Title: Network Slicing for IEEE 802.11 Wireless Networks
Student: Matteo Nerini

Problem description:

In recent years, the mobile traffic and the number of devices connected to the network grew exponentially. Furthermore, 5G is anticipated to pave the way for a myriad of use cases with diverging requirements and all these services will need to be simultaneously accommodate over a common infrastructure. Since the current generation of mobile networks is unable to satisfy different requirements in an optimal way, the introduction of the so-called network slicing paradigm is required. Network slicing consists in a virtual and physical division of network resources with customized functionality, in order to provide logical networks adjusted to the requirements of each use case, on top of a common network.

Due to this evolution, standard algorithms that analyse Key Performance Indicators (KPIs) and act on the system to keep them under control will no longer be effective in future networks. Instead, Machine Learning (ML) techniques will take place to deal with the complexity and the high number of variables characterizing the new communication networks. In this regard, the recommendation ITU-T Y.3172 (approved in June 2019) defines an architectural framework to implement ML in such future networks. The document defines the fundamental architectural components needed to integrate ML functionalities into future networks and guidelines for applying them.

This thesis wants to address the issue of managing and orchestrating resources in order to create separated network slices. To correctly realize slicing, orchestration and resources management processes should be automated and elastic: i.e. the configuration of network slices must take place on-demand without the need of manual intervention and the desired Service Level Agreement (SLA) must be assured even under varying network conditions. Thus, a ML algorithm for resource allocation will be developed and validated by means of simulations.

The main task of this thesis will be to simulate an IEEE 802.11 network in which network slicing is realized. In this model, to achieve the optimal slice configuration in the system, a ML algorithm, compliant to ITU-T Y.3172, interacts with the environment and manages available resources assigning them to isolated slices.

Date approved: 2020-02-04
Supervisor: David Palma, IIK

Abstract

Future networks will pave the way for a myriad of applications with different requirements. In such a context, the today's *one-size-fits-all* approach will not be able to efficiently address the different demands that verticals impose in terms of Quality of Service (QoS) and involved data volumes. To this end, network slicing is a new network paradigm which may provide the needed flexibility. It allows to offer multiple logical networks over a common infrastructure, tailored to the services which run on the network.

In today's Wi-Fi networks, all the users are connected to the same wireless channel, which allows service differentiation only at the traffic level. Thus, in this study we propose a standard-compliant network slicing approach for the radio access segment of Wi-Fi, often neglected by the literature on network slicing.

We present two algorithms to realize network slicing at the access level. The first assigns resources according to the requirements of the slices in a static way. On the other hand, the second, more advanced, dynamically configures the slices according to the network conditions and relevant KPIs. These techniques can be applied to the IEEE 802.11 standard and, in general, to all the protocols that use Carrier Sensing Multiple Access (CSMA) as channel access technique.

The proposed algorithms were validated through extensive simulations, conducted with *ns-3* network simulator and accompanied by theoretical calculations. Particular attention, often neglected in similar simulation-based works, has been paid to the electromagnetic properties of the spectrum, which play a fundamental role in radio communications.

From the conducted simulations, we found that our slicing approaches largely outperform the today's Wi-Fi access technique. They allow to reach higher goodput (i.e. a lower error probability) and lower latency, when needed. At the same time, tailored slicing saves energy to low-power devices and increases the spectrum efficiency.

Preface

This thesis has been written within the Double Degree program between University of Bologna and NTNU – Norwegian University of Science and Technology.

Thank you, David Palma, for the patient guidance and advice you have provided throughout this year at NTNU. I feel really lucky to have worked with you both for your excellent technical preparation and your extraordinary human qualities.

Contents

Appendices

List of Figures

List of Tables

List of Algorithms

List of Acronyms

3GPP 3rd Generation Partnership Project.

5GPPP 5G Public Private Partnership.

5QI 5G QoS Identifier.

ACK Acknowledgement.

AP Access Point.

API Application Program Interface.

AWGN Additive White Gaussian Noise.

BSS Basic Service Set.

BSSID Basic Service Set Identifier.

CCA Clear Channel Assessment.

CSMA Carrier Sensing Multiple Access.

CSMA/CA Carrier Sensing Multiple Access with Collision Avoidance.

CTS Clear to Send.

DCF Distributed Coordination Function.

DIFS DCF Inter-Frame Spacing.

DL Deep Learning.

DSSS Direct Sequence Spread Spectrum.

E2E End-to-End.

EDCA Enhanced Distributed Channel Access.

EIRP Equivalent Isotropically Radiated Power.

EM Element Management.

eMBB Enhanced Mobile Broadband.

EPC Evolved Packet Core.

ETSI European Telecommunications Standards Institute.

FCS Frame Check Sequence.

GI Guard Interval.

GUI Graphical User Interface.

IETF Internet Engineering Task Force.

InP Infrastructure Provider.

IoT Internet of Things.

ISM Industrial Medical Scientific.

ITU International Telecommunication Union.

KPI Key Performance Indicator.

LAN Local Area Network.

LoS Line-of-Sight.

MAC Medium Access Control.

MANO Management and Orchestration.

MCS Modulation and Coding Scheme.

MIMO Multiple-Input Multiple-Output.

