionally distributing resources in WiFi networks, by means of the airtime. The proposed mechanism (called ATERR) is based on considering the airtime as the wireless resource to be shared and allocated. We also develop an analytical model of the ATERR algorithm, which shed light on how such resources could be split and on the capacities and limitations of the proposal. The validity of the proposed model is assessed by means of a simulation-based evaluation on the `ns-3` framework.

Finally, regarding the QoS Slicing problem, we consider two different performance requirements: a guaranteed minimum bit rate and a maximum allowable delay; and a capacity requirement which can bound the amount of resources allocated. We formulate the resource allocation problem to the different slices as a stochastic optimization problem, where each slice's requirement of bit rate, delay, and capacity is modeled as a constraint. We devise a solution to the aforementioned problem using the Lyapunov drift optimization theory and obtain an approximate deterministic problem. With this solution, we develop a novel queuing and scheduling algorithm which allows implementing the obtained solution in WiFi devices.

Table of contents

List of figures

List of tables

Nomenclature

Acronyms / Abbreviations

4G	Fourth Generation
5G	Fifth Generation
AC	Access Category
AIFS	Arbitration inter-frame spacing
AP	Access Point
BER	Bit Error Rate
BS	Base Station
BTS	Base Transceiver Station
CW	Contention Window
DCF	Distributed Coordination Function
DIFS	DCF Interframe Space
DRA	Dynamic Resource Allocation
EDCA	Enhanced Distributed Channel Access
eNB	Evolved Node B, the Base Station in LTE
FDD	Frequency Division Duplex
HCCA	HCF Controlled Channel Access
HCF	Hybrid Coordination Function

IoT	Internet of Things
IP	Internet Protocol
M2M	Machine to Machine
MCF	Mesh Coordination Function
OFDM	Orthogonal Frequency-Division Multiplexing
OFDMA	Orthogonal Frequency-Division Multiple Access
PCF	Point Coordination Function
PRB	Physical Resource Block
RAN	Radio Access Network
SC-FDMA	Single Carrier - Frequency Division Multiple Access
SE	Slice Embedding
SMA/CA	Carrier Sense Multiple Access with Collision Avoidance
TDD	Time Division Duplex
TDM	Time Division Multiplexing
TTI	Transmission Time Interval
TXOP	Transmission Opportunity
VAP	Virtual Access Point
VM	Virtual Machine
VNE	Virtual Network Embedding
VNI	Visual Networking Index
WiFi	Wireless Fidelity. Commercial name of the technology defined in the IEEE 802.11 set of standards.

Chapter 1

Introduction

In the last thirty years, wireless communications have been established as a commodity and have become essential in the evolution of telecommunications. The number of devices that have wireless capabilities has increased dramatically, going from innate communication devices such as computers or phones to home appliances as televisions and fridges. Even more, wireless access has become the predominant way of connecting to the Internet, which makes it necessary for wireless networks to bear ever-increasing amounts of traffic, from web browsing to video streaming and voice calls. For example, the last update of the VNI report from Cisco [24] predicts that global IP traffic will be three times higher by the year 2022, where 81% of this traffic will proceed from non-PC devices while in 2017 it was of 59%. Moreover, traffic from tablets, smart-phones, and machine-to-machine devices will show the highest increment. So, by 2022, traffic from wireless devices will be much higher than that originated in wired devices. Also, the amount of devices connected to the Internet through wireless will increase dramatically, not only because the number of smart-phones, tablets, and wearables will increase, but also because of new paradigms as Machine-to-Machine (M2M) communications and Internet of Things (IoT), where all kind of electronic devices will have wireless communication capabilities.

These increasing demands have been considered in the design of the recent fifth-generation (5G) of mobile networks, which is expected to support the future requirements of new applications and services. The overall technical challenges of 5G can be grouped in two fundamental directions to where mobile networks and applications would evolve: (i) there will be an explosion of traffic caused by the increment of users and the demand of better user experience; (ii) the appearance of new applications and services, such as Internet of Things which will require ubiquitous connectivity and massive deployment of devices. Therefore, in comparison with current 4G technologies, 5G is expected to provide data rates 1000 times faster, to reduce latency by an order of magnitude, and to support several connected devices 10 to 100 times higher without increasing costs and energy consumption [9, 64].

Even more, in the last years, there has been a trend of increasing heterogeneity in wireless access networks with new types of communications, diverse access technologies, and various offered services. It is expected that in 5G, the heterogeneity increases, requiring new management and control techniques to cope with this diversity. As mentioned, 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. However, it will not be necessary to cope with all these requirements at the same time, everywhere, and for all users and applications. Some applications may need high data rates but not a reduced latency, or some may need extreme latency guarantees with no requirement on data rates. 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, efficient use of network resources can be achieved by individually managing and controlling each use-case [67]. This will be one of the new and different characteristics of 5G, which was not considered on 4G and previous systems.

Therefore, the *Network Slicing* concept has been proposed as a paradigm to enable future 5G networks to support all those heterogeneous requirements. With network slicing, a shared physical network infrastructure can be partitioned into multiple self-contained logical pieces (called *slices*) with customized functions established to meet specific network characteristics and requirements. Furthermore, slicing allows providing better resource isolation as well as increased efficiency of resource usage. Within this paradigm, each slice can be seen as an end-to-end virtual network which allows infrastructure operators to allocate resources to service providers (*tenants*) by creating dynamic and on-demand resource slices. The tenants have complete control over those isolated resources, and they use them to satisfy their client's demands.

As can be appreciated, network slicing encompasses a variety of different functional and performance requirements and can be applied to the entire network, which includes the data center and the cloud, the core and metro wired network, and the wireless access network. This engenders an enormous number of challenges, mainly associated with the virtualization and allocation of network resources. Therefore, this work concentrates on the difficulties of partitioning the Radio Access Network (RAN) into slices with specific performance requirements.

To implement network slicing in the RAN, specific mechanisms to allocate wireless physical resources to the slices are needed. Our research focuses on the design and implementation of resource allocation policies and mechanisms for the wireless edge of the network. However, in the process of designing these policies and mechanisms, it was noticed the lack of precise definitions of the slicing concepts as well as an absence of an accurate problem description and formulation. Therefore, this thesis also elaborates on those issues. In particular, the thesis focuses on the IEEE 802.11 (WiFi) technology, for which slicing has not been thoroughly studied, despite its doubtless relevance, given that 5G networks are being designed to be multi-technology, and they are being deployed to work with already existing infrastructures. In this context, available WiFi infrastructure, which is massively deployed, can be used to leverage 5G capabilities. This thesis intends to overcome this limitation by proposing a solution to implement network slicing in WiFi Access Points (AP). Within this technology, the proposed solution concentrates on slices that demand performance requirements to be guaranteed to its users.

The following sections provide more details about the problem and objectives covered in this dissertation.

1.1 Problem Statement

When implementing slicing on a wireless network, the principal issue to deal with is how to assign the physical resources to the different slices. This is known as the *resource allocation problem* and is well studied by computer science and networking researchers [52, 38]. The particularity of wireless slicing (in comparison with wired networks) is that it includes not only the slicing of the specific wireless hardware but also the radio spectrum that is used as the medium for communication. The difference arises because the wireless medium introduces several problems and challenges that did not exist in the wired domain, for example, loss of signal power because of propagation, interference, user mobility, different radio technologies, and standards.

The assignment of radio resources to slices presents many issues, some related specifically to the wireless networks and others related to the assignment problem itself. In this section, the problem of resource allocation and control in wireless slicing is presented.

1.1.1 Wireless Resource Management

In fixed networks, the resource management problem has gained recent attention in the context of network virtualization. In this context, two main problems are being considered: Virtual Network Embedding (VNE) and Dynamic Resource Allocation (DRA) [75]. The VNE problem consists of solving the mapping between virtual resources and physical resources, while DRA consists of opportunistically adjusting assigned resources based on the traffic demand. Network slicing shares many similarities with network virtualization, and the same problem division can be considered. However, in wireless, because of the wireless channel particularities mentioned before, new challenges are added to both tasks, and also new issues appear.

Regarding the embedding problem, we have identified two new challenges: (i) the heterogeneity and variability of resources and (ii) the variability of the final user's location. In the case of DRA, also new challenges appear: in wireless, not only the traffic load can change, but also the link capacity and the network load (number of connected devices). Hence, the allocation of resources needs to be adjusted considering this new dynamism. Even more, within a single slice, different users can have different link capacities, and then resource allocation has to consider each user's link characteristics. Therefore, dynamic resource allocation becomes essential in wireless networks to guarantee that the embedding is respected, that isolation between slices is guaranteed, and that resources are used efficiently.

Therefore, we also propose to consider those two problems for slicing: the Slice Embedding (SE) and the Dynamic Resource Allocation. Nevertheless, we have also divided both problems (SE and DRA) into sub-tasks that deal with some of the mentioned challenges (see Figure 1.1). The Slice Embedding problem has been divided into the *Modelling of Resources and Requests* task, which includes the definition and modeling of physical resources and resource requests and in the *Mapping Algorithm* task, which maps the resources requested by the slices into physical resources. Dynamic Resource Allocation has also been divided into two sub-tasks: the *Decision Making* and the *Enforcement and Control.* Decision Making is the task of deciding how much resources and of what kind must be given to each slice at each moment, and the Enforcement and Control task covers the aspects of implementing the decided allocation and of controlling that the assignment is preserved. Following, we introduce in more detail both problems and show some of the challenges a wireless slicing solution will need to undertake.

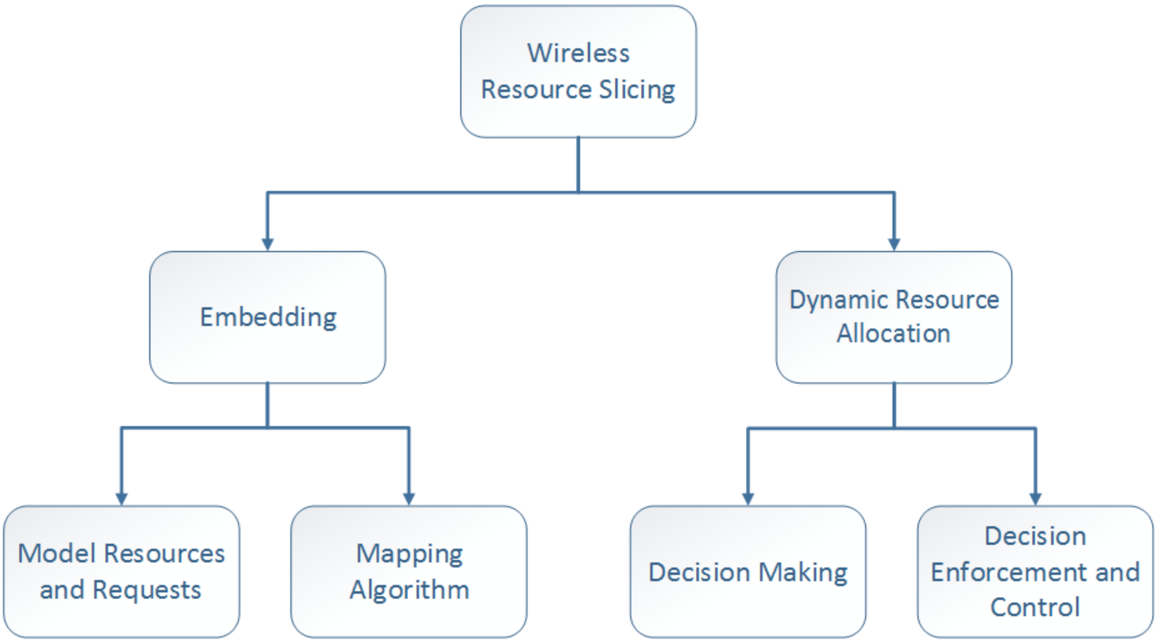
Wireless Resource Slicing
Embedding
Dynamic Resource Allocation
Model Resources and Requests
Mapping Algorithm
Decision Making
Decision Enforcement and Control

Fig. 1.1 Wireless Slicing Problem Taxonomy

1.1.2 Embedding

As stated previously, for the wireless resource slicing approach, we have divided the Embedding problem into two sub-tasks: the *Modeling of Resources and Requests* and the *Mapping Algorithm*.

The definition and modeling of resources and requests consider these aspects:

1. Definition of what the resources are.

2. If necessary, the definition of a model for representing the resources.

3. Selection of a way to partition the resources.

4. Definition of a way for modeling the request of resources.

Regarding aspects 1 and 2, in the wireless domain, what the resources are and how they are modeled could vary depending on the point of view. The resources can be the available radio spectrum (also divided in time, frequency, or space), the available transmission time, or the capacity of the medium. For points 3 and 4, existent works propose different models of the allocation problem, from low-level and highly technology-dependent models (resource-based models) to more general high-level models (throughput-based models) [59]. Detailed analysis of this issue is provided in Section 2.4.

After deciding what the resources are, how they are modeled, and how to model the requests, the next step is to define an algorithm to assign the resources to fulfill the requests, the Mapping Algorithm. The algorithm will be tightly coupled with the defined models and has to consider some important challenges:

- *On-line requests of slices:* Requests can come in different moments.
- *Users location:* The location of the users belonging to the slices is an important restriction as wireless devices can only have access to physical resources on their proximity.
- *Heterogeneous link bandwidths*: It is possible that two devices connected to the same antenna need a different quantity of resources to achieve the same performance.

1.1.3 Dynamic Resource Allocation

As already mentioned, the Dynamic Resource Allocation problem consists of the design of mechanisms to keep or update the assignment in case of changes to guarantee isolation and efficient resource utilization. We divided this problem into two sub-tasks: the *Decision Making* and the *Enforcement and Control*.

The Decision Making is in charge of deciding changes on the allocated resources. For example, it includes opportunistically increasing or decreasing assigned resources based on the utilization, as proposed in [75]. But it must also make decisions based on changes in the link conditions or the number of connected devices. For example, it can decide to deallocate a slice if a sever change in the link capacity makes it impossible to maintain the slices isolated.

Because of the particularities of wireless transmissions, applying the decisions made by the Decision Making algorithm is not trivial. The Enforcement and Control task is responsible for the development of techniques and mechanisms that translate the decisions into actions in the resources. The type of actions and the actual implementation will depend mostly on the wireless technology used, but also on the models and specifications defined by the Embedding task. In general, the context and scope of the enforcement task will be limited to a single wireless device, differently from the decision making the task, which can have a more global view of the network.

Controlling that the resource allocation (and so the slice specification) is preserved is another responsibility of the Enforcement and Control task, which we call the *isolation problem.* The fundamental idea of isolation is to avoid the deterioration of the performance of one slice because of any change on another slice (like the number of devices, flows, channel conditions) or because of the destruction or creation of slices.

1.2 Research Objectives and Approach

1.2.1 Objectives

The main goal of this thesis is to design and develop resource allocation techniques and mechanisms to implement Network Slicing in the context of WiFi access networks. It is expected that the developed mechanisms allocate resources efficiently, that can guarantee different performance requirements of the slices, and that maintain the isolation of the slices.

For achieving this goal, the focus will be given to the problem of dynamic resource allocation and mainly to the Enforcement and Control task. However, for the design of a comprehensive slicing mechanism, some previous objectives have to be accomplished, so we will also consider the resource- and request-modeling problem. The rationale for focusing the research in the Enforcement and Control task is that after a detailed study of existing slicing solutions, it was found a gap between dynamic resource allocation mechanisms and the actual implementation of the allocations in the wireless devices. There exist several proposals for frameworks and management solutions to implement dynamic resource allocation, which assumes that the resource allocations decisions taken can be implemented in the wireless devices. However, this is not entirely true and is a complicated task.

The specific objectives proposed for this research are:

1. To study the resource model possibilities available in the context of WiFi technology as well as to analyze possible models for the slices' requests of resources in WiFi.

2. To design, develop, and evaluate enforcement and control mechanisms for the allocation of radio resources to slices in WiFi.

3. To focus the design, development, and evaluation of the proposed mechanisms in terms of efficiency, quality of service guarantees, and isolation.

1.2.2 Methodology

Overall, this thesis is a feasibility study, with theoretical and empirical parts, intending to answer the following question: is it possible to implement slicing with specific Quality of Service guarantees on WiFi access networks?

For answering this question, we followed a set of steps based on a literature survey, analysis, formal modeling, design, prototyping, and simulation. The obtained results were both qualitative and quantitative.

The research started with a comprehensive analysis of the slicing paradigm and on the main challenges of slicing a wireless network with QoS guarantees. Then it follows with a thorough study of the state of the art in the field of wireless network virtualization and slicing with an emphasis on embedding, resource allocation, and isolation problems. This contributed to the necessary background on the problem, exposing the limitations of existent solutions and providing open research directions.

Afterward, analytical models were developed for the study of the problem, with the focus set on the impact of the WiFi technology on the slicing approach. Later, the strategy consists of following a *proof by construction* method, developing new techniques and mechanisms, and designing and implementing prototype solutions in simulation frameworks. An iterative-incremental process was followed for the analysis, design, and implementation of the solutions.

For the assessment of the proposed solutions, specific evaluation scenarios were defined, to replicate typical scenarios and use cases of wireless networks. Furthermore, evaluation and validation metrics were defined with a focus on the points of view of the network operator and the final user. Evaluation, as well as prototyping, was based on network simulation tools.

1.3 Contributions and Publications

This thesis contributes to the area of resource management and control in wireless networks. Specifically, it proposes new approaches for dynamic resource allocation in WiFi networks for achieving Quality of Service Slicing. The main contributions are:

- A deep analysis of the Network Slicing concept with a focus on the problems and challenges of wireless slicing.

- A survey on wireless slicing approaches, which results as an outcome of the literature review. It contributes with a comprehensive review of existent work and with a discussion of current limitations and open research problems.

- A management architecture for achieving slicing in a wireless network.
- A resource model and allocation mechanism for slicing in WiFi. To achieve the objective of QoS Slicing, as a previous step, we develop a model for resource allocation in WiFi. As a result, we propose a model and a mechanism to partition and allocate the transmission time (airtime) in WiFi Access Points.
- An enforcement and control mechanism for QoS Slicing in WiFi. A mechanism that implements the resource allocation in the wireless hardware, guarantees isolation, and ensures the slice specification is met.

In the process of elaborating this thesis, the following publications were produced:
Journals

- Matías Richart, Javier Baliosian, Joan Serrat, Juan-Luis Gorricho, Ramón Agüero, “Slicing in WiFi Networks Through Airtime-Based Resource Allocation", in Journal of Network and Systems Management, pp. 1-31, Nov. 2018.
- Matías Richart, Javier Baliosian, Joan Serrat, Juan-Luis Gorricho, “Resource Slicing in Virtual Wireless Networks: A Survey," in IEEE Transactions on Network and Service Management, vol. 13, no. 3, pp. 462-476, Sept. 2016.

Conferences

- Matías Richart, Javier Baliosian, Joan Serrat, Juan-Luis Gorricho, Ramón Agüero, “Guaranteed Bit Rate Slicing in WiFi Networks", IEEE Wireless Communications and Networking Conference (WCNC), Marrakech, Morroco, 2019.
- Matías Richart, Javier Baliosian, Joan Serrat, Juan-Luis Gorricho, Ramón Agüero, Nazim Agoulmine, “Resource allocation for network slicing in WiFi access points", 13th International Conference on Network and Service Management (CNSM), Tokyo, 2017, pp. 1-4.

The body of this thesis is strongly based on the contents of the journal publications listed above.

1.4 Thesis Layout

The rest of the thesis is organized as follows. Chapter 2 presents an introduction to Wireless Network Slicing, providing an exhaustive definition as well as the main motivations and enablers for this new paradigm. In the same chapter, a detailed analysis of the wireless slicing problem is provided, and a wireless slicing management architecture is depicted. In Chapter 3 state of the art in wireless slicing is presented, with a particular focus on the resource allocation problems. Chapter 4 proposes a resource allocation strategy for WiFi networks, based on considering the transmission time (airtime) as the assignable resource. In Chapter 5, a mechanism to achieve QoS Slicing in WiFi networks, is presented. The mechanism is built over the airtime allocation mechanism developed in the previous chapter and is based on a scheduling strategy to provide a guaranteed bit rate and guaranteed bounded delay to each flow of a slice. Finally, Chapter 6 concludes the thesis and provides some insights into open research directions in wireless slicing.

Chapter 2

Wireless Network Slicing

This chapter introduces in detail the concept of network slicing, with the focus on the different definitions, the main motivations, and enablers. It also elaborates on the problem of slicing a wireless network and describes the wireless network slicing overall architecture used in this thesis as framework.

2.1 Network Slicing Definition

Network Slicing is a new network paradigm developed within the context of recent 5G networks which proposes the partition of the physical network infrastructure into multiple self-contained logical (or virtual) networks called slices. According to 3GPP [2], a *Network Slice* is a logical network that provides specific network capabilities and network characteristics. Even more, a *Network Slice Instance* is defined as a set of *Network Function Instances* and the required physical resources which compose a deployed Network Slice. During the last years, the network slicing approach has consolidated as a main enabler for 5G networks. As explained in [64], network slices leverage deploying services with contradictory requirements over a shared infrastructure, easing the management of the network. In particular, 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 [33] slicing is proposed as one of the key capabilities of 5G to manage the expected heterogeneous requirements of future mobile networks.

Despite their end-to-end nature, and the intrinsic intention of slices of being deployed to support particular services, seen from the point of view of an AP, we regard a slice as a set of traffic flows with some common features, and demanding some performance requirements[1]. That is complementary to the previous definition, and, as in our previous work [93], is the working perspective for this thesis.

Within this perspective, some slice examples would be: 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. Moreover, a slice can support flows from multiple *clients* (mobile devices connected to the network), 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. As mentioned, this perspective of a network slice is complementary to existing definitions and would help on the design of a network slicing solution.

With network slicing, *infrastructure providers* can lend their network resources to new business players, such as virtual mobile network operators (VMNOs) or Over-The-Top (OTT) services. 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.

Regarding the possible requirements of a slice, in [1] the most relevant network slicing requirements are identified, describing two main types:

- Slices may be tailored to provide different functional requirements like priority, charging or mobility that can be achieved, for example, by implementing only the necessary functions for a slice, while avoiding functions that are not required for that specific service.

- Slices may also have different performance requirements such as latency, reliability, or bit rate.

Nevertheless, because of the novelty of the paradigm, some other approaches to network slicing have also been considered. During the work developed in this research, we have proposed to 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. For example, a slice can be created to assure a minimum guaranteed latency or to provide a minimum guaranteed bandwidth to a given service.

- *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

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

(e.g., Mobile Virtual Network Operator) has complete control over the network infrastructure and network functions within the slice.

Hence, the *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. Although the ISS approach is not explicitly included in the previous definitions given it provides neither performance or functional guarantees, we envision it has potential to be used in some scenarios, as can be seen in recent works which propose a similar approach: [50, 25, 100].

An example scenario for applying the QoS Slicing is given in [32]. This scenario consists of a future 5G network operator offering differentiated types of services depending on the specific use case. For example, a high-throughput service for smart-phones, a low-rate non-critical service for Internet of Things (IoT) or Machine to Machine (M2M) communications and a low-latency service for critical real-time communications. So, the scenario is a combination of these use cases, each one with its specific requirements, and the operator has to jointly provide service and management for all of them. To cope with these requirements, isolated slices are defined, each one giving service to a specific group of users or devices (see Figure 2.1).

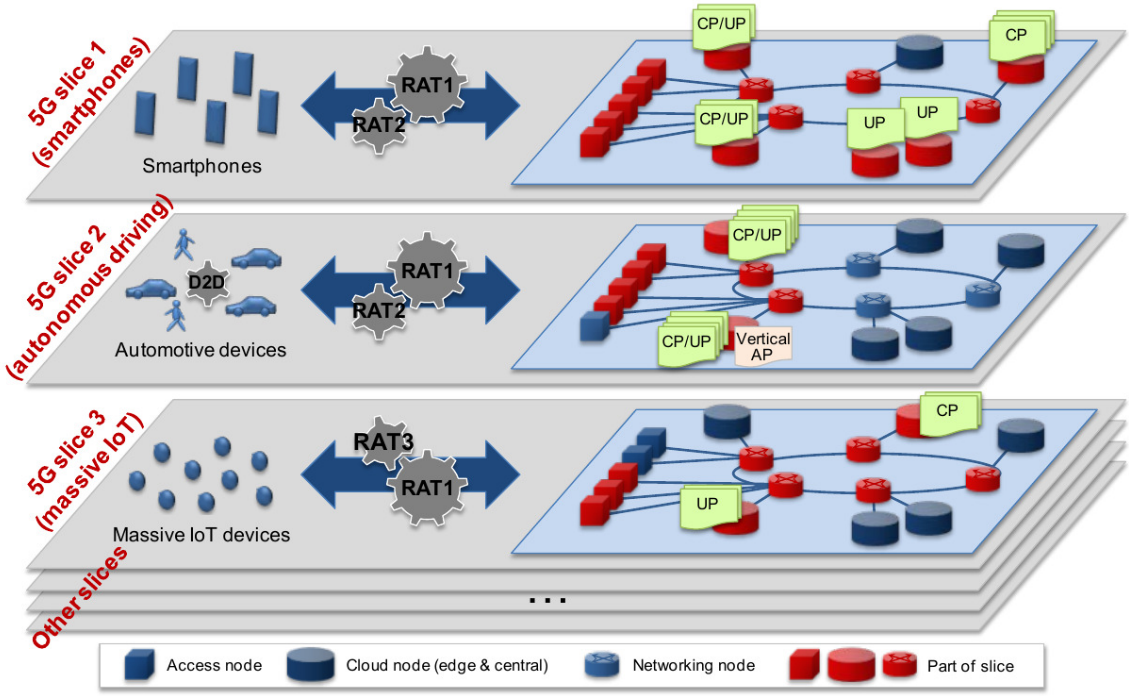

Fig. 2.1 Example of slices in a 5G scenario. Source: [32].

2.2 Motivation

We have already mentioned some of the motivations for implementing slicing on the wireless domain. In what follows, we thoroughly detail the major benefits the slicing approach brings to networks and in particular to wireless networks.

2.2.1 Heterogeneous Service Differentiation

In the current context, where there is a wide variety of services and devices that wireless networks have to deal with, slicing becomes a way to isolate and accomplish different requirements simultaneously. On sharing resources, slicing enables the creation of customized services with fine control features of QoS [27]. The idea is to divide the network into slices made of different resources and capacities to offer differentiated services for heterogeneous use cases. Even more, in [33], slicing is presented as one of the key enablers of future 5G Systems to manage the expected heterogeneous requirements.

It is also possible to define slices for specific applications, which may require customized network capabilities [117]. With virtualization, a layer of abstraction over the slice can be defined to control the network as a *black box* to easily specify application requirements. Another possible approach is to have customized slices per type of device or per type of user requirement. Network slices offer efficient resource utilization as each slice can be customized for a specific service and on a dynamic on-demand way. This dynamism, is the key difference from existing similar proposals as VPNs.

2.2.2 Network Management

As said in [64], “The management of different applications with contradictory requirements on a common infrastructure can be performed via separated network slices”. Slicing the network allows to individually configure the slices *edge-to-edge* and define specific functions for each case, while sharing the same infrastructure and avoiding higher costs. For example, slicing allows to allocate only the necessary functions and to reserve resources on the entire path of the communication, allowing the network configuration to be tailored for each case.

Slicing also provides flexibility to dynamically create and destroy slices depending on the operators policies, with the help of *Network Function Virtualization* (NFV) and *Software Defined Networks* (SDN). The objective is to virtualize as many functions as possible, and those that cannot be virtualized should be programmable and configurable [32]. Even more, in the case of slices defined per type of service or device, as it is known which service each slice is servicing, the network can be simplified by removing functions that are not necessary. For example, if a slice is giving access to static sensors, mobility management can be reduced to a minimum. This way, management is simplified, becoming easier to develop autonomic management for each specific slice.

2.2.3 Heterogeneous Radio Access Technologies

Slicing can also help on the management of networks using heterogeneous Radio Access Technologies (RATs). It is becoming more common to have different RATs working on the same network as a way to alleviate the spectrum scarcity problem. For example, WiFi has become an important player on the mobile business as a way to *offload* data transmissions of mobile devices like smart-phones or tablets. This way, end users avoid the extra cost penalty when exceeding the contracted data usage limit. For instance, in the 2017 VNI report from Cisco [24] it is reported that more than half of data traffic from mobile devices is offloaded to WiFi Access Points. Resource allocation from different technologies can be handled from a slicing perspective where, depending on parameters such as: throughput, user location or costs, the best RAT is assigned to each slice.

The spectrum efficiency can also be improved with slicing, as it is possible to match different requirements to the best available radio resources [113]. To allow this possibility, the network must encompass virtualized or programmable wireless interfaces, as well as different wireless technologies, as expected in future wireless networks. Then, as predicted in [116], the future leads to the coexistence and convergence of different wireless technologies composing a service-oriented infrastructure. Slicing appears as one possible solution to allow this coexistence by simplifying the management.

2.2.4 Infrastructure Sharing

Another important motivation for slicing is infrastructure sharing. It is similar to the service differentiation concept but, in this case, each slice can be used by a different operator offering its own services. For example, in 2014 there were in the UK 41 mobile virtual operators, which are customers of mobile infrastructure providers [87]. Most of them offer similar services of voice, SMS and data as the incumbent operator. Slicing facilitates the infrastructure management and provides isolation between different operators.

From a different point of view, the idea of sharing the infrastructure gives operators more flexibility to change their logical network and efficiently use their resources [122].

This idea is also backed by the Telemanagement Forum [39], which quoted: “The expectation is that 5G will offer multiple virtual networks with different cost/performance characteristics across a shared infrastructure”.

2.2.5 Flexibility for New Services and Business Models

From a business point of view, network slicing promotes the introduction of new use cases without increasing costs thanks to the ability to share the infrastructure by different slices. This can allow to provide service to devices with low traffic demands on highly dense areas (e.g. IoT) without increasing costs, as 5G needs to do. Besides, as a standardized API for programming the network could be offered, slicing leverages the Everything as a Service (XaaS) business model and allows third parties to explore new opportunities.

2.3 Wireless Slicing Enablers

In this section the concepts of *Wireless Network Virtualization* (WNV), *Network Function Virtualization* (NFV) and *Software Defined Networks* (SDN) are briefly described and some existing works on these topics are reviewed, showing their contribution to the implementation of slicing. This concepts are considered fundamental enablers for the wireless slicing purpose.

2.3.1 Wireless Network Virtualization

WNV aims to share a common network infrastructure, including the radio resources, among different virtual networks. We can identify five different goals behind this paradigm:

- The definition of an abstraction layer to simplify the provisioning of wireless access from heterogeneous networks.
- High-level management and programmability of wireless networks.
- Network slicing by service, user or application.
- Infrastructure sharing.
- Radio spectrum sharing.

Wireless virtualization, in comparison with wired network virtualization, encompasses the virtualization of specific wireless hardware and the radio spectrum as well. The virtualization of the wireless medium introduces a number of challenges that do not exist in the wired domain, e.g.: the signal propagation, the interference, the user mobility or the considered radio access technology. All these particularities will be the focus of WNV.

Virtualization of a wireless network can be applied at different layers and degrees, from only virtualizing the core network to virtualizing the radio spectrum and physical layer of base stations. Even more, the motivations for virtualizing a wireless network can be very diverse: from enabling the infrastructure sharing among several operators, to offering a layer of abstraction in order to simplify the network management. There is an extended bibliography devoted to WNV, treating the subject under different perspectives, tackling a specific problem or using a particular technology. The works of Wen et. al. [116] and Liang and Yu [68] offer a comprehensive view on WNV and present existing works on this field.

WNV as an Enabler for Slicing

Virtualization and slicing are two concepts so coupled that virtualization becomes the principal technology enabler for slicing. Nowadays, all slicing proposals consider each slice as some kind of virtual network in order to achieve the objectives behind wireless network slicing.

Some frameworks for virtualizing wireless networks have been proposed in the last years: [97, 47, 78, 88]. In general, these proposals do not provide details about their implementation, but they present candidate design guidelines. These works provide the foundations for a wireless network slicing design. Common to all proposals, there are two requirements wireless virtual networks will need to satisfy:

- the coexistence of different virtual networks mapped onto the same physical network,
- the isolation of the virtual networks to avoid conflicts between coexistent virtual networks.

2.3.2 Software Defined Networking

The main idea of SDN is to decouple the data and control planes, moving the control plane from the network devices to a central location [51]. Then, the forwarding devices (switches or routers) just apply the forwarding rules programmed by a controller element. In this regard, one of the principal technologies used to implement SDN is OpenFlow [72]. OpenFlow is a protocol which specifies the communication between a SDN controller and network switches. In particular it allows the controller to configure the packet forwarding policies in the network switches.

These SDN ideas imply a separation between the network's policies definition, their implementation in hardware, and the forwarding of data. With this separation, considerable flexibility is achieved, which allows a simpler management of the network [61].

Applying SDN to the wireless domain, some works have proposed different designs, frameworks and tools [120, 90]. Many of the SDN proposals have concentrated on decoupling the management from the hardware and the technology, to be able to give a unique interface to control a heterogeneous network. As we will show next, several works have focused on abstracting from the wireless technology and on allowing network programmability, but do not consider the isolation and coexistence of virtual networks over a shared infrastructure. However, both paradigms, SDN and virtualization, seem to be necessary for achieving a complete cross-layer solution to manage, program, share and slice a heterogeneous wireless network.

SDN as an Enabler for Slicing

Deploying and managing a sliced wireless network are complex tasks if not handled correctly. In our opinion, SDN is the necessary tool for easing these tasks, and it is crucial for achieving the needed flexibility and programmability a sliced wireless network will need. Even though SDN does not appear as the solution for the slicing problem itself, hereafter are mentioned some existing ideas of how SDN would be helpful for wireless network slicing.

A good example of how SDN can be helpful is FlowVisor [104]. This slicing tool is designed and used in wired networks to achieve slicing and flow isolation. The FlowVisor architecture consists on a hypervisor which acts as a virtualization layer between the network switches and the SDN controller. Specifically, it enables to have multiple OpwnFlow controllers over the same physical infrastructure. Although in a wireless scenario the problems are different, the main ideas of *flow-based slicing* and *control message isolation* can be used in the wireless domain. In summary, the idea of *flow-based slicing* consists on considering a slice as a set of traffic flows which are controlled by a single OpenFlow controller. Moreover, FlowVisor also visualize and isolate the control channel of OpenFlow so that each controller can only access its own traffic flows configuration.

In the cellular domain, SoftCell [53] focuses on the core network of cellular providers. It proposes the use of the SDN paradigm at the providers network by using common switches and middle-boxes instead of specific proprietary hardware. This approach tackles particular problems of this kind of networks, such as scalability and high bandwidth requirements. In a similar way, MobileFlow [89] proposes an architecture for deploying SDN onto mobile operators, in particular with 3GPP infrastructure. The architecture is called Software-Defined Mobile Network (SDMN) and its main idea is “to provide maximum flexibility, openness, and programmability to future carriers without mandating any changes in user equipment”. In this architecture, the data and control planes are decoupled and the functions of each plane are virtualized.

The works above mentioned are the most direct application of the SDN paradigm on a cellular network. Although, not the focus of our work, the slicing of the backbone of a cellular network is an important aspect of any slicing approach. Increasing the flexibility and programmability of this part of the network is essential.

A more ambitious approach is proposed in SoftRAN [40]. It defines a *virtual big-base station* that logically groups geographically close physical base stations. The idea is that these physical base stations can be centrally managed to facilitate the radio resource allocation and the interference mitigation. For this, the authors propose an abstraction of the radio resources through the virtualization of the physical resources. Also, an Application Programming Interface (API) is proposed to export the state of the network to an external manager which can program the control plane. In this case, the control plane of the wireless devices is decoupled from hardware.

In [35] FlexRAN is presented, which is a SDN solution for RAN slicing. The proposed slicing framework is implemented in the open-source platform OpenAirInterface [86] which is the main difference and advantage from previous works. This allows the solution to be deployed in testbeds and evaluated in real field scenarios. In FlexRAN the control and data planes of base stations are decoupled by providing a novel API (*FlexRAN Agent API*) between a FlexRAN local agent (control plane) and the base station (data plane). The proposed control plane consists of the FlexRAN agents which are instantiated on each base station and a unified global controller which communicates with a custom-designed protocol. This platform is design to allow a flexible and programmable control plane which would enable slicing applications to be easily implemented.

For WiFi systems, some works have been proposed in the last few years, introducing the SDN paradigm to wireless LANs [29]. In [121] it is described an approach for slicing a home network. In this work several of the already mentioned use cases and motivations are discussed for the specific case of slicing a WiFi network of a house. The presented scheme is based on SDN to split a home network into different slices. However, the authors did not provide many details on how the partition is made or the resources shared.

Odin [108, 103] is a framework for enabling the programmability of a wireless network. The most interesting idea is the definition of an abstraction called Light Virtual Access Point (LVAP). This formalizes a logical connection between a client and an AP to maintain the status of the association. The idea is that these LVAPs can be allocated on any kind of hardware (WiFi AP or LTE eNodeB). Then an API is defined to access network parameters and to reconfigure the network. The objective of this approach is to build high-level applications to manage the network. Similar works that extend, use and improve these ideas are: Empower [95] and AeroFlux [101, 102].

The previously mentioned ideas of wireless resource abstraction can be used in a slicing approach to set-up slices and to specify their resources. Also, the decoupling of control from hardware and the centralized management are necessary to develop different control planes for the different slices. In relation to WiFi, SDN approaches are essential for implementing slicing on WiFi networks. For example, the LVAP idea could be extended or modified to be used in slicing, having an LVAP per slice and so, easing the wireless configuration of each slice.

2.3.3 Network Function Virtualization

The main idea behind NFV is the decoupling of network functions from the physical network equipment where they run on [74]. This is achieved by removing their execution from specific hardware and, by means of virtualization, run on standalone hardware, with the additional possibility to be deployed on any location.

Hence, a network service can be decomposed into a set of network functions, which are then virtualized and executed on general purpose servers. This way, the Virtualized Network Functions (VNFs) can be easily created, moved or destroyed, anywhere and at anytime, giving flexibility and lowering costs to the network operator. With NFV, more dynamic and service-aware networks are possible with lower operating and capital expenditures [74].

NFV as an Enabler for Slicing

As already mentioned, we see NFV as an enabler for slicing a wireless network. It will make the creation and management of slices easier to perform if some functions can be taken from proprietary hardware, virtualized and run centrally. In the following, we describe some works where we found ideas that can be applied to slicing a wireless network.

The approach of Software Defined Radio (SDR), where signal processing functions are run in a centralized manner by general purpose hardware is clearly a NFV approach. Cloud-RAN [23, 43] is a SDR architecture with the objective of taking away the signal processing functions from the Base Stations to put them in the cloud. This way, these heavy processing functions can be run on general purpose hardware, and therefore this solution reduces capital costs and promotes the deployment of new technologies. Another example is the work in [44] where the authors apply NFV to the Evolved Packet Core (EPC) of a LTE network. EPC is the core network for LTE systems consisting on a number of entities in charge of functions such as: mobility, routing and forwarding, access control, pricing, etc. The objective of this work is to virtualize the functions of all these entities on the cloud, but grouping some of the functions on the same server to reduce transactions over the network.

CloudMAC [26] proposes to move all MAC processing functions currently run by WiFi APs to the cloud. In this proposal, the functionality of APs is limited to forwarding frames, consequently all the processing is done on a central server where Virtual APs are running as virtual machines. To connect the physical AP (now called Wireless Termination Point, WTP) to the Virtual Access Point (VAP), tunnels and OpenFlow switches are used.

Having network functions decoupled from hardware and grouped at a single location could be a solution to many of the slicing problems we will describe later. For example, with an SDR approach or with the idea of a decoupled MAC from CloudMAC, it could be possible to modify the MAC-layer or PHY-layer implementation to adapt it to new requirements (e.g. prioritizing some traffic over other). Also, with NFV at the core of the network, it would be possible to assign to each slice specific network functions and to remove others to tailor the slice for a specific scenario.

2.4 Analysis of the Wireless Slicing Problem

This section provides more details about the different problems tackled in this dissertation and introduces some concepts that will be used through the document. It is important to note that all the challenges mentioned here are highly dependant on the radio access technology and, in particular, to the medium access technique used, therefore a brief introduction to this matter can be found on Appendix A.

2.4.1 Modelling of Resources and Requests

As already mentioned in Section 1.1, one of the issues to be tackled so as to implement wireless slicing is to define the resources to be assigned to the different slices as well as how a tenant would request resources for a slice.

In the wired domain it is common practice to allocate resources such as link bandwidth or CPU usage. This is possible because the available resources (as link capacity) are fixed and known. However, in the wireless domain the bandwidth of a link between the antenna and the client is unknown as it depends on the wireless channel characteristics. This makes that sharing a resource as link bandwidth in the wireless domain to be much more complex and dependant on variables external to the managed network. Given that sharing bandwidth is not possible under this scheme, the possible shareable resources are limited. Hence, we conceive the available radio spectrum (divided also in time, frequency or space) and the available transmission time (airtime) as the shareable resources.

In the case of LTE cellular networks, different techniques for allocating radio spectrum are used in existent slicing proposals, extending the usual scheduling of LTE Physical Resource Blocks (PRBs) [123]. PRBs are allocation units which represent a portion of the time and frequency available for transmission (more details are given in Appendix A). For example, a tenant can request a given quantity of PRBs avaibale in the transmission device to be assigned to a slice. Then, a PRB scheduler will allocate the feasible PRBs for every slice transmission. This model has the advantage that the requests are given in allocation units, facilitating the implementation of the allocation.

On the other hand, a more high-level model may consider a fraction of resources per request. In this case, there is not an exact demand of a number of PRBs but a relative percentage of resources. In the case of LTE, this would translate easily to PRBs, but this model could also be used for other technologies where resources are quantified differently. Also, an advantage of this more abstract model is that low-level requests (like specific number of PRBs) are difficult to handle from high-level management entities such as operators or service providers, and specifying a Service Level Agreement (SLA) in relation of a specific number of PRBs does not appears adequate.

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 between transmissions. Instead, each node transmits when it senses the medium idle, utilizing the entire assigned frequency band. Therefore, a different approach is to partition the time a resource is used instead of partitioning the actual resource. The idea is that, given an interval of time, the interval is divided into slots to be used by each slice. For example, in WiFi, sharing the transmission time (airtime) is a possible alternative. Airtime sharing implies distributing the transmission time, assigning a fraction to each slice. This approach can also be used for similar technologies, which implement a random access to the medium where splitting resources is much more complex than in cellular networks.

Hence, a slicing airtime-based approach would consist in splitting the transmission time of the device, giving a fraction of the total airtime to each slice. The implementation of this approach involves some complexity. The airtime consumed by a given communication depends on its transmission data-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. Moreover, because of variable data-rate transmissions, there is a minimum assignable airtime which depends on the amount of data being sent and the data-rate used.

The scheduling of transmissions among the connected clients within a slice is also a challenge. It is expected that, while guaranteeing the airtime of the slice, each client receives a fair amount of this airtime. This is difficult considering that each client can have different data-rates and low data-rates consume more airtime than higher data-rates for the same amount of data.

In summary, we envision two different approaches to model the resources and requests for wireless slicing:

- A *resource-based model*, where the tenant demands a set of the total resources for its service. This demand could be as a number of physical resources (such as PRBs) or as a percentage of the total resources.
- An *airtime-based model*, where the tenant demands a ratio of the total wireless transmission airtime for its service.

The availability of these models on a specific network would depend on the physical infrastructure and the technology. For example, as previously mentioned, we do not see feasible implementing a resource-based model in WiFi devices. It is important to mention that all previously mentioned models share their dependency on the channel conditions, and consequently the resulting performance for each slice cannot be guaranteed by just assigning a fixed number of resources.

2.4.2 Analysis of Quality of Service Slicing

As already mentioned, the main objective of this research is to implement a mechanism to provide slicing in WiFi networks with performance requirements. Hereafter, it is analyzed two different aspects of this problem which need to be considered for the development of a slicing solution.

Wireless Medium Challenges

In wireless networks, achieving performance guarantees is challenging mainly because of the variability of wireless links' capacity and because of limited resources. In wireless communications, the capacity of the link depends on the Signal-to-Interference and Noise Ratio (SINR) and on the available bandwidth of the link. The SINR can be understood as the power with which the signal arrives to the receiver and depends mainly on the transmission power, the attenuation of the signal while it is transmitted through the wireless medium, the noise and the interference. As lower is the SINR of the signal at a receiver, lower is the maximum data rate at which the receiver can decode a signal[2]. Hence, for a given wireless communication the transmission data rate is variable over time due to, for example, the distance between the transmitter and receiver, the location of the interferers or the obstacles in between the communicating nodes.