ML Machine Learning.

mMTC Massive Machine-Type Communications.

MPDU MAC Protocol Data Unit.

MSDU MAC Service Data Unit.

MU-MIMO Multi-User MIMO.

NF Network Function.

NFV Network Function Virtualization.

NFVI Network Function Virtualization Infrastructure.

NGMN Next Generation Mobile Networks.

NSI Network Slice Instance.

OFDM Orthogonal Frequency Division Multiplexing.

OFDMA Orthogonal Frequency Division Multiple Access.

ONF Open Network Foundation.

OS Operating System.

PHY Physical.

PSD Power Spectrum Density.

QoS Quality of Service.

RAN Radio Access Network.

RL Reinforcement Learning.

RTS Request to Send.

RU Resource Unit.

SDM Spatial Division Multiplexing.

SDMA Spatial Division Multiple Access.

SDN Software Defined Networking.

SLA Service Level Agreement.

SLO Service Level Objective.

SNR Signal-to-Noise Ratio.

SSID Service Set Identifier.

STA Station.

TCP Transmission Control Protocol.

TPC Transmit Power Control.

UDP User Datagram Protocol.

UE User Equipment.

URLLC Ultra-Reliable Low-Latency Communications.

VM Virtual Machine.

VNF Virtual Network Function.

WDM Wavelength Division Multiplexing.

WSN Wireless Sensor Network.

Chapter 1 Introduction

In this introductory chapter we clarify the objective of this study and the relative motivations. The adjustments made to the problem description, which was approved in February 2020, are discussed and justified.

1.1 Motivations

In the current generation of mobile networks, vertical industries adapt their connectivity requirements to fit the so called *one-size-fits-all* mobile network. This does not offer much flexibility and, thereby, it is unable to address the dissimilar demands in terms of performance and dependability that different verticals impose.

5G is anticipated to pave the way for a myriad of use cases with diverging requirements. According to the International Telecommunication Union (ITU) in [IR15], all the possible applications that the future generation of mobile networks will support can be organized, depending on their characteristics, in a triangular structure as shown in Figure 1.1.

Three main classes of services can be identified:

- **Enhanced Mobile Broadband (eMBB)**: necessitates support for very high data rates and high data traffic volumes. The main issues taken into account are high user mobility, security and coverage.

- **Massive Machine-Type Communications (mMTC)**: a large number of devices transmit low volumes of data that are not sensitive to delay and do not involve mobility.

- **Ultra-Reliable Low-Latency Communications (URLLC)**: class of services requiring ultra-low latency, high reliability and availability.

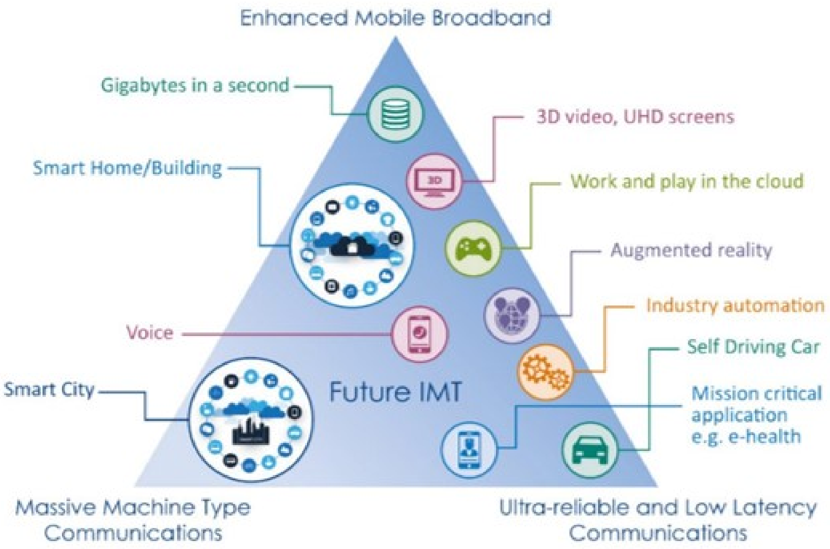

Enhanced Mobile Broadband
Gigabytes in a second
3D video, UHD screens
Smart Home/Building
Work and play in the cloud
Augmented reality
Industry automation
Voice
Self Driving Car
Smart City
Future IMT
Mission critical application e.g. e-health
Massive Machine Type Communications
Ultra-reliable and Low Latency Communications

Figure 1.1: 5G usage scenario [IR15].

All these service categories need to be simultaneously accommodated over a common infrastructure. Since the current generation of mobile networks is unable to satisfy different requirements in an optimal way, a new paradigm offering the needed programmability, flexibility and modularity is required. The solution to this issue, can be provided by network slicing, able to serve users allocating different types of resources according to their needs.

Given the high complexity of future networks, a possible solution could be provided by ML approaches. The architectural framework for implementing ML in future networks is provided by ITU-T [IT19].

1.2 Problem Scope and Methodology

The entire thesis consists in a single iteration through the so-called design cycle, represented in Figure 1.2 [Wie14]. The phase of treatment implementation aims to evaluate the designed artifact in its real context and, for this reason, it is out of the scope of this research study. At the end of this cycle, the resulting artifact is an approach to implement network slicing in the radio access segment of a Wi-Fi network.