Besides, the available radio spectrum is not only limited by the technology but also it is regulated. This implies a bound on the bandwidth of the wireless channel that can be used and limits the possibilities to be increased, in contrast to the usual deployment of wired networks.

Therefore, given that there is an upper bound on the capacity given mainly by the technology, one of the main parameters which determines the obtained throughput of a wireless communication is the rate at which the data can be transmitted. This parameter is not directly controllable, is variable on time and depends mainly on the channel conditions. Besides, current wireless technologies include an adaptation mechanism which selects the best Modulation and Coding Scheme (MCS) (which defines the data rate) for the given channel conditions so as to minimize errors at the reception of the data.

Consequently, to guarantee any performance requirement which requires to control the amount of data to be transmitted in a given period of time (such as throughput), the main parameter to adjust would be how much of the available resources to use. As

[2] A more comprehensive explanation of this matter is given in Appendix A

previously analyzed, this resources would depend on the technology. In LTE systems the main parameter to adjust will be the number of resource blocks, and in WiFi systems the transmission time (and so, the access to the medium).

In summary, the achievable performance of a wireless communication will depend on:

- *The channel capacity:* which depends mainly on the channel characteristics between the sender and the receiver (the distance, the obstacles or the interference).
- *The network load:* the number of clients associated to the transmitting device will influence how the resources can be allocated into the different clients.
- *The configuration and characteristics of the system:* which indirectly affects the channel capacity, this includes for example: the number of antennas used, the link bandwidth or the data rates available.

All this parameters will be information inputs for a solution which allocates resources to guarantee any performance requirement.

Quality of Service in Wireless Slicing

As introduced in Section 2.1, in the QoS-enabled Wireless Slicing approach, tenants require slices with some Quality of Service performance objectives, which should be provided by the network infrastructure. QoS can be expressed through different indexes, 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 be taken in the design of this slicing variant 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.

Our approach is to consider the QoS requirement of a slice as the QoS guaranteed to each flow within the slice. We deem this approach as more interesting for a tenant which asks a QoS guarantee for its users. This is consistent with current design definitions in 5G, where the concept of QoS flows is introduced [3]. However, there exists other approaches in the literature where performance guarantees are requested for the entire slice and not for each flow [57].

Another important design aspect in 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. Furthermore, a policy that defines the actions to take when such requirements cannot be met is also desirable. There are two main reasons for this: first, in wireless communications it may happen that the capacity of a 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. Another example could be a delay requirement, which demands the guarantee to be enforced in a percentage of the total transmitted packets and not on every packet: “The delay must be at most of 50ms for the 90% of the transmitted packets".

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 user connection should be dropped, notifying the service provider. This is very important to guarantee isolation, as clients with low channel capacity can affect the performance of all the connected clients.

A different approach, which in some extent narrows QoS Slicing is to limit the resources a QoS request can use. In this case QoS requirements are respected until a threshold on the used resources is reached. For example, the negotiated slicing SLA can define that a QoS requirement will be guaranteed while the necessary resources for that requirement are less than the 20% of the total resources of the device (or network). This type of SLA could be of interest for a network operator as it eases the resource allocation and management and does not depend so strongly on the behaviour of the environment or the users.

Finally, it is important to note that in QoS Slicing defining SLAs to achieve the requirements imposed by the expected slicing use cases is not trivial. As the slicing concept involves the network end-to-end, it is necessary a deep knowledge of networking to specify the performance requirements at each part of the network. Hence, beyond the already mentioned characteristics of slicing, an important aspect that should be implemented is automation. This means that the slice should auto-configure and should be capable of autonomously defining the requirements for a given use-case.

Isolation

Controlling that the resource allocation (and so the slice specification) is not violated over the time is another difficulty for any slicing approach. We call this the *isolation* *problem.* The fundamental idea of isolation is to ensure that slices do not disturb each other, avoiding performance degradation or appropriation of resources allocated to any slice despite of any change on other slices (like the number of connected clients, traffic flows or channel conditions) or because of the removal or set-up of slices. The complexity of assuring isolation in wireless networks appears because of the high variability of the channel conditions and because of users' mobility.

The isolation problem can be more or less difficult to solve depending on how resources are modeled and on the used wireless technology. Because of the different models, isolation can be interpreted as the maintenance of the assigned resources or as the maintenance of the requested throughput or bandwidth, despite of changes on any other slice. Also, in some slicing approaches, isolation is implicit to the resource allocation solution, for example, if while assigning resources it is guaranteed that there are no overlapping of resources.

Special consideration must be taken in the implementation of QoS Slicing, as clients with low channel capacity can indeed jeopardize the performance of the whole network by hoarding all the resources. The approach mentioned before, which imposes a limit on the resources assigned to a slice and also to a QoS flow, would help to provide QoS Slicing but avoid isolation issues.

In summary, an important aspect of wireless resource slicing is how to keep satisfying the requests in spite of this variability. Nevertheless, the isolation issue is the less treated aspect in the literature in spite of being one of the major challenges of research in this field. Dynamic resource allocation techniques which use the information of the channel conditions to take fast and accurate decisions will be needed.

2.5 A Slicing Management Architecture

Network slicing involves the slicing of both the core network and the Radio Access Networks (RANs) to achieve an *end-to-end slicing.* Hereafter there is a description of a simple architecture for the management of network slices over the RAN. It is important to note that we avoid implementation details of this architecture as it is out of the scope of our work. This proposal can be considered as the necessary framework for our resource allocation solution presented in the following Chapters.

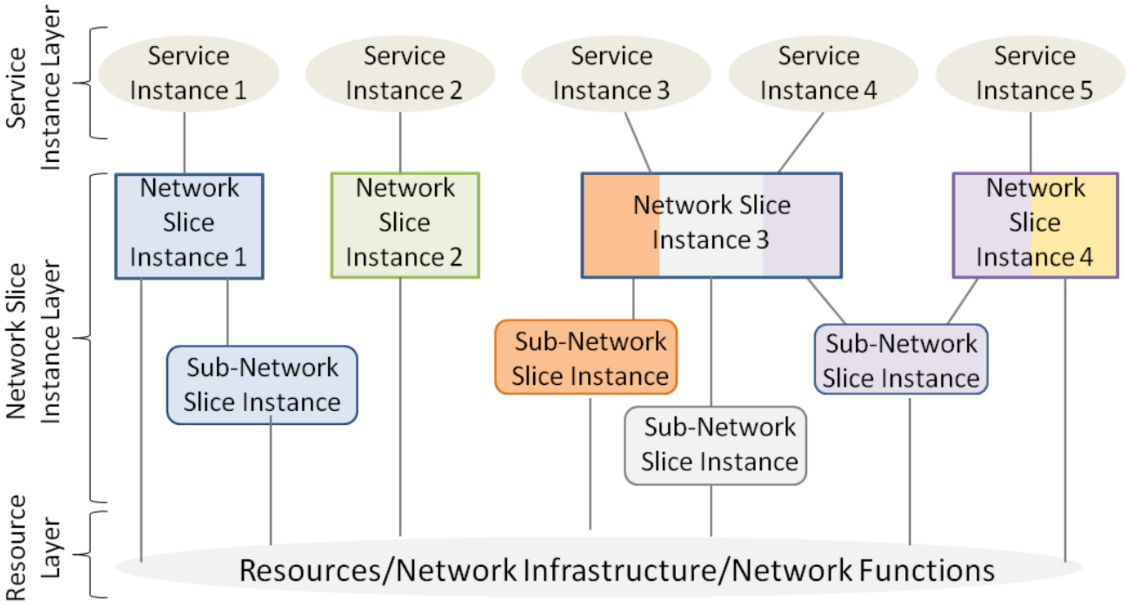

Fig. 2.2 NGMN Slicing Conceptual Outline. From [6].

2.5.1 Slicing Conceptual Outline

This architectural proposal is based on the *Network Slicing Conceptual Outline* introduced by the Next Generation Mobile Network (NGMN) Alliance where 3 layers are defined: the *Service Instance Layer*, the *Network Slice Instance Layer* and the *Resource Layer* [84]. The Service Instance Layer represents the services to be supported. A *Service Instance* could be any service provided by a tenant that needs a slice. A Network Slice Instance (or simply a *slice*) represents the network resources and network functions that form a complete end-to-end logical network defined to support the requirements of a Service Instance. The Network Slice Instances can be shared by multiple Service Instances. A Network Slice Instance may be composed of *Sub-network Instances* which are smaller constructs which also represent a set of resources and network functions. This separation in network and sub-network instances allows more control on specific parts of the network as a Sub-network Instance does not need to form an end-to-end self-contained logical network. It also enables to share Sub-network Instances among different Network Instances and also to deaggregate a Network Instance in several Sub-network Instances. Finally, in the Resource Layer there are all the physical resources of the network infrastructure and the network functions.

Network and Sub-network Slice Instances are defined by *Blueprints* which describe how to configure, instantiate, deploy and control an Instance. A *Network Slice Blueprint* includes the required network resources and functions as well as specifies the Sub-network Blueprints if necessary.

2.5.2 Management Architecture

Given the conceptual outline, we propose a management architecture for an infrastructure provider which manages a RAN with several devices and diverse technologies. We envision three levels of slice management and control:

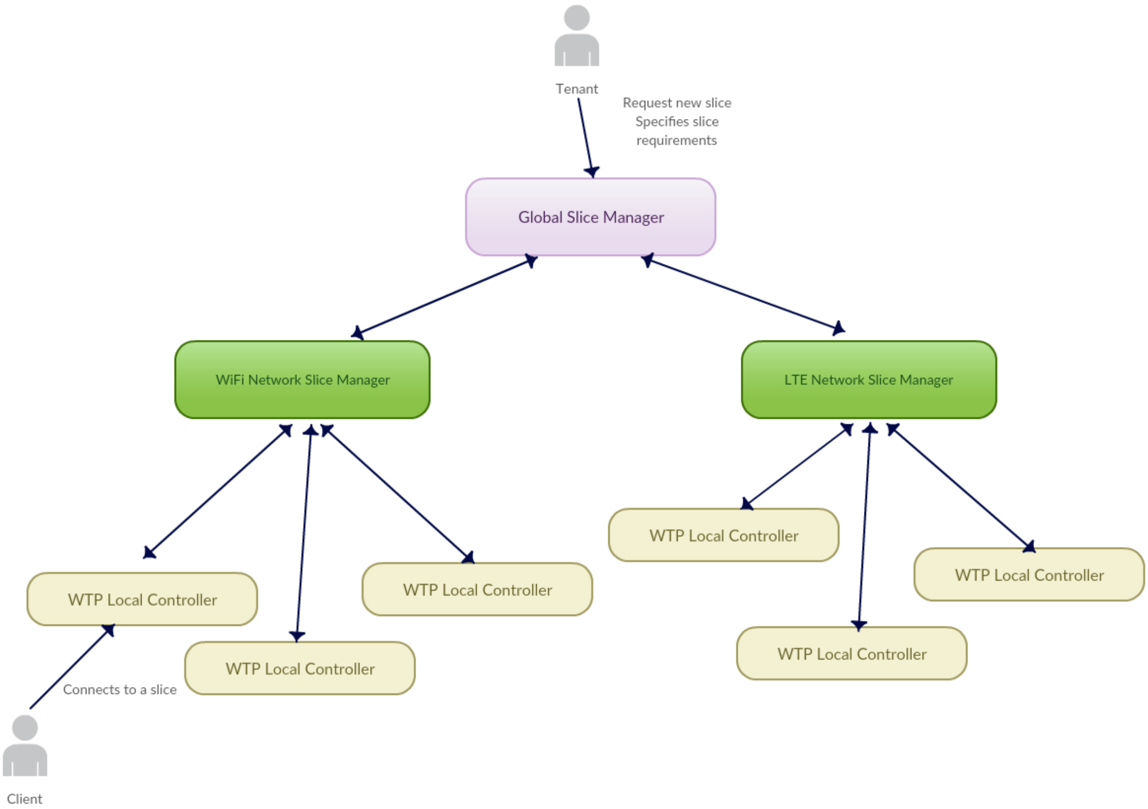

Fig. 2.3 An Example of Our Management Architecture Proposal.

- A *Global Slice Manager*, which is in charge of the orchestration of the entire system, managing the different RAN devices. It administers the Network Slice Instances defining the necessary Blueprints from the tenants Service Instance requirements. For example, it decides (from the entire infrastructure which includes different technologies and locations) the necessary network resources and functions to create the slices. It may also make admission control decisions given the available resources.

- A *Network Slice Manager*, which has a global view of a network specific technology (or any other logical partition of the RAN), for example the WiFi network. Given a Sub-Network Blueprint, this manager is in charge of instantiating a Sub-network Slice Instance. For example, it is in charge of allocation decisions, choosing the WTPs[3] where slices are to be created. In addition, it can move slices between WTPs using global information of the network such as available resources and user locations. This manager also controls admission of slices, informing the *Global Slice Manager* if a slice can be accepted or not in the network.

- A *WTP Local Controller*, is in charge of the enforcement and control of the slices in the wireless device. For example, given a resource requirement by a Network Instance it conducts the actual allocation in the physical device.

As can bee seen, the proposed three-level architecture provides control with different granularities. Fine-grained control is done at the WTP Local Controller, which are technology specific and can use low level knowledge for the control. On the other side, the Global Controller makes high-level decisions, tuning the parameters of the Network and WTP Controllers.

Moreover, the three proposed entities will also perform *admission control.* The objective of admission control at the three levels will be to maximize network utilization while respecting the requested performance objectives of each slice. All three managers also implement monitoring functions to control the performance objectives.

2.5.3 Managers Interactions

So as to illustrate how this architecture would work, hereafter we analyze three possible use cases.

Slice creation and specification:

1. The tenant requests a new slice and specifies the traffic characteristics and QoS requirements.

2. The Global Slice Manager evaluates the request, decide if enough resource are available and informs the decision to the tenant (a many-steps negotiation between the tenant and the Global Manager is possible).

[3] WTP, Wireless Termination Point. Is the wireless device that provides access to the network like Access Point in WiFi or eNodeB in LTE

3. If the slice is accepted, it is informed to the Network Slice Managers. The Network Slice Managers receive the slice Blueprint and keep a list of current deployed slices (the slice is not instantiated until a client connects to the network).

Client connection to a slice:

1. A client requests a connection to a slice.
2. The request is received by a WTP Local Controller.
3. The Controller informs the Network Slice Manager of the connection request. It also sends information of the current available resources and information of the client channel capacity.
4. The Network Slice Manager decides if the connection can be accepted. The connection can be accepted in the device which received the request or in any other WTP.
5. The Network Slice Manager mandates the WTP Local Controller to create a new slice connection for the client and specifies the performance objectives to control or the resources to assign.

Slice decommission:

1. A WTP Local Controller detects it can no longer meet the requirements of current connections and informs the Network Slice Manager.
2. The Network Slice Manager decides to remove one or more connections from the device.
   (a) If there is a WTP in the same network which can host the connection/s, it moves the connection/s to the new WTP.
   (b) If no more resources are available in the network, it informs the Global Slice Manager.
3. The Global Slice Manager decides to move one or more connections.
   (a) If there is a WTP in another network which can host the connection/s, it moves the connection/s to the new WTP.
   (b) If no more resource are available in the system, selects one or more connections to drop, and informs the tenant/s.

2.6 Conclusions

In the first section of this chapter, we provide a comprehensive definition of slicing and in particular of wireless slicing. Given the novelty of the slicing paradigm and the evolution of the term in the last years, there exist several definitions that provide different views of the paradigm. Therefore, we have collected and summarized the main definitions to clearly understand the concept of slicing as an integral part of future 5G networks. In Section 2.2, we analyze the main advantages and motivations for implementing slicing in a wireless access network, while in Section 2.3, we review and summarize existent works on current technological trends which are fundamental enablers for slicing. In this regard, this revision contributes to a better understanding of the slicing paradigm and how it could be used in a wireless network.

Complementarily, the objective of Section 2.4 and Section 2.5 is to provide the necessary context for the solution proposed in this dissertation and to present our work on the modeling and analysis of the slicing problem.

In this respect, in Section 2.4, we contribute with an analysis of the different possibilities for modeling resources to be allocated to slices as well as on how the slices can request resources. We conclude this analysis with a proposal of two different models: a *resource-based model* and an *airtime-based model.* As a second step, we analyze the main challenges for slicing a wireless network with Quality of Service guarantees. In particular, we describe how the wireless medium (and mainly its variable capacity) affects and complicates the implementation of slicing. Moreover, we reason on the different decisions that must be taken when designing and implementing QoS slicing, emphasizing on the two major problems of slicing wireless resources: resource allocation and isolation. We believe that this thorough analysis of the different aspects of slicing is an important contribution to the correct design and implementation of a QoS Slicing solution. Finally, in Section 2.5, we depict a simple proposal for an architecture where our slicing solution would work. As will be clear in the following Chapters, this architecture provides the working context of our slicing solution and clarifies the interaction of our solution with other parts of the network.

Chapter 3

State of the Art

Facing the issues and complexities we have mentioned regarding the implementation of an efficient slicing, some ideas and mechanisms have been proposed in the last few years. In this section, we review these proposals and explain their characteristics, advantages and drawbacks. We focus on the works that deal with low-level resource allocations and propose ideas related to the Enforcement and Control approach previously defined. A summary of these works is given in Table 3.1. In this table, we classify the works based on the considered technology and summarize their main characteristics and objectives.

3.1 A Classification of Current Solutions

Current proposals for resource allocation and isolation of slices in wireless networks are significantly dependent on the wireless technology, focusing on 3GPP LTE or IEEE 802.11 (WiFi) standards. Although the focus of this thesis is in the WiFi technology, we also review proposals for cellular networks as we consider that many of those solutions provide interesting ideas about resource allocation and slicing. Even more, because of it similarities, we also review works on resource allocation for network virtualization, which can be considered as a precursor of network slicing. In Appendix A we provide a description of the main characteristics of both technologies.

For the case of LTE (or other cellular technologies such as WiMAX) the vast majority of approaches modify the frame scheduler to assign Physical Resource Blocks (PRBs) to the slices (*PRB scheduling*). Some works, trying to avoid such a low-level strategy, propose mechanisms which schedule the use of resources between slices in a higher layer. This approach is generally made at the MAC-layer or the Network-layer, we call it *Slice scheduling*. The third category is *Traffic shaping*, controlling the traffic (packets) that is sent to the scheduler, with the implicit intention to modulate the schedulers' behavior. So, for LTE three strategies prevail:

- *PRB scheduling*
- *Slice scheduling*
- *Traffic shaping*

For WiFi, similar approaches are proposed. In our review, we identify three predominant strategies:

- *EDCA control*: This strategy adjusts the EDCA parameters (*Contention Window*, *Arbitration inter-frame spacing* and *Transmission Opportunity*) of the MAC layer to control how the slices access to the wireless medium.
- *Slice scheduling*: These strategies consist of scheduling resources usage among the slices. For example, scheduling transmission opportunities. The idea is to have each slice as a virtual machine over the physical AP and schedule the use of transmission resources between the slices.
- *Traffic shaping*: In this case, the traffic coming from different slices is shaped before sending the data to the MAC-layer to conform performance requirements.

In Table 3.1 we explicitly mention which of these classifications are used by the reviewed works along with their main objectives. In fact, some works only focus on achieving some grade of slicing, others on resource allocation or resource embedding while others only on isolation. Also, the earliest works on the area of slicing were focusing on experimental testbeds, hence, the use cases and objectives of these early works are different from present approaches.

Table 3.1 Summary of Wireless Virtualization and Slicing Proposals

	Technology	**Objective**	**Resource Allocation and/or Isolation Mechanism**		**Implementation**
			Uplink	**Downlink**	
Xia et. al. [118]	WiFi	Seamless wireless virtualization	None	None	WiFi device with several hardware modifications
Aljabari et. al. [8]	WiFi	Open-source wireless virtualization	None	None	Real Wifi device
C-VAP [14]	WiFi	Slicing Airtime fairness	EDCA parameters	None	Matlab simulation
Nakauchi et. al. [79]	WiFi	Slicing. Airtime control.	EDCA parameters	EDCA parameters	Simulation in QualNet
ViFi [41]	WiFi	Slicing. Airtime control.	EDCA parameters	Slice scheduling and traffic shaping.	Real, WiFi prototype
Derakhshani et.al. [28]	WiFi	Airtime control	EDCA parameters	None	Matlab simulation
Smith et. al. [106]	WiFi	Isolate experiments in testbed	Slice scheduling	Slice scheduling	Real in ORBIT testbed
Katsalis et. al. [57]	WiFi	Guarantee throughput ratios	None	Slice scheduling	Real, Wifi prototype
Mahindra et. al. [69]	WiFi	Testbed. Multiple concurrent experiments	None	Traffic shaping	Real in ORBIT testbed
Virtual WiFi [118]	WiFi	Client virtualization	None	Slice scheduling	Real, WiFi prototype
Zaki et. al. [123, 124]	LTE	Assign resources based on predefined contracts	None	Scheduling PRBs	OPNET simulation
Li et. al [66]	LTE	Load balancing in multi-cell slicing.	None	Scheduling PRBs	OPNET simulation
Karnaugh-map embedding [119]	LTE	Embedding requests	None	Scheduling PRBs	None
Dynamic embedding[114]	LTE	Dynamic embedding	None	Scheduling PRBs	Simulation
Kamel et. al. [56]	LTE	Optimal resource allocation. Fairness among users	None	Scheduling PRBs	Matlab simulation
Ksentini & Nikaein [62]	LTE	RAN Slicing. PRB virtualization.	None	Two-level PRB scheduler	Real, OAI prototype
Orion [36]	LTE	RAN Slicing. PRB virtualization.	None	None	Real, OAI prototype
NVS [59]	WiMAX	Slicing framework	Slice and flow scheduling	Slice and flow scheduling	Real, WiMAX prototype
Guo & Arnott [42]	LTE	Resource sharing	None	Slice scheduling	Simulation
Jumba et. al. [55]	Generic OFDMA	Slicing. Minimum bit rate per slice	None	Bandwidth allocation	Simulation
Malanchini et. al. [70]	LTE	Resource sharing among virtual operators	None	Assign ratio of resources	Simulation
Tang et. al. [110]	Generic OFDMA	Slicing. Minimum bit rate per slice	None	Bandwidth allocation	Simulation
CellSlice [60]	WiMAX	Slice isolation	Slice scheduling and traffic shaping	Sustained rate control	Real, WiMAX prototype
Virtual Basestation [18]	WiMAX	Slice isolation	None	Traffic shaping	Real, WiMAX prototype
VNTS [17]	WiMAX	Airtime fairness	None	Traffic shaping	Real, WiMAX prototype

3.2 WiFi Slicing Solutions

In this section, we detail existent works dealing with the slicing problem for the WiFi technology. As explained previously, because of the distributed nature of the medium access control, these solutions have to deal with problems different from those of cellular networks. At the end of the review, we briefly discuss the advantages and drawbacks of each approach.

Most of the current works which deal with resource allocation for slicing (or virtualization) in WiFi consider traffic flows as the entities to assign resources. In those works, the goal is to share physical devices at the flow-level and do not implement resource allocation specifically. In this context, [4] introduces the idea of Virtual Access Points (VAPs), which is used by many following works on this area. For example, [118, 8] 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).

3.2.1 EDCA Control

Control-theoretic optimization of Virtual APs (C-VAP) [14] is a control-theory approach for adjusting the *Contention Window* (CW) of the clients in a sliced WLAN to provide optimized throughput and fairness to virtual slices. The slicing mechanism uses a Proportional Integral (PI) controller adapting the CW parameter of each client. This way, the mechanism achieves the same throughput in all slices independently of the number of associated clients. Although the authors present a complete formal approach to the problem, the solution does not provide any type of guarantee of throughput or airtime to each slice, just fairness among them.

In [79] a mechanism is presented to create VAPs over a physical AP, in such a way that each VAP has its own MAC transmission queue and virtual machine. Thus, there is a set of EDCA parameters specific to each queue, enabling an airtime-based mechanism to isolate the slices. The mechanism adjusts the parameters of each queue (each slice) and the parameters of the associated clients, all based on previously defined requirements for each slice and on the number of devices in the network. The mechanism allows the definition of target airtime ratios for each VAP and adjusts the CW_{min} to achieve those ratios. Although the solution is said to consider variable rates, this is not shown in the simulations done. Besides, the description of the algorithm is not clear about the control mechanism of the CW_{min} parameter and the proposal requires several modifications on the devices to implement the virtual APs.

The work in [41] (ViFi) proposes a similar idea to that found in [79] for the uplink traffic, and a slice scheduling mechanism for the downlink traffic. The uplink control mechanism configures two EDCA parameters, the CW_{min} and the transmission opportunity τ for each client. The authors argue that the joint control of both parameters provides better fine-grained tuning of the throughput. The final goal of the control mechanism is to guarantee pre-established airtimes for the uplink flows of each client. For the downlink, it uses a scheduler which, in a first stage, schedules packets per-slice depending on the requirements of each slice and, in a second stage, schedules packets per-user of each slice in a round-robin manner. In this work, the management of variable rates is peculiar. There is no real isolation, when the rate drops on one client, the throughput is increased on other clients of another slice. The evaluation is made on a real implementation, which is an important progress from previous works. However, more extensive evaluations, where a more dynamic scenario is tested, would be interesting. Also, an evaluation of the convergence time of the algorithm and the airtime usage would be needed.

The work in [28] 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 focuses 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 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 a description of 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. Besides, in the article it is not clear how the wireless capacity variability is managed and the optimization appears to be based on a static deployment of clients.

3.2.2 Slice Scheduling

In [106] the authors propose a mechanism for isolating experiments over a shared WiFi infrastructure. The objective is to separate a testbed of APs into independent virtual testbeds to be able to run simultaneous and isolated experiments. The isolation is achieved by a mechanism similar to Time Division Multiplexing (TDM) where different experiments are allocated in separated time slots. This approach has two major drawbacks, the synchronization for enabling and disabling all virtual nodes of one experiment, and the context switch cost at the devices when switching between different experiments.

Virtual WiFi [118] is a proposal designed for *client virtualization*. It tackles the problem where a client runs several virtual machines (VMs) on its device and these virtual machines handle the connections to an AP independently. The objective is to have many VMs within a single device with a single wireless interface connected to different networks simultaneously. The work contributes with a very valuable analysis on the problems of sharing and slicing a wireless interface: the support of all the functionalities of the wireless interface inside the VMs, and the ability for each VM to establish its own connections with its own credentials. The proposed architecture enables access to the physical interface from inside any VM to have the same management functions and to create isolated connections. Nevertheless, it needs to modify the device driver at the host and the firmware of the wireless card. The issues tackled in this work are important for full wireless virtualization where slices can access low-level wireless functions.

The work of Katsalis et. al. [57] also falls in the *Slice Scheduling* category. In this work, it is proposed a scheduling mechanism with feedback control 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 of 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 to account for the current transmitted bytes.

3.2.3 Traffic Shaping

In [69] an empirical comparison of different approaches for isolation in concurrent experiments on a shared testbed is conducted. In particular, the authors study the efficiency of space and time isolation and conclude that no mechanism gives sufficient isolation if the bandwidth of the different slices is not controlled. Then, a mechanism for traffic shaping and admission control is proposed to enforce slices to specific bandwidths. No details are given on how the bandwidths are selected or how the admission control mechanism takes decisions.

SplitAp [19] is a proposal to assure isolation for the uplink traffic in a virtual WiFi network. The method applies traffic shaping on the client side based on commands sent by the AP. For this, special software has to be installed on the client: a traffic shaping module and a control and reporting module. The control and reporting module is responsible for two tasks: reporting usage parameters (as the MCS and packet size) to the AP; and controlling the shaping module. The algorithm at the AP uses the information sent by the clients to estimate the uplink airtime usage of each slice, and if the predefined policies for each slice are not kept, it broadcasts a command to adjust the airtime usage at each client. However, in the article it is not explained the mechanism at the client which takes the command sent by the AP and converts it to a traffic shaping rate. In our opinion, this would be the actual slicing mechanism. Besides, as other similar works that control uplink traffic, changes are needed at the client-side, an approach that currently appears unfeasible.

3.2.4 Discussion

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. For example, in most devices the EDCA parameters are coupled to the hardware queues, and the number of those hardware queues is fixed. Even more, there is a total lack of analysis of the real feasibility to adapt the EDCA parameters on the hardware. The possible values the EDCA parameters can take is limited and this is not considered, neither is the time granularity to which these values can be modified. The variability of the channel conditions is also not well considered, for example, the convergence time or accuracy of the mechanisms are important metrics to show how fast the algorithms can adapt the resource assignment to new channel conditions.

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 to use resources efficiently. This cross-layer communication is not always easy to implement if access to the firmware is not available. Moreover, existent slice scheduling techniques do not consider isolation but only tackle sharing problems.

3.3 LTE and WiMAX Slicing Solutions

In the following, we review the main proposals to the slicing problem in the context of cellular networks. We focus on the two most popular technologies in the literature: LTE and WiMAX. Although WiMAX has lost attention and LTE is evolving with the introduction of 5G systems, these solutions are an interesting approach to the problem. After the description of the proposals we briefly compare the different approaches. Similarly to the case of WiFi, many of the reviewed works are previous to the slicing concept and deal with the network virtualization problem.

3.3.1 PRB Scheduling

Zaki et. al. [123, 124] present a framework for LTE virtualization. The authors propose an architecture for virtualizing the LTE Base Stations (called *eNodeBs* (eNB) in LTE architecture) with the objective of having different operators sharing the same physical resources. The solution is based on a Hypervisor (as in CPU virtualization), which hosts virtual instances of eNBs, allocates the resources and is responsible of the spectrum sharing and data multiplexing. The Hypervisor will accomplish two tasks: (i) it will host several virtual eNBs onto a physical eNodeB, scheduling the physical resources among them; (ii) it will schedule the wireless resources among the different virtual eNodeBs. For this second task, the solution uses the Physical Resource Block (PRB) as the minimum resource granularity that can be allocated, and assigns them among the different virtual eNBs, instead of among the users (as done typically by a scheduler). The PRBs are scheduled to the different virtual eNodeBs based on previously arranged contracts, which specify different guarantees for the operator owning a virtual eNodeB. The contracts can set different PRBs allocation policies:

- a fixed amount of PRBs,
- a maximum amount of PRBs to be allocated dynamically according to the current estimated demand or,
- a best effort allocation with no guarantees.

After the Hypervisor allocates PRBs to the virtual eNBs, each virtual eNB allocates the PRBs to its users.

In this work, a comparison between the fixed and dynamic approaches is done using the OPNET simulator. For showing the benefits of the approach it is assumed that multiple operators have their traffic peaks at different moments of time. In our opinion, this assumption is unrealistic or not well founded. Also, although the scheduler handles the coexistence of different eNBs over a shared physical eNB, the mechanism does not offer explicit isolation. If the demand exceeds the available resources, the assignment is reduced proportionally. If we consider that each eNB could represent a slice, this solution would not respect the slicing isolation requirement.

In [66], the framework from [123] is used and extended through a more detailed algorithm for scheduling PRBs for the virtual eNBs. The objective of the solution is to dynamically allocate PRBs based on the estimated demand of the virtual networks. The demand is estimated separately for Guaranteed Bit Rate (GBR) traffic and non-GBR traffic. The allocation goal is to satisfy the GBR demands and then to allocate PRBs to non-GBR traffic proportionally. If the total demand overloads the amount of available resources, the assignment is done proportionally to each virtual network demand. An interesting aspect of this solution is the addition of a load balancing mechanism to distribute the load of a virtual network among different eNodeBs (in a multi eNB scenario). The idea is that if a eNB is overloaded and a neighbor eNB has available resources, a user is selected to be migrated to the unloaded eNB.

The Karnaugh-map-like Embedding Algorithm (KEA) considered in [119] deals with the problem of embedding virtual wireless networks (slices) requests onto the physical wireless resources. Specifically, this work concentrates on the allocation of resources to slices when the requests come along the time (on-line requests). This approach presents some obvious drawbacks when compared to resource allocation with all requests arriving at once (off-line requests). To handle this dynamic scenario, requests are grouped within a time window, and the embedding is done at the end of each window. The spectrum is modeled as a two-dimensional (frequency and time) grid of assignable resources and an algorithm based on Karnaugh-map is used to make the assignment. In the article there is no explanation on how this mechanism would be implemented in real hardware or in current wireless technologies. For example, how this can be implemented by the LTE scheduler seems complicated.

In [114] an extended version of the previous mechanism [119] is presented. The proposal in this work considers *dynamic embedding* to avoid requests rejections due to topological constraints. This type of rejections happen when, sufficient resources (time-frequency slots) are available for a request, but due to the arrangement of existing assignments in the 2-D grid of time and frequency, there is not enough contiguous space for that request. The mechanism of dynamic embedding rearranges the assignments on each time slot, giving priority to already assigned requests. Through simulations it is shown that with dynamic embedding the number of rejections is significantly reduced in comparison to a static embedding. Although an interesting approach, it suffers the same problems as in [119]. Even more, this mechanism can seriously affect the scheduling times if too elaborate calculations are needed every time.

Kamel et. al. [56] propose a scheduling mechanism to slice an LTE network into several virtual networks owned by different Service Providers (SP). For each SP, a contract is agreed, which defines the minimum amount of PRBs that will be assigned. Differently from previous works, in this case the scheduler assigns PRBs to users (as LTE generally does) but it is modified to follow a specific optimization strategy to allow slicing. The optimization problem objective is to maximize the transmission rate obtained by each user subject to a set of constraints: not to exceed the total

BS power, not to assign the same PRB to more than one user, to assign at least the minimum agreed PRBs to each SP and to keep certain fairness among users. The solution is numerically evaluated by a Matlab simulation, which shows the effectiveness of the proposed heuristic when compared to other solutions. However, the lack of more realistic simulations or deployments, with variable traffic and channel conditions, make the proposal difficult to compare to others.

The work in [62] presents a two-level resource scheduler. An important contribution of this work is the concept of *virtual Resource Blocks* (vRBs), which are an abstraction (virtualization) of the PRBs. Therefore, the two-level scheduling consists of a scheduler per slice called *Slice Resource Manager* (SRM) which allocates vRBs to each of the flows in the slice. Then, a second inter-slice scheduler named *Resource Mapper* (RM) maps the vRMs to the PRBs translating the SRM allocation to the actual resources. The RM is configured with a policy referred as *Slice Dedicated Bandwidth* which specifies the amount of resources allowed to each slice. The authors argue that with this approach it is avoided the joint scheduling of previous works where complex (NP-hard) multi-objective optimization problems are formulated and solved with heuristics.

A similar resource abstraction approach is proposed in [36] but also considers the physical layer constraints such as frequency-dependencies in scheduling. Following an SDN approach, in this work the network slicing problem is attacked by separating the control plane from the data plane. This is achieved by a adding an entity called *Base Station Hypervisor*. With this separation it is achieved functional isolation between slices allowing each slice to have its own controller for network functions. Also, for efficient resource sharing and allocation, the Hypervisor provides to the slice controller virtualized resources which the slice will use to schedule to its users. Then, the Hypervisor would translate those virtual resources to the actual physical resources. In this work it is provided a detail explanation on how to virtualize the physical resources while assuring that physical layer constraints such as frequency-dependencies are respected. However, it does not provide specific scheduling algorithms to achieve slice requirements but it leaves this task to be defined by the slice. Therefore, this work could have also been classified in the *Slice Scheduling* category.

3.3.2 Slice Scheduling

Network Virtualization Substrate (NVS) [59] proposes an architecture and algorithms for slicing a WiMAX (IEEE 802.16) network. The main contribution of this work is a mechanism for scheduling slices, which guarantees the requested resources or bandwidth demand while keeping isolation between slices. In this case, the scheduling is implemented by modifying the WiMAX flow scheduler which is located at a higher level than the PRB scheduler. The scheduler decides at each time interval which slice should use the transmission resources by formulating and solving an optimization problem. Then, each slice can decide how to schedule its own flows with the given resources following different allowed strategies to finally send the packets to the frame scheduler. Therefore, the frame scheduler is not changed in the way PRBs are assigned, however, modifications at the MAC layer of the base station will be needed. This causes the approach to face similar deployment constraints as the PRB Scheduling proposals, as in general, this software is proprietary and manufacturer dependent.

In [42], the previous NVS proposal is extended and applied for LTE eNodeBs. The approach is very similar but it is added the possibility to consider *partial resource reservation*. This possibility implies that unused resources can be shared among all slices. In the solution this is achieved by adding an extra shared slice which is assigned those remaining resources. This solution provides extra flexibility in resource allocation and is more efficient in resource usage.

In [70] it is proposed a resource allocation mechanism for sharing a base station among several operators. In the system model considered, each operator selects a ratio of the total resources to be allocated for its exclusive use, but the agreement includes a tolerance (deviation) on actual assigned resources. Then, a resource scheduler is designed to maximize an utility function considering the requested ratios and possible deviations. This scheduler then communicates the allocated quantity of resources to each operator to the low-level PRB scheduler which is in charge of the final assignment of resources.

The proposals in [55] and [110] also consider a high-level approach where the slice's allocations are made in resource ratios, specifically allocating fractions of the total available bandwidth. Differently from previous works, in these approaches, on of the main objectives is to guarantee a minimum average bit rate to each slice. Also, on both cases the problem is formulated as an stochastic optimization problem and a solution is developed following the Lyapunov optimization theory.

3.3.3 Traffic Shaping

Virtual Basestation [18] is an architecture for the virtualization of WiMAX base stations (BTS) to achieve resource sharing and isolation between multiple virtual network slices. The proposal adds a new layer over the WiMAX network called virtual BTS substrate. This substrate acts as a virtualization layer and provides a platform where virtual machines (VMs) are created and executed for each slice. These VMs operate as virtual

BTSs, and emulate an isolated private BTS for each slice. This framework presents two interesting aspects: the definition of the virtual BTS as an entity separated from the physical BTS and an isolation mechanism based on traffic shaping decoupled from the BTS. This idea makes the proposal feasible and independent of hardware. However, modifications on network components such as in the ASN-GW are needed for control, data tunneling and isolation.

An implementation of the isolation mechanism is discussed in [17]. The mechanism (called Virtual Network Traffic Shaper (VNTS)) obtains information from the wireless interface about the current transmission rate and uses this value, jointly with the number of clients and the weight of the slice, to shape the traffic. The shaping is implemented outside the BTS to control the offered load to the frame scheduler and to assure the fraction of resources assigned to each slice. Some important limitations can be foreseen with this proposal: (i) the traffic shaping is done independently of the number of available resources, resulting in an inefficient use of resources if some slice does not provide traffic; (ii) the mechanism focuses on isolation and not on resource allocation, there is no explanation on how to assign resources when new slice requests arrive. It also lacks an isolation mechanism for the uplink traffic and no study is done on the latency performance.

CellSlice [60] is another resource slicing proposal which does not need to modify the scheduling algorithms at the base stations but instead proposes a shaping mechanism at the gateway. The ideas are similar to other works, the network is divided into slices and each slice specifies a reservation of a fraction of the total resources. The objective of the mechanism is to assure that the requests are satisfied, while maintaining isolation among slices and using resources efficiently. The work focuses mainly on uplink flows, which are difficult to control, as traffic originates from the clients. The method used for this control is based on the adaptation of a specific parameter of the WiMAX standard, the *maximum sustained rate*. Hence, although an interesting mechanism, it highly depends on the BTS scheduler capability to control the rate of a flow through an adaptable parameter.

3.3.4 Discussion

Differently from WiFi technology, in cellular networks and particularly in LTE, the advances in slicing the wireless network are significant. Although in some of the initial slicing and virtualization works in the PRB Scheduling category the descriptions of the algorithms are vague and not clearly explained, more recent works, like [62] and [36], are much more comprehensive and propose good solutions. Moreover, in several reviewed works slicing agreements are based on the number of PRBs that are guaranteed to the operator, which, in our opinion, is a too low-level approach. This appears problematic for a slice tenant which could not have enough knowledge or information to decide the correct number of PRBs to request. Moreover, this does not guarantee a fixed performance when considering variable rates and variable channel conditions. Agreements related to more high-level variables such as a percentage of the total resources would be more appropriate. In addition, PRB Scheduling slicing will require a modification of the scheduler, which can be a difficult task because of the complexity of the scheduling algorithms. Also, an important aspect of the schedulers are the tight time constraints that the algorithms must satisfy, an issue not considered by some of the proposed solutions. Nevertheless, in our opinion the works in [62, 36] have improved those two main disadvantages of previous works: in these works it is proposed an abstraction from the physical resources, which allows the slice's owners to avoid dealing with low-level resources; and secondly, the proposal are implemented in an open-source software implementation of LTE called OpenAirInterface (OAI) [86].

On the other hand, Slice Scheduling and Traffic Shaping techniques do not need to modify the PRB scheduler and are generally easier to deploy. Also, as these are higher level mechanisms, the allocations are made on fractions of the total resources or on guaranteed bandwidths. This allows to propose more complex algorithms and also to formulate optimization problems to tackle several objectives simultaneously. Nevertheless, the reviewed proposals also face some drawbacks which appear as the main reason why these approaches have not been considered in more recent works:

- Without controlling the scheduling, it is more difficult to control the traffic coming from end users. Only [60] proposes a solution to this issue, but it highly depends on the technology and the hardware.

- Slice Scheduling at upper layers does not always guarantee that the scheduling will be kept at low layers, for example, queue buildup can happen at the MAC layer or the frame scheduler which can disrupt the scheduling.

- In the case of LTE, PRB scheduling has specific technology constraints which are not consider in Slice Scheduling proposals.

- Traffic Shaping can increase latency if queue management is not made properly or jointly considered with the shaping.

3.4 Open Research Directions

Because of the novelty of wireless slicing and more generally of wireless virtualization many challenges remain not addressed or at least not solved properly. In this section, we explore some of the challenges that make wireless slicing an interesting and promising research topic. As will be shown in the following chapters, some of these challenges have been tackled in this thesis. However, many others are not the focus of the thesis but are also presented here for completeness.

3.4.1 Isolation in Random Access Networks

For the particular case of technologies that use random access methods (e.g. WiFi) the isolation between slices is a complex and not fully studied problem. As already stated, two major challenges prevail in this technology: the randomness and the distributed nature of the access control. The most comprehensive works that deal with these problems include traffic shaping and EDCA control for the downlink and uplink slicing (Section 3.2.1). However, many aspects of isolation such as variable traffic, mobility and variable channel capacity are not deeply treated. Therefore, designing a mechanism for effectively slicing the uplink and downlink with strictly assured isolation is still a challenge.

3.4.2 Technology Agnostic Solutions

One of the biggest research challenges is to obtain a mechanism that could perform resource allocation and isolation of wireless slices independently of the wireless technology. The air-interface, the spectrum, the protocols for wireless and for the backbone are different for the different technologies. Then, there is not yet a unified approach dealing with any of the above mentioned factors. This technological dependency will be a problem when slicing a heterogeneous network.

When dealing with this issue, there is a trade-off between flexibility (or abstraction) and performance [117]. It seems to be difficult to use the same approach to virtualize and slice different wireless technologies without affecting the performance, as each technology has its own particularities and mechanisms for optimization. Also, for allowing virtualization to offer slicing and abstraction, a common language would have to be defined to be able to specify, manage and control the heterogeneous infrastructure. For example, assuring certain throughput to a given slice appears complex without knowing the underneath technology, or at least the channel access method used (deterministic or random access).

Ideas like modeling a “general” wireless network or developing layers of abstraction could be useful to reach this objective. For instance, approaches for network programmability [90] such as *Software-Defined Radio (SDR)* [30, 20, 15] or *programmable MAC protocols* [83, 112, 54] may be used to circumvent this challenge. In these proposals a generic API is provided to develop new wireless protocols (or modify existing ones) over generic radio transmitters or vendor-specific wireless devices.

3.4.3 Dynamics and Time Constraints

The existing proposals have not given enough attention to the impact of the adaptation of transmission parameters on the slicing techniques. As the link quality varies dynamically over time, many wireless standards incorporate an autonomous modulation and coding scheme (MCS) adaptation mechanism and/or a transmit power adaptation mechanism, to select the best transmission parameters for the current conditions. These reconfigurations of parameters are done with short latency, for example, in WiFi the MCS control algorithm takes decisions within $100ms$ intervals. Hence, slicing mechanisms have to be efficient and fast when reacting to changes or when searching for reallocations.