According to Wieriga, each step in the Design Cycle can open one or more empirical cycles, reported in Figure 1.3 [Wie14]. The objective of an empirical cycle

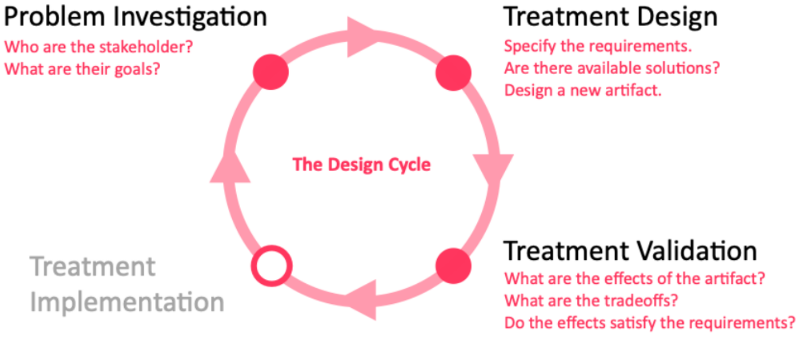


Figure 1.2: Design cycle adapted from [Wie14].

is in general to address a research question. More precisely, it can investigate the problem context, answer knowledge questions or conduct a validation. The research questions that we addressed in this study are summarized in Figure 1.4.

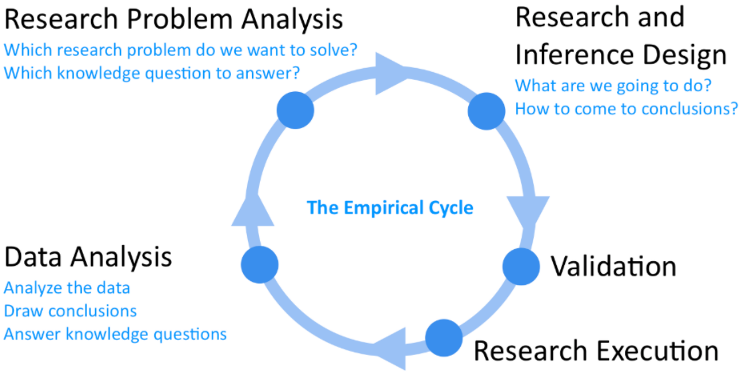

Research Problem Analysis
Which research problem do we want to solve?
Which knowledge question to answer?
Research and
Inference Design
What are we going to do?
How to come to conclusions?
The Empirical Cycle
Validation
Data Analysis
Analyze the data
Draw conclusions
Answer knowledge questions
Research Execution

Figure 1.3: Empirical cycle adapted from [Wie14].

The initial goal of our study was to realize a framework, compliant with the ITU-T Y.3172 model ([IT19]), in which a ML algorithm assigns dynamically resources to realize slicing.

However, a deep literature review revealed that the process of defining network slicing has just begun, and only a few concrete approaches have been proposed for Wi-Fi networks. Additionally, ML approaches must be employed when a well defined model does not exist or it is practically infeasible. Thus, this empirical cycle performed in the problem investigation phase, allowed us to refine the design problem. It proved necessary to study how network slicing can be implemented using a traditional algorithmic method, because considering ML was premature.

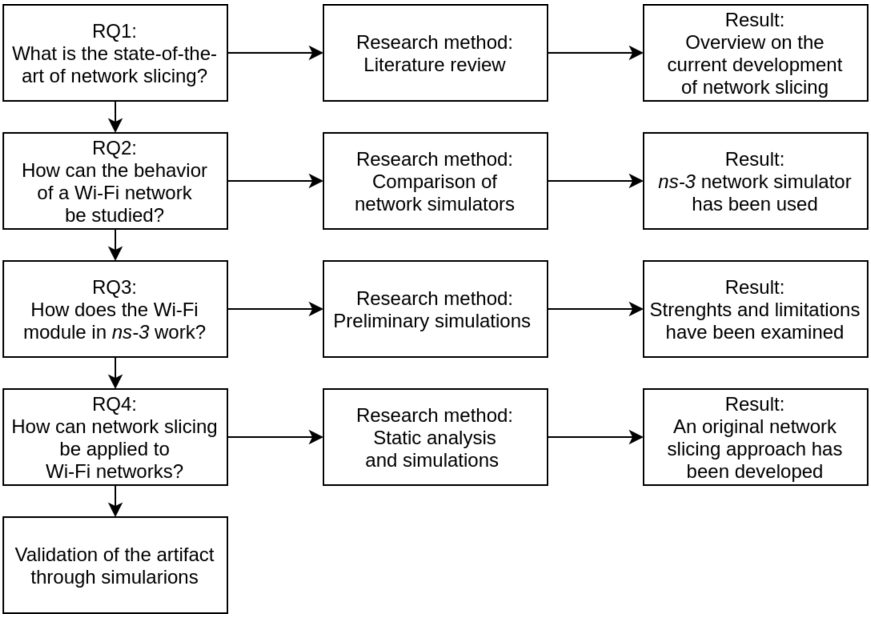

RQ1:
What is the state-of-the-art of network slicing?
Research method:
Literature review
Result:
Overview on the current development of network slicing
RQ2:
How can the behavior of a Wi-Fi network be studied?
Research method:
Comparison of network simulators
Result:
ns-3 network simulator has been used
RQ3:
How does the Wi-Fi module in *ns-3* work?
Research method:
Preliminary simulations
Result:
Strenghts and limitations have been examined
RQ4:
How can network slicing be applied to Wi-Fi networks?
Research method:
Static analysis and simulations
Result:
An original network slicing approach has been developed
Validation of the artifact through simularions

Figure 1.4: Research questions in our design cycle, with relative research method and result.

We decided to focus on the radio access segment of Wi-Fi networks, i.e. based on the IEEE 802.11 standard. This communication protocol has been selected, instead of 5G, because there is still lack of literature on network slicing applied to Wi-Fi networks. Furthermore, the available 5G millimeter waves simulators are still today in an early phase which does not allow to conduct realistic experiments. The *ns-3* simulator has been chosen to carry out our study after a careful comparison of the available networks simulators.

1.3 Contributions

This report has multiple contributions. They are the results of the all research questions encountered, and are summarized in the following list:

- A detailed study on the state-of-the-art of network slicing has been performed. The current development process carried out by standardization bodies is described and particular attention is paid to the challenges opened by this new networking paradigm.

- A comparison on the leading network simulators capabilities has been conducted with the aim of finding the most suitable simulator to use in our study.

- The selected network simulator (*ns-3*), has been deeply analyzed. After a profound examination of the *ns-3* Wi-Fi module, we report important properties and limitations which are often omitted in the official documentation and research works.
- A realistic and complete Wi-Fi scenario has been implemented in *ns-3*. Our network is fully customizable and adaptable. It can be used to evaluate resource allocation algorithms for network slicing and also to train and test ML based slicing techniques. It is possible to read KPIs at run-time, and act on the network by dynamically allocating radio resources and changing network parameters. This implementation can be used as a sandbox network in the ITU-T ML architectural framework[1].
- Finally, the main contribution consists in a network slicing proposal for the Wi-Fi radio segment. Our solution, extendable to every kind of CSMA communications, has been deeply evaluated through simulations and compared with the today's approach where network slicing is not implemented.

1.4 Outline

The report is organized as follow. The first two chapters cover the background concepts needed in our work: network slicing and IEEE 802.11 standard respectively.

Chapter 4 contains a comparison of the available network simulators and in Chapter 5 the main characteristics of the selected simulator, *ns-3*, are analysed through preliminary simulations.

In chapter 6 we describe the Wi-Fi scenario which we want to study, and our proposed slicing solutions are presented in Chapter 7.

Finally, the numerical results and the main findings are analysed in Chapter 8, while possible future developments are discussed in the concluding chapter.

[1] According to ITU-T Y.3172, a sandbox is a network in which ML applications can be trained and tested. In this way, their effects on the underlay network can be evaluated. A sandbox is needed to prevent a ML model from affecting the underlay network with unexpected actions.

Chapter 2 Network Slicing

This chapter provides background information on the network slicing paradigm. We illustrate the standardization work that has been done to date and the challenges that this technology is opening. The study on the state of the art of network slicing presented in this chapter has been carried out by the author in parallel with the course TTM4517 – Advanced Topics in Networking [Ner19].

2.1 An Introduction on Network Slicing

The concept of slice appeared in networking since the late ’80s, and it was introduced in the 5G context for the first time in 2015 by Next Generation Mobile Networks (NGMN) in the “5G White Paper” [NGM15]. Network slicing consists in a virtual and physical division of network resources with customized functionality, in order to provide logical networks adjusted to the requirements of each use case, on top of a common network.

In [3GP16], 3rd Generation Partnership Project (3GPP) defines network slicing as the technology that enables the operator to create networks, tailored to provide optimal solutions for different market scenarios which demand diverse requirements, e.g. in terms of functionality, security and QoS.

ITU-T provides another definition, stressing the importance of separation between slices, and considers the slices as logical isolated network partitions composed of multiple virtual resources, isolated and equipped with programmable control and data plane [IT12].

Even if specific names may vary, standardization bodies agree that the network slicing concept consists of three layers: Service Instance Layer, Network Slice Instance Layer, and Resource Layer. This model is represented in Figure 2.1. The Service Instance Layer contains the end-user services for which the network slices are created. Such services can be provided by a vertical segment, by an application provider or directly by a mobile network operator. The Network Slice Instance Layer can be seen as a library of Virtual Network Functions (VNFs), and resources to connect and run them on. A Network Slice Instance (NSI) provides the network characteristics which are required by one or more service instances. An NSI may be composed by none, one or more sub-network instances, which may be shared by another NSI. Finally, the Resource Layer represents the physical infrastructure of one ore more operators. Based on this 3-layered model, high level network slicing architectures are proposed by NGMN in [NGM16] and by Internet Engineering Task Force (IETF) in [IET17].