Related to this, many proposals lack a detailed performance study in terms of resource consumption (processor, memory or storage) and in terms of execution times. In wireless networks, changes can happen very fast and the time interval between transmissions is of the order of milliseconds. This way, new mechanisms with better reaction to changes need to be developed for slicing. For example, feedback control theory [31] and machine learning [76] techniques should be considered. Control theory would help with the design of controllers with guarantees of performance and stability. Furthermore, machine learning can help to learn control policies without the need to model a very complex environment.

3.4.4 Real Deployments

Only few works have deployed and tested their slicing proposals on real networks. In the wireless domain, doing a real deployment is critical for the evaluation of solutions. Furthermore, in real deployments it is common to find scenarios with multiple BSs or APs belonging to the same access network. Then, resource allocation and isolation on a multi-cell (multi-AP) network needs to be considered carefully. Deploying slices sharing multiple BSs or APs can bring new issues such as: interference between slices or load unbalance. For instance, sharing the spectrum could be accomplished cooperatively considering the interference among the cells and some load estimation mechanism. Then, resource assignment to each slice inside each cell could be more accurate [124].

Additionally, in a real deployment it becomes necessary to decide up to which level virtualization should be applied to achieve an efficient sliced solution. Slicing can be done at different levels, namely: from application and flow slicing to hardware and spectrum slicing. As stated in [115], virtualization can be considered at different levels with respect to providers and operators:

- *Universal Virtualization*, where the network is viewed as a cloud of BSs where the tenant has to choose and configure all the components to provide the desired service, and is totally transparent to the resource provider.
- *Cross-infrastructure Virtualization*, the idea of this paradigm is to share resources among infrastructure providers, where there is a pool of resources, and the tenants can choose the resources which best fit their needs.
- *Limited intra-infrastructure virtualization*, is the virtualization inside a single infrastructure provider, in this case there is spectrum sharing only between tenants inside the network of the provider.

Deciding which of these approaches better fits current network deployments is an open challenge.

3.4.5 User Mobility and Interference

The mobility of users is a particular feature of wireless networks that brings new challenges to slicing. Not only because mobility generates variations in links capacity and performance, or because it makes the number of users on a network to vary significantly, but also because it adds management complexity. Wireless networks have to deal with the management of user mobility, handle handovers and assure QoS despite of the location of the user.

It is clear that new problems are introduced when allowing the user mobility in a sliced network. In this case, not only a user will change the BS or AP it is connected to, but also it could change of slice (if changing of operator or service is needed). Then, handover mechanisms to move a user across slices are necessary. As slices may be owned by different entities and can belong to totally independent virtual networks, implementing this seems complex. A possible approach could share a central mobility manager across slices, however, this may need to be a third party agent with an open interface controller. In addition, centralization would add latency in a task with strict time constraints. Alternatively, a distributed solution could also be considered. However, the distribution of mobility management can add new problems such as more signaling overhead between the management entities. In summary, a good solution to the mobility problem in a virtual sliced scenario will need to support handoffs between BSs, slices and technologies while maintaining the service quality.

3.4.6 Control at the End Users

Another major challenge in wireless resource slicing is the access control of end user devices to the medium. The complexity of this problem depends greatly on the wireless technology used. For example, the most used medium access control in the IEEE 802.11 standard is totally distributed, i.e. the AP does not have any possibility to control how and when an end user will transmit. In this case, the allocation and isolation of resources used by end users becomes complex as there is little control over users' devices. In contrast, in 3GPP LTE, the scheduler at the BS aside from scheduling the downlink traffic, it also schedules the resources for the uplink traffic, having complete control of the resources on both links. However, information from the end users is needed in order to gather knowledge about the traffic that is generated, as well as the channel conditions.

3.4.7 Complex Wireless Management Functions and Configurations

Most wireless equipment has complex management functions, and is manufacturer specific, involving the programming of drivers and low level software. When sharing a base station by multiple slices, these specific functions have to be used with care as commands from different slices can conflict with each other. Also, in general, each wireless link has its particular configuration parameters like its frequency of operation, bitrate or transmit power, which can be very different from another link sharing the same infrastructure.

Also, in architectures with central controllers (e.g. [40, 101]) special care has to be taken on local functions at the devices. The possible delays between a central controller and the physical devices imply that the physical devices have a more updated view of the local state. Consequently, the physical devices, in certain scenarios, can manage in a better way their resources locally. So, the controller will have to manage the network globally while each device could take local decisions, without interfering nearby devices.

3.4.8 Compatibility with Other New Technologies

To satisfy the ever increasing requirements of future wireless networks, other new technologies, aside from slicing, are being proposed. For example:

- *Extreme Densification and Offloading*, which consists of massively deploying base stations on a given area, and complementary, using offloading techniques to redirect the traffic through different networks.
- *Millimeter Wave* consists of the use of higher frequencies of the spectrum, those of millimeter wavelength, to overcome the spectrum scarcity. These frequencies are more affected by path-loss and do not have good penetration through walls, therefore, they are expected to be used for indoor communications, along with dense deployments.
- *Massive MIMO* to spatially increase the spectral efficiency by using multiple antennas of Multiple Input Multiple Output (MIMO) technology.

How these technologies interact with a sliced design of a wireless network has still to be studied. For instance, slicing a MIMO interface introduces new challenges as several transmissions can happen in parallel, and then multiple slices would be “transmitting” at the same time.

3.4.9 Security

One of the main features of slicing is the abstraction process where the slice is viewed as a whole network, and the slice tenant can manage and configure it in its own way. This flexibility introduces higher security risks to wireless networks, as the slices share the same physical infrastructure. Hence, it is of crucial importance to offer security and isolation at the configuration, management and programming levels. However, to the best of our knowledge, there is a complete lack of research in security for wireless slicing.

In addition, wireless networks offer authentication and encryption on the air-interface, but with slicing, those functions need to be splitted between the slices. Furthermore, all the security issues related to virtualization and hardware sharing, are also relevant to the slicing approach. Extensive research efforts will need to undertake these challenges with extreme care before virtualization and slicing in wireless networks can be deployed.

3.5 Conclusions

In this chapter, we present a review of existing works in resource allocation for virtualization and slicing. We also propose a classification of these works based on the resource allocation strategy followed and on the wireless technology of application. Although in this thesis we concentrate on the WiFi technology, we also analyzed existing proposals for cellular technologies like LTE and WiMAX. This extensive review provides a good understanding of the current trends in resource allocation techniques in the wireless domain. We also compare the different strategies and highlight their advantages and drawbacks.

As a result of the review, we have found that the majority of existing approaches for WiFi take advantage of the adaptation of the EDCA parameters provided by the 802.11e standard to control how the devices use the wireless medium. By adapting these parameters, it is possible to control the access to the medium, prioritizing a given traffic flow over others. However, this strategy has some crucial drawbacks like a limited number of possible flows and may need complex modifications at the driver of the device. Other solutions concentrate on resource sharing but do not consider the specific problems of slicing like performance guarantees or isolation. On the other hand, in cellular and specifically in LTE, more advanced and complete solutions exist. In this case, the most predominant approach is to control the scheduling of the PRBs to allocate different bandwidths to the slices. This technique provides fine-grained control of the resources, which can be used by high-level allocation strategies to assign resources to slices. Although not directly applicable to WiFi, some of these scheduling strategies are of interest as they could be adapted to work on WiFi.

Nevertheless, in both technologies, the resource allocation for Network Slicing and specifically for slicing with specific QoS guarantees is not well investigated. In this regard, in Section 3.4, we analyzed some research challenges and improvements needed for wireless resource slicing to become a reality. Some of these challenges and drawbacks of current solutions motivate and support the research performed in this thesis. Specifically, guaranteeing isolation between slices (3.4.1) is a key aspect of slicing, which is overlooked in many current proposals and is explicitly considered in this thesis in Chapters 4 and 5. For our QoS Slicing approach proposed in Chapter 5, we take into account the dynamics of the wireless medium (3.4.3) as well as some of the issues introduced by the mobility of final users (3.4.5) like channel capacity variability. Even more, our QoS Slicing design considers the need for fast and efficient resource allocation decisions (3.4.3).

Chapter 4

An Airtime Allocation Mechanism for WiFi

4.1 Introduction

As mentioned in the previous chapters, the first step in the process of slicing a wireless network is to define how to model the resources and the requests. In Section 2.4.1, it was anticipated that for the WiFi technology, the best approach is to consider the transmission time (*airtime*) as the resource to partition. However, achieving airtime sharing and implementing the allocation of airtime to different slices is a complex task.

The idea of controlling the airtime usage in WiFi has been mainly studied as a means to overcome the *performance anomaly* problem [45]. This problem appears in multi-rate WiFi networks and causes the transmission throughput of any client to be bounded by the lowest rate of all clients associated with the same AP. Hence, several works have proposed different mechanisms for a fair allocation of the airtime among the stations associated with the same AP (see [49, and references therein]). In particular, several works ([94, 37, 49]) have suggested the use of a modified Deficit Round Robin (DRR) [105] scheduling to achieve fairness.

Moreover, in the context of wireless slicing and virtualization, some of the works described in Chapter 3 also propose airtime sharing. The works [79] and [41] proposed an airtime resource allocation method through the management of some EDCA parameters such as the *Contention Window* and the *Transmission Opportunity.* In those works, each slice defines its target airtime ratio, and an allocation algorithm adjusts the parameters to achieve those ratios.

As can be observed, airtime allocation has already been proposed for network slicing; however, many unsolved challenges remain. Therefore, we propose a novel mechanism (called *ATERR*) for slicing a WiFi Access Point (AP) by considering the transmission time as the resource to share. ATERR is based on modifying how packets are queued and scheduled for transmission at the MAC layer of WiFi devices. Our resource allocation proposal is efficient, as each slice only receives the exact amount of resources required for its current load so that free resources can be used by other slices. With the proposed mechanism, it is possible to implement the *Infrastructure-Sharing Slicing* described earlier, as the solution allows to split and allocate network resources into slices, proportionally to the requested demand.

With our solution we aim to overcome open 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 EDCA 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 behavior, 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 resembling real devices, and we also make our simulation code available, so that the community would be able to use it freely.

The rest of the Chapter is organized as follows: Section 4.2 presents the airtime allocation proposal describing in detail its architecture and design. In Section 4.3, we theoretically analyze the proposed mechanism to find the necessary conditions for the accurate allocation of airtime. In this analysis, we also show that our proposal fairly allocates airtime among clients of the same slice. We also identify the parameters that influence the latency caused by the scheduling algorithm. In Section 4.4, we describe how the proposed allocation mechanism can be used to implement Infrastructure-Sharing Slicing and briefly present a SLA to achieve this goal. The analysis of the allocation mechanism is completed with simulated experiments in Section 4.5, where the theoretical results are validated, and the behavior of the mechanism is shown in an illustrative use case. Finally, Section 4.6 concludes the Chapter.

4.2 ATERR Architecture and Design

To implement airtime allocation with the objective of deploying slices in a WiFi AP, we propose a queuing structure and a scheduling mechanism inspired on works from other authors like in [105] and [5] and their extension as proposed in [49] for airtime fairness among clients of a WiFi AP. The work in [49] showed promising results on achieving airtime fairness for solving the *performance anomaly* problem, including an implementation on real hardware (*Atheros ath9k* driver). Our proposal adapts and extends this airtime scheduling mechanism to be used for network slicing. We call our mechanism *Adaptive Time-Excess Round Robin* (ATERR). ATERR is envisioned to be deployed onto the APs of a WiFi network, replacing the queuing and scheduling algorithms of the AP. In this context, we use the term *client* to refer to any WiFi device connected to an AP (also known as *station* in WiFi terminology).

The proposed airtime allocation mechanism can be separated in three different parts (see Figure 4.1): the *Controller*, the *Classifier and Queues* and the *Scheduler* which are all implemented as part of the WiFi device. The Controller is in charge of managing the flows and clients on each slice and on configuring the required resources to each slice. The Classifier identifies the current traffic flows and assigns a queue to each flow while the Scheduler decides which queue to provide service to guarantee each slice requirements.

4.2.1 ATERR Design Objectives

The design of ATERR follows three main objectives:

- To allocate the share of airtime requested by each slice.
- To allocate airtime to each client within a slice fairly.
- To use resources efficiently.

As expected, one of the main objectives is to obtain a mechanism that can allocate and control an airtime share to a given slice. In this sense, the enforcement and control of the allocation is essential to maintain the airtime shares given the traffic load and channel characteristics variability. Even more, ATERR is designed to use resources efficiently so that if a slice is not consuming all its requested airtime share, it can be used by other slices with extra traffic. In this regard, an analogy with circuit switching and packet switching can be made. In circuit-switching, the physical resources are reserved for the entire use of a tenant and cannot be used by others. This strict allocation guarantees that the requested resources are always available by the tenant but may generate inefficiencies when the tenant is not consuming all the resources. On the other hand, in packet-switching, there are no resources granted to any tenant, and resources are allocated on demand (statistical multiplexing of resources). This allows a better and more efficient resource utilization but does not guarantee that the resources are available when needed. For our airtime allocation solution, we propose an intermediate approach where the airtime resources are allocated and reserved for the slice, but we also implement statistical multiplexing allowing unused airtime not consumed by a slice to be assigned to other slices with extra load. However, in our approach, it is not possible to allocate more than a 100% of the available airtime. In this regard, a possible extension to the proposed approach, but which falls out of the scope of the ATERR design, is to allow over-provisioning and take advantage of the statistical multiplexing to use resources more efficiently. This must be implemented with care as it can generate situations where the requested airtime shares are not guaranteed.

Given that in this approach we do not consider specific requests per client of a slice but just a request of airtime share for the entire slice, our objective is to guarantee to each client of the slice the same airtime share. We propose airtime fairness among clients within a slice to avoid the *performance anomaly* problem mentioned earlier. This way, the slice resources are allocated fairly independently of the channel characteristics of the clients. Nevertheless, our system could be easily extended to consider requests per client of a slice by changing how the quantum sizes are computed in Section 4.2.4. For this, the model must also be extended to differentiate the quantum values of each client of the same slice.

4.2.2 ATERR Controller

The *ATERR Controller* can be seen as the implementation of the *WTP Local Controller* described in Section 2.5 and is in charge of communicating with the *Network Slice Manager*. The ATERR Controller receives the requests of deploying new slices in the WiFi AP with a set of configuration parameters. These configuration parameters

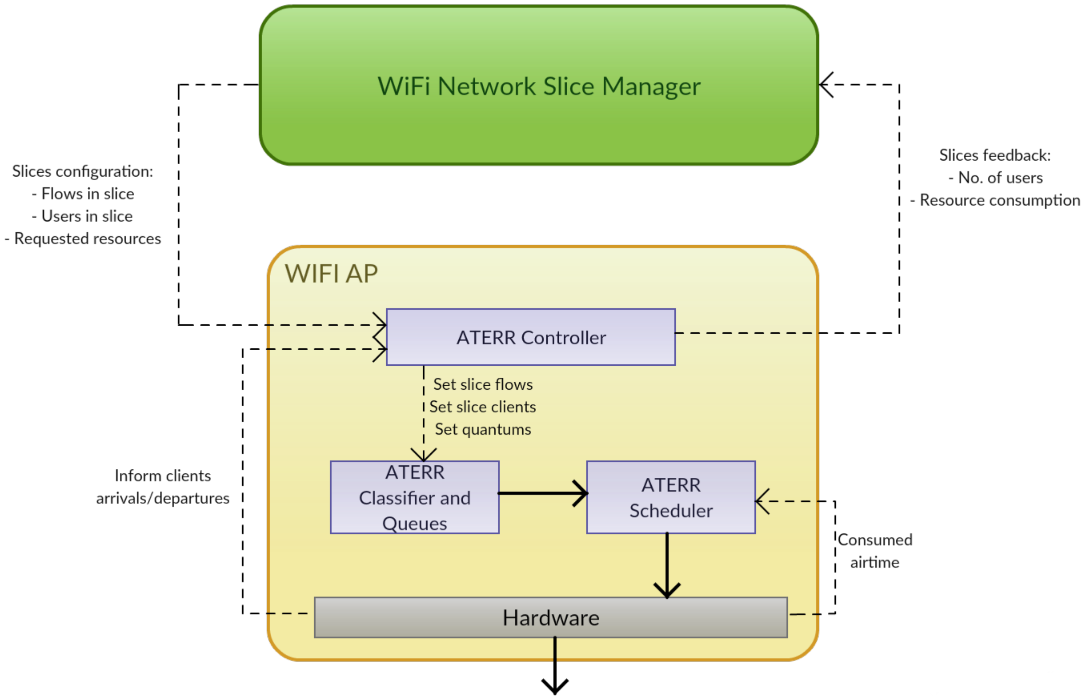

Fig. 4.1 ATERR Architecture.

can be divided into two sets, which describe different aspects of the slice. One set of parameters contains the description of the traffic flows which belong to the slice, to be able to classify and assign the flows to its corresponding slice. Remember that in Section 2.1 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). Secondly, there is a set of parameters to define the percentage of airtime that must be allocated to the slice (we provide more details of these parameters in Section 4.3).

As mentioned, the ATERR Controller is the interface between the AP and the Network Slice Manager. For example, in the simplest model, a slice tenant can request a percentage of the total airtime in an AP to be assigned to a particular slice. This command is directly sent to the Network Slice Manager and from there to the ATERR Controller, which given that enough resources are available, enforce the airtime assignment to the slice. However, the proposed architecture supports more complex scenarios. For example, the slice tenant may request a percentage of the resources of the entire WiFi network. In this case, the Network Slice Manager can configure the ATERR Controller of each AP of the slice with different airtime requirements using its global view of the network.

Even more, the ATERR Controller can provide feedback information to the Network Slice Manager, such as the current number of connected clients, the amount of airtime allocated, or low-level information such as the wireless channel occupancy. With this information, the Network Slice Manager can implement a decision process to dynamically adjust the airtime percentages assigned to the slices and achieve different requirements.

4.2.3 ATERR Classifier and Queuing Structure

ATERR maintains a queue for each client's traffic within a slice. Hence, we have a queue per client and per slice. This means that if a client participates in three slices, three queues are created in the system for such a client, one per slice. In Figure 4.2, we show in detail the queuing structure within the ATERR Classifier and Queues. To fill the different queues, a Classifier is implemented, which, given the slice's descriptions from the Controller, assigns each traffic flow with its corresponding queue.

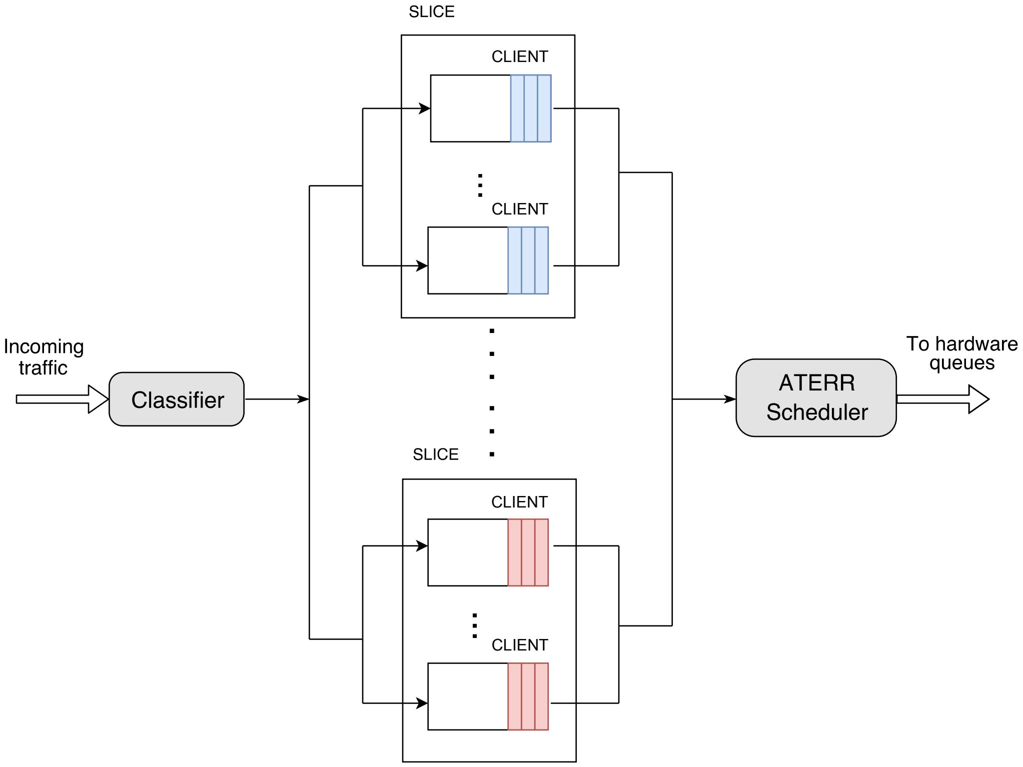

Fig. 4.2 ATERR Queuing Structure with the Classifier and the Scheduler.

For our scheduling algorithm, ATERR keeps a list of queues with backlogged traffic (*Active* queues), which are the ones that need to be served. Furthermore, we add an

Inactive state to identify queues without traffic. If a queue is in the Inactive state for a predefined period (1 second in our implementation), it is removed from the slice, and each quantum is recalculated. The queues are not removed immediately after being empty to soften quantum changes.

Whenever a packet is received from the upper layer, a *Classify* procedure is executed, which determines the client and slice to which the packet belongs and enqueues it in the corresponding queue. The first time a new client's flow is received, a new empty queue is created and identified by the client and slice. This new queue is added to the list of Active queues. In particular, there are two lists of Active queues: a list of new queues (*newQueuesList*) and a list of old queues (*oldQueuesList*). When a queue is just created, it is added to the end of the list of new queues. Also, when a queue evolves from an Inactive state to an Active state, the queue is added to the *newQueuesList*. As we will describe afterward, a Round Robin scheduling process is run every time a new packet arrives, or a transmission is completed. In this scheduling process, the queues in the *newQueuesList* are always served first than the queues in the *oldQueuesList*. However, if a queue from the *newQueuesList* remains with backlogged traffic after being served, it is changed to the *oldQueuesList*. Dealing with two different lists is a contribution from the FQ-CoDel algorithm [46], to reduce the latency of sparse flows[1]. The rationale is that in the case of clients with very low traffic rates and which queues are emptied in just one round, its transmissions are given priority in the scheduling.

4.2.4 ATERR Scheduling

The previously described 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 and transmitted, 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

[1] The reader might refer to [46] for a detailed description of its operation.

Algorithm 1: ATERR scheduling pseudocode. For every new transmission, selects the queue to dequeue a packet from.

```
1  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.
2      /* Get the next queue from the lists of Active queues                    */
3      var queue;
4      if newQueues is not empty then
5          queue ← GetFront(newQueues);
6      else if oldQueues is not empty then
7          queue ← GetFront(oldQueues);
8      else
9          return;
10     end
11     /* If queue surpasses its quantum, update and move to the end of list    */
12     if $queue.excess \geq 0$ then
13         $queue.excess \leftarrow queue.excess - queue.quantum$;
14         if $queue \in newQueues$ then
15             RemoveFront(newQueues);
16         else
17             RemoveFront(oldQueues);
18         end
19         InsertTail(queue, oldQueues);
20         goto 4;
21     end
22     /* If queue got empty on the previous round                              */
23     if queue is empty then
24         if $queue \in newQueues$ then
25             /* If queue is on the newQueues list, move to oldQueues list     */
26             RemoveFront(newQueues);
27             InsertTail(queue, oldQueues);
28         else
29             /* If queue is on the oldQueues list, set Inactive               */
30             RemoveFront(oldQueues);
31             setQueueInactive(queue);
32         end
33         goto 4;
34     end
35     setAvailableToLowerLayer(queue);
36 end
```

needs to be updated after each packet transmission. This may cause an allocated airtime "excess" in a round.

The complete rationale of the scheduling and queuing scheme is presented in Algorithm 1. This scheduling process is executed every time a new packet arrives to the system from the upper layer or when a transmission is completed. Note that the airtime accounting in the excess variable is performed in another function (outside the *ScheduleTransmission()* function) after transmission has completed.

Given that the slice requests are expressed in percentage of airtime to be allocated, the quantum of each queue must be dynamically computed based on the requested airtime and the system state. In the following we explain how we compute the size of each quantum.

Quantum Calculation

The quantum computation must be done to satisfy the objectives we previously mentioned:

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

Let us define $\mathcal{S}$ as the set of all slices instantiated on an AP and $\mid \mathcal{S} \mid$ 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 $p_s \mid 0 < p_s \leq 1$. Let us also define $\mathcal{U}$ as the set of clients associated with the AP, $\mathcal{U}_s$ the subset of those clients that belong to slice s, and $N_s =\mid \mathcal{U}_s \mid$ the number of clients in $\mathcal{U}_s$.

To achieve fairness among clients of the same slice, each quantum 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 every quantum 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:

$$Q_s = p_s \sum_{j=1}^{|\mathcal{S}|} Q_j \quad \forall s \in \mathcal{S} \tag{4.1}$$

For the second objective, as each quantum of a slice must 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 clients is:

$$q_s = \frac{Q_s}{N_s} \quad \forall s \in \mathcal{S} \tag{4.2}$$

Substituting (4.1) in (4.2) we have:

$$q_s = \frac{p_s \sum_{j=1}^{|\mathcal{S}|} Q_j}{N_s} = \frac{p_s \sum_{j=1}^{|\mathcal{S}|} N_j q_j}{N_s} \quad \forall s \in \mathcal{S} \tag{4.3}$$

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

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

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

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

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

$$q_s = \frac{p_s \sum_{j=1, j \neq s}^{|\mathcal{S}|} N_j q_j}{N_s(1 - p_s)} \quad \forall j \in \mathcal{S} \tag{4.4}$$

The above result shows the adaptive nature of the algorithm as each queue's quantum need to be recalculated every time a queue is created or removed (i.e. when a client connects or disconnects) on the AP. Even more, it is important to notice that the previous result does not provide an absolute value for each quantum, but a relationship between each slice's quantum. To solve this indeterminate system of equations, we fix the minimum allowable quantum in the system and assign it to the lower q_s. Therefore, in Algorithm 2 we show a pseudo-code of the procedure to recalculate each quantum every time a slice or client arrives or departures.

Algorithm 2: ATERR Quantum Recalculation.

```
function RecalculateQuantum() is
    input  : $\mathcal{S}$, $N_s, p_s\ \forall s \in \mathcal{S}$
    output : $q_s\ \forall s \in \mathcal{S}$
    $minX \leftarrow MAX\_INT$;
    $minS$;
    /* Find the slice which uses the smallest quantum */
    foreach $s \in \mathcal{S}$ do
        $x \leftarrow \frac{p_s}{N_s}$;
        if $x < minX$ then
            $minX \leftarrow x$;
            $minS \leftarrow s$;
        end
    end
    /* Set the quantum to the obtained slice */
    $q_{minS} \leftarrow MIN\_QUANTUM$;
    $Q_{minS} = MIN\_QUANTUM * N_{minS}$;
    /* Set the remaining quantum */
    foreach $s \in \{\mathcal{S} - minS\}$ do
        $q_s \leftarrow \frac{p_s}{p_{minS}} * \frac{Q_{minS}}{N_s}$;
    end
end
```

4.2.5 Implementation Details

Airtime measurement

In WiFi, the time consumed by a packet depends on many different aspects of the transmission process. First, depending on the congestion in the wireless channel, the packet may need to wait before being actually transmitted. As we explain in Appendix A, this waiting time is regulated by a backoff procedure and is generally not considered as airtime. Secondly, when the packet is transmitted, two different possibilities can occur: (1) the packet correctly arrives at the destination, and an acknowledgment is replied or (2) the packet is not received (because of low power signal, interference or collision), and retransmission is needed. When the packet is correctly received, the airtime consumed depends on some parameters such as the packet size, the MCS used, and some extra time waits given by the WiFi standard. However, when the packet is not received, it is retransmitted, adding more airtime consumption.

Therefore, given that in our approach we consider the airtime consumed by a packet after it has correctly been received or discarded (because it was not received after several retransmissions), all these different aspects of the wireless medium can be taken into account. Note that this is one of the main differences of our approach with some recent works, which also consider airtime allocation ([25, 7]) but need to estimate the airtime before transmission.

An interesting compromise exists when deciding if the waiting time before transmission (because of congestion in the medium and the backoffs) should be taken into consideration in the airtime measurement. If this time is not considered and only the actual transmission time is measured, it has the advantage that only effective transmission time is being allocated. However, it would affect the allocations as the actually assigned ratio of transmission time to a slice would be less than the requested. On the other hand, if the waiting time is considered, there is no certainty on the amount of actual airtime allocated, but the amount of used resources (time of the transmission device) is precisely measured. In summary, two types of allocation and sharing policies can be defined; in one case, the slices only share the available transmission airtime in the medium (time the medium is free) while, in the other case, the total usage time of the transmitting device is shared.

Uplink traffic

As we already mentioned, the mechanism proposed only controls downlink transmissions (from the AP to the clients). However, traffic from the clients to the AP (uplink) also consumes the available airtime in the medium and should be controlled to avoid isolation problems. Therefore, in our proposal, it is possible to also consider the airtime of received packets and to decrement this value from the quantum of the slice. This provides an indirect control of the uplink traffic, given that most traffic is bidirectional (TCP or HTTP) and ATERR controls the rate of downlink traffic.

Resource assignment

The allocation strategy proposed in ATERR does not guarantee instantaneous satisfaction of the requested share given that on each particular transmission the quantum could be exceeded. However, 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 4.3.

On the other hand, 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.

Hardware implementation

Finally, it is important to point out that the proposed solution is implementable in the 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 4.2, the work in [49] has already implemented individual queues per client and an airtime scheduler in this driver. 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 calculate the quantum's values dynamically.

4.3 Analytical Study

4.3.1 System Model

So as to manage the sharing of resources, we consider that Service Level Agreements (SLAs) are subscribed between the slice providers and the tenants. We propose a simple mechanism to define the requested resource share of a slices s based on three parameters: p_s, K_s and W_s. A slice tenant requests a share of the total airtime, as a quantified value between 0 and 1, identified by p_s ($p_s \in (0, 1]$), and is what the tenant expects to be guaranteed by the provider. Also, to provide flexibility to the provider, it is possible to define a *tolerance* K_s which measures the possible maximum deviation from the expected resource share. In wireless this is particularly important and useful, as the medium is variable and guaranteeing a strict resource share could be complex to implement. W_s is a time window over which the airtime allocation is computed and where the resource share plus the deviation must be guaranteed. Hence, the SLA requires that, in a time window of size W_s the slice s receives, in average, a ratio of resources between $(p_s - K_s, p_s + K_s)$. 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 for a given allocation window.

Therefore, in this section we present an study with the following objectives: (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 clients of a slice, and

(iii) to study the latency introduced by the ATERR scheduling. The notation used is summarized in Table 4.1.

Table 4.1 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.
$\mid \mathcal{S} \mid$	Total number of slices on an AP.
$\mathcal{U}$	The set of clients associated to an AP.
$\mathcal{U}_s$	The set of clients that belong to slice s.
N_s	Number of clients that belong to slice s.
q_s	Quantum of every queue of the slice s.
Q_s	The sum of every quantum of slice s.
Q	The sum of every quantum of the system.

4.3.2 Allocation-window Size

Regarding objective (i), as a first step, in Lemma 4.1, we found the airtime a slice s gets in any time interval. Then, in Lemma 4.2, using this result we bound the ratio between a slice's airtime and the total allocated airtime. Finally, in Theorem 4.3, we propose a lower bound for the allocation-window's size. For the analysis we consider the set *active clients* as the subset of clients associated with the AP with pending incoming traffic. Hence, the queues assigned to the active clients at the AP always have packets to be dequeued.

Lemma 4.1. *In any interval of time (t_1, t_2) such that the set of active clients 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:*

$$mQ_s - N_s T_{max} \leq T_s(t_1, t_2) \leq mQ_s + N_s T_{max} \quad \forall s \in \mathcal{S} \tag{4.5}$$

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:

$$0 \leq e_{i,s} < T_{max} \tag{4.6}$$

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:

$$e_{i,s}(k) = e_{i,s}(k-1) + t_{i,s}(k) - q_s \tag{4.7}$$

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:

$$t_{i,s}(k) = q_s + e_{i,s}(k) - e_{i,s}(k-1) \tag{4.8}$$

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:

$$T_{i,s}(t_1, t_2) = \sum_{k=1}^{m} t_{i,s}(k) \tag{4.9}$$

Replacing (4.8) in (4.9):

$$T_{i,s}(t_1, t_2) = \sum_{k=1}^{m} (q_s + e_{i,s}(k) - e_{i,j}(k-1)) \tag{4.10}$$

$$T_{i,s}(t_1, t_2) = mq_s + \sum_{k=1}^{m} (e_{i,s}(k) - e_{i,s}(k-1)) \tag{4.11}$$

$$T_{i,s}(t_1, t_2) = mq_s + e_{i,s}(m) - e_{i,s}(0) \tag{4.12}$$

Finally, from (4.6) and (4.12) we have:

$$mq_s - T_{max} \leq T_{i,s}(t_1, t_2) \leq mq_s + T_{max} \tag{4.13}$$

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:

$$m \sum_{i=1}^{N_s} q_s - N_s T_{max} \leq T_s(t_1, t_2) \leq m \sum_{i=1}^{N_s} q_s + N_s T_{max} \tag{4.14}$$

□

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 rounds. 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 4.2. *In any interval of time (t_1, t_2) such that the active clients 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:*

$$\frac{mQ_s - N_s T_{max}}{mQ + \hat{N} T_{max}} \leq r_s(t_1, t_2) \leq \frac{mQ_s + N_s T_{max}}{mQ - \hat{N} T_{max}} \quad \forall s \in \mathcal{S} \tag{4.15}$$

where $N = \sum_{l=1}^{|\mathcal{S}|} N_l$ is the total amount of queues in the system and $\hat{N} = N - 2N_s$.

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:

$$r_s(t_1, t_2) = \frac{T_s(t_1, t_2)}{\sum_{l=1}^{|\mathcal{S}|} T_l(t_1, t_2)} \tag{4.16}$$

Applying the bounds from Lemma 4.1 we obtain:

$$\frac{T_s(t_1, t_2)}{\sum\limits_{l=1}^{|\mathcal{S}|} T_l(t_1, t_2)} \geq \frac{mQ_s - N_s T_{max}}{m \sum\limits_{l=1}^{|\mathcal{S}|} Q_l + \sum\limits_{l=1, l \neq s}^{|\mathcal{S}|} N_l T_{max} - N_s T_{max}} \tag{4.17}$$

$$\frac{T_s(t_1,t_2)}{\sum\limits_{l=1}^{|\mathcal{S}|} T_l(t_1,t_2)} \leq \frac{mQ_s + N_sT_{max}}{m\sum\limits_{l=1}^{|\mathcal{S}|} Q_l - \sum\limits_{l=1,l\neq s}^{|\mathcal{S}|} N_lT_{max} + N_sT_{max}} \tag{4.18}$$

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:

$$\frac{T_s(t_1,t_2)}{\sum\limits_{l=1}^{|\mathcal{S}|} T_l(t_1,t_2)} \geq \frac{mQ_s - N_sT_{max}}{mQ + (N-2N_s)T_{max}} \tag{4.19}$$

$$\frac{T_s(t_1,t_2)}{\sum\limits_{l=1}^{|\mathcal{S}|} T_l(t_1,t_2)} \leq \frac{mQ_s + N_sT_{max}}{mQ - (N-2N_s)T_{max}} \tag{4.20}$$

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 every quantum 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.

Theorem 4.3. *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:*

$$W_{s,K} \geq \frac{T_{max}}{Kp_s}\left(p_s\hat{N} + N_s + \sqrt{(p_s\hat{N} + N_s)^2 + (Kp_s\hat{N})^2}\right) - NT_{max} \tag{4.21}$$

Proof. In this proof, for the sake of simplicity, we assume the following:[2]

Assumption 4.1. *The provided airtime measurement considers all the time consumed, counting from the packet being schedule for transmission until it is correctly received or discarded.*

[2] This assumption considers the time that a packet may be waiting for the transmission medium to be clear, with an impact on the provided airtime allocation as discussed in Section 4.2.5. Excluding such time from our model is left for further work.

From Lemma 4.2 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:

$$\frac{mQ_s + N_s T_{max}}{mQ - \hat{N} T_{max}} - \frac{mQ_s - N_s T_{max}}{mQ + \hat{N} T_{max}} \leq K p_s \tag{4.22}$$

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

$$\frac{2QT_{max}(p_s \hat{N} + N_s)m}{Q^2 m^2 - \hat{N}^2 T_{max}^2} \leq K p_s \tag{4.23}$$

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

$$-K p_s Q^2 m^2 + 2QT_{max}(p_s \hat{N} + N_s)m + K p_s \hat{N}^2 T_{max}^2 \leq 0 \tag{4.24}$$

The positive solution of (4.24) is:

$$m \geq \frac{T_{max}}{K p_s Q}(p_s \hat{N} + N_s + \sqrt{(p_s \hat{N} + N_s)^2 + (K p_s \hat{N})^2}) \tag{4.25}$$

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 4.1, we have that the time consumed by the m rounds is all accounted in the airtime measurement.

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

$$T(m) = T(t_1, t_2) = \sum_{l=1}^{S} T_l(t_1, t_2) \geq mQ - NT_{max} \tag{4.26}$$

Then, the theorem follows from substituting the minimum m from (4.25) in (4.26).

□

Hence, Theorem 4.3 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 queues (slices and clients) in the AP (N) and the number of clients 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 clients or slices to an AP. Given the directly proportional relation between the number of slices and clients per slice in the system and the size of the allocation window, it is expected that the slicing mechanism limits the amount of simultaneous clients to assure the slice's requirements. This can be achieved by a client access control mechanism which can reject new clients' connections to the AP if current slice's allocations can not be achieved with the requested window sizes. Moreover, the slicing mechanism can reject or renegotiate new slice requirements if with the current system status (current slices' allocations and number of clients per slice) the new slice requirement cannot be fulfilled.

In what follows we illustrate this result with numerical values.

Graphical Illustration

In Figure 4.3 we plot numerical values for the lower bound of W given in Theorem 4.3 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 [107] for an aggregate frame.

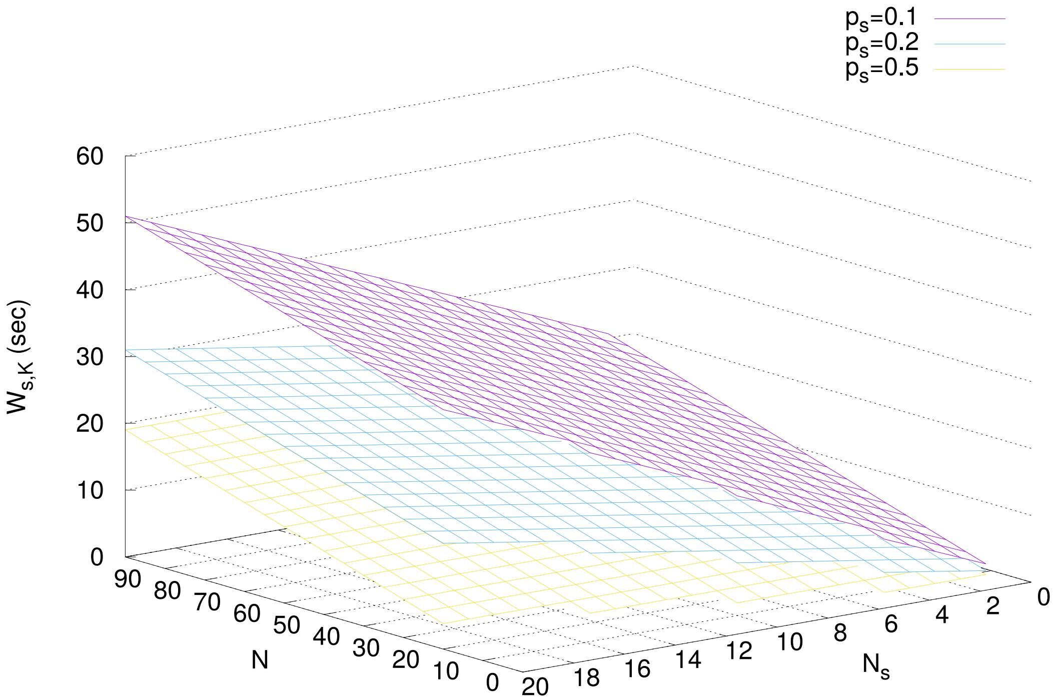

Fig. 4.3 Worst case lower bounds for $W_{s,K}$ with a tolerance of 0.1 as a function of the number of associated clients N and number of clients in the slice N_s

Therefore, the Figure shows the minimal time window (in seconds) in which the requested airtime allocation is guaranteed for airtime ratios of 0.1, 0.2 and 0.5. As expected, when the number of clients in the system increases the needed time window becomes larger. 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 4.5, in more common scenarios, the requested assignments are correctly achieved in smaller allocation windows.

4.3.3 Fairness Among clients of one Slice

Regarding the fairness between clients of a slice, following a similar analysis to the one proposed in [105] we can show that within a slice all queues receive a fair share of the airtime allocated to that slice.

Lemma 4.4. *In any interval of time (t_1, t_2) such that the active client 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:*

$$mq_s - T_{max} \leq T_{i,s}(t_1, t_2) \leq mq_s + T_{max} \quad \forall i \in \mathcal{U}_s \;\; \forall s \in \mathcal{S} \tag{4.27}$$

The proof is analogue to the one of Lemma 4.1.

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

$$T_{i,s}(t_1, t_2) - T_{j,s}(t_1, t_2) \leq q_s + 2T_{max} \tag{4.28}$$

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 4.4 that:

$$T_{i,s}(t_1, t_2) \leq mq_s + T_{max} \tag{4.29}$$

In that same interval, the algorithm can serve queue l for m' times, then from Lemma 4.4 we have:

$$T_{j,s}(t_1, t_2) \geq m'q_s - T_{max} \tag{4.30}$$

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:

$$T_{i,s}(t_1, t_2) - T_{j,s}(t_1, t_2) \leq mq_s + T_{max} - m'q_s - T_{max} \tag{4.31}$$

$$T_{i,s}(t_1, t_2) - T_{j,s}(t_1, t_2) \leq mq_s - (m - 1)q_s + 2T_{max} \tag{4.32}$$

Finally we get:

$$T_{i,s}(t_1, t_2) - T_{j,s}(t_1, t_2) \leq mq_s - (m - 1)q_s + 2T_{max} \tag{4.33}$$

$$T_{i,s}(t_1, t_2) - T_{j,s}(t_1, t_2) \leq q_s + 2T_{max} \tag{4.34}$$

□

Hence, the difference in airtime allocations between any two queues (and thus, clients) of a slice is bounded. For a given system, the bound depends on two constants: the quantum size and the maximum packet airtime.

4.3.4 Latency Analysis

Regarding the latency introduced by ATERR, in this section we find an upper 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 an upper bound, which depends on the number of queues and on the value of the quantum.

Theorem 4.6. *The time l_i between two consecutive service rounds to a queue i in a slice s, is bounded by:*

$$l_i < Q - q_s + (N - 1)T_{max} \tag{4.35}$$

Proof. Let us consider a packet P which was left at the head of the queue i after round R. Let us 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 (4.8) and (4.6) 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[3]

[3] 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}.

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

$$l_i < \sum_{j=1}^{|\mathcal{S}|} \sum_{\substack{k=1 \\ k \neq i}}^{N_j} q_k + T_{max} \tag{4.36}$$

$$l_i < \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{N_j} q_k + T_{max} - (q_s + T_{max}) \tag{4.37}$$

$$l_i < \sum_{j=1}^{|\mathcal{S}|} \sum_{k=1}^{N_j} q_k + (N-1)T_{max} - q_s \tag{4.38}$$

$$l_i < Q - q_s + (N-1)T_{max} \tag{4.39}$$

□

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 4.6 is that queues with a bigger *quantum* will have a lower latency.

In summary, the latency is given by:

- The number of queues in the system, which is given by the number of slices, and clients per slice.
- The size of the *quantum*.
- 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 analyzed before 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 as will be discussed in the next chapter.