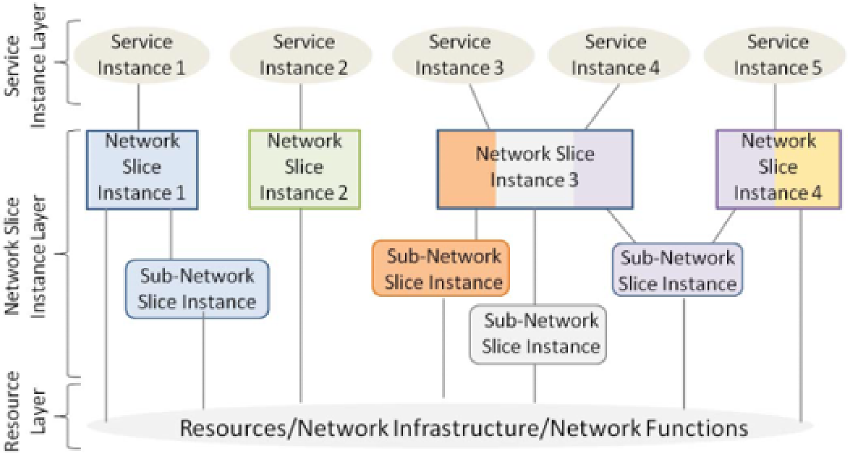


Figure 2.1: 3-layered model for network slicing [ATS$^+$18].

2.1.1 Slicing Principles

In order to better delineate the concept of network slicing, five main characteristic elements and principles are described in the following paragraphs.

Resources

In a network slicing environment, resources are units, defined by a set of attributes and capabilities, that can be used to deliver a specific service. Two types of resources can be identified:

- Network Functions (NFs): functional blocks that provide certain network capabilities;

- Infrastructure resources: radio resources, hardware and necessary software for hosting and connecting network functions.

Virtualization

A crucial aspect related to the previously defined resources is virtualization. Thanks to this property, resources can be abstracted, i.e. represented in terms of only relevant aspects. Virtualization enables the creation of multiple virtual networks that are completely decoupled from the underlying physical network. The introduction of virtualization in this framework defines new business model with three kinds of actors playing roles at different levels:

- Infrastructure Provider (InP): owns and controls a physical network.
- Tenant: leases virtualized resources from one or more InPs to create customized slices.
- End-user: consumes services supplied by a tenant.

A graphical representation of the interactions between these actors is given in Figure 2.2. In this scenario, virtualization allows recursion, also known as hierarchical abstraction: resources allocated to a particular tenant, can be further leased (either partially or fully) by another third player. This procedure facilitates the creation of another network slice service on top of the previous one.

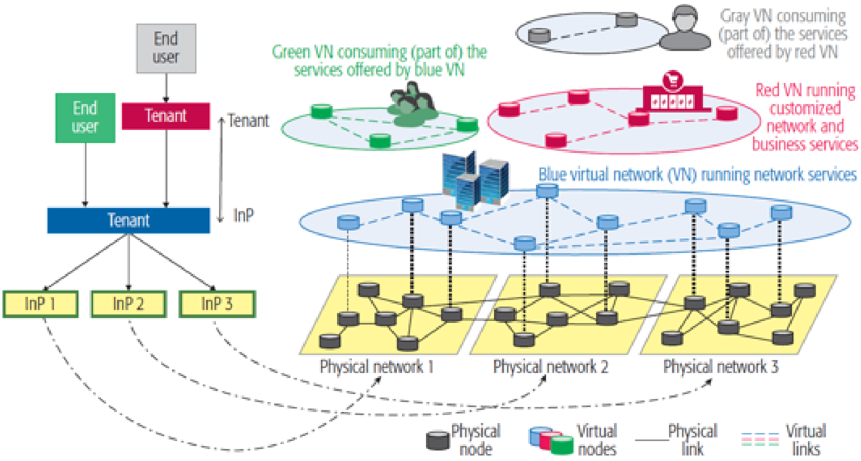

End user
End user
Tenant
Tenant
InP
Tenant
InP 1
InP 2
InP 3
Green VN consuming (part of) the services offered by blue VN
Gray VN consuming (part of) the services offered by red VN
Red VN running customized network and business services
Blue virtual network (VN) running network services
Physical network 1
Physical network 2
Physical network 3
Physical node
Virtual nodes
Physical link
Virtual links

Figure 2.2: Network slicing actors in a multi-layered structure [OLAL$^+$17].

Orchestration

Orchestration is the continuing process of selecting resources to fulfil client service demands in an optimal manner. This process is performed by dedicated entities that will be described later. To correctly realize slicing, orchestration and resources management processes should be automated and elastic: i.e. the configuration of network slices must take place on-demand without the need of manual intervention and the desired SLA must be assured even under varying network conditions.

Isolation

Isolation is a requirement that must be satisfied to operate parallel slices on a common shared underlying substrate. The isolation must be understood in terms of:

- Dependability and performance: specific requirements on dependability and performance must always be met on each slice, regardless of the failures and performance levels on fellow slices.
- Security and privacy: attacks or security failures in one slice must not have an impact on other slices.
- Management: each slice must be independently managed as a separate network.

In other words, network slicing requires that each slice is not affected by activities within other slices. Isolation is a decisive aspect since if this property is not properly met, all efforts made to create slices will be in vain.

End-to-End

To facilitate a service delivery all the way from the service providers to the end-users, a network slice must be End-to-End (E2E), i.e. it stretches across different administrative domains, unifies various network layers and heterogeneous technologies. In the following, the consequences of this aspect will be analysed.

2.1.2 Slicing Enabling Technologies

Virtualization and softwarization technologies are the key enablers for network slicing since they can provide the required flexibility and modularity. First of all, a layer between the physical infrastructure and the Operating System (OS) running on the top is needed for producing, controlling and managing Virtual Machines (VMs). Such system, called hypervisor, enables and supervises the sharing of hardware resources between NSIs.

VMs and containers are capable of running VNFs, providing the effect of a physical resource that runs its own OS. Each VM shares resources while its operation is completely isolated from that of the host and fellow guest VMs.

Software Defined Networking (SDN) transforms network management, introducing programmability and decoupling the control plane from the data plane. In addition, Network Function Virtualization (NFV) allows the deployment of network functions on virtual environments, leveraging the cost efficiency the benefits of cloud computing instead of having custom hardware for each network function. European Telecommunications Standards Institute (ETSI) analyses how network slicing use cases can be addressed using jointly SDN and NFV technologies in the dedicated report [ETS17].

Finally, cloud computing can offer storage, computational, and networking facilities needed to enable slicing, and, in particular, edge computing offers computing applications in close proximity to end-users, assuring ultra-low latency and high data rates.

The following sections are organized as follows: Section 2.2 gives a detailed description of the network slicing proposed architectures and methodologies; in Section 2.3, the issues raised by the implementation of a sliced environment are treated; Section 2.4 presents isolation with further detail; and finally, in Section 2.5, the future research perspectives for network slicing are presented.

2.2 Slicing Implementation

After having defined the general principles of network slicing, in this section, the main proposed architectures to realize slicing are analysed. Finally, we focus on how to enable slicing in the different segments of the whole mobile network: in particular, in the Radio Access Network (RAN) and in the core network.

2.2.1 SDN Network Slicing Architecture

As already mentioned, SDN is an appropriate technology to support slicing. The SDN architecture recommended by Open Network Foundation (ONF) is presented in [ONF16b, ONF16a], with particular attention on how it can be applied to enable slicing in 5G context. Such architecture is shown in Figure 2.3.

The major SDN architectural components are resources and controllers. As seen earlier, resources are utilized to provide services in response to client requests. To allow this to happen, a controller consists in a control plane entity, which manages SDN resources at run-time to deliver services in an optimized manner. It mediates between clients and resources, acting simultaneously as server and client via client and server contexts, respectively. The client context contains all the information the controller needs to support and communicate with a given client while, on the other side, the server context represents the information needed to the controller to interact

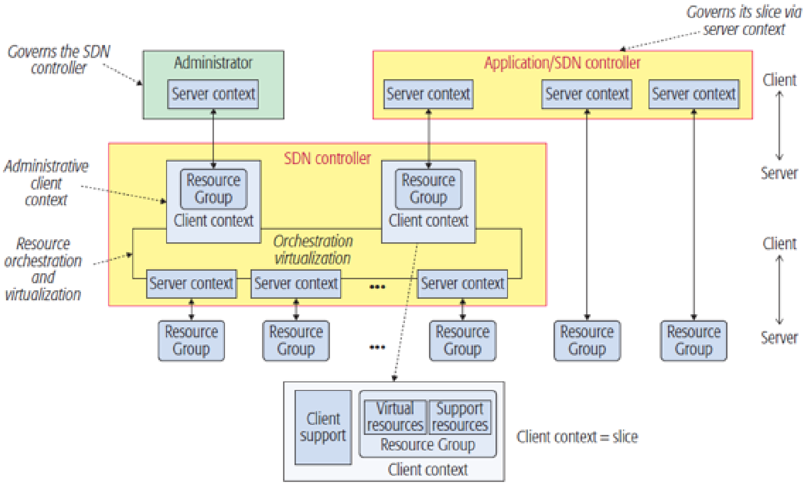


Figure 2.3: SDN network slicing architecture proposed by ONF [ONF16a].

with a set of underlying resources. The resource groups accessed through server contexts have to be transformed in those defined in separate client contexts. To this end, every SDN controller performs two operations: virtualization and orchestration. Finally, an administrator is needed to instantiate and configure the entire controller (i.e. define the contexts and their policies). One remarkable aspect, which makes this SDN framework suitable for slicing, is recursion. In fact, thanks to resources abstraction provided by client contexts, a controller can access resources acting as a client over another controller, and deliver new scaled resources (and hence new slices) to its clients.

2.2.2 NFV Architectural Framework

A NFV architectural framework must be employed to separate network functions from the hardware they run on. Furthermore, cooperation between SDN and NFV is required in order to take advantage of the management and orchestration capabilities of NFV. A possible realization of such framework, comprising the two technologies, is proposed by ETSI in [ETS15], and it is represented in Figure 2.4.

The ETSI NFV functional architectural infrastructure is described in [ETS14] and it includes the following entities:

- VNFs: software implementations of network functions. They run over the infrastructure and can be connected to form slices.