4.4 A Use Case for Infrastructure-Sharing Slicing

In this section, we briefly explain how ATERR can be used to implement *Infrastructure Sharing Slicing* (ISS) in WiFi APs. The primary motivation for implementing ISS is an efficient use of radio resources. Sharing the wireless resources among clients, service providers, or network operators, contributes to the utilization of idle resources when possible, reducing the need for broadening network deployments.

In WiFi, it is becoming more popular to share residential APs for public use or to cooperate between different WiFi networks to provide service to a dense user area [121, 125, 100, 22]. For example, 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.

We envision an ISS approach where the slices negotiate a SLA with the infrastructure operator to obtain a portion of the network resources. The SLA could be defined with the airtime sharing parameters described earlier: p_s, K_s, and W_s. The SLA can also consider a fourth parameter D_s, which specifies the duration (lifetime) of the slice, which means that during this period, the slice provider guarantees the allocation of the resources. In this context, the ATERR mechanism can be used at the WiFi APs to provide the requested allocations and to guarantee the isolation between slices at the resource level. Even more, the results from the analytical study are fundamental for the negotiation of the SLA. For example, Theorem 4.3 imposes some limits on the possibilities of the tolerance value and allocation window size for a given requested ratio. Even more, this result can be used to decide if new slices or clients can be accepted in the AP, as the number of queues also impact on the allocation.

Finally, it is important to mention that it is not possible to implement ISS only with ATERR scheduling. A high-level management framework is also needed to coordinate the allocations at the different APs of the network. Our simple architecture presented in Section 2.5 would be an initial approach to this matter. However, there exist more elaborated and complex solutions that could ease the management of ISS in WiFi networks. In this respect, the recent work in [29] comprehensively review several SDN approaches to this issue.

4.5 Experimental Evaluation

As a proof of concept we have implemented ATERR in the NS3 Network Simulator [96], by modifying the existent transmission queuing structure of the wireless module. The simulation code is available at [91]. 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.

4.5.1 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 clients in a slice and the airtime-allocation requests.

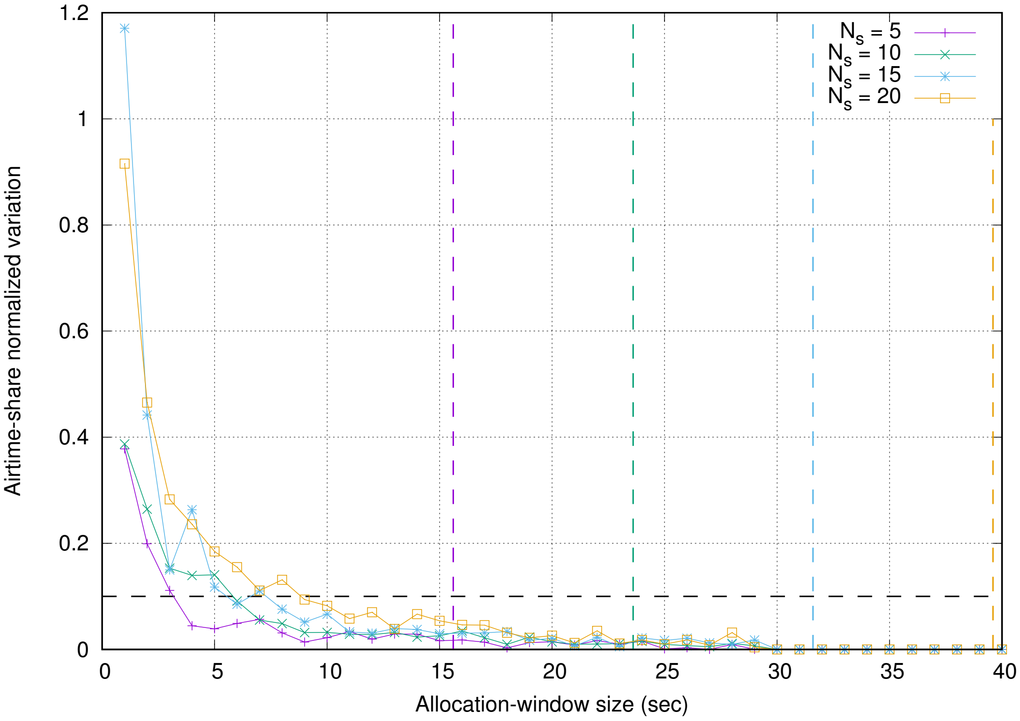

Fig. 4.4 Allocated airtime-share variation vs. allocation-window size for a requested airtime-share of 0.1.

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 $25m \times 25m$, to generate variability in the transmission rates and thus in the consumed airtime.

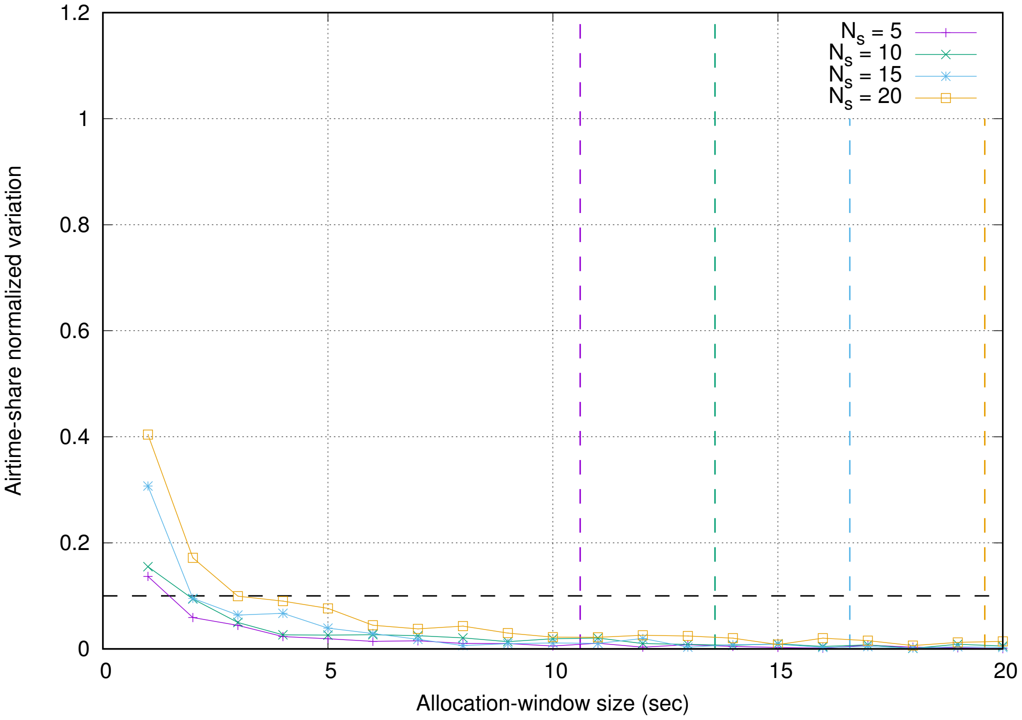

Fig. 4.5 Allocated airtime-share variation vs. allocation-window size for a requested airtime-share of 0.2.

We run experiments for two different airtime requests (p_s): 0.1 and 0.2. Figures 4.4 and 4.5 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 4.3.2.

There are four plots on each figure which depict the results obtained with different number of clients 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 computed from the result of Theorem 4.3.

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 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 clients in the slice and the airtime-share requested. It can be seen that as the number of clients 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.

4.5.2 Evaluation with UDP Traffic

The experiment consists on a deployment comprising one AP and 10 clients randomly located around the AP. Three slices are defined:

- Slice 1 requests 20% of the airtime ($p_s = 0.2$) and there are four clients associated with the slice (Clients 1, 2, 3 and 4).

- Slice 2 requests 20% of the airtime ($p_s = 0.2$) and there are four clients associated with the slice (Clients 4, 5, 6 and 7).

- Slice 3 requests 60% of the airtime ($p_s = 0.6$) and there are four clients associated with the slice (Clients 7, 8, 9 and 10).

It is worth noticing that Clients 4 and 7 belong to two slices.

In this experiment we evaluate two scenarios, one with static clients and one with mobile clients. For the static case, clients are randomly deployed around the AP with an average distance from the AP of 5 m. For the other case, clients 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 client with a high rate to maintain all queues backlogged. Simulations are repeated 20 times, varying the clients' 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 4.6 and 4.7 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

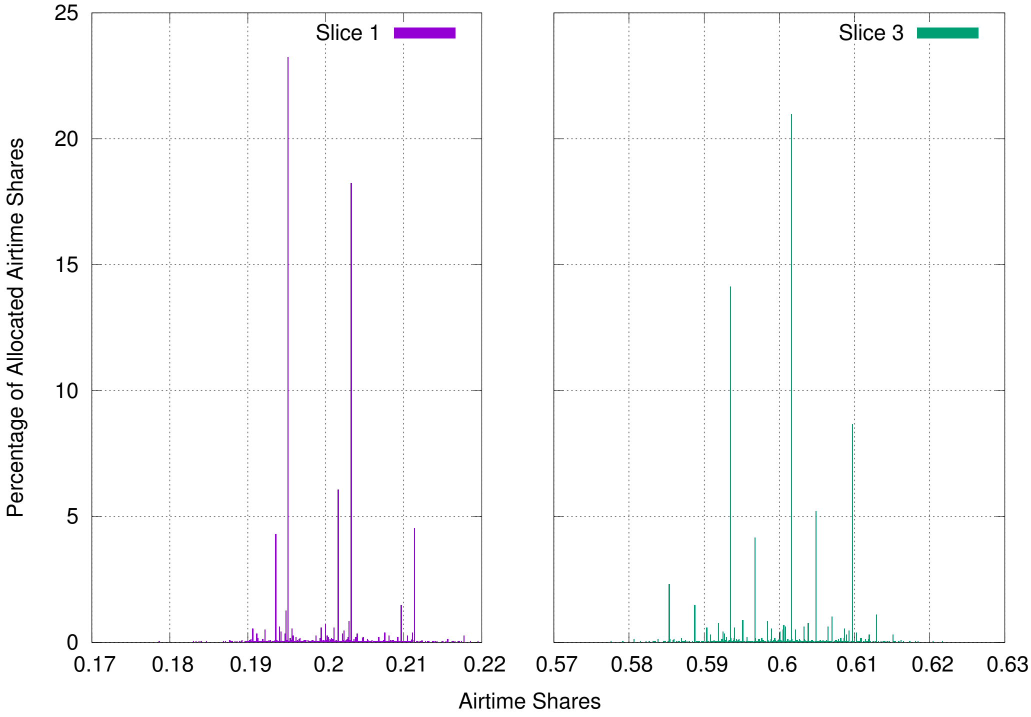

Fig. 4.6 Histogram of the airtime-shares allocated in the static scenario for Slice 1 ($p_s = 0.2$) and Slice 3 ($p_s = 0.6$).

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 client on every slice to assess the fairness among clients of the same slice. The results are shown in Figure 4.8. For the sake of clarity the figure shows the result for one simulation execution. The figure highlights that the assignments for each client are noticeably fair. To better assess the fairness, we calculated the Jain's fairness index of the assigned airtime to each client 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.

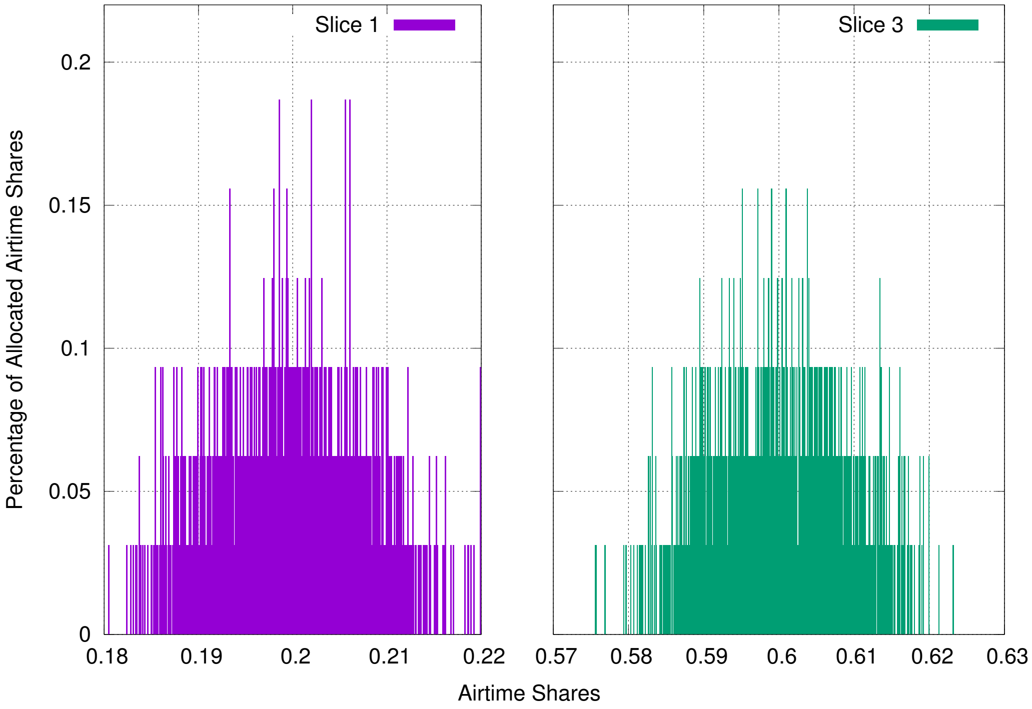

Fig. 4.7 Histogram of the airtime-shares allocated in the mobile scenario for Slice 1 ($p_s = 0.2$) and Slice 3 ($p_s = 0.6$).

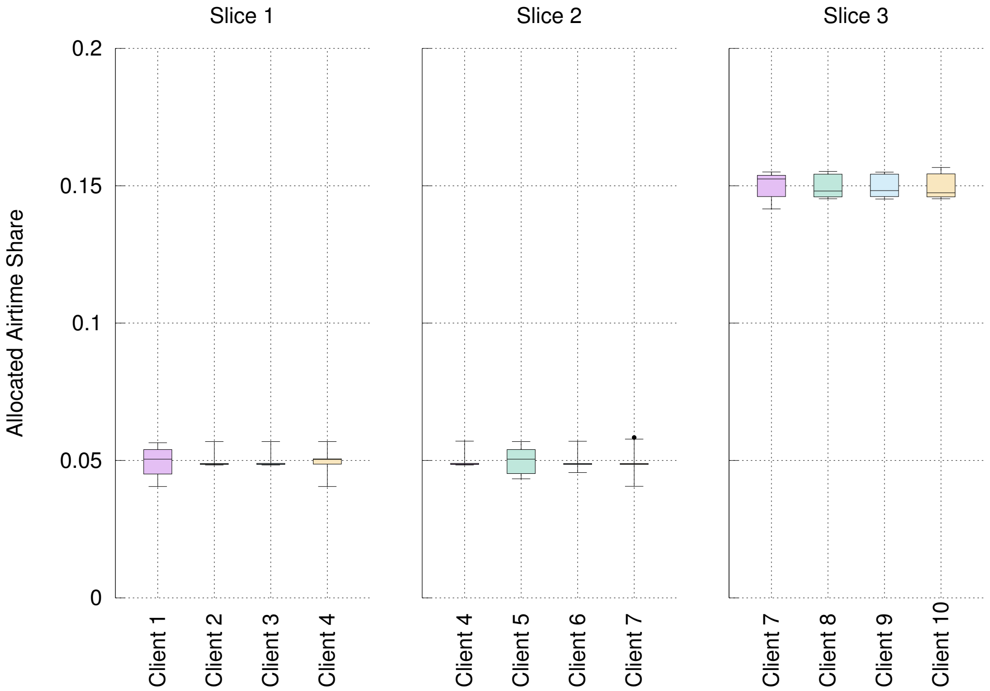

Fig. 4.8 Allocated airtime-share to each client of the slices.

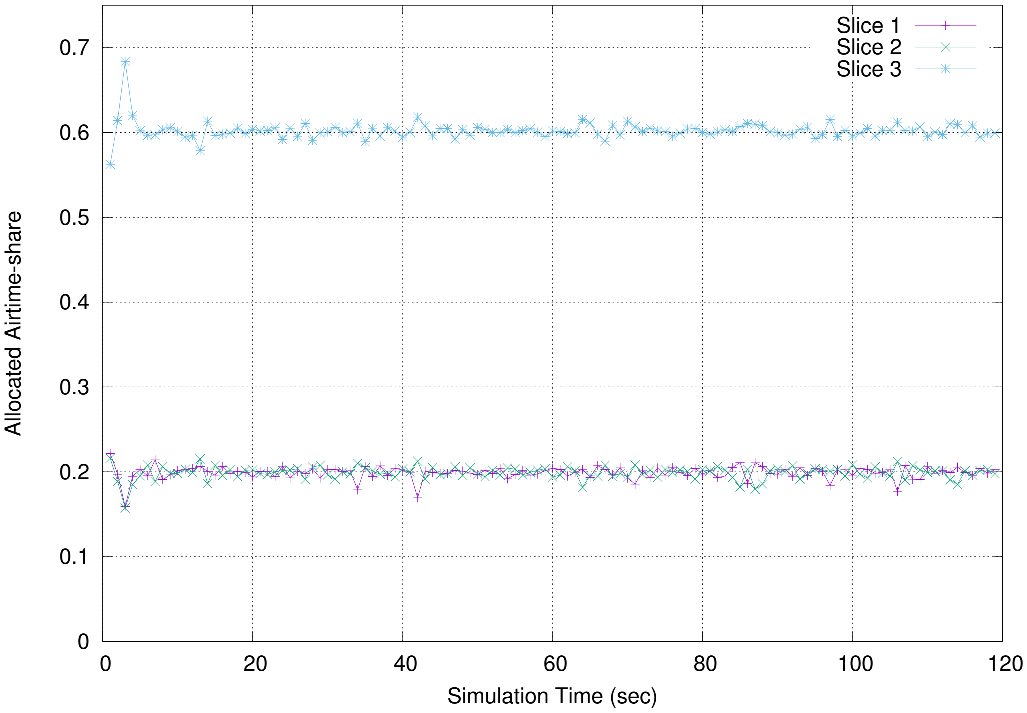

Fig. 4.9 Airtime-share allocation for TCP traffic.

4.5.3 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 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 4.9 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 4.10 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 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 4.11.

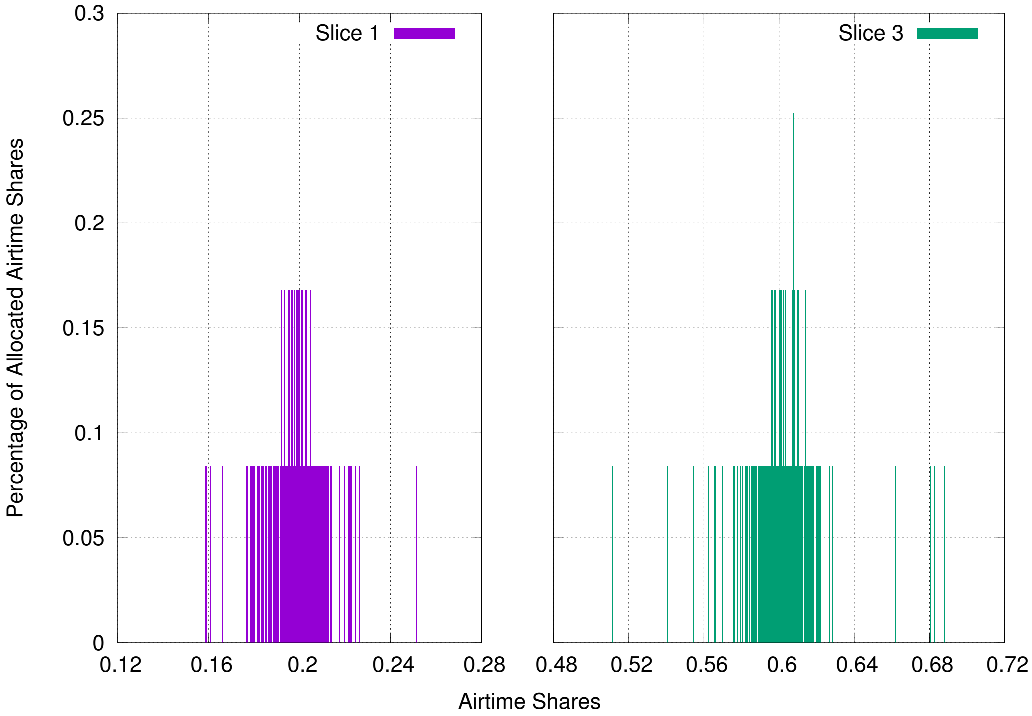

Fig. 4.10 Histogram of airtime-shares allocated in the TCP scenario for Slice 1 ($p_s = 0.2$) and Slice 3 ($p_s = 0.6$). Allocation-window of 1s.

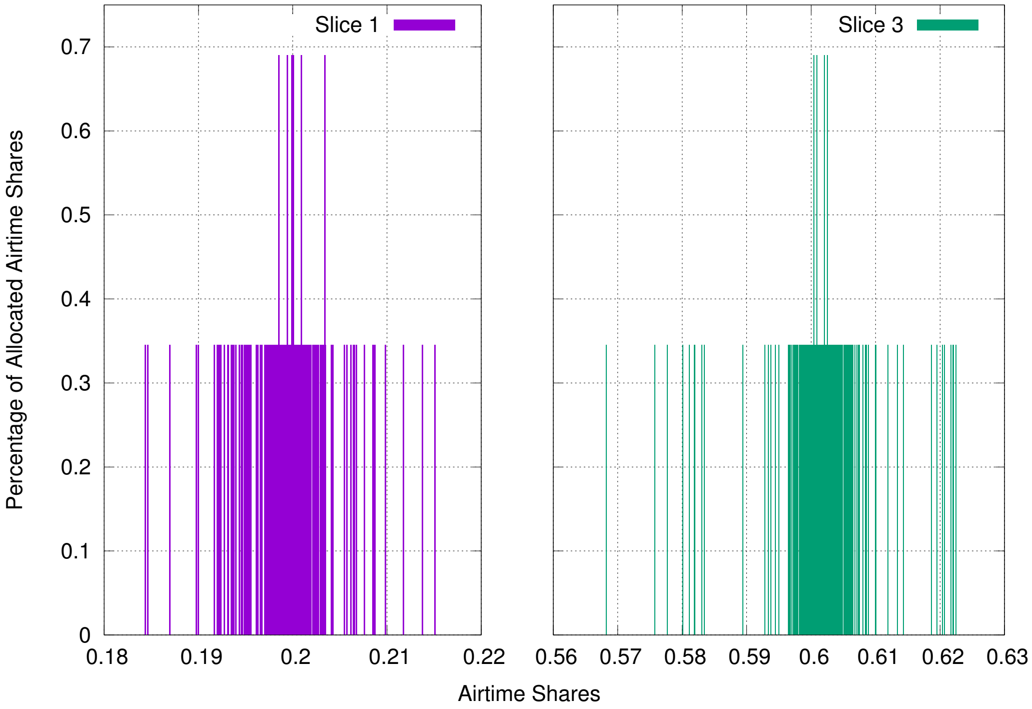

Fig. 4.11 Histogram of airtime-shares allocated in the TCP scenario for Slice 1 ($p_s = 0.2$) and Slice 3 ($p_s = 0.6$). Allocation-window of 4s.

4.5.4 Evaluation with UDP Fluctuating Traffic

This experiment is very similar to the previous one, but in this case, we use UDP 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.

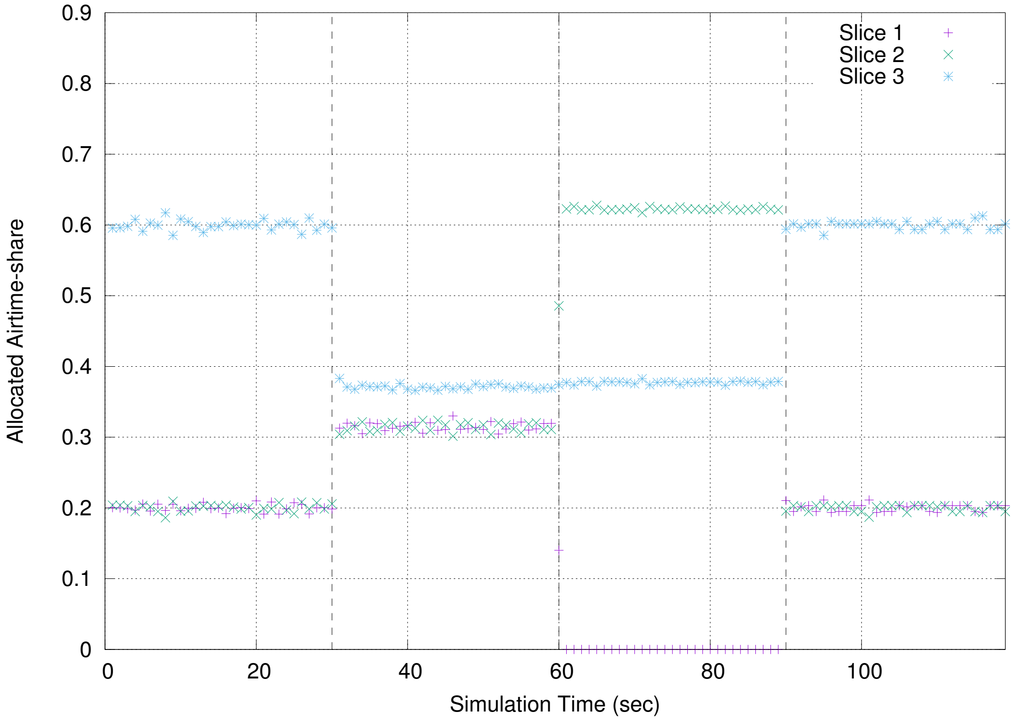

Fig. 4.12 Allocated airtime-share vs. simulation time for fluctuating UDP traffic.

The traffic configuration is as follows:

- At time 0, the traffic load to the three slices is high so that all slices use all the requested resources.
- At 30 seconds, the traffic to all clients of Slice 3 is reduced while maintaining a high load on the other slices.
- At 60 seconds, the traffic to all clients of Slice 1 is stopped while maintaining a medium load on Slice 3 and a high load on Slice 2.
- At 90 seconds, all traffic is restored to its initial values.

Figure 4.12 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 the traffic of each slice is high enough to use all the allocated 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. It is important to note that this is possible because the traffic load of Slices 1 and 2 is high enough to occupy the freed resources. 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. Again, this is possible because the traffic load of Slice 2 is high. 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.

4.6 Conclusions

This chapter describes the design, implementation, and evaluation of ATERR, a resource allocation mechanism based on airtime assignment to achieve slicing and infrastructure sharing in WiFi Access Points. The approach is aimed to be simple and has the potential to be easily used in different slicing scenarios. The solution is designed to work within an AP, and its main objective is to enforce and control airtime allocations to the slices deployed in the AP. The architecture is designed in a modular way, consisting of a controller that receives commands from a network manager and configures the system, a classifier to distribute the traffic to the different slices, and a scheduler that controls the packet transmissions.

The main novelty of the proposed 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 main differences between ATERR and previous works on airtime allocation 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 a deficit. This allows considering in the airtime assignment, the wireless medium variability produced by user mobility, interference, or congestion.

- Quantum sizes are adaptive and are computed based on the requested airtime share, the current number of slices, and the clients per slice. This way, ATERR supports slices' and clients' connections and disconnections dynamically.

The designed architecture also envisions two levels of control: a *fine-grained control* preformed by the ATERR Scheduler which enforces and controls that the airtime ratios requested are guaranteed and a *coarse-grained control* which is performed by a decision process at the *Network Slice Manager* which can take network-level decisions with global information and interfaces with the Scheduler through the ATERR Controller. In this regard, we show how ATERR (within this management architecture) can be used to implement Infrastructure Sharing Slicing. For this, we contribute with a possible set of parameters to define a slice's airtime allocation request, which considers an average airtime guarantee and a tolerance on this guarantee.

We also contribute 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 analytical model, it is possible to determine if an airtime-share request can be accepted. This would help to manage the slices, allowing the implementation of client access-control. It is also a valuable 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. As a first step, we perform experiments to validate the analytical results regarding the impact of the allocation-window size on the airtime assignment. Secondly, we studied the key characteristics of the proposal under different scenarios. The experiment results specifically illustrate the accurate airtime-share allocation based on slice requests, acceptable airtime fairness among clients of a slice, and the efficient utilization and isolation of resources between slices.

In summary, we have proposed ATERR, an enforcement and control mechanism which fulfills the objectives proposed in this thesis to efficiently allocate resources to slices and to provide slice isolation. On the other hand, ATERR, on its own, does not provide performance guarantees, which is also an objective of this thesis. However, as will be shown in the next chapter, some of the concepts of ATERR can be used to implement QoS Slicing and provide throughput and delay guarantees.

Chapter 5

Quality of Service Slicing for WiFi

5.1 Introduction

The airtime resource allocation proposal from Chapter 4 implements static allocation in that it just fulfills the allocation requests it receives. It allocates a fixed amount of resources only based on the requested ratio of each slice, which, given the characteristics of the wireless medium, makes it impossible to guarantee any performance metric. In other words, it does not consider the possibility of adapting the resource allocation to the different slices based on the achieved throughput or delay of the slice's traffic. As previously discussed in Chapter 2, the variability in the capacity of the wireless channel makes that the necessary airtime to achieve a given performance (e.g., throughput) to also variate. For example, if the channel conditions deteriorate, then the capacity decreases, which causes that more airtime is needed to achieve the same performance. Therefore, to be able to implement slices that can guarantee some performance requirements, it is necessary to dynamically allocate airtime to slices based on the current channel conditions.

The approach of slicing with performance guarantees has been called *Quality of Service Slicing*, to differentiate it from other slicing variants. In this slicing approach, 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. In Section 2.4.2, we have already analyzed the requirements and challenges of *Quality of Service Slicing*. Briefly, the objective is to support slices with specific performance guarantees where the performance requirement must be guaranteed to every flow within the slice. The main challenge is that the capacity of the transmission medium (wireless channel) is dynamic and, in general, unknown beforehand and unpredictable.

In this chapter, we propose a solution to implement QoS Slicing in WiFi APs, which considers two performance metrics: bit rate and delay. More specifically, the solution seeks to provide a minimum guaranteed transmission bit rate and guaranteed bounded queuing delay at the AP. The proposed approach consists of a dynamic airtime allocation mechanism that uses the information of the channel conditions to decide the necessary allocation to achieve and maintain the required performance guarantees on each slice. The solution reuses the classifier and queuing structure proposed in Chapter 4, and a new scheduling mechanism is developed for jointly implementing slices that can guarantee a minimum bit rate and bounded delay in the AP transmissions. Similarly to the ATERR Scheduler from Chapter 4, the solution described in this chapter also considers the transmission time (airtime) as the resource to share and implements a scheduling algorithm with airtime quantum allocation. However, the allocation is based on the slices' performance requirements and wireless channel characteristics. Even more, differently from ATERR, the allocation of the airtime to the slices is dynamic. This way, the mechanism keeps track of the evolution of the channel conditions and reallocates the airtime to fulfill the performance requests.

Nevertheless, the solution described in this chapter maintains two crucial characteristics from the ATERR solution: (1) the resource allocation is efficient, as each slice receives the exact amount of resources to achieve the required performance, so additional slices can use available resources; and (2) it provides isolation between slices, to protect assigned resources and hence the performance.

The rest of the chapter is organized as follows. In Section 5.2, we analyze the characteristics of the guaranteed bit rate and bounded delay problem. Section 5.3 presents a brief introduction to *Lyapunov Optimization Theory* which will be used in the design of our solution. Section 5.4 describes the system model developed to represent the QoS Slicing problem while Section 5.5 presents the problem formulation as a stochastic optimization problem. In Section 5.6 we propose a solution based on the drift-plus-penalty method from the Lyapunov Optimization Theory while in Section 5.7 we analyze the proposed solution and show its main performance characteristics. In Section 5.9 we discuss several considerations concerning the implementation of our solution for WiFi technology while in Section 5.8 we describe our mechanism to detect unfeasible scenarios and guarantee isolation. In Section 5.10 we present an experimental evaluation of our solution. Finally, in Section 5.11 we conclude the chapter by identifying the main contributions of this work.

5.2 Bit Rate and Delay Guarantees

Even though network slicing is a rather new concept, the problem of providing Quality of Service in IEEE 802.11 networks has been thoroughly studied during the last twenty years. Although in radio access networks the most predominant traffic is downlink (from the AP to the stations), most of the research on this subject has focused on providing QoS on the uplink (the transmissions from the stations to the AP), or between stations [71]. This is because WiFi networks have been traditionally seen as an extension of the wired local area network, and not as an Internet access alternative. Moreover, in many of those previous works, QoS provisioning is based on service differentiation and prioritization, but not on performance guarantees. Regarding bit rate (or throughput) guarantees for traffic from stations to the AP, the works of Banchs et al. [12, 13] are worth mentioning. They foster throughput guarantees through access management schemes, controlling the Contention Window (CW) size. Differently, 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 delay control, one of the main components affecting packet delay in a network is queuing. Therefore, this proposal concentrates on the problem of controlling the queuing delay at the AP. Queuing delay is caused by the accumulation of packets in a queue, originated by the difference between the arrival and departure rates of packets and is affected by two main factors: the arrival and departure traffic rates and the formation of *standing queues* because of traffic bursts. If the arrival rate is continuously higher than the departure rate, queues will build-up, and the delay would increase to infinity. Nevertheless, in more benevolent situations, queues can also introduce delays. In today's networks, queues play an essential role in absorbing traffic bursts, which are commonly generated on the Internet (for example, because of the TCP transmission window). However, these traffic bursts can cause *persistent queues* if, after a burst, the queue is not completely emptied. This problem is called the *persistent full buffer problem* or *bufferbloat*, and is one of the leading causes of queuing delay in the network edge. Moreover, in wireless networks, persistent queues can also be formed because of the burstiness of departures, since they use a shared medium and also due to the variable bit rate of the wireless transmissions. This makes queues to accumulate packets while waiting for a transmission opportunity and, consequently, to send them all in a burst, when the access to the medium is granted.

Hence, controlling queuing delay implies keeping the length of the queues, yet taking into account all the above considerations. Notice that if the queue length is too short, the delay could be reduced, but at the cost of high packet loss and so throughput reduction. On the other hand, with too long queues, no packets would be lost, maximizing the throughput, but leading to excessive delays. In this regard, some approaches have dealt with bufferbloat in WiFi by controlling the queue length through Active Queue Management (AQM) techniques. AQM mainly consists of intelligently dropping packets to avoid queues to grow indefinitely. In general, the main objective of AQM is to reduce network congestion by signaling (either implicitly, by dropping packets, or explicitly, by sending special packets) the sender to reduce the flow rate. Nevertheless, AQM has also a clear impact on queuing delay. A recent AQM technique (which has been very well received by academia and industry) is CoDel (from Controlled Delay) [85], which, instead of using the more traditional method of measuring the queue length to make a dropping decision (like the well-known RED algorithm [34]), uses the waiting time of packets in the queue (the queuing delay). However, AQM techniques have proved not to achieve good latency reductions in WiFi [48] because of the queuing at the lower layers of the WiFi stack. Therefore, the work in [49] implements a queue management mechanism at the queuing structure of the lower layers. The objective of this work is to reduce queuing delay while achieving airtime fairness. Nevertheless, it does not provide a bounded delay guarantee, nor it considers the slicing concept.

A different approach to provide QoS guarantees (particularly bit rate and delay) in wireless networks is to manage how the packets are scheduled for transmission. In the context of scheduling algorithms, in the last twenty-five years, the concept of *opportunistic scheduling* has been thoroughly studied to provide QoS guarantees [11]. These schedulers take advantage of physical layer information, such as the client channel capacity or local system information, such as the queue lengths. Nevertheless, all of the existing works have concentrated on theoretical proposals or cellular technologies, but, to the best of our knowledge, they have overlooked WiFi. In [10] a scheduler called *Modified Largest Weighted Delay First* (M-LWDF) is proposed to achieve average delay guarantees to traffic flows. The method is based on scheduling the packet with the largest product of queue length and transmission rate. A similar approach is proposed in [65], where users are prioritized based on an exponential formula using queue length and transmission rate. In [98] an approach is discussed where the scheduling function is based on a logarithmic expression of the queue lengths and the transmission rate. In [58] a scheduler is proposed where each QoS flows receive a fixed average throughput per slot, but other QoS objectives such as delay are not considered. However, most of those works consider priorities between different services, and the objective is to reduce the latency of the high priority flows. Conversely, in the slicing scenario, all slices are equally important but have different QoS requirements, which all have to be guaranteed simultaneously.

The solution presented in this chapter is based on the works from Michael J. Neely on opportunistic scheduling [82, 80, 81]. The work of Neely shares some similarities with the works previously mentioned but has some particular characteristics which make it more adequate for the slicing problem. The approach followed by Neely is to formulate the scheduling problem as a stochastic optimization problem, where the channel conditions and the arrival rates are unknown stochastic processes. Then, using the *Lyapunov Optimization Theory*, an optimization framework is proposed to solve this problem. The framework permits to obtain an equivalent deterministic problem, which provides an approximate bounded solution to the original stochastic one. This scheme provides the advantage that if the scheduling problem can be formulated as an optimization problem, a set of theoretic tools can be applied, and an approximate solution can be obtained. Even more, the distance between the optimal solution and the approximate solution is bounded by an adjustable parameter.

Therefore, one of the main contributions of this thesis is to consider the QoS Slicing problem as a stochastic optimization problem. We develop a system model and formulate the guaranteed bit rate and bounded queuing delay slicing problem as a stochastic optimization problem. Even more, after applying the Lyapunov Optimization Theory, we were able to build a very efficient scheduling algorithm from the obtained solution. The main contributions and differences of our proposal with previous works on opportunistic scheduling consist on that all existent works consider slotted time; however, in WiFi, there are no time slots predefined. Hence, we needed to develop a mechanism to adapt the scheduling solution to WiFi. Also, all existent previous works have focused on the cases where there is always a feasible solution and have disregarded unfeasible situations. In WiFi, because of the transmission medium characteristics, unfeasible cases may emerge during the scheduling process. Therefore we have implemented a solution to tackle this issue (see Section 5.8). Furthermore, we use the queuing model proposed in Chapter 4, which considers the particularities of the hardware behavior, to avoid queue buildup at lower layers and to allow packet aggregation.

5.3 Stochastic Network Optimization

This section provides a brief explanation of the theory used in the QoS Slicing solution adopted in this chapter. For a formal presentation as well as a generalization of the theory we recommend the texts [38] and [80]. This presentation is strongly based on [38], [80] and [82] where Stochastic Network Optimization and the Lyapunov Optimization Theory are presented and applied to Communications and Queuing Systems.

5.3.1 The Model

The theory we are presenting considers a system with K queues and define $Q_k(t)$ as the queue backlog of queue k on time t. It also defines $\mathbf{Q}(t) = (Q_1(t), ..., Q_K(t))$ as the queue backlog vector. The vector backlog $\mathbf{Q}(t)$ can be seen as a partially controllable stochastic process which evolves over slots t and depends on a *network state variable* $\omega(t)$ and a *control action* $\alpha(t)$. The network state variable $\omega(t)$ varies randomly depending on the network conditions and represents all uncontrollable properties of the network that affect the system. The control action $\alpha(t)$ takes values in a set of possible control actions $\mathcal{A}_{\omega(t)}$ which depend on the network state $\omega(t)$.

Queue dynamics are given by:

$$Q_k(t+1) = [Q_k(t) - b_k(t)]^+ + a_k(t) \tag{5.1}$$

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

where $a(t)$ and $b(t)$ are functions of $\omega(t)$ and $\alpha(t)$. $a(t)$ is the amount of new arrivals that enter the queue during slot t and $b(t)$ represents the amount of data that can be processed and removed from the queue on slot t. Both $a(t)$ and $b(t)$ are potentially random functions, as $a(t)$ can be composed of exogenous and uncontrollable arrivals and $b(t)$ may depend on the random network state. We can show this dependence in the following way:

$$\begin{aligned} a(t) &= \hat{a}(\alpha(t), \omega(t)) \\ b(t) &= \hat{b}(\alpha(t), \omega(t)) \end{aligned} \tag{5.2}$$

In this regard, the concept of queue stability will be of utmost importance in this theory. Therefore, [80] introduces two important definitions:

Definition 5.1. A discrete time process $Q(t)$ is *mean rate stable* if:

$$\lim_{t\to\infty} \frac{\mathbb{E}\{|Q(t)|\}}{t} = 0 \tag{5.3}$$

Definition 5.2. A discrete time process $Q(t)$ is *strongly stable* if:

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{|Q(\tau)|\} < \infty \tag{5.4}$$

In this context we can define that a system of K queues is *strongly stable* if all queues of the system are strongly stable. Even more, in [80] it is shown that under some mild boundless assumptions, strong stability implies mean rate stability. Also, to simplify the presentation, during this description we will assume all taken limits exist. Raising this assumption is very easy by accordingly replacing lim by lim sup or lim inf.

5.3.2 The Problem

The theory we are using focuses on problems which seek to minimize the time average expectation of a network penalty function, subject to network stability and additional time average constraints.

Let us consider a stochastic penalty process $y(t) = \hat{y}(\alpha(t), \omega(t))$ which we want to minimize. The process $y(t)$ can represent penalties incurred by control actions on slot t, such as power consumption, packet drops, etc. Consider also a set of additional stochastic penalty process e_j for $j \in \{1, ..., J\}$ which act as constraints in our system and which also depend on the network state $\omega(t)$ and on the control action $\alpha(t)$ in slot t. There exists a controller which, on every slot t, observes the current values of $\omega(t)$ and $\mathbf{Q}(t)$ and makes a control action $\alpha(t)$. The objective is to find a control policy (a sequence of control actions $(\alpha(1), \alpha(2), ...)$) that solves the following optimization problem:

$$\begin{aligned} &\text{minimize} && \overline{y} && (5.5)\\ &\text{subject to} && \overline{e}_j \leq E_j \;\; \forall\, j \in (1, ..., J), && (5.6)\\ & && \lim_{t\to\infty} \frac{\mathbb{E}\{Q_k(t)\}}{t} = 0 \;\; \forall\, k \in (1, ..., K). && (5.7) \end{aligned}$$

where:

$$\begin{aligned} \overline{y} &= \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{y(\tau)\} \\ \overline{e}_j &= \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{e_j(\tau)\} \end{aligned} \tag{5.8}$$

In constraint (5.6) E_j are average penalty bounds and constraint (5.7) implies that all queues are mean rate stable. In summary, the objective is to minimize the time average expected penalty while assuring the time average expectation of the constraints and assuring the queues are stable.

As mentioned before, $y(t)$, $e_j(t)$ and $Q_k(t)$ are stochastic processes which depend on $\alpha(t)$ and $\omega(t)$ random variables. Hence, we have formalized the minimization problem as a *stochastic optimization problem.* It is important to note that the problem can be formulated in an analog way if $y(t)$ is considered a *reward* that we want to maximize.

5.3.3 Virtual Queues

To solve the problem (5.5)-(5.7), the first step of the method we are describing is to transform the constraints into queue stability problems. For each of the J constraints in (5.6) it is defined a *virtual queue* $Z_j(t)$ with update equation:

$$Z_j(t+1) = [Z_j(t) + e_j(t) - E_j]^+ \tag{5.9}$$

These queues are totally virtual and do no represent the real network queues. In the following, we will show that requiring mean rate stability on these virtual queues guarantees that the constraints are all satisfied.

From the previous queue dynamics definition it is easy to show that, for any $\tau \geq 0$:

$$Z_j(\tau+1) - Z_j(\tau) \geq e_j(t) - E_j \tag{5.10}$$

Summing the above equation over $\tau \in \{0, ..., t-1\}$, using the law of telescoping sums and dividing by t yields:

$$\frac{Z_j(t)}{t} - \frac{Z_j(0)}{t} \geq \frac{1}{t}\sum_{\tau=0}^{t-1} e_j(t) - E_j \tag{5.11}$$

Taking expectations and taking the limit $t \to \infty$ over the previous equations shows:

$$\lim_{t\to\infty} \frac{\mathbb{E}\{Z_j(t)\}}{t} \geq \lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{e_j(t)\} - E_j \tag{5.12}$$

Then, from the definition of mean rate stability we know that the left-hand side of the previous equation is 0, therefore:

$$\lim_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{e_j(\tau)\} \leq E_j$$

which are the initial constraints of the problem.