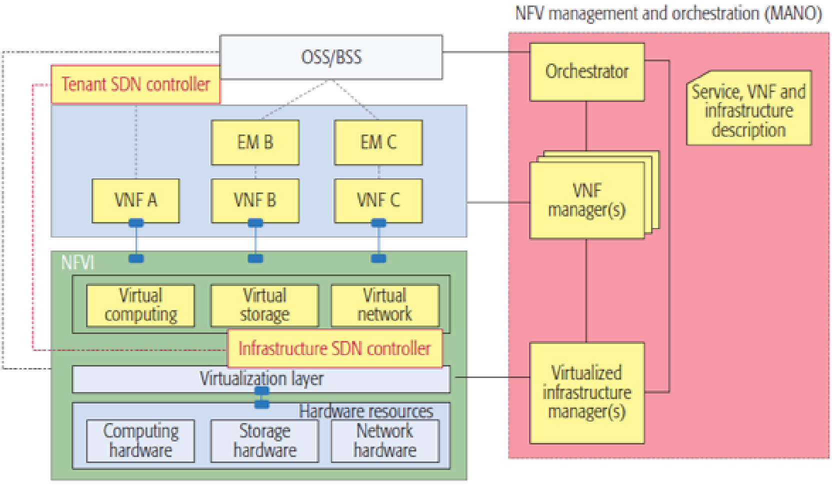


Figure 2.4: NFV architectural framework proposed by ETSI [ETS15].

- Element Management (EM): performs the typical management functionalities for one or several VNFs.

- Network Function Virtualization Infrastructure (NFVI): consists of the totality of hardware and software components which build up the environment in which VNFs are deployed, managed and executed.

- Management and Orchestration (MANO): performs all tasks related to the management, coordination, and automation of VNFs. This entity is composed of:

  - Virtualized Infrastructure Manager: responsible for controlling and managing the NFVI resources.
  - VNF Manager: responsible for configuration and life cycle management of the VNFs.
  - Orchestrator: is in charge of managing the network connectivity topology and the forwarding paths leveraging the fact that it has a broad view of the network.

The ETSI proposal includes two SDN controllers that act on the architecture. The Infrastructure SDN Controller configures and controls the underlying networking resources and provides the required connectivity to communicate with the VNFs. It is not aware of the number of slices that utilize the VNFs it connects, nor the tenants which operate such slices.

On the other hand, from a higher perspective, the Tenant SDN Controller dynamically manages the relevant VNFs used to realize the tenant's network services. It sees the network as abstracted in terms of VNFs, without notions of how those VNFs are physically deployed.

2.2.3 Life Cycle Management

On top of the described framework, the process of creating and utilizing slices is not straightforward. 3GPP defined a precise life cycle management for NSIs to enable scalability in providing network slices independently from service instances [3GP18]. Such life cycle management, represented in Figure 2.5, is administrated by the network operator, and it is decoupled from the corresponding on-the-top service instances, which use it. The following four phases can be identified in a NSI life cycle:

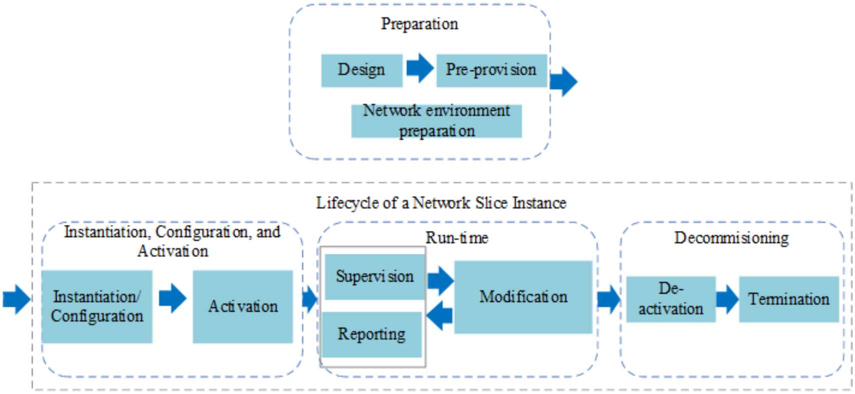

Preparation
Design
Pre-provision
Network environment preparation
Lifecycle of a Network Slice Instance
Instantiation, Configuration, and Activation
Instantiation/ Configuration
Activation
Run-time
Supervision
Reporting
Modification
Decommisioning
De-activation
Termination

Figure 2.5: NSI life cycle [3GP18].

1. **Preparation**. The NSI does not exist in this phase. The preparation phase includes the creation of the network slice template, and the preparation of necessary network environment to support the NSI.

2. **Instantiation, Configuration and Activation**. During instantiation and configuration, resources are created or configured if already existing (in case they are shared). The activation actions make the NSI active diverting traffic to it and provisioning databases. This phase can include instantiation and configuration of other shared network functions.

3. **Run-time**. Now, the NSI is capable of traffic handling and can support a certain type of communication services. To keep the slice working as designed, a closed-loop process analyses continuously the service performance (e.g. through

KPI monitoring) to assure that the desired requirements are met. To this end, the run-time phase includes supervision and reporting, as well as modification activities. Modification step comprises run-time tasks as reconfiguration, NSI scaling, changes of NSI capacity, changes of NSI topology, association and disassociation of network functions with NSI.