Hence, the time average constraints are transformed into a queue stability problem. Therefore, the problem (5.5) can be written as:

$$\begin{aligned} \text{minimize} \quad & \overline{y} \\ \text{subject to} \quad & \lim_{t\to\infty} \frac{\mathbb{E}\{|Z_j(t)|\}}{t} = 0 \ \ \forall\, j \in (1, ..., J), \\ & \lim_{t\to\infty} \frac{\mathbb{E}\{Q_k(t)\}}{t} = 0 \ \ \forall\, k \in (1, ..., K). \end{aligned} \tag{5.13}$$

In this new formulation all the constraints are queue stability requirements. As we will show next, this allows to apply the Lyapunov Optimization Theory to this problem. Note that in this problem formulation it is required that the queues are mean rate stable. As shown before, mean rate stability is enough to ensure constraint satisfaction. However, the theory presented here can also guarantee (under some additional assumptions) strong stability over all queues.

5.3.4 Lyapunov Drift Theorem

Generalizing the previous model, the theory considers a system of N queues (which includes real queues $Q_k(t)$ and virtual queues $Z_j(t)$) with queue backlog $\Theta_n(t)$. Consider also a vector of queue backlogs $\mathbf{\Theta}(t) = (\Theta_1(t), ..., \Theta_N(t))$ and let us define a *Lyapunov function* as a measure of the size of this vector:

$$L(\mathbf{\Theta}(t)) = \frac{1}{2} \sum_{n=1}^{N} \Theta_n(t)^2 \tag{5.14}$$

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

$$\Delta(\mathbf{\Theta}(t)) = \mathbb{E}\{L(\mathbf{\Theta}(t+1)) - L(\mathbf{\Theta}(t)) | \mathbf{\Theta}(t)\} \tag{5.15}$$

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

In this regard, [80] presents the following theorem which we reproduce here for the sake of completeness.

Theorem 5.1. *(Lyapunov Drift) Consider the quadratic Lyapunov function defined above, and assume* $\mathbb{E}\{L(\mathbf{\Theta}(0))\} < \infty$*. Suppose there are constants* $B > 0$*,* $\epsilon \geq 0$ *such that the following drift condition holds for all slots* $\tau \in \{0, 1, 2, ...\}$ *and all possible* $\mathbf{\Theta}(\tau)$*:*

$$\Delta(\boldsymbol{\Theta}(\tau)) \leq B - \epsilon \sum_{n=1}^{N} |\Theta_n(\tau)| \tag{5.16}$$

Then:

1. *If* $\epsilon \geq 0$ *then all queues* $\Theta_n(\tau)$ *are mean rate stable.*
2. *If* $\epsilon > 0$ *then all queues are strongly stable and:*

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n=1}^{N} \mathbb{E}\{|\Theta_n(\tau)|\} \leq \frac{B}{\epsilon} \tag{5.17}$$

In what follows we also reproduce (with some modifications for clarity) the proof of part (2) of the theorem as it provides some insight about the Lyapunov theory applied.

Proof. Taking expectation on both sides of (5.16) yields:

$$\mathbb{E}\{\Delta(\boldsymbol{\Theta}(\tau))\} \leq \mathbb{E}\{B\} - \mathbb{E}\{\epsilon \sum_{n=1}^{N} |\Theta_n(\tau)|\}$$

$$\mathbb{E}\{\mathbb{E}\{L(\boldsymbol{\Theta}(\tau+1)) - L(\boldsymbol{\Theta}(\tau))|\boldsymbol{\Theta}(\tau)\}\} \leq B - \epsilon \sum_{n=1}^{N} \mathbb{E}\{|\Theta_n(\tau)|\}$$

Using the *law of iterated expectations* we have:

$$\mathbb{E}\{L(\boldsymbol{\Theta}(\tau+1)) - L(\boldsymbol{\Theta}(\tau))\} \leq B - \epsilon \sum_{n=1}^{N} \mathbb{E}\{|\Theta_n(\tau)|\}$$

$$\mathbb{E}\{L(\boldsymbol{\Theta}(\tau+1))\} - \mathbb{E}\{L(\boldsymbol{\Theta}(\tau))\} \leq B - \epsilon \sum_{n=1}^{N} \mathbb{E}\{|\Theta_n(\tau)|\}$$

Summing the above over $\tau \in \{0, 1, ..., t-1\}$ for some $t > 0$ and using the law of telescoping sums yields:

$$\mathbb{E}\{L(\boldsymbol{\Theta}(t))\} - \mathbb{E}\{L(\boldsymbol{\Theta}(0))\} \leq Bt - \epsilon \sum_{\tau=0}^{t-1} \sum_{n=1}^{N} \mathbb{E}\{|\Theta_n(\tau)|\} \tag{5.18}$$

Now assume that $\epsilon > 0$. Rearranging terms and dividing by $t\epsilon$ we get:

$$\epsilon \sum_{\tau=0}^{t-1} \sum_{n=1}^{N} \mathbb{E}\{|\Theta_n(\tau)|\} \leq Bt + \mathbb{E}\{L(\boldsymbol{\Theta}(0))\} - \mathbb{E}\{L(\boldsymbol{\Theta}(t))\}$$

$$\frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n=1}^{N} \mathbb{E}\{|\Theta_n(\tau)|\} \leq \frac{B}{\epsilon} + \frac{\mathbb{E}\{L(\boldsymbol{\Theta}(0))\}}{t\epsilon} - \frac{\mathbb{E}\{L(\boldsymbol{\Theta}(t))\}}{t\epsilon}$$

Using the fact that $\mathbb{E}\{L(\boldsymbol{\Theta}(t))\} \geq 0$ yields:

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{n=1}^{N}\mathbb{E}\{|\Theta_n(\tau)|\} \leq \frac{B}{\epsilon} + \frac{\mathbb{E}\{L(\mathbf{\Theta}(0))\}}{t\epsilon} \tag{5.19}$$

Taking limit as $t \to \infty$ proves the part (2) of the theorem. □

In summary, the above theorem shows that if the Lyapunov drift is bounded as required in (5.16) then all queues are mean rate stable. Also, if $\epsilon > 0$, then all queues are strongly stable with time average expected queue backlog bounded by $^{B}/_{\epsilon}$. As it was described previously, to assure the problem's constraints are respected, mean rate stability on the virtual queues is required. Therefore, this theorem transforms the problem of queue stability into a problem of bounding the Lyapunov drift of the queues. Moreover, from equation (5.19) it is easy to observe that with an additional initial condition of all queues being empty at $t = 0$ (i.e. $L(\mathbf{\Theta}(0)) = 0$) then the $^{B}/_{\epsilon}$ bound is satisfied for all t without needing to take limits.

5.3.5 Lyapunov Optimization Theorem

Now, as it was explained in Section 5.3.2, in addition to stabilizing the queues $\mathbf{\Theta}(t)$, the formulated problem has an associated stochastic penalty process $y(t)$. Suppose that the objective is to make the time average of $y(t)$ less than some target value y^*. Assume the expected penalty is lower bounded by a finite (possibly negative) value y_{min}, so that for all t and all possible control actions, we have:

$$\mathbb{E}\{y(t)\} \geq y_{min} \tag{5.20}$$

where the expectation is with respect to the $\omega(t)$ random process and the control decision $\alpha(t)$.

We reproduce here the following theorem, which is presented in detail in [80]. In this case we omit the proof as it is very similar to the previous theorem and is provided in [80].

Theorem 5.2. *(Lyapunov Optimization) Suppose $L(\mathbf{\Theta}(t))$ and y_{min} are defined by (5.14) and (5.20), and that $\mathbb{E}\{L(\mathbf{\Theta}(0))\} < \infty$. Suppose there are constants $B \geq 0$, $V > 0$, $\epsilon > 0$, and y^* such that for all slots $\tau \in \{0, 1, 2, ...\}$ and all possible values of $\mathbf{\Theta}(\tau)$, we have:*

$$\Delta(\mathbf{\Theta}(\tau)) + V\mathbb{E}\{y(\tau)|\mathbf{\Theta}(\tau)\} \leq B + Vy^* - \epsilon\sum_{n=1}^{N}|\Theta_n(\tau)| \tag{5.21}$$

Then all queues $\Theta_n(t)$ are mean rate stable and the time average expected penalty and queue backlog satisfy:

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{y(\tau)\} \leq y^* + \frac{B}{V} \tag{5.22}$$

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{n=1}^{N} \mathbb{E}\{|\Theta_n(\tau)|\} \leq \frac{B + V(y^* - y_{min})}{\epsilon} \tag{5.23}$$

Therefore, if the condition (5.21) can be satisfied on every slot τ and for any parameter $V > 0$, then the time average expected penalty satisfies (5.22). This means that the obtained penalty is less than the value y^* plus a given constant. In other words, the obtained penalty differs from the value y^* by no more than $^B/_V$. Even more, this difference can be made arbitrarily small as V is increased. On the other hand, from (5.23) follows that the bound of the time average queue backlog increases linearly with V. In [80] this is defined as a *performance-backlog tradeoff* of $[O(1/V), O(V)]$.

Even more, an intermediate result (which appears in the demonstration of the previous theorem) provides a notion on how the initial backlog of the queues influence on the bound. We reproduce it here because we consider it important for our solution:

$$\frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{y(\tau)\} \leq y^* + \frac{B}{V} + \frac{\mathbb{E}\{L(\mathbf{\Theta}(0))\}}{Vt} \tag{5.24}$$

As can be seen, the influence of the initial backlog in the obtained utility decreases linearly with t.

It is easy to observe that the previous theorem allows us to transform the stochastic optimization problem into a problem that must be solved on every slot τ. Hence, the objective is to design a control algorithm that on every slot τ takes a control decision that, subject to $\mathbf{\Theta}(\tau)$ (which on instant τ can be observed and therefore is known), minimizes the expression on the left-hand-side of (5.21): $\Delta(\mathbf{\Theta}(\tau)) + V\mathbb{E}\{y(\tau)|\mathbf{\Theta}(\tau)\}$.

5.3.6 The Drift-Plus-Penalty Algorithm

Given that $\mathbf{\Theta}(t)$ is the concatenated vector of the real queues vector $\mathbf{Q}(t)$ and the virtual queues vector $\mathbf{Z}(t)$. Then:

$$L(\mathbf{\Theta}(t)) = \frac{1}{2} \sum_{k=1}^{K} Q_k(t)^2 + \frac{1}{2} \sum_{j=1}^{J} Z_j(t)^2 \tag{5.25}$$

Given this backlog vector, the objective of the presented method is, as discussed previously, to find a control policy which minimizes the *drift-plus-penalty* expression on every time slot:

$$\Delta(\mathbf{\Theta}(t)) + V\mathbb{E}\{y(t)|\mathbf{\Theta}(t)\} \tag{5.26}$$

An Upper Bound on the Drift-Plus-Penlaty Expression

Therefore, in the following, we will schematically show a derivation of an upper bound on the previous drift expression which is satisfied by any control algorithm, for all t and for all $V \geq 0$.

By definition we have that:

$$\begin{aligned}
\Delta(\mathbf{\Theta}(t)) &= \mathbb{E}\{L(\mathbf{\Theta}(t+1)) - L(\mathbf{\Theta}(t)) \mid \mathbf{\Theta}(t)\} \\
&= \mathbb{E}\left\{\frac{1}{2}\sum_{k=1}^{K} Q_k(t+1)^2 + \frac{1}{2}\sum_{j=1}^{J} Z_j(t+1)^2 - \frac{1}{2}\sum_{k=1}^{K} Q_k(t)^2 - \frac{1}{2}\sum_{j=1}^{J} Z_j(t)^2 \mid \mathbf{\Theta}(t)\right\} \\
&= \mathbb{E}\left\{\frac{1}{2}\sum_{k=1}^{K} Q_k(t+1)^2 - \frac{1}{2}\sum_{k=1}^{K} Q_k(t)^2 + \frac{1}{2}\sum_{j=1}^{J} Z_j(t+1)^2 - \frac{1}{2}\sum_{j=1}^{J} Z_j(t)^2 \mid \mathbf{\Theta}(t)\right\} \\
&= \mathbb{E}\left\{\frac{1}{2}\sum_{k=1}^{K}[Q_k(t+1)^2 - Q_k(t)^2] + \frac{1}{2}\sum_{j=1}^{J}[Z_j(t+1)^2 - Z_j(t)^2] \mid \mathbf{\Theta}(t)\right\}
\end{aligned}$$

Hence:

$$\Delta(\mathbf{\Theta}(t)) = \mathbb{E}\left\{\sum_{k=1}^{K}\frac{Q_k(t+1)^2 - Q_k(t)^2}{2} + \sum_{j=1}^{J}\frac{Z_j(t+1)^2 - Z_j(t)^2}{2} \mid \mathbf{\Theta}(t)\right\} \tag{5.27}$$

Let us consider the update expression of the queues $Q_k(t)$:

$$Q_k(t+1) = [Q_k(t) - b_k(t)]^+ + a_k(t) = \max[Q_k(t) - b_k(t), 0] + a_k(t)$$

Squaring the previous expression and using the fact that $\max[q-b,0]^2 \leq (q-b)^2$ we get:

$$\begin{aligned}
Q_k(t+1)^2 &= \max[Q_k(t) - b_k(t), 0]^2 + a_k(t)^2 + 2\max[Q_k(t) - b_k(t), 0]a_k(t) \\
&\leq (Q_k(t) - b_k(t))^2 + a_k(t)^2 + 2\max[Q_k(t) - b_k(t), 0]a_k(t) \\
&\leq Q_k(t)^2 + b_k(t)^2 - 2Q_k(t)b_k(t) + a_k(t)^2 + 2\max[Q_k(t) - b_k(t), 0]a_k(t)
\end{aligned}$$

Let us define $\tilde{b}_k(t) = \min[Q_k(t), b_k(t)]$, then $\max[Q_k(t) - b_k(t), 0] = Q_k(t) - \tilde{b}_k(t)$ which yields:

$$Q_k(t+1)^2 \leq Q_k(t)^2 + b_k(t)^2 - 2Q_k(t)b_k(t) + a_k(t)^2 + 2(Q_k(t) - \tilde{b}_k(t))a_k(t)$$

Therefore:

$$\begin{aligned} \frac{Q_k(t+1)^2 - Q_k(t)^2}{2} &\leq \frac{a_k(t)^2 + b_k(t)^2}{2} - Q_k(t)b_k(t) + (Q_k(t) - \tilde{b}_k(t))a_k(t) \\ &\leq \frac{a_k(t)^2 + b_k(t)^2}{2} - \tilde{b}_k(t)a_k(t) + Q_k(t)(a_k(t) - b_k(t)) \end{aligned} \tag{5.28}$$

Using the same argument for $Z_j(t)$ yields:

$$\frac{Z_j(t+1)^2 - Z_j(t)^2}{2} \leq \frac{E_j^2 + e_j(t)^2}{2} - \tilde{e}_j(t)E_j + Z_j(t)(E_j - e_j(t)) \tag{5.29}$$

Using the results of (5.28) and (5.29) in (5.27) we get:

$$\begin{aligned} \Delta(\mathbf{\Theta}(t)) \leq \mathbb{E}\Bigg\{ &\sum_{k=1}^{K} \left[\frac{a_k(t)^2 + b_k(t)^2}{2} - \tilde{b}_k(t)a_k(t) + Q_k(t)(a_k(t) - b_k(t)) \right] + \\ &\sum_{j=1}^{J} \left[\frac{E_j^2 + e_j(t)^2}{2} - \tilde{e}_j(t)E_j + Q_k(t)(E_j - e_j(t)) \right] \mid \mathbf{\Theta}(t) \Bigg\} \end{aligned}$$

Rearranging terms, using the linearity of $\mathbb{E}$ and using the fact that $\mathbb{E}\{Q_k(t) \mid \mathbf{\Theta}(t)\} = Q_k(t)$ and $\mathbb{E}\{Z_j(t) \mid \mathbf{\Theta}(t)\} = Z_j(t)$ yields:

$$\begin{aligned} \Delta(\mathbf{\Theta}(t)) \leq &\frac{1}{2} \sum_{k=1}^{K} \mathbb{E}\left\{ a_k(t)^2 + b_k(t)^2 \mid \mathbf{\Theta}(t) \right\} - \sum_{k=1}^{K} \mathbb{E}\left\{ \tilde{b}_k(t)a_k(t) \mid \mathbf{\Theta}(t) \right\} + \\ &\sum_{k=1}^{K} Q_k(t)\mathbb{E}\left\{ a_k(t) - b_k(t) \mid \mathbf{\Theta}(t) \right\} + \\ &\frac{1}{2} \sum_{j=1}^{J} \mathbb{E}\left\{ E_j^2 + e_j(t)^2 \mid \mathbf{\Theta}(t) \right\} - \sum_{j=1}^{J} \mathbb{E}\left\{ \tilde{e}_k(t)E_j \mid \mathbf{\Theta}(t) \right\} + \\ &\sum_{j=1}^{J} Z_j(t)\mathbb{E}\left\{ E_j - e_j(t) \mid \mathbf{\Theta}(t) \right\} \end{aligned}$$

Finally, we get:

$$\begin{aligned}\Delta(\boldsymbol{\Theta}(t)) \leq & B + \sum_{k=1}^{K} Q_k(t)\mathbb{E}\left\{a_k(t) - b_k(t) \mid \boldsymbol{\Theta}(t)\right\} + \\ & \sum_{j=1}^{J} Z_j(t)\mathbb{E}\left\{E_j - e_j(t) \mid \boldsymbol{\Theta}(t)\right\}\end{aligned} \tag{5.30}$$

where B is a positive constant such that:

$$\begin{aligned}B \geq & \frac{1}{2}\sum_{k=1}^{K}\mathbb{E}\left\{a_k(t)^2 + b_k(t)^2 \mid \boldsymbol{\Theta}(t)\right\} - \sum_{k=1}^{K}\mathbb{E}\left\{\tilde{b}_k(t)a_k(t) \mid \boldsymbol{\Theta}(t)\right\} + \\ & \frac{1}{2}\sum_{j=1}^{J}\mathbb{E}\left\{E_j^2 + e_j(t)^2 \mid \boldsymbol{\Theta}(t)\right\} - \sum_{j=1}^{J}\mathbb{E}\left\{\tilde{e}_k(t)E_j \mid \boldsymbol{\Theta}(t)\right\}\end{aligned} \tag{5.31}$$

It is important to note that for the B constant to exist it is required that the second moments of $a_k(t)$, $b_k(t)$ and $e_j(t)$ to be bounded by a finite constant. This requirement can be interpreted as a constraint on the variance of the arrival and departure rates on the actual and virtual queues.

Adding $V\mathbb{E}\{y(t) \mid \boldsymbol{\Theta}(t)\}$ to both sides of (5.30) gives:

$$\begin{aligned}\Delta(\boldsymbol{\Theta}(t)) + V\mathbb{E}\{y(t) \mid \boldsymbol{\Theta}(t)\} \leq & B + V\mathbb{E}\{y(t) \mid \boldsymbol{\Theta}(t)\} + \\ & \sum_{k=1}^{K} Q_k(t)\mathbb{E}\left\{a_k(t) - b_k(t) \mid \boldsymbol{\Theta}(t)\right\} + \\ & \sum_{j=1}^{J} Z_j(t)\mathbb{E}\left\{E_j - e_j(t) \mid \boldsymbol{\Theta}(t)\right\}\end{aligned} \tag{5.32}$$

The Drift-Plus-Penalty Algorithm for Minimization

Hence, we have shown that there exists an upper bound on the drift expression. The *drift-plus-penalty algorithm* consists on minimizing the upper bound given in (5.32) rather than directly minimizing the drift expression. For implementing the minimization the algorithm is based on the *framework of opportunistically minimizing an expectation* (see Appendix B). Consider a stochastic function $f(\omega(t), \alpha(t))$, such that the objective is to minimize $\mathbb{E}\{f(\omega(t), \alpha(t))\}$, then, a solution to this minimization problem is to, on each time t, observe the value of $\omega(t)$ and choose the action $\alpha(t)$ which minimize $f(\alpha(t), \omega(t))$.

Hence, the *drift-plus-penalty algorithm* can be described as: On each slot t, observe $\mathbf{\Theta}(t)$ and the random event $\omega(t)$, and find the control action $\alpha(t)$ to:

$$\begin{aligned} &\text{minimize} \quad B + Vy(t) + \sum_{k=1}^{K} Q_k(t)(a_k(t) - b_k(t)) + \sum_{j=1}^{J} Z_j(t)(E_j - e_j(t)) \\ &\text{subject to} \quad \alpha(t) \in \mathcal{A}_{\omega(t)}. \end{aligned} \tag{5.33}$$

Note that in the above formulation B and $Z_j(t)E_j$ are non-negative constants, hence they can be removed from the minimization problem. Therefore:

$$\begin{aligned} &\text{minimize} \quad Vy(t) + \sum_{k=1}^{K} Q_k(t)(a_k(t) - b_k(t)) - \sum_{j=1}^{J} Z_j(t)e_j(t) \\ &\text{subject to} \quad \alpha(t) \in \mathcal{A}_{\omega(t)}. \end{aligned} \tag{5.34}$$

The importance of this algorithm lies in that it does not need to know the distribution of the process $\omega(t)$. Further, on each slot t, the process $\omega(t)$, the real queues' backlogs $Q_k(t)$ and the virtual queues' backlogs $Z_j(t)$ are known constants for the minimization problem. The complexity of the problem will depend on the structure of $y(t)$, $a_k(t)$, $b_k(t)$ and $e_j(t)$. Nevertheless, it is important to note that no other knowledge (such as the arrival rate of data) is needed.

To finalize this presentation of the Lyapunov Optimization Theory and the Drift-Plus-Penalty method, we reproduce a very important theorem which shows that the previous algorithm finds an approximate solution which is close to the optimal and which difference is bounded [1]. Also, it shows that the algorithm ensures that all queues are mean rate stable and under some extra assumptions the limiting expected average is bounded. Although we do not provide the proof of this theorem, in Section 5.7 we demonstrate a similar theorem applied to our proposed solution.

Theorem 5.3. *(Performance of Drift-Plus-Penalty Algorithm) Suppose that $\omega(t)$ is i.i.d. over slots with probabilities $\pi(\omega)$, the problem (5.5)-(5.7) is feasible, and that $\mathbb{E}\{L(\mathbf{\Theta}(0))\} < \infty$. If we use the drift-plus-penalty algorithm, then:*

1.

$$\overline{y} = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{y(\tau)\} \leq y^{opt} + \frac{B}{V} \tag{5.35}$$

where y^{opt} is the infimum time average cost achievable by any policy that meets the required constraints, and B is defined in 5.31.

[1] For the sake of simplicity we simplify the theorem presentation. For the complete version see [80].

2. All queues $Q_k(t)$ and $Z_j(t)$ are mean rate stable, and all required constraints (5.6) are satisfied.

5.4 Qos Slicing System Model

As already mentioned, our objective is to implement *QoS Slicing* on WiFi APs by allocating the necessary resources to the different slices. We devise the problem of guaranteeing a minimum bit rate to each client of a slice jointly with providing an upper bound on the queuing delay, as a dynamic resource allocation problem that can be optimized. Therefore, we first define the network and system model to be used and then formulate the optimization problem.

Regarding practical and implementation considerations, we assume that the traffic arrival rate to the different slices and clients is an unknown stochastic process. Moreover, we also assume that the AP is operating in a stochastic environment, which is challenging to model, so we consider it unpredictable and uncontrollable. In particular, given the medium access control of WiFi, we consider the existence of interference and congestion in the wireless medium as well as the possibility of packet collisions. Furthermore, in WiFi, nodes transmit randomly, depending on the channel state (idle or busy), and thus the concept of time slots (needed to consider the previously described Lyapunov Theory) does not explicitly exist. Hence, to be able to implement our scheduling proposal on a WiFi node, we need to simulate a time-slotted behavior. In this scenario, to tackle all the uncertainties mentioned above of the wireless medium as well as to simulate time slots, we adopt the airtime allocation mechanism from Chapter 4.

In this regard, the approach followed is to use the idea of airtime allocation and the *Adaptive Time-Excess Round Robin* (ATERR) algorithm. In ATERR, all the possible variations of the unknown environment are accounted in the airtime, providing an exact measurement of the time consumed in a transmission (see Section 4.2.5). Hence, all the unknown parameters that may affect our system are contemplated in the allocated airtime. For achieving time slots, we propose using the ATERR mechanism but with a fixed quantum size. In other words, 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 the 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 the quantum does not precisely 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 one. 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 length, but leveraging a mechanism that yields, in the long run, the correct time assignment.

Even more, we also adopt the queuing structure proposed in Chapter 4 consisting of a queue per client and per slice. In summary, the ATERR Architecture is mostly reused with an extension of the Scheduler to add QoS guarantees. In this regard, the rest of this chapter concentrates mainly on the design and implementation of this Scheduler.

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

- $\mathcal{S} = \{1, 2, ..., S\}$ is the set of slices instantiated in the AP, such that S is the number of slices defined in the AP. A slice is identified by a value $s \in [1, S]$.
- $\mathcal{C}_s = \{1, 2, ..., C_s\}$ is the set of clients associated to the AP which belong to slice s and C_s is the total number of clients in slice s. We identify a client from slice s as $n_s \in [1, C_s]$.
- K_s is the minimum bit rate requirement, which must be guaranteed to every client of the slice s.
- H_s is the slice s capacity limit, given as a ratio of the total resources.
- A_{n_s} is the arrival rate of the client n_s (client n of slice s). This rate is dynamic and evolves with time but we assume it is invariant within a time slot. We define $A_{n_s}(t)$ as the arrival rate of client n_s in time slot t.
- $\mathbb{C}_{n_s}$ is the channel capacity between the AP and the client n_s. This capacity is variable, and it depends on the channel conditions. We assume the capacity to be invariant within a time slot, hence, we define $\mathbb{C}_{n_s}(t)$ as the channel capacity between the AP and client n_s in time slot t. We also assume this is not the maximal theoretical capacity but a measured capacity which includes all the possible delays introduced by the unknown wireless environment.

5.4.1 Bit Rate Modelling

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

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

$$R_{n_s}(t) = \mathbb{C}_{n_s}(t) \times x_{n_s}(t) \tag{5.36}$$

where $x_{n_s}(t)$ is the proportion (or ratio) of the time slot t assigned for transmitting to client n_s. Note that $\mathbb{C}_{n_s}(t)$ and $R_{n_s}(t)$ are respectively the channel capacity and the bit rate measured in bits per slot (for a generic time slot size). Therefore, $R_{n_s}(t)$ also represents the amount of data transmitted to client n_s in time slot t.

The problem to solve is thus finding the ratios x_{n_s} that satisfy the slice requests. However, we have not yet exactly defined the type of requests that can be made, 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 a very stringent one, 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 a time window.

In this proposal, we consider the second option, where slice requests consist of a minimum average bit rate to be assured to each client of the slice. Therefore, the problem is to guarantee that the average bit rate of each client (R_{n_s}) would be within an interval of the requested average bit rate K_s. In other words, we propose an approach based on three parameters which define the slice request (similar to the approach of Chapter 4): K_s, Δ_s and W_s. A slice tenant requests a minimum bit rate identified by K_s (in this work we assume the bit rate measured in bits per slot, but the SLA can be defined in any metric previously agreed), and is what the tenant expects to be guaranteed by the provider. Also, to provide flexibility to the provider, it is possible to define a *tolerance* Δ_s which measures the possible maximum deviation from the expected minimum bit rate. This parameter is also measured in bits per slot. Lastly, W_s is a time window over which the average bit rate is computed and where the minimum bit rate plus the deviation must be guaranteed. Hence, the SLA requires that, in a time window of size W_s the slice s receives, in average, a bit rate in the interval $(K_s - \Delta_s, K_s + \Delta_s)$.

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

$$\overline{R}_{n_s} = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{R_{n_s}(\tau)\} \tag{5.37}$$

$$= \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mathbb{C}_{n_s}(\tau) x_{n_s}(\tau)\} \tag{5.38}$$

Note that considering the expected average rewards is a classical utility function in infinite-horizon stochastic decision problems [109]. In Section 5.7 we will show that the proposed solution with this bit rate modelling approach provides the requested slice's guarantees under some necessary conditions.

5.4.2 Delay Modelling

As a delay guarantee, we propose to consider an upper bound on the delay of every packet. To achieve this objective, we adapted the approach described in [81], where the solution jointly manages the queue and the scheduler[2]. In summary, our proposed model manages two decision parameters (on each time slot) to guarantee that the delay of each packet is below a given threshold. The decisions to make are two:

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

As already analyzed in Section 5.2 one major factor on the queuing delay is the queue length. Hence, it becomes necessary to model the queue dynamics to consider its length on the problem formulation. As previously described, to guarantee a bound on the queuing delay, the scheduling mechanism we are proposing includes the possibility of dropping packets at the head of the queue. Therefore, the queues' dynamics can be expressed as:

$$Q_{n_s}(t+1) = [Q_{n_s}(t) - R_{n_s}(t) - D_{n_s}(t)]^+ + A_{n_s}(t) \tag{5.39}$$

where $R_{n_s}(t)$, $D_{n_s}(t)$ and $A_{n_s}(t)$ is the amount of data transmitted, dropped and received, respectively, in slot t.

[2] The work in [81] considers the possibility of rejecting incoming packets from the upper layers. In our solution we disregard this possibility as this would move in fact the delay problem to the upper layers.

Persistent-Service Queues

To achieve bounded delay guarantees, we consider *ϵ-persistent service queues* into the problem. These queues are virtual, and they do not represent real network queues, but add new constraints to the problem to assure delay guarantees. These new virtual queues are defined by the following update equation for each client n_s:

$$Z_{n_s}(t+1) = \begin{cases} [Z_{n_s}(t) + \epsilon_{n_s} - R_{n_s}(t) - D_{n_s}(t)]^+ & \text{if } Q_{n_s}(t) > 0 \\ [Z_{n_s}(t) - D_{n_s}(t) - R_{n_s}^{max}]^+ & \text{if } Q_{n_s}(t) = 0 \end{cases} \tag{5.40}$$

where ϵ_{n_s} are pre-defined constants.

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

Then, with this approach, the guaranteed bounded delay problem is transformed into a problem of bounding queues. Therefore, if the QoS Slicing solution we are implementing can maintain these queues bounded it will also bound the delay.

5.4.3 Slice Capacity Limit

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

Let us consider $X_s(t) = \sum_{n_s=1}^{C_s} x_{n_s}$ as the airtime ratio allocated to slice s on time slot t. Then, following the expected average approach used in this model, we define the expected time average allocated airtime ratio to a slice s as:

$$\overline{X}_s = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{X_s(\tau)\} \tag{5.41}$$

5.5 QoS Slicing Problem Formulation

In the formulation that follows, which just focuses on the allocation problem, we assume that 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. Nevertheless, in Section 5.8 we develop a mechanism to deal with unfeasible situations.

Based on the described model, 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:

$$\overline{R}_{n_s} \geq K_s, \ \forall s \in [1, S], \ \forall n_s \in [1, C_s]. \tag{5.42}$$

$$\overline{X}_s \leq H_s, \ \forall s \in [1, S]. \tag{5.43}$$

$$\lim_{t \to \infty} \frac{\mathbb{E}\{Q_{n_s}(t)\}}{t} = 0, \ \forall s \in [1, S], \ \forall n_s \in [1, C_s]. \tag{5.44}$$

$$\lim_{t \to \infty} \frac{\mathbb{E}\{Z_{n_s}(t)\}}{t} = 0, \ \forall s \in [1, S], \ \forall n_s \in [1, C_s]. \tag{5.45}$$

The first constraint guarantees that a minimum average bit rate is provided to each client of each slice. The second constraint provides a limit on the average airtime ratio allocated to each slice. The last two constraints indicates that the stochastic processes $Q_{n_s}(t)$ and $Z_{n_s}(t)$ must be *mean rate stable* (as defined in Section 5.3).

With the above conditions we can formulate an optimization problem with the objective of finding the resource allocation and dropping decisions for maximizing the average expected total throughput of the AP. Considering a system where the queues are stable, the transmission bit rate (measured in bits per slot) in a time slot t can be expressed as the amount of arrived data minus the amount of data dropped in slot t. We thus define our throughput function as:

$$u_{n_s}(t) = A_{n_s}(t) - D_{n_s}(t), \tag{5.46}$$

and its expected time average as:

$$\overline{u}_{n_s} = \lim_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{u_{n_s}(\tau)\}. \tag{5.47}$$

However, to consider fairness in the dropping decisions, we define the following fair utility function, which approximates proportional fairness (as suggested in [38]):

$$\phi(\boldsymbol{u}(t)) = \sum_{s=1}^{S} \sum_{n_s=1}^{C_s} \log(1 + \omega u_{n_s}(t)). \tag{5.48}$$

for some constant $\omega > 0$ and where $\boldsymbol{u}(t) = [u_{1_1}, ..., u_{n_S}]$ is the vector of throughputs. Note, that with this utility function we are indeed minimizing the packet drops in relation to the arrival rates.

Then, we can formulate a *stochastic optimization problem* that maximizes the average expected total throughput subject to the previous constraints:

$$\underset{\boldsymbol{x},\boldsymbol{D}}{\text{maximize}} \quad \phi(\overline{\boldsymbol{u}}) \tag{5.49}$$

$$\text{subject to} \quad \overline{R}_{n_s} \geq K_s, \ \forall s \in [1, S], \ \forall n_s \in [1, C_s], \tag{5.50}$$

$$\overline{X}_s \leq H_s, \ \forall s \in [1, S], \tag{5.51}$$

$$\lim_{t \to \infty} \frac{\mathbb{E}\{Q_{n_s}(t)\}}{t} = 0, \ \forall s \in [1, S], \ \forall n_s \in [1, C_s], \tag{5.52}$$

$$\lim_{t \to \infty} \frac{\mathbb{E}\{Z_{n_s}(t)\}}{t} = 0, \ \forall s \in [1, S], \ \forall n_s \in [1, C_s], \tag{5.53}$$

$$0 \leq u_{n_s}(t) \leq A_{n_s}^{max}, \ \forall s \in [1, S], \ \forall n_s \in [1, C_s], \ \forall t, \tag{5.54}$$

$$0 \leq x_{n_s}(t) \leq 1, \ \forall s \in [1, S], \ \forall n_s \in [1, C_s], \ \forall t, \tag{5.55}$$

$$\sum_{s=1}^{S} \sum_{n_s=1}^{C_s} x_{n_s}(t) \leq 1, \tag{5.56}$$

$$0 \leq D_{n_s}(t) \leq D_{n_s}^{max}, \ \forall s \in [1, S], \ \forall n_s \in [1, C_s], \ \forall t. \tag{5.57}$$

where

$$\phi(\overline{\boldsymbol{u}}) = \sum_{s=1}^{S} \sum_{n_s=1}^{C_s} \log(1 + \omega \overline{u}_{n_s}). \tag{5.58}$$

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

Constraint (5.50) considers the minimum average expected bit rate K_s of each slice. It represents in fact a set of constraints, since there is a bit rate requirement for each client of each slice. Constraint (5.51) limits the airtime ratio assigned to each slice to its capacity limit H_s. Constraints (5.52) and (5.53) are the stability conditions for the packet and the ϵ-persistent service queues, respectively. Constraint (5.54) guarantees that the objective function is non-negative, by not dropping more packets than those that had arrived. Both (5.55) and (5.56) ensure that assigned resources do not surpass the available ones, by limiting the possible values of the $\boldsymbol{x}$ variables. Finally, constraint (5.57) limits the amount of data to drop at each time slot.

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

As was discussed in Section 5.3, the main complexity of the optimization problem (5.49)-(5.57) resides in the variability of the channel capacity $\mathbb{C}(t)$ and the arrival rate $A_{n_s}(t)$. As already mentioned, this functions can be considered stochastic processes, for which we do not know its distribution and so cannot be predicted.

5.6 Proposed Solution for QoS Slicing

Our proposal consists of solving the previous stochastic problem by applying the *drift-plus-penalty* method described in Section 5.3. This method allows us to build a new deterministic problem, which provides an approximate solution to the original one. Even more, as previously explained, such solution can be made arbitrarily close to the optimal one, but with a trade-off on how the constraints are fulfilled.

5.6.1 Problem Transformation

Since problem (5.49)-(5.57) consists in the optimization of a concave non-linear function of time averages, we need to transform it before applying the drift-plus-penalty method. Note that this problem differs from the problem formulation described in Section 5.3 as it involves the optimization of a *function* of time averages. As the function $\phi(u(t))$ proposed is not linear, in general, it is not the same to maximize a time average of a function than to maximize a function of time averages. Therefore, we follow the auxiliary variable technique described in [80], to adapt the formulated problem into a traditional time average optimization problem.

With this technique, the stochastic network optimization problem (5.49)-(5.57) can be transformed using a vector of auxiliary variables $\boldsymbol{\gamma}(t) = (\gamma_1(t), ..., \gamma_{n_S}(t))$, which are chosen at every slot, according to the constraints $0 \leq \gamma_{n_s}(t) \leq A_{n_s}^{max}$, where $A_{n_s}^{max}$ is the maximum possible arrival rate at each client and slice. The modified problem is therefore:

$$\begin{aligned}
&\underset{\boldsymbol{x},\boldsymbol{D}}{\text{maximize}} && \overline{\phi(\boldsymbol{\gamma})} && (5.59)\\
&\text{subject to} && \overline{R}_{n_s} \geq K_s,\ \forall s \in [1,S],\ \forall n_s \in [1,C_s], && (5.60)\\
& && \overline{X}_s \leq H_s,\ \forall s \in [1,S], && (5.61)\\
& && \lim_{t\to\infty} \frac{\mathbb{E}\{Q_{n_s}(t)\}}{t} = 0,\ \forall s \in [1,S],\ \forall n_s \in [1,C_s], && (5.62)\\
& && \lim_{t\to\infty} \frac{\mathbb{E}\{Z_{n_s}(t)\}}{t} = 0,\ \forall s \in [1,S],\ \forall n_s \in [1,C_s], && (5.63)\\
& && \overline{\gamma}_{n_s} \leq \overline{u}_{n_s},\ \forall s \in [1,S],\ \forall n_s \in [1,C_s], && (5.64)\\
& && 0 \leq \gamma_{n_s}(t) \leq A_{n_s}^{max},\ \forall s \in [1,S],\ \forall n_s \in [1,C_s],\ \forall t, && (5.65)\\
& && 0 \leq x_{n_s}(t) \leq 1,\ \forall s \in [1,S],\ \forall n_s \in [1,C_s],\ \forall t, && (5.66)\\
& && \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} x_{n_s}(t) \leq 1, && (5.67)\\
& && 0 \leq D_{n_s}(t) \leq D_{n_s}^{max},\ \forall s \in [1,S],\ \forall n_s \in [1,C_s],\ \forall t. && (5.68)
\end{aligned}$$

where

$$\overline{\phi(\boldsymbol{\gamma})} = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\phi(\boldsymbol{\gamma}(\tau))\}. \qquad (5.69)$$

Intuitively, we can explain the previous transformation as follows. First, it is worth noting that the original constraints are a subset of the new ones, so any solution to the transformed problem will also satisfy the original constraints. Suppose we have decisions $\boldsymbol{x}^*(t)$ and $\boldsymbol{D}^*(t)$, which are a solution to the original problem. Let $\overline{\boldsymbol{u}}^*$ be the expected utilities obtained by the clients under those decisions, which yield a maximum utility value $\phi(\overline{\boldsymbol{u}}^*) = \phi^{opt}$. Then, we can build a solution to the transformed problem with the same $\boldsymbol{x}^*(t)$ decisions, selecting $\boldsymbol{\gamma}(t) = \overline{\boldsymbol{u}}^*$ for all t. Note that this solution satisfies constraint (5.64), as we enforce equality, and new constraint (5.65) equals (5.54). As $\boldsymbol{\gamma}(t) = \overline{\boldsymbol{u}}^*$ is enforced for all t, we have that $\overline{\phi(\boldsymbol{\gamma})} = \phi(\overline{\boldsymbol{u}}^*) = \phi^{opt}$. Hence, we have a solution to the transformed problem with optimal value ϕ^{opt}, which is also a solution to the original problem. Therefore, a solution to the transformed problem ensures that the constraints of the original problem are satisfied, and it obtains an utility that approximates the original problem utility, as constraint (5.64) is forced to equality. A discussion on the distance between the solution of the original and transformed problem is provided in Section 5.7.

5.6.2 Application of the Drift-Plus-Penalty Approach

To solve the problem (5.59)-(5.68) using the drift-plus-penalty method, we first transform the constraints into queue stability problems. For constraints (5.60), (5.61) and (5.64) we define *virtual queues*, with update equations:

$$G_{n_s}(t+1) = [G_{n_s}(t) - R_{n_s}(t) + K_s]^+ \ \ \forall\, s \in [1, S],\ \forall\, n_s \in [1, C_s] \tag{5.70}$$

$$U_s(t+1) = [U_s(t) + X_s(t) - H_s]^+ \ \ \forall\, s \in [1, S] \tag{5.71}$$

$$Y_{n_s}(t+1) = [Y_{n_s}(t) + \gamma_{n_s}(t) - u_{n_s}(t)]^+ \ \ \forall\, s \in [1, S],\ \forall\, n_s \in [1, C_s] \tag{5.72}$$

As already mentioned, these queues are virtual and so do not represent real network queues. Intuitively, they can be seen as queues that accumulate the difference between the requested bound of the constraint and the actual value achieved. As we have seen in Section 5.3, by ensuring mean rate stability on these three queues and in the already described $Q_{n_s}(t)$ and $Z_{n_s}(t)$ queues, we guarantee that the constraints (5.60)-(5.64) are met.

Remember that the drift-plus-penalty strategy consists in minimizing the queue backlogs, as well as in minimizing an utility function called *penalty*. As in our case we have a reward maximization rather than a penalty minimization, we consider the opposite of our utility function $\phi(\boldsymbol{\gamma}(t))$ as a penalty. Therefore, the strategy consists on minimizing on each time slot t the following expression:

$$\Delta(\boldsymbol{\Theta}(t)) - V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} \tag{5.73}$$

where $\boldsymbol{\Theta}(t) = [\boldsymbol{Q}(t), \boldsymbol{Z}(t), \boldsymbol{G}(t), \boldsymbol{U}(t), \boldsymbol{Y}(t)]$ is the concatenated vector of queue backlogs and $\Delta(\boldsymbol{\Theta}(t))$ is the one-slot conditional Lyapunov drift defined as $\Delta(\boldsymbol{\Theta}(t)) = \mathbb{E}\{L(\boldsymbol{\Theta}(t+1)) - L(\boldsymbol{\Theta}(t))|\boldsymbol{\Theta}(t)\}$ and V is a non-negative constant that will affect the trade-off between the drift and the penalty. Moreover, given that the proposed problem formulation consists of five different queues, we have the following Lyapunov function, as a measure of the length of all the queues:

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

$$\frac{1}{2}\sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Y_{n_s}(t) + \frac{1}{2}\sum_{s=1}^{S} U_s(t) \quad (5.74)$$

Hence, following the same reasoning as in Section 5.3 the previous drift-plus-penalty expression can be upper bounded as:

$$\begin{aligned}\Delta(\boldsymbol{\Theta}(t)) - V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} \leq B - V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} \\ + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} G_{n_s}(t)\mathbb{E}\left\{-R_{n_s}(t) + K_s \mid \boldsymbol{\Theta}(t)\right\} \\ + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Q_{n_s}(t)\mathbb{E}\left\{A_{n_s}(t) - R_{n_s}(t) - D_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} \\ + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Z_{n_s}(t)\mathbb{E}\left\{\epsilon_{n_s} - R_{n_s}(t) - D_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} \\ + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Y_{n_s}(t)\mathbb{E}\left\{\gamma_{n_s}(t) - A_{n_s}(t) + D_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} \\ + \sum_{s=1}^{S} U_s(t)\mathbb{E}\left\{X_s(t) - H_s \mid \boldsymbol{\Theta}(t)\right\} \quad (5.75)\end{aligned}$$

Rearranging terms we have:

$$\begin{aligned}\Delta(\boldsymbol{\Theta}(t)) - V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} \leq B - V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} \\ + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} [G_{n_s}(t) + Q_{n_s}(t) + Z_{n_s}(t)]\mathbb{E}\left\{-R_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} \\ + \sum_{s=1}^{S} U_s(t)\mathbb{E}\left\{X_s(t) \mid \boldsymbol{\Theta}(t)\right\} + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} [Q_{n_s}(t) - Y_{n_s}(t)]\mathbb{E}\left\{A_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} \\ + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} [Y_{n_s}(t) - Q_{n_s}(t) - Z_{n_s}(t)\mathbb{E}\left\{D_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} \\ + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Y_{n_s}(t)\mathbb{E}\left\{\gamma_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} G_{n_s}(t)\mathbb{E}\left\{K_s \mid \boldsymbol{\Theta}(t)\right\} \\ + \sum_{s=1}^{S} U_s(t)\mathbb{E}\left\{-H_s \mid \boldsymbol{\Theta}(t)\right\} + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Z_{n_s}(t)\mathbb{E}\left\{\epsilon_{n_s} \mid \boldsymbol{\Theta}(t)\right\} \quad (5.76)\end{aligned}$$

As explained in Section 5.3 we can find an approximate solution to the problem by minimizing, at each slot t, the right-hand side of (5.76). Therefore, our new problem is to minimize this expression as a function of decision variables $x_{n_s}(t)$, $D_{n_s}(t)$ and $\gamma_{n_s}(t)$. Removing the constant values (the parameters which do not depend on the decision variables) the expression to minimize reduces to:

$$\begin{aligned}
&- V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Y_{n_s}(t)\mathbb{E}\left\{\gamma_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} \\
&\quad - \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} [G_{n_s}(t) + Q_{n_s}(t) + Z_{n_s}(t)]\mathbb{E}\left\{R_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} + \sum_{s=1}^{S} U_s(t)\mathbb{E}\left\{X_s(t) \mid \boldsymbol{\Theta}(t)\right\} \\
&\qquad + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} [Y_{n_s}(t) - Q_{n_s}(t) - Z_{n_s}(t)\mathbb{E}\left\{D_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} \qquad (5.77)
\end{aligned}$$

Even more, following the approach of opportunistically minimizing an expectation described in Appendix B, the problem transforms into a minimization, on each time slot t, of the following expression:

$$\begin{aligned}
&- V\phi(\boldsymbol{\gamma}(t)) + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Y_{n_s}(t)\gamma_{n_s}(t) \\
&\quad - \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} [G_{n_s}(t) + Q_{n_s}(t) + Z_{n_s}(t)]R_{n_s}(t) + \sum_{s=1}^{S} U_s(t)X_s(t) \\
&\qquad + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} [Y_{n_s}(t) - Q_{n_s}(t) - Z_{n_s}(t)]D_{n_s}(t) \qquad (5.78)
\end{aligned}$$

Using the fact that $R_{n_s}(t) = \mathbb{C}_{n_s}(t)x_{n_s}(t)$ and that $X_s(t) = \sum_{n_s=1}^{C_s} x_{n_s}(t)$ (to show the different decision variables) yields:

$$\begin{aligned}
&- V\phi(\boldsymbol{\gamma}(t)) + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Y_{n_s}(t)\gamma_{n_s}(t) \\
&\quad - \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} [G_{n_s}(t) + Q_{n_s}(t) + Z_{n_s}(t)]\mathbb{C}_{n_s}(t)x_{n_s}(t) + \sum_{s=1}^{S} U_s(t)\sum_{n_s=1}^{C_s} x_{n_s} \\
&\qquad + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} [Y_{n_s}(t) - Q_{n_s}(t) - Z_{n_s}(t)]D_{n_s}(t) \qquad (5.79)
\end{aligned}$$

Then, to minimize the expression in (5.79), we separate the minimization problem into three different goals:

1. a minimization of the γ_{n_s} terms,
2. a minimization of the x_{n_s} terms,
3. a minimization of $D_{n_s}(t)$ terms.

Optimization of the Auxiliary Variable

First, we find the optimal auxiliary variables, by observing the values of $Y_{n_s}(t)$:

$$\begin{aligned} \underset{\boldsymbol{\gamma}(t)}{\text{minimize}} \quad & -V\phi(\gamma_{n_s}(t)) + \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Y_{n_s}(t)\gamma_{n_s}(t) \\ \text{subject to} \quad & 0 \leq \gamma_{n_s}(t) \leq A_{n_s}^{max} \;\; \forall\, s \in [1,S], \;\; \forall\, n_s \in [1,C_s]. \end{aligned} \tag{5.80}$$

We can find a closed-form solution for problem (5.80). For this, we transform the single minimization problem into a multiple problem by minimizing each term of the sum (remember that $\phi(\gamma_{n_s}(t)) = \sum_{s=1}^{S}\sum_{n_s=1}^{C_s}\log(1+\omega\gamma_{n_s}(t)))$. Therefore, for each slice $s \in [1,S]$ and for each client $n_s \in [1,C_s]$ the problem is:

$$\begin{aligned} \underset{\boldsymbol{\gamma}(t)}{\text{minimize}} \quad & -V\log(1+\omega\gamma_{n_s}(t)) + Y_{n_s}(t)\gamma_{n_s}(t) \\ \text{subject to} \quad & 0 \leq \gamma_{n_s}(t) \leq A_{n_s}^{max}. \end{aligned} \tag{5.81}$$

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

$$\gamma_{n_s}(t) = \frac{V}{Y_{n_s}(t)} - \frac{1}{\omega}. \tag{5.82}$$

Optimization of the Airtime Allocation Variables

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

$$\begin{aligned} \underset{\boldsymbol{x}(t)}{\text{minimize}} \quad & -\sum_{s=1}^{S}\sum_{n_s=1}^{C_s}[G_{n_s}(t)+Q_{n_s}(t)+Z_{n_s}(t)]\mathbb{C}_{n_s}(t)x_{n_s}(t) + \sum_{s=1}^{S}U_s(t)\sum_{n_s=1}^{C_s}x_{n_s} \\ \text{subject to} \quad & \sum_{s=1}^{S}\sum_{n_s=1}^{C_s}x_{n_s}(t) \leq 1, \\ & 0 \leq x_{n_s}(t) \leq 1 \;\forall s \in [1,S], \;\forall n_s \in [1,C_s]. \end{aligned} \tag{5.83}$$

Removing constants and changing the sign, the previous formulation can be transformed into:

$$\begin{aligned}
\underset{\boldsymbol{x}(t)}{\text{maximize}} \quad & \sum_{s=1}^{S}\sum_{n_s=1}^{C_s}[\mathbb{C}_{n_s}(t)(G_{n_s}(t)+Q_{n_s}(t)+Z_{n_s}(t))-U_s(t)]x_{n_s}(t) \\
\text{subject to} \quad & \sum_{s=1}^{S}\sum_{n_s=1}^{C_s} x_{n_s}(t) \leq 1, \\
& 0 \leq x_{n_s}(t) \leq 1 \; \forall s \in [1,S], \; \forall n_s \in [1,C_s].
\end{aligned} \tag{5.84}$$

Optimization of the Dropping Decision Variables

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

$$\begin{aligned}
\underset{\boldsymbol{x}(t)}{\text{minimize}} \quad & \sum_{s=1}^{S}\sum_{n_s=1}^{C_s}[Y_{n_s}(t)-(Q_{n_s}(t)+Z_{n_s}(t))]D_{n_s}(t) \\
\text{subject to} \quad & 0 \leq D_{n_s}(t) \leq D_{n_s}^{max} \; \forall s \in [1,S], \; \forall n_s \in [1,C_s].
\end{aligned} \tag{5.85}$$

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

$$D_{n_s}(t) = \begin{cases} D_{n_s}^{max} & \text{if } Q_{n_s}(t)+Z_{n_s}(t) > Y_{n_s}(t) \\ 0 & \text{otherwise} \end{cases} \tag{5.86}$$

Therefore, we get a deterministic optimization problem that, at every slot t, calculates the control actions $x_{n_s}(t)$ and $D_{n_s}(t)$ by observing the backlogs of the real queues $Q_{n_s}(t)$, the virtual queues $Z_{n_s}(t)$, $G_{n_s}(t)$ and $U_s(t)$ and the random channel capacities of each client $\mathbb{C}_{n_s}(t)$. Note that the channel capacities and the queue backlogs on time slot t act as constants in the optimization problem. After each iteration, the real and virtual queues are updated, as defined in (5.39), (5.40), (5.70), (5.71) and (5.72). In Section 5.7, we provide a proof which shows that this solution satisfies all constraints in (5.59)-(5.68), and that the obtained utility differs from the target utility by no more than B/V, which can be made arbitrarily small as V is increased. However, the time average queue backlogs bound and the delay bound increases linearly with V. In the case of our QoS Slicing problem, this trade-off translates into a compromise between the optimal utility achieved and the satisfaction of the bit rate and delay guarantees.

5.6.3 Proposed Scheduling Algorithm

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

Resolving the previous airtime allocation optimization problem at each time slot can be complex given that the size of the time slot is in general small and hence, the resolution must be fast. However, given that we are using the airtime-quantum based approach and we can define the value of the quantum, we can consider that at each slot the AP is allowed to transmit to just one client. This makes the scheduling easier, as it does not need to calculate the ratios for all clients at each slot, but only to decide to which client to transmit. Hence, the new optimization problem to find the airtime allocation variables is:

$$\begin{aligned} \underset{\boldsymbol{x}(t)}{\text{maximize}} \quad & \sum_{s=1}^{S} \sum_{n_s=1}^{C_s} [\mathbb{C}_{n_s}(t)(G_{n_s}(t) + Q_{n_s}(t) + Z_{n_s}(t)) - U_s(t)] x_{n_s}(t) \\ \text{subject to} \quad & \sum_{s=1}^{S} \sum_{n_s=1}^{C_s} x_{n_s}(t) \leq 1, \\ & x_{n_s}(t) \in 0, 1. \end{aligned} \tag{5.87}$$

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 $\mathbb{C}_{n_s}(t)(G_{n_s}(t) + Q_{n_s}(t) + Z_{n_s}(t)) - U_s$, where all items weight 1 and the maximum capacity is also 1. It is straightforward to note that the solution of this problem is the item with the highest reward.

Then, from the previous analysis, we design the scheduling algorithm shown in Algorithm 3. The scheduler only considers non-empty queues at each slot so as to ensure that resources are assigned beyond the capacity limit H_s only when free resources are available. The rationale is that, if a client has an empty queue, it implies that all offered traffic has been already served, and so, if there were other queues with traffic, they should be served although having a lower utility. Also, in the algorithm shown, we adapt the quantum-based airtime allocation from ATERR so as to be used for this specific case.

As can be seen in Algorithm 3, each iteration of the algorithm corresponds to a time slot where the traffic queue of the client n_s with the highest value of $\mathbb{C}_{n_s}(t)(G_{n_s}(t) + Q_{n_s}(t) + Z_{n_s}(t)) - U_s$ is assigned for transmission. Then, packets are dequeued and transmitted until the quantum is totally consumed. After the transmission has ended, each client's queue is checked to decide if any packet drops are necessary. Finally, all virtual queues are updated accordingly.

Algorithm 3: QoS Slicing Scheduler Pseudocode.

```
function Scheduler() is
    input  : $V$, $Q_{n_s}$, $H_s$, $K_s$,$D_{n_s}^{max}$, $\gamma_{n_s}^{max}$, $\epsilon_{n_s}$, $\omega$  $\forall s \in [1, S]$, $\forall n_s \in [1, C_s]$
    output : Scheduling and dropping decisions on each slot $t$
    /* Initialize queues */
    foreach $n_s \mid s \in [1, S], n_s \in [1, C_s]$ do
        $Z_{n_s} \leftarrow 0; G_{n_s} \leftarrow 0; Y_{n_s} \leftarrow 0; U_s \leftarrow 0;$
    end
    while true do
        /* Observe the current channel capacity and arrivals */
        foreach $n_s \mid s \in [1, S], n_s \in [1, C_s]$ do
            $\mathbb{C}_{n_s} \leftarrow getCapacity(n_s);$
            $A_{n_s} \leftarrow getArrivals(n_s);$
        end
        /* Compute the vector of benefits $\boldsymbol{B}$ using only clients with queued
           packets */
        $\boldsymbol{B} = \mathbb{C} * (\boldsymbol{G} + \boldsymbol{Q} + \boldsymbol{Z}) - \boldsymbol{U};$
        /* Find the client $i$ with the maximum benefit */
        $i \leftarrow arg\,max\,\boldsymbol{B};$
        /* Get the queue of client $i$ */
        $queue \leftarrow GetQueue(queueList, i);$
        /* Transmit packets for a quantum period */
        while $queue.excess < 0$ do
            $airtime \leftarrow transmitPacket(queue);$
            $queue.excess \leftarrow queue.excess + airtime$
        end
        $queue.excess \leftarrow queue.excess - QUANTUM;$
        foreach $n_s \mid s \in [1, S], n_s \in [1, C_s]$ do
            if $Q_{n_s} + Z_{n_s} > Y_{n_s}$ then
                $dropPackets(n_s, D_{n_s}^{max});$
            end
            /* Calculate the auxiliary variable values */
            $\gamma_{n_s} \leftarrow \min\{\frac{V}{Y_{n_s}} - \frac{1}{\omega}, \gamma_{n_s}^{max}\};$
            /* Update queue backlogs */
            $Z_{n_s} \leftarrow [Z_{n_s} - \epsilon_{n_s} - R_{n_s} - D_{n_s}]^+;$
            $G_{n_s} \leftarrow [G_{n_s} - R_{n_s} + K_s]^+;$
            $U_s \leftarrow [U_s + X_s - H_s]^+;$
            $Y_{n_s} \leftarrow [Y_{n_s} + \gamma_{n_s} - u_{n_s}]^+;$
        end
    end
end
```

5.7 Analysis of the Solution

The objective of this section is to demonstrate that the proposed mechanism of minimizing the right-hand side of Equation (5.75) finds an approximate solution to the problem in (5.49)-(5.57) which utility is close to the optimal utility, all the queues are stabilized and all the constraints and delay bounds are satisfied. The following proofs are based on the proofs provided by M. Neely in [80], however, we present them here with the purpose of better understanding the solution performance and because our approach differs in some particularities with the original work of Neely.

For the following demonstrations some assumptions we already mentioned in previous sections are needed as well as some bounds on the system parameters. Let us first review some of the bounds we have previously defined and are satisfied by our solution:

$$\begin{aligned} A_{n_s}(t) &\leq A_{n_s}^{max} \quad \forall s \in [1, S],\ \forall n_s \in [1, C_s],\ \forall t \\ R_{n_s}(t) &\leq R_{n_s}^{max} \quad \forall s \in [1, S],\ \forall n_s \in [1, C_s],\ \forall t \\ D_{n_s}(t) &\leq D_{n_s}^{max} = A_{n_s}^{max} \quad \forall s \in [1, S],\ \forall n_s \in [1, C_s],\ \forall t \\ X_s(t) &\leq 1 \quad \forall s \in [1, S],\ \forall n_s \in [1, C_s],\ \forall t \\ \gamma_{n_s}(t) &\leq \gamma_{n_s}^{max} = A_{n_s}^{max} \quad \forall s \in [1, S],\ \forall n_s \in [1, C_s],\ \forall t \\ \phi(\boldsymbol{\gamma}(t)) &\leq \phi^{max} = \log(\gamma_{1_1}^{max}, ..., \gamma_{n_s}^{max}) \quad \forall t \end{aligned} \tag{5.88}$$

Let us also summarize the necessary assumptions:

Assumption 5.1. *Let us consider $\omega(t) = [\boldsymbol{A}(t), \mathbb{C}(t)]$ as the network state consisting of the arrivals $A_{n_s}(t)$ and the channel capacities $\mathbb{C}_{n_s}(t)$ at each client. We assume $\omega(t)$ is an stochastic process which is i.i.d. over slots and is stationary with distribution probability $\pi(\omega)$.*

Assumption 5.2. *The problems defined in (5.49)-(5.57) and in (5.61)-(5.68) are feasible. In other words, for any vector of possible arrival rates there exists an allocation of transmission airtime ratios which solves the problem.*

Assumption 5.3. *All the real and virtual queues are initially empty.*

Assumptions 5.1 and 5.3 are needed for a clear theoretical demonstration. However, in [80] it is shown that the drift-plus-penalty method also works in scenarios were the channel is modeled as a Markovian process. Even more, Assumption 5.3 is used in this thesis to simplify the demonstration, however, similar results are obtained without the Assumption. Finally, in Section 5.8 we have proposed a mechanism to deal with cases when the problem is not feasible and transform it into a feasible problem.

5.7.1 Delay Bounds

At first, we will demonstrate that the proposed algorithm guarantees that there exists a bound on both $Q_{n_s}(t)$ and $Z_{n_s}(t)$ queues. Therefore, we propose the following theorem.

Theorem 5.4. *Assuming the arrivals and drops are bounded by $A_{n_s}^{max}$ and that $\epsilon_{n_s} \leq A_{n_s}^{max}$, then the proposed algorithm ensures that:*

$$Q_{n_s}(t) \leq Q_{n_s}^{max},\ Z_{n_s}(t) \leq Z_{n_s}^{max},\ Y_{n_s}(t) \leq Y_{n_s}^{max}\ \forall t \tag{5.89}$$

where:

$$\begin{aligned} Q_{n_s}^{max} &= V\omega + 2A_{n_s}^{max} \\ Z_{n_s}^{max} &= V\omega + A_{n_s}^{max} + \epsilon_{n_s} \\ Y_{n_s}^{max} &= V\omega + A_{n_s}^{max} \end{aligned} \tag{5.90}$$

Proof. Let us first observe that the virtual queue $Y_{n_s}(t)$ is bounded. From the result (5.82) which defines the optimal auxiliary variables and the bound $\gamma_{n_s}(t) \geq 0$ it is easy to ascertain that when $Y_{n_s}(t) \geq V\omega$, the algorithm will choose $\gamma_{n_s}(t) = 0$. Then, the virtual queue defined in (5.72) will not grow in time slot $t + 1$. On the other hand, when $Y_{n_s}(t) < V\omega$ the virtual queue will follow the dynamics defined in (5.72) and given that $\gamma_{n_s}(t) \leq A_{n_s}^{max}$, it can increase at most by $A_{n_s}^{max}$. Therefore, we have an upper bound of the virtual queue, given by $Y_{n_s}^{max} = V\omega + A_{n_s}^{max}$.

Next, we demonstrate that there exist constants $Q_{n_s}^{max}$ and $Z_{n_s}^{max}$ which are respectively upper bounds for $Q_{n_s}(t)$ (as defined in (5.39)) and $Z_{n_s}(t)$ (as defined in (5.40)). As mentioned, for this proof we assume that $0 \leq \epsilon_{n_s} \leq D_{n_s}^{max}$ and $D_{n_s}^{max} = A_{n_s}^{max}$ for all clients and slices.

It is easy to see from the dropping decision (5.86) that when $Q_{n_s}(t) > Y_{n_s}^{max}$, the mechanism will drop $A_{n_s}^{max}$ packets from the queue. Therefore, as the maximum arrival rate in one slot is given by $A_{n_s}^{max}$, the queue will not grow at the next slot. On the other case, when $Q_{n_s}(t) \leq Y_{n_s}^{max}$, the queue will grow at maximum by $A_{n_s}^{max}$, such that $Q_{n_s}(t+1) \leq Y_{n_s}^{max} + A_{n_s}^{max}\ \forall t$. Hence, we can conclude that $Q_{n_s}(t) \leq Y_{n_s}^{max} + A_{n_s}^{max}\ \forall t$.

For $Z_{n_s}(t)$ the proof is very similar. If $Z_{n_s}(t) > Y_{n_s}^{max}$, then the decision is to drop $D_{n_s}^{max}$ packets from the queue. As $0 \leq \epsilon_{n_s} \leq D_{n_s}^{max}$, the queue cannot grow at the next slot. If $Z_{n_s}(t) \leq Y_{n_s}^{max}$, then $Z_{n_s}(t+1) \leq Y_{n_s}^{max} + \epsilon_{n_s}$.

□

As we know from Section 5.4.2, the delay bound is given by $W_{n_s}^{max} = \left\lceil \frac{Q_{n_s}^{max} + Z_{n_s}^{max}}{\epsilon_{n_s}} \right\rceil$ then, substituting we have:

$$W_{n_s}^{max} = \left\lceil \frac{2V\omega + 3A_{n_s}^{max} + \epsilon_{n_s}}{\epsilon_{n_s}} \right\rceil \tag{5.91}$$

Therefore, we have obtained an upper bound on the queuing delay of each client which depends on a set of configuration parameters of the solution (V, ω and ϵ_{n_s}) and on an upper bound on the traffic arrival rate. This result is important as it can be used to adjust the scheduler parameters to a given delay bound requirement.

5.7.2 Optimal Utility and Constraint Satisfaction

Hereafter, we demonstrate that the proposed algorithm achieves an utility which approximates the optimal utility by a factor depending on V. We also show that all the constraints of the initial problem (5.49)-(5.57) are satisfied and that the queue backlogs are bounded.

Theorem 5.5. *Supposing that Assumptions (5.1)-(5.3) hold. If the proposed algorithm is implemented with a constant $V > 0$, then all queues are mean rate stable and the obtained utility satisfies:*

$$\phi(\overline{\mathbf{u}}) \geq \phi^{opt} - \frac{B}{V} \tag{5.92}$$

where ϕ^{opt} is the maximum utility of the problem (5.49)-(5.57).

Before presenting the proof of this theorem, we will introduce the definition of a special kind of control policies. Let us consider a particular type of control policy that solves the problem in (5.49)-(5.57) and which does not consider the queue backlog on the decisions. These type of policies are called ω-only policies, as they observe the network state $\omega(t)$ at each slot t and independently choose a control action $x(t)$ as a function only of the observed $\omega(t)$. The following demonstration is based on an important result (which demonstration exceeds the scope of this document) from [80] which shows that the optimal utility and the constraints satisfaction of our stochastic optimization problem can be achieved by ω-only policies.

Proof. Because the proposed drift-plus-penalty algorithm minimizes the bound on the right-hand side of equation (5.75) on every slot, we have that it is less or equal than the bound obtained by any other policy and any vector $\boldsymbol{\gamma}$. In particular we have that:

$$\Delta(\boldsymbol{\Theta}(t)) - V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} \leq B - V\phi(\boldsymbol{\gamma}^{*})$$

$$
\begin{aligned}
&+\sum_{s=1}^{S}\sum_{n_s=1}^{C_s} G_{n_s}(t)\mathbb{E}\left\{-R^*_{n_s}(t)+K_s \mid \boldsymbol{\Theta}(t)\right\} \\
&+\sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Q_{n_s}(t)\mathbb{E}\left\{A_{n_s}(t)-R^*_{n_s}(t)-D^*_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} \\
&+\sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Z_{n_s}(t)\mathbb{E}\left\{\epsilon_{n_s}-R^*_{n_s}(t)-D^*_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} \\
&+\sum_{s=1}^{S}\sum_{n_s=1}^{C_s} Y_{n_s}(t)\mathbb{E}\left\{\gamma^*_{n_s}(t)-A_{n_s}(t)+D^*_{n_s}(t) \mid \boldsymbol{\Theta}(t)\right\} \\
&+\sum_{s=1}^{S} U_s(t)\mathbb{E}\left\{X^*_s(t)-H_s \mid \boldsymbol{\Theta}(t)\right\}
\end{aligned}
\tag{5.93}
$$

where $\boldsymbol{\gamma}^*$ is any feasible vector, where $x^*(t)$, $D^*(t)$ are the decisions of any other alternative policy and where $R^*(t)$, $X^*(t)$ are the resulting attribute values of the system under those decisions.

Now let us consider that $x^*(t)$ are the decisions taken by the ω-only policy which yields the optimal utility and satisfies all the constraints of our problem. Considering that this is a ω-only policy and that from Assumption 5.1 we know that $\omega(t)$ is i.i.d., the values of $R^*(t)$, $D^*(t)$, $X^*(t)$ are independent of the queue backlogs $\boldsymbol{\Theta}(t)$. Hence, for example $\mathbb{E}\left\{-R^*_{n_s}(t)+K_s \mid \boldsymbol{\Theta}(t)\right\} = \mathbb{E}\left\{-R^*_{n_s}(t)+K_s\right\}$ where this new expectations are with respect to the network state $\omega(t)$ and the actions $x(t)$. Given that this policy obtains optimal utility and satisfies all constraints (it maintains all queues mean rate stable), it is true that:

$$
\begin{aligned}
\phi(\boldsymbol{\gamma}^*) &= \phi^{opt} \\
\mathbb{E}\{-R^*_{n_s}(t)\} &\le -K_s \\
\mathbb{E}\{A_{n_s}(t)\} &\le \mathbb{E}\{R^*_{n_s}(t)+D^*_{n_s}(t)\} \\
\epsilon_{n_s} &\le \mathbb{E}\{R^*_{n_s}(t)+D^*_{n_s}(t)\} \\
\gamma^*_{n_s}(t) &\le \mathbb{E}\{A_{n_s}(t)-D^*_{n_s}(t)\}
\end{aligned}
\tag{5.94}
$$

Therefore, using the i.i.d. condition of $\omega(t)$ and substituting the previous result in (5.93) yields:

$$
\Delta(\boldsymbol{\Theta}(t)) - V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} \le B - V\phi^{opt} \tag{5.95}
$$

Hence, we have the conditions to apply the Lyapunov Optimization Theorem (Theorem 5.2). Then, from this theorem we know that all queues $\boldsymbol{\Theta}(t) = [\boldsymbol{Q}(t), \boldsymbol{Z}(t), \boldsymbol{G}(t),$

$\boldsymbol{U}(t), \boldsymbol{Y}(t)]$ are mean rate stable and we also obtain for all $t > 0$:

$$\frac{1}{t}\sum_{\tau=1}^{t-1}\mathbb{E}\{\phi(\boldsymbol{\gamma}(\tau))\} \geq \phi^{opt} - \frac{B}{V} - \frac{\mathbb{E}\{L(\boldsymbol{\Theta}(0))\}}{Vt} \tag{5.96}$$

By definition we know that $\phi(\boldsymbol{\gamma}(t))$ is a concave function, then by Jensen's inequality we know that $\frac{1}{t}\sum_{\tau=1}^{t-1}\mathbb{E}\{\phi(\boldsymbol{\gamma}(\tau))\} \leq \phi(\frac{1}{t}\sum_{\tau=1}^{t-1}\mathbb{E}\{\boldsymbol{\gamma}(\tau)\})$, hence we have:

$$\phi(\overline{\boldsymbol{\gamma}}(t)) \geq \phi^{opt} - \frac{B}{V} - \frac{\mathbb{E}\{L(\boldsymbol{\Theta}(0))\}}{Vt} \tag{5.97}$$

where $\overline{\boldsymbol{\gamma}}(t) = \frac{1}{t}\sum_{\tau=1}^{t-1}\mathbb{E}\{\boldsymbol{\gamma}(\tau)\}$. Taking limits yields:

$$\lim_{t\to\infty}\phi(\overline{\boldsymbol{\gamma}}(t)) \geq \phi^{opt} - \frac{B}{V} \tag{5.98}$$

On the other hand, from the definition of the queue dynamics (5.72) it is easy to observe that for any slot $\tau \geq 0$:

$$\begin{aligned} Y_{n_s}(\tau+1) &\geq Y_{n_s}(\tau) + \gamma_{n_s}(\tau) - u_{n_s}(\tau) \\ Y_{n_s}(\tau+1) - Y_{n_s}(\tau) &\geq \gamma_{n_s}(\tau) - u_{n_s}(\tau) \end{aligned} \tag{5.99}$$

Summing the above equations over $\tau \in \{0, ..., t-1\}$ and using the law of telescoping sums we obtain:

$$Y_{n_s}(t) - Y_{n_s}(0) \geq \sum_{\tau=0}^{t-1}\gamma_{n_s}(\tau) - \sum_{\tau=0}^{t-1}u_{n_s}(\tau) \tag{5.100}$$

Then, dividing all by t yields:

$$\begin{aligned} \frac{Y_{n_s}(t)}{t} - \frac{Y_{n_s}(0)}{t} &\geq \frac{1}{t}\sum_{\tau=0}^{t-1}\gamma_{n_s}(\tau) - \frac{1}{t}\sum_{\tau=0}^{t-1}u_{n_s}(\tau) \\ \frac{Y_{n_s}(t)}{t} - \frac{Y_{n_s}(0)}{t} &\geq \overline{\gamma}_{n_s}(t) - \overline{u}_{n_s}(t) \\ \overline{u}_{n_s}(t) &\geq \overline{\gamma}_{n_s}(t) - \frac{Y_{n_s}(t)}{t} + \frac{Y_{n_s}(0)}{t} \end{aligned} \tag{5.101}$$

Taking limits and using the previous result of $Y_{n_s}(t) \leq Y_{n_s}^{max}\ \forall t$ we have:

$$\lim_{t\to\infty}\overline{u}_{n_s}(t) \geq \lim_{t\to\infty}\overline{\gamma}_{n_s}(t) \tag{5.102}$$

Therefore, given that $\phi(\boldsymbol{u})$ is continuous and non-decreasing, we can show that:

$$\lim_{t\to\infty} \phi(\overline{\boldsymbol{u}}(t)) \geq \lim_{t\to\infty} \phi(\overline{\boldsymbol{\gamma}}(t)) \tag{5.103}$$

Finally, using this in (5.98) we get:

$$\lim_{t\to\infty} \phi(\overline{\boldsymbol{u}}(t)) \geq \phi^{opt} - \frac{B}{V} \tag{5.104}$$

which proves the theorem.

□

It is remaining to show that all the constraints of the initial problem in (5.49)-(5.57) are satisfied by the proposed algorithm. From the previous delay bound analysis we have found that the queues $Q_{n_s}(t)$ and $Z_{n_s}(t)$ are bounded for all t. Therefore it is easy to show that this queues are mean rate stable as requested by constraints (5.53) and (5.54). From the previous theorem, we also know that the virtual queues $G_{n_s}(t)$ and $U_{n_s}(t)$ associated to the constraints (5.50) and (5.51) are mean rate stable. Then, from the analysis provided in Section 5.3.3 (mean rate stability of virtual queues imply constraint satisfaction) it derives that the constraints are satisfied.

Even more, given an additional assumption on the behaviour of the control algorithm, we can obtain bounds on the backlog of the queues. The assumption needed is that there exists an ω-only policy that satisfies a *Slater Condition* on the constraints satisfaction. We will show this result applied to the virtual queue $G_{n_s}(t)$ which controls the constraint of minimum average bit rate. The same approach can be followed for the other queues of the system.

Let us assume that exists a value $\zeta > 0$, a value of the utility function ϕ_ζ (which may depend on ζ) and an ω-only policy which yields decisions $x^*(t)$, $D^*(t)$ and satisfies (5.94) but with the following changes:

$$\begin{aligned} \phi(\boldsymbol{\gamma}^*) &= \phi_\zeta \\ \mathbb{E}\{-R^*_{n_s}(t)\} - K_s &\leq -\zeta \end{aligned} \tag{5.105}$$

Therefore, by substituting this ω-only policy in (5.75) we get:

$$\Delta(\boldsymbol{\Theta}(t)) - V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t)) \mid \boldsymbol{\Theta}(t)\} \leq B - V\phi_\zeta - \zeta \sum_{s=1}^{S} \sum_{n_s=1}^{C_s} G_{n_s}(t) \tag{5.106}$$

Fixing a slot τ, using the definition of the drift, taking expectations and using the law of iterated expectations we have:

$$\mathbb{E}\{L(\boldsymbol{\Theta}(\tau+1))\} - \mathbb{E}\{L(\boldsymbol{\Theta}(\tau))\} - V\mathbb{E}\{\phi(\boldsymbol{\gamma}(t))\} \leq B - V\phi_\zeta - \zeta \sum_{s=1}^{S} \sum_{n_s=1}^{C_s} \mathbb{E}\{G_{n_s}(t)\} \tag{5.107}$$

Summing with $\tau \in \{0, 1..., t-1\}$ and using the law of telescoping sums yields:

$$\mathbb{E}\{L(\boldsymbol{\Theta}(t))\} - \mathbb{E}\{L(\boldsymbol{\Theta}(0))\} - V\sum_{\tau=0}^{t-1} \mathbb{E}\{\phi(\boldsymbol{\gamma}(t))\} \leq (B - V\phi_\zeta)t - \zeta \sum_{\tau=0}^{t-1} \sum_{s=1}^{S} \sum_{n_s=1}^{C_s} \mathbb{E}\{G_{n_s}(\tau)\} \tag{5.108}$$

Given that $L(\boldsymbol{\Theta}(t)) \geq 0 \; \forall t$ and rearranging terms we obtain:

$$\zeta \sum_{\tau=0}^{t-1} \sum_{s=1}^{S} \sum_{n_s=1}^{C_s} \mathbb{E}\{G_{n_s}(\tau)\} \leq (B - V\phi_\zeta)t + V\sum_{\tau=0}^{t-1} \mathbb{E}\{\phi(\boldsymbol{\gamma}(t))\} + \mathbb{E}\{L(\boldsymbol{\Theta}(0))\} \tag{5.109}$$

Rearranging and defining $\frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{\phi(\boldsymbol{\gamma}(t))\} = \overline{\phi(\boldsymbol{\gamma}(t))}$:

$$\frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{s=1}^{S} \sum_{n_s=1}^{C_s} \mathbb{E}\{G_{n_s}(\tau)\} \leq \frac{B + V(\overline{\phi(\boldsymbol{\gamma}(t))} - \phi_\zeta)}{\zeta} + \frac{\mathbb{E}\{L(\boldsymbol{\Theta}(0))\}}{\zeta t} \tag{5.110}$$

Finally, taking limits and noting that $\lim_{t\to\infty} \overline{\phi(\boldsymbol{\gamma}(t))} \leq \phi_\gamma^{opt}$ where ϕ_γ^{opt} is the maximum value of the objective function of the transformed problem gives:

$$\lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{s=1}^{S} \sum_{n_s=1}^{C_s} \mathbb{E}\{G_{n_s}(\tau)\} \leq \frac{B + V(\phi_\gamma^{opt} - \phi_\zeta)}{\zeta} \tag{5.111}$$

Therefore, we have obtained a bound on the expected time average of the sum of all the backlogs of the virtual queues $G_{n_s}(t)$ which is directly proportional to the value of V.

5.7.3 Discussion

To conclude this section, we briefly discuss and analyze how the previous results affect our proposal. From the result and proof of Theorem 5.5 we can derive some interesting observations. First of all, the direct result of the theorem is that the proposed solution achieves an approximate utility in which the distance to the optimal utility of the problem is bounded and depends on a configurable parameter. However, in our opinion, it is more interesting the result that can be derived from the equation

(5.97), which provides a bound on the obtained utility for every slot t. As can be observed, the influence of the initial queue backlog decreases at a rate of Vt. In other words, the proposed solution converges to the $\phi^{opt} - {}^{B}\!/_{V}$ utility in order $O(V)$. Although considering the initial queues empty, this is important because it also shows how fast the algorithm can react to a network state change, where it needs to find a new airtime allocation solution after already running. In this case, it is clear that the queue's backlogs are not empty, and though the influence of these backlogs decreases at a rate of Vt.

On the other hand, from the result of the bound for the virtual queues, we have a different situation. As already mentioned, this result shows that we can bound the backlogs if we have a control policy, which, on average, provides an allocation that satisfies the constraint with a slackness of ζ. On one side, this result shows that the sum of the backlogs is proportional to the value V, meaning that higher the value of V, more difference between the expected and achieved constraint bound (the minimum average bit rate for example) is allowed. This result is fundamental for our slicing scenario where the main objective is to guarantee all the constraints the best as possible, and therefore it may be important to maintain V small, even with a cost on the total throughput of the AP. The other important observation from this result is related to how the initial queue's backlogs influence on the bound for every t. Similarly to the utility analysis, in this case, this value decreases at a rate of ζt. Hence, the speed of convergence to the bound depends on the ζ achieved by the control algorithm.

In summary, all the previous results provide a strategy to choose the system configuration parameters. From the result on delay bounds, we can calculate the necessary values for V and ϵ_{n_s} given the requested delay bound of each slice $W_{n_s}^{max}$. However, given there may be several possibilities for these values, the approach is to select the smallest possible value for V, to guarantee the best constraint satisfaction possible. As mentioned, it is critical to note that this has an impact on the total throughput of the AP.

5.8 A Mechanism for Guaranteeing Isolation

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

Therefore, we propose to classify the possible isolation violations in two different types: *Excess of offered load*, and *Lack of resources*. A possible solution to prevent the first type is that the slicing architecture must control and limit the traffic to conform to the parameters in the corresponding slice agreement. A simple yet practical solution to this issue is to perform *traffic shaping* on the incoming flows. Traffic shaping techniques are widely used as they are beneficial for the correct operation of networks with constrained resources. In particular, several techniques for rate limiting purposes already exist [99]. Hence, in this section, we focus on the second case of isolation issues, where there are not enough resources.

5.8.1 Proposed Solution for Guaranteeing Isolation

This isolation issue of *Lack of resources* emerges when there are not enough resources to provide the agreed guarantees to each client, thus affecting their performance. Although admission control mechanisms may prevent this from happening when instantiating new clients or slices into the devices, the channel conditions of a client might worsen after the initial connection, causing the scheduler to take resources from other slices to provide the agreed QoS. From the perspective of the optimization problem formulated in Section 5.6, this issue generates a situation of unfeasibility. This means that there is no possible airtime allocation that can meet all the required constraints.

As described in Section 5.4, in the proposed QoS Slicing model, the isolation between different slices is provided by setting a limit on the number of resources that a slice can use. With this approach, two objectives are achieved: (1) the expected time average airtime usage ratio of each slice is limited, to avoid consuming resources from other slices; and (2) if extra resources are available (because one or more slices use fewer resources than requested), these are distributed among slices that have traffic to be served. Nevertheless, as we need to provide guarantees to each flow of a slice, the behavior of a client or a traffic flow may affect other flows of the same slice. Therefore, the issue of isolation among flows within the same slice must also be tackled by the slicing mechanism. Our isolation proposal consists in integrating the isolation management to the scheduler described in Algorithm 3, to deal with the isolation issue when a client's channel conditions deteriorate, and more resources than available are necessary. We propose a solution in two stages:

- A monitoring stage, where we detect the situations where the available resources are not enough for the required guarantees.
- An action stage, where we chose the clients and the actions to take to move the system to a stable state.

Monitoring

In this stage, we propose to monitor the evolution of the virtual queues to detect isolation issues. We recall that the virtual queues $G_{n_s}(t)$ model the bit rate constraints and they can be conceived as buffers of the amount of bit rate not currently satisfied. In the analysis of Section 5.6, we showed that the objective of the proposed optimization solution is to stabilize the virtual queues, and it has been shown that when virtual queues are stable, the constraints are satisfied. Hence, if we continuously monitor the virtual queue lengths to detect when they are not stable, we can infer that the guarantees are not necessarily being satisfied. Therefore, our solution consists of adding a mechanism that monitors the evolution of virtual queues, so in the case when it detects a situation of a constant increment on the size of any of the virtual queues, it can emit an alarm to take the appropriate action.

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

As already mentioned, this mechanism detects an isolation problem when the performance requirements and the channel conditions make the solution not feasible. However, with the queues dynamic proposed in previous sections, it does not consider the current traffic of the clients. For example, if a client requires a guaranteed bit rate of $3Mbps$, but it is only using $1Mbps$, the system will alert when the $3Mbps$ guarantee is not feasible, even a $1Mbps$ traffic would be. This is due to the update equation of the virtual queues:

$$G_{n_s}(t+1) = [G_{n_s}(t) - R_{n_s}(t) + K_s]^+ \tag{5.112}$$

where $R_{n_s}(t)$ is the achievable bit rate and not the actual bit rate obtained given the current data on the queues. A solution to this issue is to modify the previous equation and to also consider the arrival rate of data:

$$G_{n_s}(t+1) = [G_{n_s}(t) - R_{n_s}(t) + \min[K_s, A_{n_s}(t)]]^+ \tag{5.113}$$

With this change, the scheduler is still required to guarantee K_s but if the input traffic is lower than K_s, it guarantees the input traffic. Our solution incorporates this change.

Enforcing Isolation

For the second stage, our proposal to solve the isolation issue and to obtain a feasible problem is to disconnect some clients from the AP (or to degrade its performance), in a controlled manner. The choice of which clients to disconnect may depend on several factors, such as the number of consumed resources, the slice to which the client belongs, or the associated revenue. In a scenario where the Slicing Architecture described in Section 2.5 is available, this decision should be performed by the Global or Network Slice Manager, which has a complete view of the available resources. Even more, with this approach, these managers may move the client to a different AP or network with more resources available to continue serving it. However, this decision problem is complex and is out of the scope of our work. Hereafter, we just introduce a simple strategy for the actions to be taken we implemented when an isolation violation is detected.

For our solution, we propose to select a client and to remove its QoS guarantees, downgrading it to a *best-effort* client. This strategy must be accompanied by a message to the service provider to take any appropriate action if necessary. With this approach, we allow the scheduler to assign resources to the client, but only if they are available. We propose selecting the client with the highest use of resources to be downgraded. If this still does not solve the isolation situation, we continue downgrading clients until it is resolved. As mentioned, the use of this policy is arbitrary, and other options are perfectly suitable.

5.9 Implementation Details

In this section, we present brief descriptions about particular details that may need to be considered when implementing the proposed solution. Some of these details have already been considered in our simulated prototype we analyze in Section 5.10, but others are specific for hardware implementation.

5.9.1 Variable Time Slots

As was described in the system model, the proposed QoS Slicing solution is built over the allocation mechanism of the ATERR algorithm. For this, the ATERR is used with a fixed size quantum for all queues and with no adaptation mechanism. Hence, we can have a scheduler which assumes slotted time and on each slot assigns a client for transmission. Although the slots are of variable length because the ATERR mechanism does not guarantee that the quantum is exactly respected, it guarantees that, in the long run, the differences are compensated.

However, because of this particular slotting mechanism that we implemented, the time slots can be of variable size. Although this does not impact on the average queue length, since the variations are compensated by the ATERR mechanism, it should be taken into account on the queue updates of each slot. Therefore, our scheduler implementation receives feedback from the transmission module with the actual amount of time consumed. This feedback allows using the exact time slot size consumed in the (virtual and real) queue updates, for example, by calculating the exact amount of data transmitted.

Variations in time slot lengths may also happen because there was not enough data in the queues to fill a complete time slot. In these cases, when a queue empties before completing a time slot, the scheduler is executed to find a new queue for transmission.

5.9.2 Channel Capacity Estimation

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

On the one hand, as we explain in Appendix A, most WiFi devices implement a MCS adaptation mechanism to find the most appropriate MCS for transmission given the current channel conditions. Also, for each MCS, we have a transmission bit rate that can be achieved with it. Then, from this MCS adaptation mechanism, it is easy to obtain the current transmission bit rate used by the AP. Nowadays, the most widespread mechanism in WiFi is called *Minstrel* [16], which uses the frame loss rate to estimate the channel capacity. This mechanism, although not being optimal, provides a good estimation and has the advantage that can be obtained from real data, with almost no overhead.

However, the above approach provides just an upper bound on the actual channel capacity obtained. As was previously discussed, congestion at the wireless channel as well as retransmission because of collisions or packet losses will reduce the channel capacity finally obtained. Hence, to improve the channel capacity estimate, we also use the airtime consumed by the transmissions and the actual data that was transmitted. Therefore, we can estimate the capacity of the next slot by measuring the amount of data transmitted and the airtime consumed in the previous slot.

5.9.3 Maximum Arrival Rate

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

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

5.9.4 Parameter Calculation and Minimum Delay Bound

The proposed mechanism has two fundamental parameters that condition its performance and provide a trade-off between delay and throughput: V and ϵ_{n_s}. As mentioned before, V determines the distance to the optimal utility. In our case, this means that as V is reduced, the total throughput obtained by the AP increases. On the other hand, the delay bound linearly grows with it. Remember that we have previously shown that the upper bound on the packet delay is given by:

$$W_{n_s}^{max} = \left\lceil \frac{2V\omega + 3A_{n_s}^{max} + \epsilon_{n_s}}{\epsilon_{n_s}} \right\rceil \tag{5.114}$$

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

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

$$W_{n_s}^{max} = \left\lceil \frac{2V\omega + 4A_{n_s}^{max}}{A_{n_s}^{max}} \right\rceil \tag{5.115}$$

Note that the previous result provides a bound on the delay measured in time slots. Hence, the actual delay measured in seconds will depend on the particular time slot selected. In our case, two parameters will have a substantial impact on the time slot value, the *quantum* of the ATERR mechanism, and the maximum packet transmission time, which, as we have previously seen in Chapter 4, affects the excess from the predefined quantum. This last factor is really important since the channel capacity of the wireless medium is variable, and very low capacity can generate long delays. For example, if the wireless medium allows a channel capacity of 2Mbps, a packet of 1500B would take 6ms to be transmitted, limiting the achievable queue delay to a value higher than that.

Therefore, the minimum channel capacity to allow in the wireless device should be controlled, and clients with very low capacity might need to be dropped or moved to another device. Note that a client with low capacity would affect all clients, and would not only jeopardize its performance.

5.10 Evaluation of the QoS Slicing Solution

In this section, we evaluate the behavior and performance of the proposed slicing mechanism by implementing a prototype on the MATLAB Simulink [111] software. In the prototype we modeled the queue and scheduling operations, the input traffic patterns, as well as the variable channel conditions of the wireless links. The implemented model is available at [92]. The goal of the evaluation is to show how our solution provides the QoS guarantees to slices with different requirements, when deployed on a WiFi network. We tested slices with different traffic patterns and different QoS requirements, having clients with different and variable channel conditions.

5.10.1 Simulation Setup

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

The QoS requests of each slice and the traffic patterns of each flow are summarized in Table 5.1. Slice 1 requires a minimum guaranteed average bit rate of 300Kbps for each flow, with a maximum allowed delay (delay bound) of 50ms. This must be respected while the required resources do not exceed the 30% of the total resources of the AP. Slice 2 requests a guaranteed average bit rate of 3Mbps, with a maximum delay of 25ms, and with a resource limit of 60%. For both slices the average bit rate guarantee must be measured in a time window of 1 second and include a tolerance of a 10% from the requested average. Finally, Slice 3 only requests a maximum of 10% of the AP resources, with no QoS guarantees. This slice will bear TCP flows with varying load. The TCP traffic generator consists of an application which sends data as fast as possible, as it would be the case of a bulk file transfer. In addition, when the lower layer buffer is full, it waits until some packets are dequeued to send more data, as it would be the case of a real elastic service.

Table 5.1 Slice and Traffic Configuration.

	Slice 1	**Slice 2**	**Slice 3**
Application	Live Video	Real-Time Gaming	Bulk
Traffic pattern	Constant Bit Rate of 300 kbps	Poisson Process[3] with 3 Mbps of mean rate.	TCP Bulk
Guaranteed bit rate	300 kbps	3 Mbps	none
Guaranteed delay	50 ms	25 ms	none
Capacity limit	30%	60%	10%
Number of flows	6	3	3

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

To emulate wireless channel variability, we used a simplified wireless channel model, where packet transmissions can be delayed, mimicking a busy medium situation, and channel capacities are variable, depending on the distance between the client and the AP. A packet is delayed in the output interface for a given period of time d with probability p. Hence, the channel will provide variable capacity, with an average capacity lower than its maximum value. In our simulations, the model is configured to delay each packet by a transmission period, with probability $p = 0.1$.

For the system configuration, we need to determine the value of the parameter V for the entire system and the values of ϵ for each client and slice. Given that all the analytical results of the previous sections are given in time slot units and in our simulation we are measuring delay in seconds, we need to define the time slot size and transform the values accordingly. Therefore, the the first step is to determine the quantum size and found the maximum possible slot size. As the minimum channel capacity is of $5Mbps$ and we have defined a maximum packet size of $1500B$, the maximum airtime of a packet is of $2.4ms$. Hence, for this simulation we have chosen a quantum size of $q = 2.5ms$ for all slices and clients. Under this configuration, the maximum possible airtime consumption of a client on a round is of $4.8ms$ (using the results in Section 4.3). Then, this determines the maximum possible size of a time slot and must be considered on the parameter selection for the delay.

Then, using the previous value of $4.8ms$ for the maximum time slot size we select the values $V = 3$, $\epsilon_1 = 1$ for all clients of Slice 1 and $\epsilon_2 = 4.55$ for all clients of Slice 2. Using the delay bound result from Section 5.7, the defined parameter values yield an upper delay bound for Slice 1 of $48ms$ and for Slice 2 of $30ms$.

Furthermore, the system setup also includes the CoDel [85] queue management technique on the queues associated to Slice 3. Although CoDel does not provide a bound on the delay, it helps on controlling the queue lengths and on minimizing the delay.

In the following we show the results for three different resource usage scenarios (different offered loads):

- **Scenario 1 (Loose Resource Usage)** Given the clients' capacities and the offered load, all QoS guarantees can be accomplished without using the 100% of the requested resources.

- **Scenario 2 (Tight Resource Usage)** Similar to the previous scenario, but QoS Slices use almost all the requested resources.

- **Scenario 3 (Lack of Resources)** After an initial resource assignment, because of a variation in the channel capacities, the amount of available resources is not sufficient to comply with all QoS requirements.

Table 5.2 Channel Capacities

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

5.10.2 Results for Scenario 1

For the first scenario, Loose Resource Usage, we show results with and without traffic in Slice 3. The objective is to assess how our solution correctly manages isolation and maintains Slice 1 and Slice 2 QoS requirements guaranteed. So as to facilitate visualization we do not show the performance of all 12 clients but we show, for each slice, the highest obtained delay and the lowest throughput (the worst client). The channel capacities of each client are depicted in Table 5.2. Clients 1 to 6 belong to Slice 1, Clients 7 to 8 to Slice 2, and Clients 10 to 12 to Slice 3.

The results for this scenario are shown in Figures 5.1 to 5.6. In Figures 5.1 and 5.2 it can be observed that in both cases (with and without traffic on Slice 3), the QoS requirements of delay and bit rate are always guaranteed. When traffic is generated in Slice 3, there are some variations on the average delay, but the maximum delay is kept below the required bound. In Figure 5.4 we depict the packet drop ratio for the clients of Slices 1 and 2, which, as can be seen, are below 10^{-2}. Although we have not considered this metric in the proposed solution, it is important to note that the obtained drop ratios are very low.

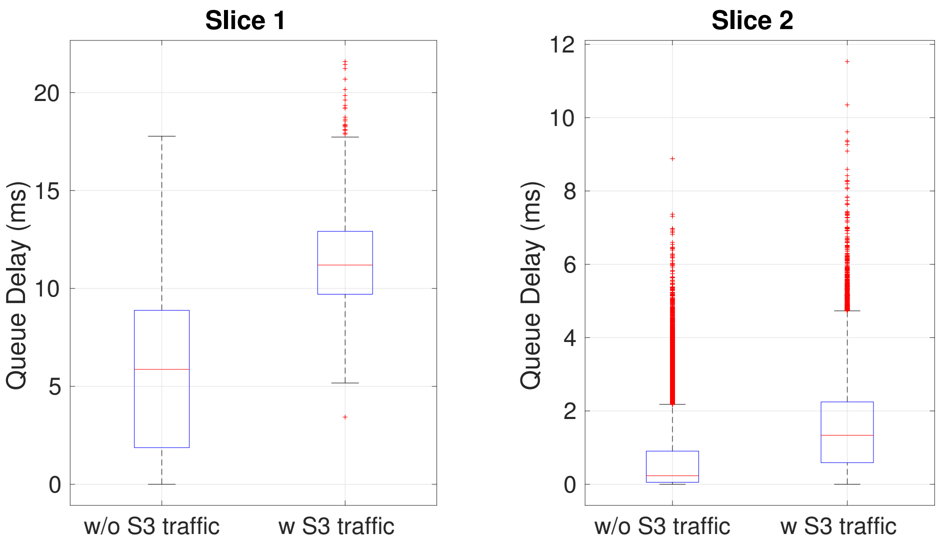

Fig. 5.1 Worst Client Delay for Slices 1 and 2 with and without Traffic on Slice 3.

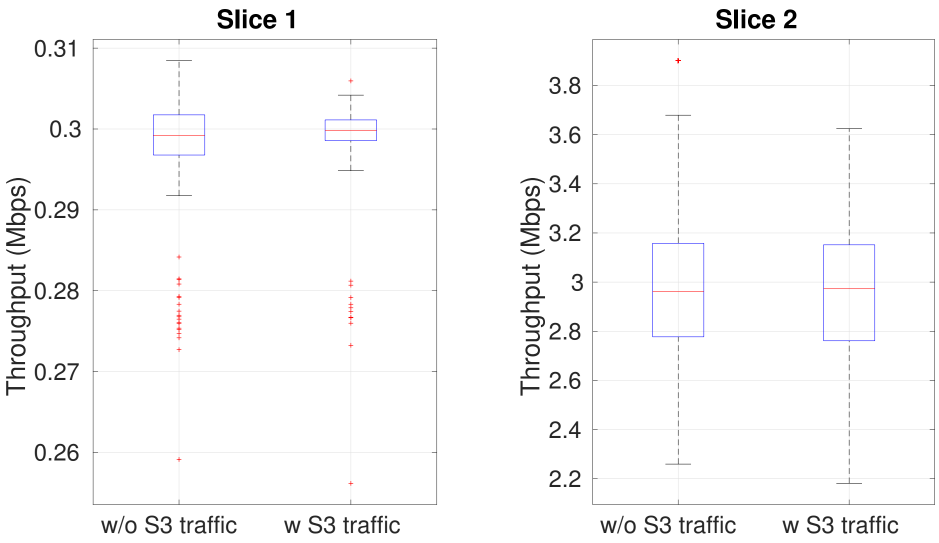

Fig. 5.2 Worst Client Throughput for Slices 1 and 2 with and without Traffic on Slice 3.

Figure 5.3 shows how the resources are allocated to the different slices. Given that Slices 1 and 2 require fewer resources than the maximum capacity to fulfill the QoS requirements, the remaining resources are then exploited by Slice 3. We can also see in Figures 5.5 and 5.6 how extra available resources are used by Slice 3 to obtain throughputs between 1 and 4 Mbps for its clients. As previously shown, this is achieved without affecting the performance of the other slices.

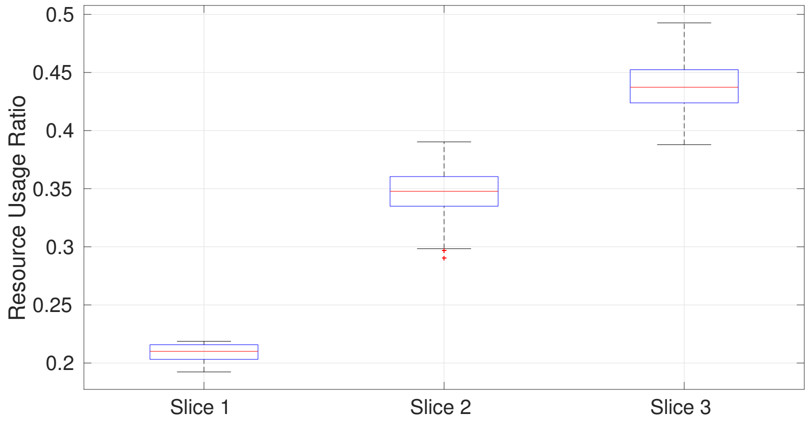

Fig. 5.3 Resource Usage Ratios with Slice 3 Traffic.

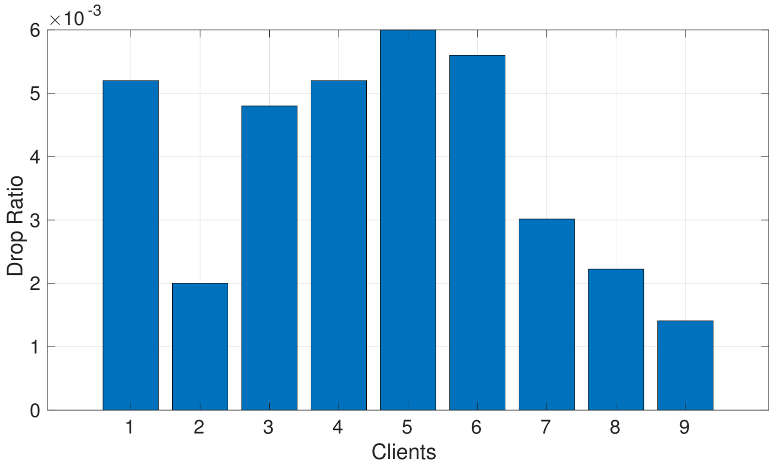

Fig. 5.4 Drop Ratios with Slice 3 Traffic.

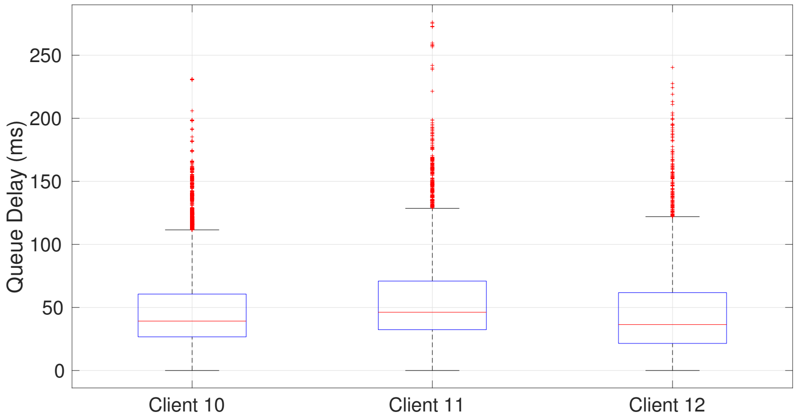

Fig. 5.5 Delay for Slice 3.

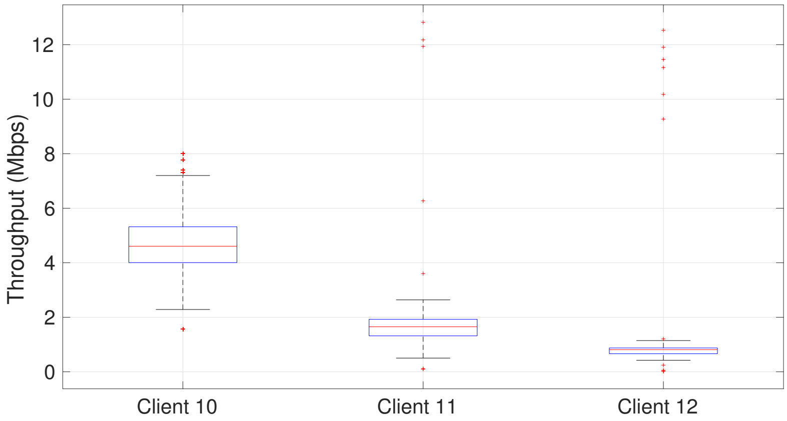

Fig. 5.6 Throughput for Slice 3.

5.10.3 Results for Scenario 2

As previously explained, we also analyzed a scenario where Slices 1 and 2 require more resources to fulfill the QoS requirements. In this case, the average client capacities are depicted in Table 5.2. Figures 5.7 to 5.12 show the results that were obtained for this scenario.

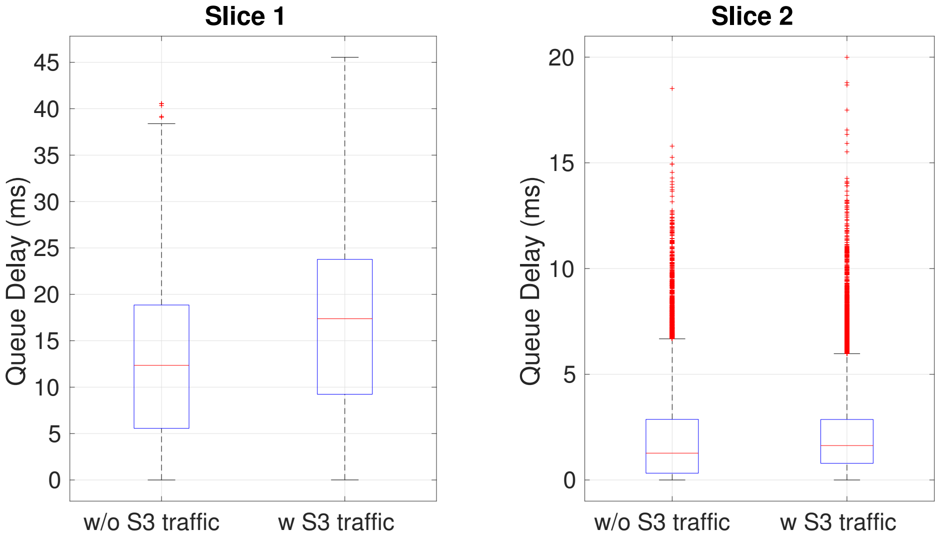

Fig. 5.7 Worst Client Delay for Slices 1 and 2 with and without Traffic on Slice 3.

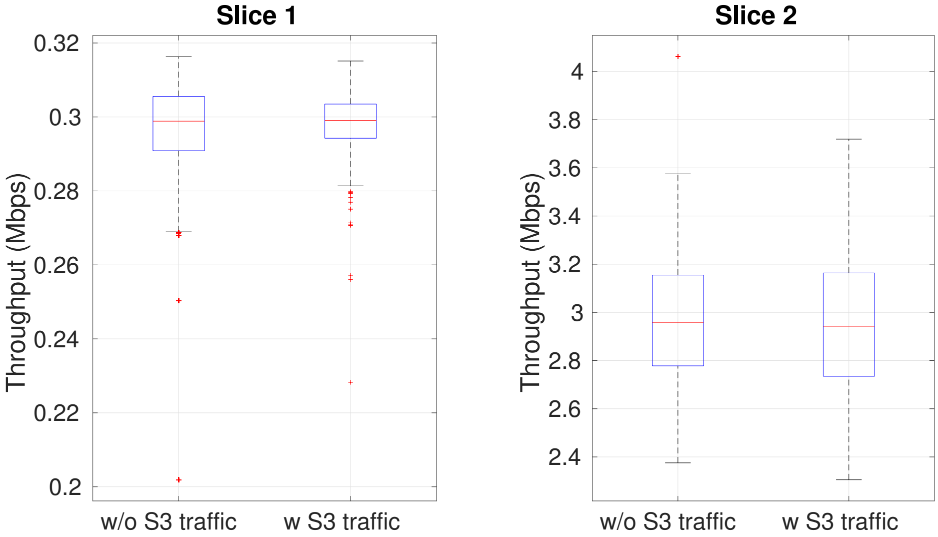

Fig. 5.8 Worst Client Throughput for Slices 1 and 2 with and without Traffic on Slice 3.

As in the previous case, all QoS requirements are correctly met. However, we can observe a slightly worse overall system performance. This is due to the fact that the total channel capacity is lower, and so more resources are needed to achieve the required performance. Nevertheless, the requested delay bound is respected for both slices and average delays are far below the requested bound. Even more, minimum average bit rate is also respected with almost any variation with respect to the previous scenario.

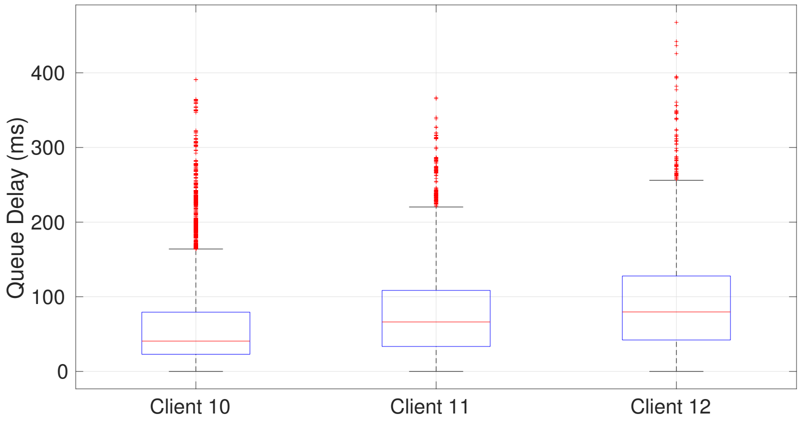

Fig. 5.9 Delay for Slice 3.

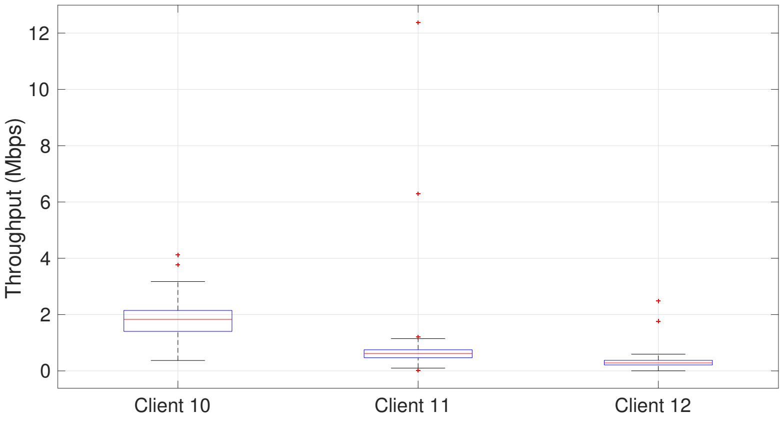

Fig. 5.10 Throughput for Slice 3.

The results in Figures 5.9 to 5.11 show that the proposed solution adapts the resource usage, and it limits Slice 3 traffic, to provide the required QoS in Slices 1 and 2. However, the scheduler still maintains a complete resource usage, allowing Slice 3 to consume more resources than its 10% limit. Finally, we can observe from Figure 5.12 that, in comparison to the Scenario 1, the drop ratio was increased to achieve the

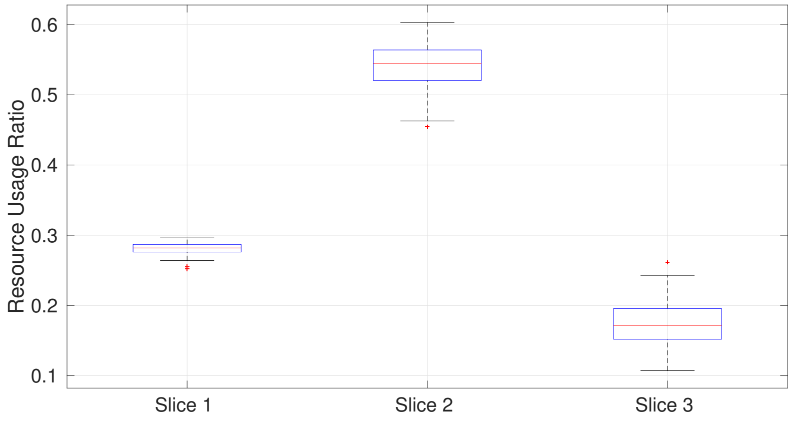

Fig. 5.11 Resource Usage Ratios with Slice 3 Traffic.

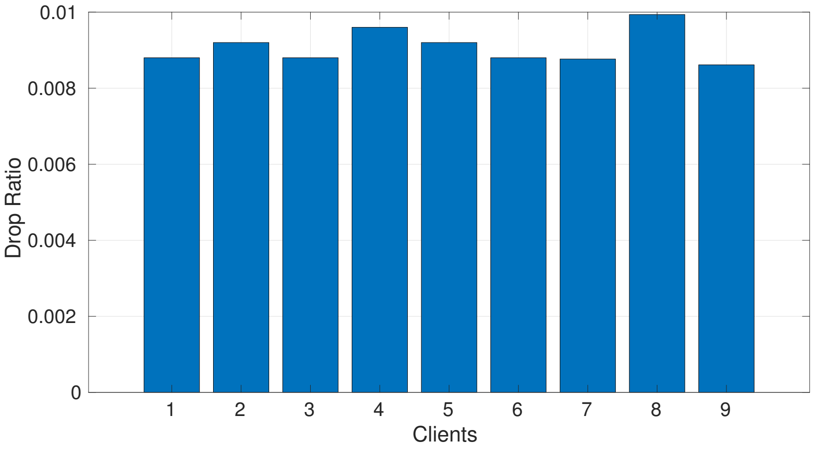

Fig. 5.12 Drop Ratios with Slice 3 Traffic.

required delay bounds. Nevertheless, it continues to be a reasonable value, being below 10^{-2}.

5.10.4 Results for Scenario 3

Finally, we also evaluate a scenario where, in a given instant, the channel capacities deteriorate up to a point such that there are not enough resources to comply with all the QoS requirements. We start the simulation with the same channel capacities used in Scenario 2 and after 50 seconds, we move Client 9 further away from the AP, so that the average channel capacity decreases from 20 Mbps to 5 Mbps.

It is worth noting that this scenario generates a case of isolation violation, as there is not any possible allocation of resources that can accomplish the required guarantees (there is no feasible solution to the optimization problem). Hence, our goal with this experiment is to show two aspects of our isolation solution:

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

For the first aspect, we provide in Figure 5.13 the performance results of Slice 1, before and after the channel variation of Slice 2. As expected, the variation does not affect the QoS requirements of Slice 1, guaranteeing the isolation between slices. As we previously explain, the solution proposed in this thesis to avoid the deterioration of the performance of the clients, is to remove the bad client from the slice. This approach generates that more resources are available for the other clients, which explains the small improvement in performance depicted in Figure 5.13.

Secondly, in Figures 5.14 and 5.15 we depict the evolution along the simulation time of the achieved throughput and delay of the three clients of Slice 2. In this case, as explained in Section 5.8, the scheduler detects the problem by monitoring the virtual queues, selects Client 9 and moves it to Slice 3. Hence, as can be seen, after a small transient period (4 seconds), in which all the clients of the slice are affected by the drastic change in the channel capacity of Client 9, the mechanism stops considering Client 9 QoS requirements. Then, the other slice clients recover their previous performance, while Client 9 is swapped to a best-effort client.

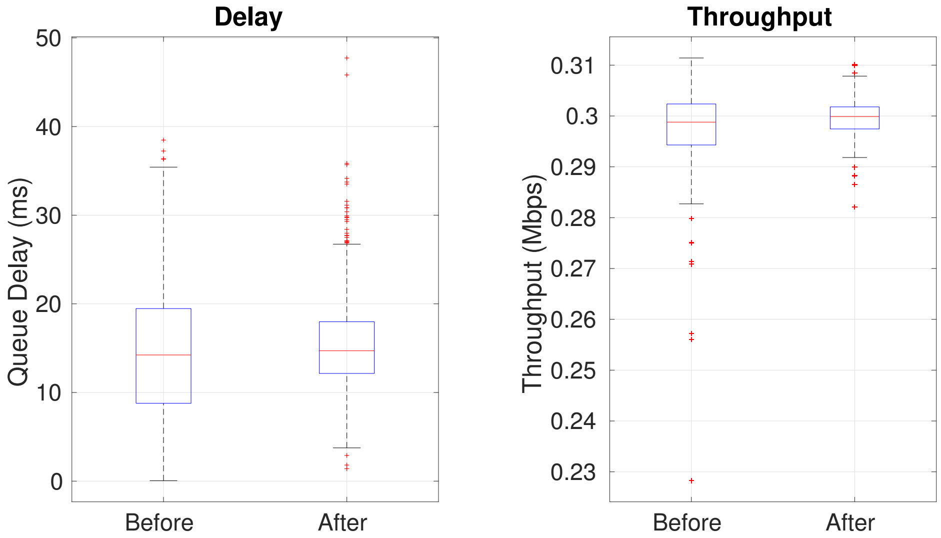

Fig. 5.13 Delay and Throughput for Slice 1 before and after Change in Slice 2.

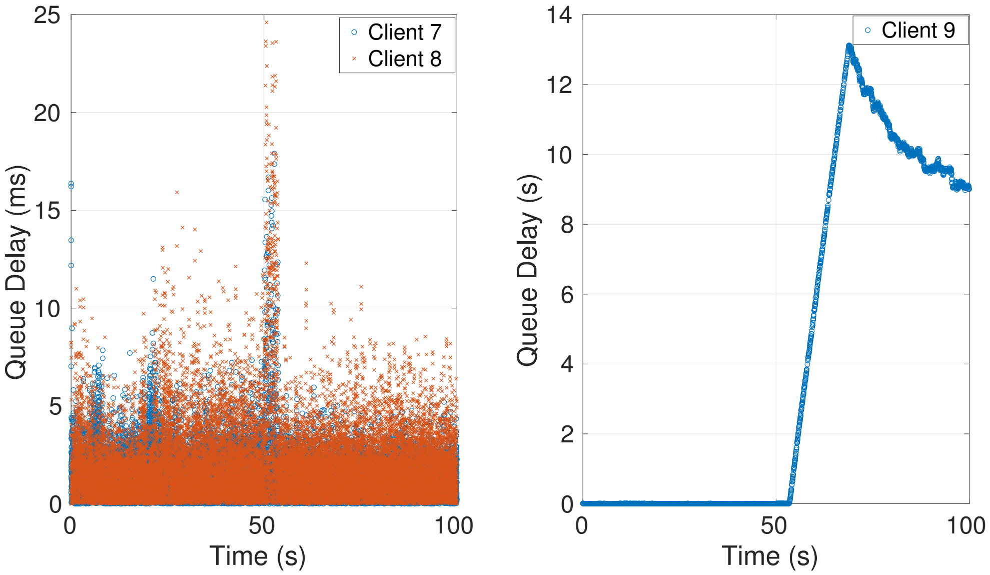

Fig. 5.14 Delay Evolution for Slice 2.

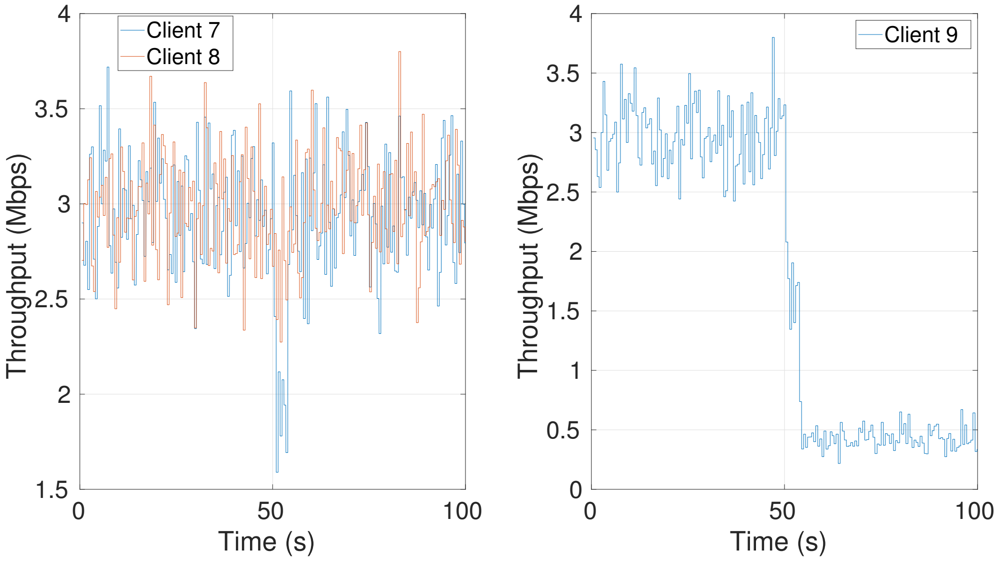

Fig. 5.15 Throughput Evolution for Slice 2.

5.11 Conclusions

In this chapter, we proposed a dynamic resource allocation mechanism to support the development of network slicing in WiFi Access Points. Through a novel packet scheduling design, slices with diverse QoS requirements can be defined. In particular, we propose a mechanism to support slices with two main performance requirements: guaranteed bit rate and delay. Therefore, in Section 5.2, we first analyze in detail the problem of guaranteeing bit rate and delay in wireless networks, and we also review existing approaches in this field. As a result, we concluded that for the problem mentioned above, we must jointly consider a packet scheduling strategy and a mechanism of queue control. We also conclude that formulating the bit rate and delay guaranteed problem as a stochastic optimization problem would be a good strategy.

Therefore, in Section 5.3, we describe the theory of *Lyapunov Optimization* applied to find a solution to the stochastic optimization problem. We also summarize its main results and provide details of some particular aspects of the theory. Although the theory is not a contribution of this thesis, we considered it important to provide those details given the theory is not well known and has some sophisticated features. A vital aspect of this theory, which differentiates it from other possible solutions, is that it provides an approximate solution to the stochastic optimization problem in which the distance to the optimal solution is known and can be adjusted.

In Section 5.4, we construct a model of the slicing problem at an AP as well as a model of the different performance constraints to be able to formulate the optimization problem. Then, in Section 5.5 the QoS Slicing problem is formulated as a stochastic optimization problem, which maximizes the total average expected throughput of the AP while satisfying the constraints of minimum average bit rate, bounded delay and average airtime usage limit. We consider this is an essential contribution of the thesis, as the problem of resource allocation to implement QoS Slicing in wireless networks (with all its complexities and particularities) is condensed in a single optimization problem. Next, in Section 5.6, the theory of Lyapunov Optimization was applied to transform the stochastic optimization problem into a deterministic problem that must be solved in each time slot. The obtained solution requires the instantaneous value of the channel capacity and the system queues status and consists of solving a deterministic optimization problem on each time slot. From the obtained solution, it was derived a scheduling algorithm that selects, in each time slot t, the client to transmit and the packets to be dropped. The proposed algorithm was theoretically analyzed in Section 5.7, showing it founds an approximate solution that satisfies the initial problem objectives of average minimum bit rate and delay bound.

Nevertheless, to implement the obtained solution in WiFi APs, more work was needed. For this, we employ the ATERR scheduler (based on airtime quantum) to provide an approximate time-slotted system. We also analyze in Section 5.9 some more details on how to implement the proposed solution in WiFi. Even more, thanks to the ATERR scheduling strategy and the system model developed, the obtained scheduling algorithm consists of finding a maximum on each time slot. This is of importance since it makes the complexity of the algorithm to be linear on the number of clients and assures the scalability of the solution. Lastly, in Section 5.8, we contribute with a mechanism to detect and control unfeasible situations, where more resources than needed are necessary to achieve the requested guarantees. This provides isolation guarantees to slices in scenarios where unexpected channel capacity variations appear.

Finally, in Section 5.10, we carried out an extensive simulation-based analysis of the proposed scheduler, evaluating the performance in a typical slicing scenario. The results show the effectiveness of our solution in guaranteeing the QoS requirements of all slices, while also providing the required isolation between slices. In summary, in this chapter, we contributed with a novel mechanism to implement QoS Slicing in WiFi APs, which guarantees a requested minimum average bit rate with bounded delay to each client of a slice. Although the mechanism was designed employing an existing technique, its application to a complex and concrete problem in the context of WiFi technology brought many new challenges that were successfully tackled.

Chapter 6

Conclusions and Future Work

In the current design of upcoming 5G networks, the paradigm of Network Slicing appears as one of the main enablers to cope with stringent new requirements. One of the fundamental challenges of Network Slicing is the allocation of physical network resources to the different slices to guarantee its functional and performance requirements. In this thesis, we have analyzed the problem of resource allocation to achieve slicing in the context of wireless access networks and particularly for the IEEE 802.11 technology.

At the beginning of the research, the Network Slicing paradigm had been recently proposed; therefore, its scope was not entirely clear, and there was a lack of a comprehensive definition. Thus, as our first research activity, we perform a comprehensive review of state of the art. As a result, it was observed that the resource allocation problem for slicing shares many similarities with network virtualization, that the paradigms of SDN and NFV are fundamental enablers for slicing and that there was a need for a formal definition of slicing in the context of wireless networks.

It was also found that there was a gap between many proposals for high-level resource management mechanisms and the actual low-level implementation of the resource allocations in the wireless device. In particular, it was observed a lack of an effective solution for implementing resource allocation in the WiFi devices, which can enforce and control the decisions taken by the resource managers. We consider this an important issue as 5G is being designed to integrate existent wireless technologies such as LTE and WiFi with new technologies such as the novel 5G New Radio.

Consequently, we set the objectives described in Section 1.2 which are replicated here:

1. To study the resource model possibilities available in the context of WiFi technology as well as to analyze possible models for the slices' requests of resources in WiFi.

2. To design, develop, and evaluate enforcement and control mechanisms for the allocation of radio resources to slices in WiFi.

3. To focus the design, development, and evaluation of the proposed mechanisms in terms of efficiency, quality of service guarantees, and isolation.

As will be shown in Section 6.1, we consider that all the initial objectives of the thesis have successfully been realized. Nevertheless, we are aware of the limitations of our proposal, so in Section 6.2, we propose possible improvements that can be performed.

6.1 Main Results

6.1.1 Problem Definition and Analysis

One of the main contributions of this work is presented in Chapters 2 and 3. As previously explained, given the novelty of the slicing paradigm, we studied in detail the problem and analyzed its relation with other technologies and paradigms. Specifically, we propose a definition and classification of slicing in two distinct variants: QoS Slicing and Infrastructure Sharing Slicing.

For the specific case of slicing in wireless networks, we introduced the two major problems of slicing wireless resources: resource allocation and isolation. In this context, we contribute with a different approach to the problem separating the resource allocation problem in two sub-tasks: the Dynamic Resource Allocation and the Enforcement and Control. We also analyzed the possible approaches for modeling the resources and requests and proposed different alternatives: a resource-based model and a performance-based model. This accomplishes the first objective proposed for this thesis.

In Chapter 3 we review existent works on slicing in wireless networks with an emphasis on the low-level allocation of resources in the devices. As a result, we presented the main open challenges we observed in this study.

As a result of this work, the survey-style article “Resource Slicing in Virtual Wireless Networks: A Survey” ([93]) was published in the journal *IEEE Transactions on Network and Service Management*.

6.1.2 Resource Allocation for WiFi

Given the complexity of allocating resources in the WiFi technology, mainly because of its medium access mechanism, in Chapter 4 we develop an approach based on the allocation of the transmission time (airtime). The mechanism is based on considering the airtime as the wireless resource that slices can request, and that can be shared and allocated.

Although airtime control has already been considered in WiFi for fairness objectives, we are the first to propose it as a way to offer resource allocation and differentiation and hence, to implement slicing. Despite its simplicity, we consider this is one of the major contributions of our research, which has already been taken and extended by other researchers [50, 25]. The main advantage of the proposal, in comparison with other works on service differentiation in WiFi, is that it avoids the modification of MAC-layer parameters.

For the airtime allocation mechanism, we propose the ATERR scheduling algorithm, which is based on the well known Deficit Round Robin (DDR). The novelty of ATERR resides in that, because of the particularities of the wireless transmissions, it allows a transmission to exceed the quantum size and compensates that excess afterward. Even more, differently from DDR, in ATERR we allow quantum size to be adaptive so as to support slice and client connections and disconnections.

Moreover, the proposed solution has been analytically evaluated so as to show its advantages and limitations. We consider this analysis as an essential contribution and a fundamental distinction with other approximate resource allocation mechanisms. The analysis contributes with a theoretical result, which provides guarantees on the airtime allocation the mechanism can achieve and is essential for the negotiation between the tenant and the entity which provides and implements the slicing.

Finally, simulation results confirmed the accurate airtime-share allocation based on slice requests, acceptable airtime fairness among users of a slice, and the efficient utilization and isolation of resources between slices. Therefore, the second objective formulated for this thesis is accomplished as we have designed, implemented, and evaluated an enforcement and control mechanism, which takes resource requests from slices and allocates resources accordingly. However, the third objective is partially accomplished, given that the solution allocates resources efficiently and guarantees isolation, but it does not offer performance guarantees.

6.1.3 QoS Slicing for WiFi

As already mentioned, slicing has become an essential part of the current 5G network design. In this context, providing WiFi networks with the ability to implement slicing further facilitates the integration of this technology in the 5G ecosystem. Consequently, we regard our QoS Slicing proposal for WiFi presented in Chapter 5 as a major contribution to the further development of 5G.

The proposed solution consists of a scheduling algorithm for Access Points, which, on each time slot, decides which client to transmit to so as to guarantee the performance requirements of all the slices deployed in the device. The algorithm is designed by formulating the slicing resource allocation problem as a stochastic optimization problem, which maximizes the total average throughput of the AP while satisfying the constraints of minimum average bit rate, bounded delay, and average resource usage limit. A solution to this problem was constructed following the Lyapunov Optimization Theory, which provides an approximate deterministic solution. A vital characteristic of this approach is that the distance between the optimal solution and the obtained approximate solution is known and bounded.

The problem formulation and the design and implementation of a solution with the Lyapunov Optimization Theory are important contributions. However, the application of this approach to the WiFi technology would not be possible without two other crucial mechanisms developed in this thesis. First, given that scheduling in WiFi is not based on time slots, we incorporate the QoS scheduler to the ATERR scheduler previously developed. This allows us to have a system that approximates a time-slotted solution and also provides feedback on the consumed airtime, which is crucial to calculate the exact channel capacity. Secondly, as the adopted theory does not consider cases when there is no feasible solution (lack of resources), we design and implement a mechanism to detect and correct unfeasible situations.

In summary, we contributed with a novel mechanism to implement slices with bit rate and delay constraints in WiFi devices. The mechanism is developed by the application of a known technique but has never been applied to the WiFi technology. This novel application brought new challenges that have been worked out to achieve a comprehensive solution.

As a conclusion, we have successfully accomplished the three objectives formulated for this research: (1) we have modeled the WiFi resources as the airtime and we have two different models for requests: one based on resources, where slices can request portions of the total airtime available; and one based on performance, where slices can impose requirements on the minimum bit rate and the maximum delay allowable to its flows; (2) for each type of slice's requests we have designed, developed and evaluated two different enforcement and control mechanisms; (3) the mechanisms use resources efficiently as the unused resources can be used by slices with more traffic, the isolation between slices is always maintained, and the QoS Slicing mechanism offers quality of service guarantees.

6.1.4 Discussion

In this section, we provide a discussion and comparison between the two variants of slicing described in this thesis: Infrastructure Sharing Slicing and Quality of Service Slicing. In particular, we compare the implementations of these variants employing the proposed solutions based on airtime allocation described in Chapters 4 and 5.

As we already mentioned, with the proposed ATERR mechanism, it is possible to implement a slicing solution where the airtime resources of a WiFi AP are partitioned and assigned to the different slices. In this scenario, a slice tenant can request a percentage of airtime given by an average airtime ratio, a tolerance on the requested average, and a time window where the average is measured. With this approach, slices are very easy to define by a tenant, and with our ATERR mechanism, they can be deployed and maintained in the APs with little management. However, it has the main disadvantage of not assuring any performance guarantees to the clients of the slices. Beyond this clear drawback, that is overcame with QoS Slicing, some other interesting limitations are worth mentioning. In this thesis, we followed a conservative strategy for the resource allocation where a slicing provider is not allowed to assign more than a 100% of resources. This assures that the requested resources are always available, but it can be an important limitation of this approach, as it may cause inefficient use of the resources. ATERR partially solves this drawback allowing a slice to consume the airtime reserved but not being used by other slices. Yet, this approach does not allow to over-allocate resources, blocking new slices, although there is available airtime. Another critical design aspect of this solution, which has some advantages and limitations, is how the airtime is measured (as discussed in Section 4.2.5). There exist two possibilities, to consider all the overhead time added by the transmitter, like waiting time in queues caused by congestion and backoffs, or only considering the actual time in the air of the packets. This provides two different sharing alternatives, and the best option may depend on the scenario and context. In summary, this slicing approach presents the advantages that the requested resources are always guaranteed and isolated for the exclusive use of the slice and that it requires minimum management. On the other hand, there is no guarantee on the performance achieved by the slice, and it may incur in inefficient use of the airtime.

With the QoS Slicing variant, the situation is very different. It provides the main advantage of QoS guarantees, which are necessary and beneficial in specific use-cases, but with some limitations. When a tenant requests the deployment of a new slice with a given performance guarantee, there is no knowledge in advance of the number of resources (airtime in our case) that are needed to satisfy this request. As already discussed in this thesis, the needed airtime depends mainly on the channel capacity of the clients using the slice. In this sense, after a slice is instantiated in an AP, clients can connect and disconnect at any time, making it very difficult to predict the number of resources needed. Moreover, user mobility and interference from other devices cause the channel capacity of a client to vary significantly. Hence, there is no previous certainty of the needed resources for a slice, neither if a performance requirement can be accomplished with the available resources through all the slice lifetime. This is an important drawback of this slicing variant, which can make it difficult to be adopted by potential slice tenants such as service providers. This generates complicated situations, where clients with adverse channel conditions cannot be served or, if served, they consume too many resources, which makes fewer clients or slices to be accepted in the AP. A partial solution proposed in this thesis is also to consider a limit on the amount of airtime consumed by a slice. This has the advantage that a slice would not hoard all the AP's available airtime, but on the other hand, it limits the number of clients a slice can serve. In summary, depending on the scenario, on the capacity of the APs, on the number of slices and on the performance requirements of the slices, providing QoS guarantees in a wireless network could need sophisticated management strategies to overcome the aforementioned limitations. In this regard, a possible solution is to have the support of a well-designed management architecture, where the different slices performances can be monitored and where clients can be moved between APs and between different networks (possibly of different technologies).

On balance, both approaches have advantages and drawbacks, and, in our opinion, there is no clear superiority of one strategy over the other. The decision on which variant to use may depend on the scenario, on the available network management functions, and on the objectives of the slicing.

6.2 Future Work

During the course of this thesis, the concept of Network Slicing has transformed from a management paradigm in 5G networks to a broad research topic in the field of computer networks. Therefore, the open directions and research challenges are vast and include management frameworks and architectures, industry standards, resource allocation solutions for the data center, the core network and the wireless access networks, scheduling algorithms at the wireless devices, and spectrum management techniques.

In the particular case of the proposals presented in this dissertation, we envision some improvements and further developments that can be carried out in future research.

Implementation in a Real Field Trial An important aspect of the research, which could not be considered within the scope of this thesis, is to implement the slicing mechanism in hardware. Given the particularities of the wireless medium, being able to evaluate the solution on a real deployment would add significant value to the proposed solution. Even more, this would help to develop more complete evaluations considering several APs, clients, and slices. As explained in Chapter 4, the implementation of the airtime allocation proposal does not appear too challenging as it is based on an existing work already implemented in hardware. On the other hand, the implementation of the QoS Slicing solution could be more complicated. In this case, it is necessary to combine information such as queue sizes and channel capacities, which may be available at different parts of the hardware driver.

Integration of the Thesis Contributions into a Global Slicing Architecture As specified in several parts of this thesis, the developed solution focus only on the Enforcement and Control of the resource allocations on a single device, i.e., a single AP. As a future research activity, it would be of interest to combine the solutions with a high-level Network Manager, which could perform resource allocation decisions with a global view of the network. For example, in a network with multiple coordinated APs which provide a slicing solution based on airtime allocation, the Network Manager must consider isolation issues between neighboring APs. Even more, a more challenging task would be to integrate the proposed wireless slicing solution to a complete 5G network so as to implement slicing end-to-end. In this case, the interfacing between the different systems could become really complicated.

Overcoming the Limitations of the Lyapunov Optimization Theory The Lyapunov Optimization Theory applied in the solution of QoS Slicing makes some assumptions that, although being appropriate for the considered problem, can be improved. The theory assumes that the arrival rate and the channel capacity stochastic processes are independent and identically distributed (i.i.d.) over time. This assumption can be improved by considering Markovian processes and developing more complex solutions that consider this fact. For example, one may consider that if the capacity of the channel in a given instant t is good, there is a good chance that it continues to be good in instant $t+1$. Therefore, this information could be used to make better decisions.

In this context, we envision the application of reinforcement learning techniques to predict the arrival traffic or the channel variations better and make more informed decisions.

Extension of the Lyapunov Optimization Theory In our QoS Slicing solution, we use the ATERR quantum scheduling so as to provide the drift-plus-penalty method with slotted time. However, ATERR does not guarantee that all slots are the same size but that, on average, the differences in size are compensated in the long run. It would be of great interest to develop a theoretical demonstration of the performance of this extension of the Lyapunov Optimization Theory. Although we have not studied this aspect in detail, we consider it is possible given that we already have some theoretical results of the ATERR performance which provides bounds on the minimum and maximum size of a time slot.

Bibliography

[1] 3GPP. Service requirements for the 5G system; Stage 1 (Release 15). Technical Specification (TS) 22.261, 3rd Generation Partnership Project (3GPP), 12 2018. URL `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3107`. Version 15.7.0.

[2] 3GPP. System architecture for the 5G System (5GS); Stage 2 (Release 15). Technical Specification (TS) 23.501, 3rd Generation Partnership Project (3GPP), 06 2019. URL `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144`. Version 15.6.0.

[3] 3GPP. NR; NR and NG-RAN Overall Description; Stage 2 (Release 15). Technical Specification (TS) 38.300, 3rd Generation Partnership Project (3GPP), 04 2019. URL `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3191`. Version 15.5.0.

[4] Bernard Aboba. Virtual access points. Technical Report IEEE 802.11-03/154r0, Microsoft, One Microsoft Way, Redmond, WA 98052-6399, 2003.

[5] Hari Adiseshu, Guru Parulkar, and George Varghese. A reliable and scalable striping protocol. *ACM SIGCOMM Computer Communication Review*, 26(4): 131–141, 1996.

[6] Ibrahim Afolabi, Tarik Taleb, Konstantinos Samdanis, Adlen Ksentini, and Hannu Flinck. Network slicing & softwarization: A survey on principles, enabling technologies & solutions. *IEEE Communications Surveys & Tutorials*, 2018.

[7] Joan Josep Aleixendri, August Betzler, and Daniel Camps-Mur. A practical approach to slicing wi-fi rans in future 5g networks. In IEEE, editor, *Wireless Communications and Networking Conference (WCNC), 2019 IEEE*, 2019, In Press.

[8] Ghannam Aljabari and Evren Eren. Virtualization of wireless lan infrastructures. In *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2011 IEEE 6th International Conference on*, volume 2, pages 837–841. IEEE, 2011.

[9] Jeffrey G Andrews, Stefano Buzzi, Wan Choi, Stephen V Hanly, Aurelie Lozano, Anthony CK Soong, and Jianzhong Charlie Zhang. What will 5G be? *Selected Areas in Communications, IEEE Journal on*, 32(6):1065–1082, 2014.

[10] Matthew Andrews, Krishnan Kumaran, Kavita Ramanan, Alexander Stolyar, Phil Whiting, and Rajiv Vijayakumar. Providing quality of service over a shared wireless link. *IEEE Communications magazine*, 39(2):150–154, 2001.

[11] Arash Asadi and Vincenzo Mancuso. A survey on opportunistic scheduling in wireless communications. *IEEE Communications Surveys & Tutorials*, 15(4): 1671–1688, 2013.

[12] Albert Banchs and Xavier Perez. Providing throughput guarantees in ieee 802.11 wireless lan. In *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, volume 1, pages 130–138. IEEE, 2002.

[13] Albert Banchs, Pablo Serrano, and Luca Vollero. Providing service guarantees in 802.11 e edca wlans with legacy stations. *IEEE Transactions on Mobile Computing*, 9(8):1057–1071, 2010.

[14] Albert Banchs, Pablo Serrano, Paul Patras, and Marek Natkaniec. Providing throughput and fairness guarantees in virtualized wlans through control theory. *Mob. Netw. Appl.*, 17(4):435–446, August 2012. ISSN 1383-469X. doi: 10.1007/s11036-012-0382-2. URL `http://dx.doi.org/10.1007/s11036-012-0382-2`.

[15] Manu Bansal, Jeffrey Mehlman, Sachin Katti, and Philip Levis. Openradio: a programmable wireless dataplane. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 109–114. ACM, 2012.

[16] Johannes Berg. *Minstrel Rate Control Algorithm Documentation*, 2016. URL `https://wireless.wiki.kernel.org/en/developers/documentation/mac80211/ratecontrol/minstrel`.

[17] Gautam Bhanage, Ronak Daya, Ivan Seskar, and Dipankar Raychaudhuri. Vnts: A virtual network traffic shaper for air time fairness in 802.16 e systems. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–6. IEEE, 2010.

[18] Gautam Bhanage, Ivan Seskar, Rajesh Mahindra, and Dipankar Raychaudhuri. Virtual basestation: architecture for an open shared wimax framework. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, pages 1–8. ACM, 2010.

[19] Gautam Bhanage, Dipti Vete, Ivan Seskar, and Dipankar Raychaudhuri. Splitap: leveraging wireless network virtualization for flexible sharing of wlans. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–6. IEEE, 2010.

[20] David A Burgess, Harvind S Samra, et al. The openbts project, 2008.

[21] Francesco Capozzi, Giuseppe Piro, Luigi Alfredo Grieco, Gennaro Boggia, and Pietro Camarda. Downlink packet scheduling in lte cellular networks: Key design issues and a survey. *Communications Surveys & Tutorials, IEEE*, 15(2):678–700, 2013.

[22] Maxweel Carmo, Sandino Jardim, Augusto Neto, Rui Aguiar, Daniel Corujo, and Joel JPC Rodrigues. Slicing wifi wlan-sharing access infrastructures to enhance ultra-dense 5g networking. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

[23] Clark Chen. C-ran: the road towards green radio access network. White paper 2.5, China Mobile Research Institute, 2011.

[24] Cisco. Cisco visual networking index: Forecast and trends, 2017–2022. White paper, Cisco Systems, February 2019.

[25] Estefanía Coronado, Roberto Riggio, José Villa1ón, and Antonio Garrido. Lasagna: Programming abstractions for end-to-end slicing in software-defined wlans. In *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*, pages 14–15. IEEE, 2018.

[26] Peter Dely, Jonathan Vestin, Andreas Kassler, Nico Bayer, Hans Einsiedler, and Christoph Peylo. Cloudmac - an openflow based architecture for 802.11 mac layer processing in the cloud. In *Globecom Workshops (GC Wkshps), 2012 IEEE*, pages 186–191. IEEE, 2012.

[27] Mahsa Derakhshani, Xiaowei Wang, Tho Le-Ngoc, and Alberto Leon-Garcia. Virtualization of multi-cell 802.11 networks: Association and airtime control. *arXiv preprint arXiv:1508.03554*, 2015.

[28] Mahsa Derakhshani, Xiaowei Wang, Daniel Tweed, Tho Le-Ngoc, and Alberto Leon-Garcia. Ap-sta association control for throughput maximization in virtualized wifi networks. *IEEE Access*, 6:45034–45050, 2018.

[29] Behnam Dezfouli, Vahid Esmaeelzadeh, Jaykumar Sheth, and Marjan Radi. A review of software-defined wlans: Architectures and central control mechanisms. *IEEE Communications Surveys & Tutorials*, 21(1):431–463, 2019. ISSN 1553-877X.

[30] Rahul Dhar, Gesly George, Amit Malani, and Peter Steenkiste. Supporting integrated mac and phy software development for the usrp sdr. In *Networking Technologies for Software Defined Radio Networks, 2006. SDR'06.1 st IEEE Workshop on*, pages 68–77. IEEE, 2006.

[31] John C Doyle, Bruce A Francis, and Allen R Tannenbaum. *Feedback control theory*. Courier Corporation, 2013.

[32] Javan Erfanian and Brian Daly. 5G White Paper. White paper, NGMN Alliance, March 2015.

[33] Ericsson. 5G systems (white paper). Technical Report Uen 284 23-3244, Ericsson, January 2015. URL `http://www.ericsson.com/res/docs/whitepapers/what-is-a-5g-system.pdf`.

[34] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on networking*, 1(4):397–413, 1993.

[35] Xenofon Foukas, Navid Nikaein, Mohamed M Kassem, Mahesh K Marina, and Kimon Kontovasilis. Flexran: A flexible and programmable platform for software-defined radio access networks. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 427–441. ACM, 2016.

[36] Xenofon Foukas, Mahesh K Marina, and Kimon Kontovasilis. Orion: Ran slicing for a flexible and cost-effective multi-service mobile network architecture. In *Proceedings of the 23rd annual international conference on mobile computing and networking*, pages 127–140. ACM, 2017.

[37] Rosario G Garroppo, Stefano Giordano, Stefano Lucetti, and Luca Tavanti. Providing air-time usage fairness in ieee 802.11 networks with the deficit transmission time (dtt) scheduler. *Wireless Networks*, 13(4):481–495, 2007.

[38] Leonidas Georgiadis, Michael J Neely, and Leandros Tassiulas. *Resource allocation and cross-layer control in wireless networks*. Now Publishers Inc, 2006.

[39] Barry Graham. NFV and SDN answer or question? http://www.cnsm-conf.org/2015/files/sdnnfv-keynote.pdf, 11 2015. URL `http://www.cnsm-conf.org/2015/files/sdnnfv-keynote.pdf`.

[40] Aditya Gudipati, Daniel Perry, Li Erran Li, and Sachin Katti. Softran: Software defined radio access network. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 25–30. ACM, 2013.

[41] Katherine Guo, Shruti Sanadhya, and Thomas Woo. Vifi: virtualizing wlan using commodity hardware. *ACM SIGMOBILE Mobile Computing and Communications Review*, 18(3):41–48, 2015.

[42] Tao Guo and Rob Arnott. Active lte ran sharing with partial resource reservation. In *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*, pages 1–5. IEEE, 2013.

[43] Mesud Hadzialic, Branko Dosenovic, Merim Dzaferagic, and Jasmin Musovic. Cloud-ran: innovative radio access network architecture. In *ELMAR, 2013 55th International Symposium*, pages 115–120. IEEE, 2013.

[44] Hassan Hawilo, Abdallah Shami, Maysam Mirahmadi, and Rasool Asal. Nfv: state of the art, challenges, and implementation in next generation mobile networks (vepc). *Network, IEEE*, 28(6):18–26, 2014.

[45] Martin Heusse, Franck Rousseau, Gilles Berger-Sabbatel, and Andrzej Duda. Performance anomaly of 802.11 b. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 836–843. IEEE, 2003.

[46] Toke Hoeiland-Joergensen, Paul McKenney, Dave Taht, Jim Gettys, and Eric Dumazet. The FlowQueue-CoDel Packet Scheduler and Active Queue Management Algorithm. Internet-Draft draft-ietf-aqm-fq-codel-06, IETF Secretariat, March 2016. URL `http://www.ietf.org/internet-drafts/draft-ietf-aqm-fq-codel-06.txt`.

[47] Marco Hoffmann and Markus Staufer. Network virtualization for future mobile networks: General architecture and applications. In *Communications Workshops (ICC), 2011 IEEE International Conference on*, pages 1–5. IEEE, 2011.

[48] Toke Høiland-Jørgensen, Per Hurtig, and Anna Brunstrom. The good, the bad and the wifi: Modern aqms in a residential setting. *Computer Networks*, 89: 90–106, 2015.

[49] Toke Høiland-Jørgensen, Michał Kazior, Dave Täht, Per Hurtig, and Anna Brunstrom. Ending the anomaly: Achieving low latency and airtime fairness in wifi. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pages 139–151, Santa Clara, CA, 2017. USENIX Association. ISBN 978-1-931971-38-6. URL `https://www.usenix.org/conference/atc17/technical-sessions/presentation/hoilan-jorgesen`.

[50] Toke Høiland-Jørgensen, Per Hurtig, and Anna Brunstrom. Polifi: Airtime policy enforcement for wifi. *arXiv preprint arXiv:1902.03439*, 2019.

[51] Fei Hu, Qi Hao, and Ke Bao. A survey on software-defined network and openflow: from concept to implementation. *Communications Surveys & Tutorials, IEEE*, 16(4):2181–2206, 2014.

[52] Joseph Y Hui. Resource allocation for broadband networks. *Selected Areas in Communications, IEEE Journal on*, 6(9):1598–1608, 1988.

[53] Xin Jin, Li Erran Li, Laurent Vanbever, and Jennifer Rexford. Softcell: Scalable and flexible cellular core network architecture. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages 163–174. ACM, 2013.

[54] Bart Jooris, Jan Bauwens, Peter Ruckebusch, Peter De Valck, Christophe Van Praet, Ingrid Moerman, and Eli De Poorter. Taisc: a cross-platform mac protocol compiler and execution engine. *Computer Networks*, 2016.

[55] Vikas Jumba, Saeedeh Parsaeefard, Mahsa Derakhshani, and Tho Le-Ngoc. Dynamic resource provisioning with stable queue control for wireless virtualized networks. In *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1856–1860. IEEE, 2015.

[56] Mahmoud Kamel, Long Bao Le, Antoine Girard, et al. LTE wireless network virtualization: Dynamic slicing via flexible scheduling. In *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th*, pages 1–5. IEEE, 2014.

[57] Kostas Katsalis, Kostas Choumas, Thanasis Korakis, and Leandros Tassiulas. Virtual 802.11 wireless networks with guaranteed throughout sharing. In *Computers and Communication (ISCC), 2015 IEEE Symposium on*, pages 845–850. IEEE, 2015.

[58] Hongseok Kim and Gustavo De Veciana. Losing opportunism: Evaluating service integration in an opportunistic wireless system. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pages 982–990. IEEE, 2007.

[59] Ravi Kokku, Rajesh Mahindra, Honghai Zhang, and Sampath Rangarajan. Nvs: a substrate for virtualizing wireless resources in cellular networks. *Networking, IEEE/ACM Transactions on*, 20(5):1333–1346, 2012.

[60] Ravi Kokku, Rajesh Mahindra, Honghai Zhang, and Sampath Rangarajan. Cellslice: Cellular wireless resource slicing for active ran sharing. In *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, pages 1–10. IEEE, 2013.

[61] Diego Kreutz, Fernando MV Ramos, P Esteves Verissimo, C Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015.

[62] Adlen Ksentini and Navid Nikaein. Toward enforcing network slicing on ran: Flexibility and resources abstraction. *IEEE Communications Magazine*, 55(6): 102–108, 2017.

[63] Anurag Kumar, D Manjunath, and Joy Kuri. *Communication networking: an analytical approach.* Access Online via Elsevier, 2004.

[64] Katsutoshi Kusume, Mikael Fallgren, Olav Queseth, Volker Braun, David Gozalvez-Serrano, Isabelle Korthals, Gerd Zimmermann, Martin Schubert, Mohammad Istiak Hossain, Ashraf A. Widaa, Konstantinos Chatzikokolakis, Reza Holakouei, SÃ©bastien Jeux, Javier Lorca Hernando, and Mauro Boldi. Deliverable D1.5. Updated scenarios, requirements and KPIs for 5G mobile and wireless system with recommendations for future investigations., April 2015. URL `https://www.metis2020.com/wp-content/uploads/deliverables/METIS_D1.5_v1.pdf`.

[65] Jang-Won Lee, Ravi R Mazumdar, and Ness B Shroff. Opportunistic power scheduling for multi-server wireless systems with minimum performance constraints. In *IEEE INFOCOM 2004*, volume 2, pages 1067–1077. IEEE, 2004.

[66] Ming Li, Liang Zhao, Xi Li, Xiaona Li, Yasir Zaki, Andreas Timm-Giel, and C Gorg. Investigation of network virtualization and load balancing techniques in lte networks. In *Vehicular Technology Conference (VTC Spring), 2012 IEEE 75th*, pages 1–5. IEEE, 2012.

[67] Qian Li, Huaning Niu, Geng Wu, Mo-Han Fong, and Apostolos Papathanassiou. Slicing architecture for wireless communication, September 26 2017. US Patent 9,775,045.

[68] C. Liang and F.R. Yu. Wireless network virtualization: A survey, some research issues and challenges. *Communications Surveys Tutorials, IEEE*, 17(1):358–380, Firstquarter 2015. ISSN 1553-877X. doi: 10.1109/COMST.2014.2352118.

[69] Rajesh Mahindra, GD Bhanage, George Hadjichristofi, Ivan Seskar, Dipankar Raychaudhuri, and YY Zhang. Space versus time separation for wireless virtualization on an indoor grid. In *Next Generation Internet Networks, 2008. NGI 2008*, pages 215–222. IEEE, 2008.

[70] Ilaria Malanchini, Stefan Valentin, and Osman Aydin. Wireless resource sharing for multiple operators: Generalization, fairness, and the value of prediction. *Computer Networks*, 100:110–123, 2016.

[71] Aqsa Malik, Junaid Qadir, Basharat Ahmad, Kok-Lim Alvin Yau, and Ubaid Ullah. Qos in ieee 802.11-based wireless networks: a contemporary review. *Journal of Network and Computer Applications*, 55:24–46, 2015.

[72] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[73] MCS Index. Modulation and Coding Schemes for IEEEE 802.11n and IEEE 802.11ac. `mcsindex.com`, 2019. Accessed: 2019-08-02.

[74] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba. Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 18(1):236–262, Firstquarter 2016. ISSN 1553-877X. doi: 10.1109/COMST.2015.2477041.

[75] Rashid Mijumbi, Joan Serrat, Javier Rubio-Loyola, Niels Bouten, Filip De Turck, and Steven Latré. Dynamic resource management in sdn-based virtualized networks. In *Network and Service Management (CNSM), 2014 10th International Conference on*, pages 412–417. IEEE, 2014.

[76] Thomas M Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[77] Andreas F Molisch. *Wireless communications*, volume 15. Wiley. com, 2010.

[78] Kiyohide Nakauchi, Kentaro Ishizu, Homare Murakami, Akihiro Nakao, and Hiroshi Harada. Amphibia: a cognitive virtualization platform for end-to-end slicing. In *Communications (ICC), 2011 IEEE International Conference on*, pages 1–5. IEEE, 2011.

[79] Kiyohide Nakauchi, Yozo Shoji, and Nozomu Nishinaga. Airtime-based resource control in wireless lans for wireless network virtualization. In *Ubiquitous and Future Networks (ICUFN), 2012 Fourth International Conference on*, pages 166–169. IEEE, 2012.

[80] Michael J Neely. Stochastic network optimization with application to communication and queueing systems. *Synthesis Lectures on Communication Networks*, 3 (1):1–211, 2010.

[81] Michael J Neely. Opportunistic scheduling with worst case delay guarantees in single and multi-hop networks. In *INFOCOM, 2011 Proceedings IEEE*, pages 1728–1736. IEEE, 2011.

[82] Michael J Neely and Rahul Urgaonkar. Opportunism, backpressure, and stochastic optimization with the wireless broadcast advantage. In *Signals, Systems and Computers, 2008 42nd Asilomar Conference on*, pages 2152–2158. IEEE, 2008.

[83] Michael Neufeld, Jeff Fifield, Christian Doerr, Anmol Sheth, and Dirk Grunwald. Softmac-flexible wireless research platform. In *Proc. HotNets-IV*, pages 1–5, 2005.

[84] NGMN Alliance. Description of network slicing concept. *NGMN 5G P1 Requirements & Architecture Work Stream End-to-End Architecture*, 1, 2016.

[85] Kathleen Nichols and Van Jacobson. Controlling queue delay. *Communications of the ACM*, 55(7):42–50, 2012.

[86] Navid Nikaein, Mahesh K Marina, Saravana Manickam, Alex Dawson, Raymond Knopp, and Christian Bonnet. Openairinterface: A flexible platform for 5g research. *ACM SIGCOMM Computer Communication Review*, 44(5):33–38, 2014.

[87] UK. Ofcom, The Office of Communications. Infrastructure report 2014. ofcom‘s second full analysis of the uk‘s communications infrastructure. Technical report, Ofcom, The Office of Communications, UK., 2014.

[88] Jignesh S Panchal, Roy D Yates, and Milind M Buddhikot. Mobile network resource sharing options: Performance comparisons. *Wireless Communications, IEEE Transactions on*, 12(9):4470–4482, 2013.

[89] Kostas Pentikousis, Yan Wang, and Weihua Hu. Mobileflow: Toward software-defined mobile networks. *Communications Magazine, IEEE*, 51(7):44–53, 2013.

[90] Junaid Qadir, Nadeem Ahmed, and Nauman Ahad. Building programmable wireless networks: an architectural survey. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):1–31, 2014.

[91] Matias Richart. mrichart/ns-3-dev-git: Airtime slicing simulation code for ns3, Jul 2018.

[92] Matias Richart. Qos slicing simulation code for matlab simulink, Apr 2019. URL `10.5281/zenodo.2649134`.

[93] Matias Richart, Javier Baliosian, Joan Serrat, and Juan-Luis Gorricho. Resource slicing in virtual wireless networks: A survey. *IEEE Transactions on Network and Service Management*, 13(3):462–476, 2016.

[94] Roberto Riggio, Daniele Miorandi, and Imrich Chlamtac. Airtime deficit round robin (adrr) packet scheduling algorithm. In *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, pages 647–652. IEEE, 2008.

[95] Roberto Riggio, Karina Mabell Gomez, Tinku Rasheed, Julius Schulz-Zander, Slawomir Kuklinski, and Mahesh K Marina. Programming software-defined wireless networks. In *Network and Service Management (CNSM), 2014 10th International Conference on*, pages 118–126. IEEE, 2014.

[96] George F Riley and Thomas R Henderson. The ns-3 network simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010. URL `http://www.nsnam.org/`.

[97] Joachim Sachs and Stephan Baucke. Virtual radio: a framework for configurable radio networks. In *Proceedings of the 4th Annual International Conference on Wireless Internet*, page 61. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[98] Bilal Sadiq, Seung Jun Baek, and Gustavo De Veciana. Delay-optimal opportunistic scheduling and approximations: The log rule. *IEEE/ACM Transactions on Networking (TON)*, 19(2):405–418, 2011.

[99] Ahmed Saeed, Nandita Dukkipati, Vytautas Valancius, Carlo Contavalli, Amin Vahdat, et al. Carousel: Scalable traffic shaping at end hosts. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 404–417. ACM, 2017.

[100] Shamik Sarkar, Christopher Becker, Josh Kunz, Aarushi Sarbhai, Gurupragaash Annasamymani, Sneha Kumar Kasera, and Jacobus Van der Merwe. Enabling wifi in open access networks. In *Proceedings of the 4th ACM Workshop on Hot Topics in Wireless*, pages 13–17. ACM, 2017.

[101] Julius Schulz-Zander, Nadi Sarrar, and Stefan Schmid. Towards a scalable and near-sighted control plane architecture for wifi sdns. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 217–218. ACM, 2014.

[102] Julius Schulz-Zander, Nadi Sarrar, and Stefan Schmid. Aeroflux: A near-sighted controller architecture for software-defined wireless networks. In *Proc. Open Networking Summit (ONS)*, 2014.

[103] Julius Schulz-Zander, Lalith Suresh, Nadi Sarrar, Anja Feldmann, Thomas Hühn, and Ruben Merz. Programmatic orchestration of wifi networks. In *2014 USENIX Annual Technical Conference (USENIX ATC 14)*, pages 347–358. USENIX Association, 2014.

[104] Rob Sherwood, Glen Gibb, Kok-Kiong Yap, Guido Appenzeller, Martin Casado, Nick McKeown, and Guru M Parulkar. Can the production network be the testbed? In *OSDI*, volume 10, pages 1–6, 2010.

[105] M. Shreedhar and George Varghese. Efficient fair queueing using deficit round robin. *SIGCOMM Comput. Commun. Rev.*, 25(4):231–242, October 1995. ISSN 0146-4833. doi: 10.1145/217391.217453. URL `http://doi.acm.org/10.1145/217391.217453`.

[106] Gregory Smith, Anmol Chaturvedi, Arunesh Mishra, and Suman Banerjee. Wireless virtualization on commodity 802.11 hardware. In *Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization*, pages 75–82. ACM, 2007.

[107] IEEE Computer Society. IEEE Std 802.11-2012, 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, March 2012.

[108] Lalith Suresh, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, and Teresa Vazao. Towards programmable enterprise wlans with odin. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 115–120. ACM, 2012.

[109] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[110] Lun Tang, Xixi Yang, Xiaolin Wu, Taiping Cui, and Qianbin Chen. Queue stability-based virtual resource allocation for virtualized wireless networks with self-backhauls. *IEEE Access*, 6:13604–13616, 2018.

[111] Inc. The MathWorks. *MATLAB Simulink version 9.2 (R2018b).* The MathWorks, Inc., Natick, Massachusetts, United States, 2018. URL `https://www.mathworks.com/products/simulink.html`.

[112] Ilenia Tinnirello, Giuseppe Bianchi, Pierluigi Gallo, Domenico Garlisi, Francesco Giuliano, and Francesco Gringoli. Wireless mac processors: Programming mac protocols on commodity hardware. In *INFOCOM, 2012 Proceedings IEEE*, pages 1269–1277. IEEE, 2012.

[113] Wen Tong and Zhu Peiying. 5G: A thechnology vision. Technical report, Huawei Technologies Co., 2013. URL `http://www1.huawei.com/en/about-huawei/publications/winwin-magazine/hw-329304.htm`.

[114] Jonathan van de Belt, Hamed Ahmadi, and Linda E Doyle. A dynamic embedding algorithm for wireless network virtualization. In *Vehicular Technology Conference (VTC Fall), 2014 IEEE 80th*, pages 1–6. IEEE, 2014.

[115] Xin Wang, Prashant Krishnamurthy, and David Tipper. Wireless network virtualization. In *Computing, Networking and Communications (ICNC), 2013 International Conference on*, pages 818–822. IEEE, 2013.

[116] Heming Wen, Prabhat Kumar Tiwary, and Tho Le-Ngoc. *Wireless virtualization.* Springer, 2013.

[117] Heming Wen, Prabhat Kumar Tiwary, and Tho Le-Ngoc. Current trends and perspectives in wireless virtualization. In *Mobile and Wireless Networking (MoWNeT), 2013 International Conference on Selected Topics in*, pages 62–67. IEEE, 2013.

[118] Lei Xia, Sanjay Kumar, Xue Yang, Praveen Gopalakrishnan, York Liu, Sebastian Schoenberg, and Xingang Guo. Virtual wifi: Bring virtualization from wired to wireless. In *Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '11, pages 181–192, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0687-4. doi: 10.1145/1952682.1952706. URL `http://doi.acm.org/10.1145/1952682.1952706`.

[119] Mao Yang, Yong Li, Lieguang Zeng, Depeng Jin, and Li Su. Karnaugh-map like online embedding algorithm of wireless virtualization. In *Wireless Personal Multimedia Communications (WPMC), 2012 15th International Symposium on*, pages 594–598. IEEE, 2012.

[120] Mao Yang, Yong Li, Depeng Jin, Lieguang Zeng, Xin Wu, and Athanasios V Vasilakos. Software-defined and virtualized future mobile and wireless networks: A survey. *Mobile Networks and Applications*, 20(1):4–18, 2015.

[121] Yiannis Yiakoumis, Kok-Kiong Yap, Sachin Katti, Guru Parulkar, and Nick McKeown. Slicing home networks. In *Proceedings of the 2nd ACM SIGCOMM workshop on Home networks*, pages 1–6. ACM, 2011.

[122] Yasir Zaki. *Future Mobile Communications: LTE Optimization and Mobile Network Virtualization*. PhD thesis, Faculty of Physics and Electrical Engineering, University of Bremen, May 2012.

[123] Yasir Zaki, Liang Zhao, Carmelita Goerg, and Andreas Timm-Giel. Lte wireless virtualization and spectrum management. In *Wireless and Mobile Networking Conference (WMNC), 2010 Third Joint IFIP*, pages 1–6. IEEE, 2010.

[124] Yasir Zaki, Liang Zhao, Carmelita Goerg, and Andreas Timm-Giel. LTE mobile network virtualization. *Mobile Networks and Applications*, 16(4):424–432, Jun 2011. ISSN 1572-8153. doi: 10.1007/s11036-011-0321-7. URL `http://dx.doi.org/10.1007/s11036-011-0321-7`.

[125] ZhenBo Zhu, Parul Gupta, Qing Wang, Shivkumar Kalyanaraman, Yonghua Lin, Hubertus Franke, and Smruti Sarangi. Virtual base station pool: towards a wireless network cloud for radio access networks. In *Proceedings of the 8th ACM international conference on computing frontiers*, page 34. ACM, 2011.

Appendix A

Background on Wireless Transmission Technologies

Two important aspects of the wireless transmissions which are considered in this thesis are: the adaptation of modulation and coding (rate adaptation) and medium access control. In general, all wireless technologies include a mechanism to adapt the transmission rate to the channel conditions and also medium access control functions to decide when each of the devices sharing the wireless medium can transmit. Therefore, we explain the general concept of rate adaptation and we briefly explain the way IEEE 802.11 standard and the 3GPP LTE standard access the medium, to easily identify the derived problems from wireless slicing.

A.1 Introduction to Rate Adaptation

In wireless transmissions, there are particular modulation formats defined for the different data-rates allowed by a standard. A modulation format define a transformation of data bits to a sequence of symbols that are, then, mapped to signal waveforms that can be transmitted over an analog channel (such as the wireless channel). These symbols can be encoded varying the frequency, amplitude or phase of the waves.

Moreover, a group of K bits can be represented by a symbol and each symbol is mapped to a different waveform which gives us $M = 2^K$ waveforms. So, the data rate (in bits) is K times the transmitted symbol rate. Hence, if the symbol rate is constant, the amount of information a modulation format can transmit depends on the possible different values a symbol can take (these different M values a symbol can take is called a constellation). However, as the ways to encode a symbol to waveform variations are limited, when the number of values increases, the decodification becomes harder. It can be demonstrated that the minimum distance (in amplitude, phase or frequency) between two values in a constellation determines the signal-to-noise ratio (SNR) needed to avoid a bit error. In summary, constellations with many values need higher SNR to decode the symbol correctly.

Additionally, *channel coding* is used as a mechanism to reduce the bit error rate (BER). It consists in adding redundancy by coding blocks of K bits in code words of length $N > K$. The relation $\frac{K}{N}$ is called the *coding rate*. The pair formed by a specific modulation format and coding rate is called a Modulation and Coding Scheme. See for example [73], where it is provided a table of the different MCSs available in the IEEE 802.11n and IEEE 802.11ac standards.

This theoretical explanation gives an idea of how the adaptation of the MCS is beneficial in scenarios where the SNR is variable. Moreover, it can be seen that each modulation format has a SNR for which it is more efficient than the rest. A detailed and formal explanation of these issues can be found in [77, 63, 107].

A.2 Medium Access Control in WiFi

The IEEE 802.11 standard [107] defines four coordination functions (or methods to arbitrate the access to the medium): the *Distributed Coordination Function* (DCF), the *Point Coordination Function* (PCF), the *Hybrid Coordination Function* (HCF) (which uses two mechanisms EDCA and HCCA) and the *Mesh Coordination Function* (MCF). Most devices, when working in infrastructure mode (several clients all connected to an AP), use DCF or EDCA by default.

DCF uses *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) to regulate the access to the medium. In this access method a device must sense the medium (physical carrier sense) before starting to transmit. If the medium is not busy, the device is able to transmit. More specifically, a device must sense the medium is idle for a period of time (called DIFS) before transmitting. If the medium is busy (on a transmission attempt), the device waits for the current transmission to end. Then, before attempting to transmit again, the device waits for a random backoff time while the medium is idle and, then, it transmits its frame (see Figure A.1). This backoff time is selected randomly from the interval $[0, CW]$ (the Contention Window). CW is a variable parameter, which is duplicated every time the device tries to transmit, and can take values between the limits CW_{min} and CW_{max}.

In EDCA, some of the access medium control parameters are adaptable, so as to prioritize the access and to provide QoS. These parameters are:

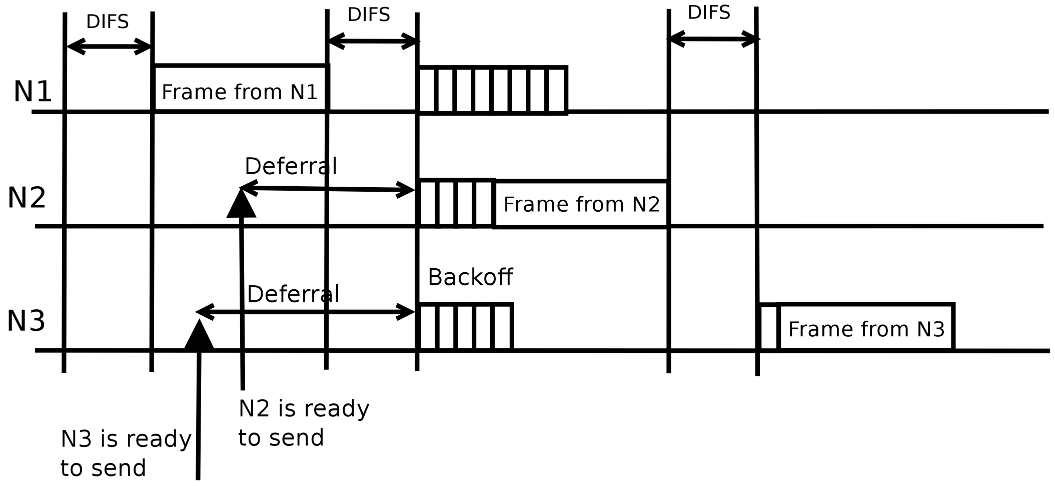

Fig. A.1 Distributed Coordination Function backoff procedure.

- The Arbitration Inter-Frame Spacing (AIFS). It defines the time between two frame transmissions.
- The backoff variables CW_{min} and CW_{max}.
- The Transmission Opportunity (TXOP) Limit, which defines the period of time a device can use the medium after it has gain access.

Then, for providing QoS, different Access Categories (ACs) are defined and each one has different configurations of these parameters so as to prioritize the access to the medium.

A.3 Medium Access Control in LTE

In LTE, medium access is performed by Orthogonal Frequency Division Multiple Access (OFDMA) on the downlink and by Single Carrier - Frequency Division Multiple Access (SC-FDMA) on the uplink. OFDMA is based on dividing the available bandwidth into orthogonal frequency sub-carriers, assigning a set of sub-carriers to each user. OFDMA uses sub-carriers distributed through the available spectrum while SC-FDMA can only use adjacent sub-carriers. The multiple access technique works this way: each time interval, called Transmission Time Interval (TTI), an assignment decision is made and each user is assigned a certain amount of radio blocks in the time and frequency domains. This task of assigning resources to users is called *scheduling*.

In the time domain, LTE supports two types of frames consisting in several TTIs: one for Frequency Division Duplex (FDD) mode and one for Time Division Duplex

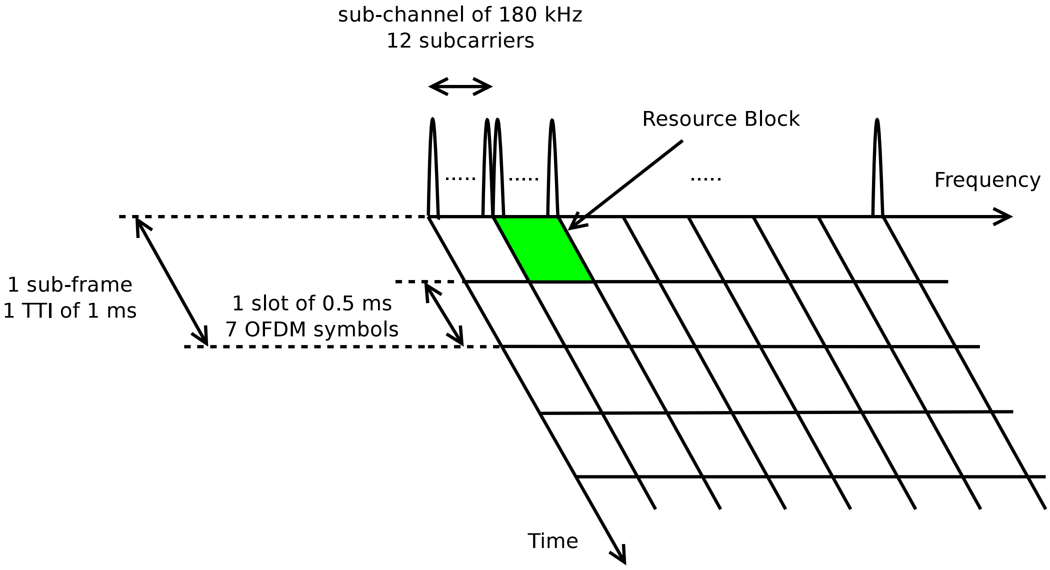

Fig. A.2 LTE time-frequency frame.

(TDD) mode. In FDD mode, the frame consists of 10 TTIs, which are divided in time slots of 0.5 ms and can carry 7 OFDM symbols (in the default configuration with short cyclic prefix). Two consecutive time slots define a sub-frame of 1 ms and, in general, the scheduling works on a sub-frame level. In TDD, which is basically designed for coexistence with legacy systems, the frame is divided into two half-frames of 5 ms.

In the frequency domain, the sub-carriers are grouped into sub-channels of 180 kHz (12 sub-carriers) and the number of sub-channels depends on the available bandwidth. The radio resource unit consisting on 1 time slot and 1 sub-channel is called a Resource Block (RB) or Physical Resource Block (PRB) and it is the basic resource unit of allocation (see Figure A.2).

In summary, the responsibility of the scheduler is to decide how to assign PRBs among users taking into account the channel conditions and QoS requirements. This complex task presents several challenges for which various scheduler designs and implementations have been proposed [21].

Appendix B

Mathematical Concepts

For the sake of completeness, in this Appendix we present some general mathematical results that are strongly used in Chapter 5 when applying the Lyapunov Optimization Theory.

Law of Telescoping Sums Given a function $f(t)$ defined over time slots $t \in \{0, 1, 2, ...\}$, it happens that:

$$\sum_{\tau=0}^{t-1}[f(\tau+1) - f(\tau)] = f(t) - f(0) \;\; \forall\, t > 0$$

Law of Iterated Expectations For any two random variables X and Y, we have that:

$$\mathbb{E}\{X\} = \mathbb{E}\{\mathbb{E}\{X \mid Y\}\}$$

Opportunistic Minimization of an Expectation Consider we have a function $f(\omega(t), \alpha(t))$ where $\omega(t)$ is a stochastic process and $\alpha(t)$ some action that may depend on ω and our objective is to minimize $\mathbb{E}\{f(\omega(t), \alpha(t))\}$. This result says that the policy which minimizes the previous expectation consists on: each time t, observe the value of $\omega(t)$ and choose the action $\alpha(t)$ which minimize $f(\alpha(t), \omega(t))$.

In [80] it is provided a simple proof: Consider $\alpha_\omega^{min}(t)$ as the action that given a value of $\omega(t)$ minimizes the function f. Now consider any other action $\alpha_\omega^*(t)$ taken by any other policy in relation to the observed $\omega(t)$. Hence, we have that: $f(\omega(t), \alpha_\omega^{min}(t)) \leq f(\omega(t), \alpha_\omega^*(t))$. Taking expectation on both sides gives $\mathbb{E}\{f(\omega(t), \alpha_\omega^{min}(t))\} \leq \mathbb{E}\{f(\omega(t),$

$\alpha^*_\omega(t))\}$. Therefore, the expectation under the control policy which selects the action that minimizes f for an observed $\omega(t)$ is less than or equal to the expectation under any other policy.

Appendix C

Proof of Bounded Delay

Hereafter we provide a simple and intuitive proof which shows that any algorithm which ensures that $Q_{n_s}(t) \leq Q_{n_s}^{max}$ $\forall t \geq 0$ and $Z_{n_s}(t) \leq Z_{n_s}^{max}$ $\forall t \geq 0$ guarantees that the delay of packets is bounded. Even more, the bound is given by $W_{n_s}^{max} = \left\lceil \frac{Q_{n_s}^{max}+Z_{n_s}^{max}}{\epsilon_{n_s}} \right\rceil$.

Proof. Lets observe that the worst case in possible delays for a packet appears when it arrives to an almost full Q_{n_s} queue. Lets assume that packet p arrives at time t, that no packet is dropped or transmitted on that time slot and that the arrival of p makes the queue full: $Q_{n_s}(t+1) = Q_{n_s}^{max}$. Therefore, packet p needs to wait until all previous packets are transmitted or dropped to leave the queue. It is easy to see that the worst case on delay for packet p is that in the following time slots, no packet is transmitted neither dropped from the queue.

On the other hand, we have the virtual queue $Z_{n_s}(t+1) = [Z_{n_s}(t) + \epsilon_{n_s} - R_{n_s}(t) - D_{n_s}(t)]^+$. In this queue, new traffic is always arriving at a rate of ϵ_{n_s} packets per slot. Therefore, when this queue is full, it enforces that a packet is transmitted or dropped. Therefore, lets assume the worst case for packet p, which is that $Z_{n_s}(t) = 0$. In this case, given that the queue gets data at a rate of ϵ_{n_s} and that the queue is bounded by $Z_{n_s}^{max}$, it will take $T_1 = \frac{Z_{n_s}^{max}}{\epsilon_{n_s}}$ to get full.

Hence, after time slot T_1, packets are transmitted or dropped at least by a rate of ϵ_{n_s} per slot so as to maintain the queue equal or below its bound. Therefore, it will take $T_2 = \frac{Q_{n_s}^{max}}{\epsilon_{n_s}}$ time slots for the packet p to be at the head of the queue. Then, the worst possible delay for any packet is given by

$$W_{n_s}^{max} = T_1 + T_2 = \frac{Q_{n_s}^{max} + Z_{n_s}^{max}}{\epsilon_{n_s}}$$

□

This is the last page.
Compiled on Monday 4$^{\text{th}}$ November, 2019.