4. **Decommissioning**. This last phase includes deactivation (the NSI is taken out of active duty) as well as the reclamation of dedicated resources (termination or re-use of network functions) and reconfiguration of shared resources. After decommissioning the NSI does not exist anymore.

2.2.4 Impact on the Network

Since a network slice is an E2E service, each part of the infrastructure should support it, as pointed in [SBT$^+$17]. The main adjustments that must be implemented in the various network domains are now discussed.

User Equipment

Information on the available slices must be provided to the User Equipments (UEs). Furthermore, a mechanism is needed to allow the end-user devices to request a connection with one or more NSIs.

Radio Access Network

In the RAN domain, to improve the efficiency of resource usage, an on-demand dynamic resource management is required. In traditional static RAN sharing methods, in fact, resources are reserved whether they are used or not, with consequent lowering of the resource utilization. Critical applications may have stringent requirement on spectrum isolation but an hard isolation could be a bottleneck for the whole network capacity. Thus, a fair balance between these two aspects must be considered.

The virtualization and programmability aspects have enabled the advent of Software Defined RAN: in this paradigm, the abstraction of the underlying RAN resources facilitates the opening of Application Program Interfaces (APIs) towards third parties. Furthermore, the notion of base station softwarization, allows certain RAN functions to run at remote cloud platforms. In Cloud-RAN, functions are split between two entities: the Base Band Unit, hosted in the cloud, and the Remote Radio Headers that provide antenna equipment and radio access. In Figure 2.6, the stack of functions performed in the RAN domain is shown, with some possible splitting options. Options 6,7 and 8 provide the highest centralization but they are achievable only with ideal fronthauls (e.g. broadband optical fibers).

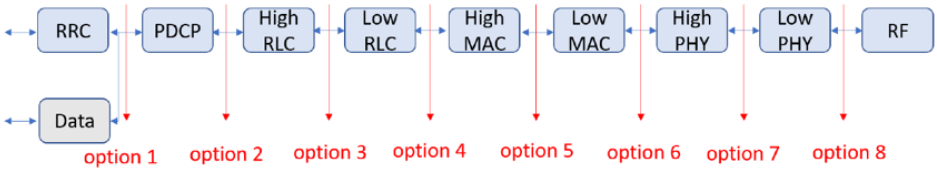


Figure 2.6: C-RAN functional splitting [ATS$^+$18].

Core Network

The core network has undergone significant changes in recent years. 3GPP has reshaped completely this domain by defining a more modular architecture, wherein the Evolved Packet Core (EPC) entities have been divided into fine-granular network functions. NFV and SDN technologies must be used to exploit commodity hardware resources for network slicing. They allow the deployment and the orchestration of multiple instances of virtual EPC running in parallel over a common infrastructure in order to fulfil different service demands.

2.3 Specific Issues

Because of the great complexity brought by network slicing, standardization bodies and recent works have pointed out several issues to be studied in order to provide and manage a properly isolated sliced environment for future networks, as reported in [GM18]. One of the first documents identifying these key problems is the 3GPP Technical Report [3GP16]. As described in the previous Section, NFV and SDN technologies can offer the needed modularity and softwarization to combine network functions and build tailored slices fulfilling given performance and dependability requirements. However, while «*what*» these architectures have to execute is becoming clear, there is still work to be done on «*how*» to perform those specific tasks. Table 2.1 summarizes the main issues that must be addressed regarding 5G network slicing according to the most recent literature.

All network slicing issues can be grouped in three classes. Firstly, when a slice is created, many difficulties arise due to the E2E nature of slices. As shown in Figure 2.7, proper resources have to be allocated in every network domain, and suitable interfaces have to be configured to perform slice chaining between them.

Once a slice has been established, other issues regard its usage: both from the client as well as from the service provider point of view. Finally, separation between slices need to be handled to achieve the required level of isolation. The aforementioned 3GPP Technical Report, after identifying the most important key issues concerning a sliced scenario, presents general solutions and frameworks to deal

Slices creation and management
- How to decide which network functions may be included in a specific NSI according to its performance and dependability requirements? - How to use resources and network functions sharing between NSI? - How to manage different trust domains of VNFs and virtualized or non-virtualized infrastructures? - How to provide an appropriately secure infrastructure? - How to realize slice chaining to enable E2E slices?
Slices exploitation
- How to enable an UE to discover, select, and access the most appropriate network slice? - How to enable an UE to simultaneously obtain services from one or more specific NSIs of one operator? - How to enable operators to efficiently support multiple third parties that require similar network characteristics?
Slices separation
- How to achieve performance isolation between NSIs? - How to achieve dependability isolation between NSIs? - How to achieve security isolation between NSIs? - How to define levels and types of isolation/separation?

Table 2.1: Current Network slicing issues.

with them. Nevertheless, the integration of all the potential proposals in order to achieve a full sliced environment is still an open challenge that needs to be addressed. In the following subsections, the issues of satisfying performance and dependability requirements in a network slicing system is presented and discussed.

2.3.1 Performance in a Shared Infrastructure

The main benefit brought by network slicing consists in the deployment of use cases with differing performance demands on the same underlying physical communication infrastructure. If a tenant only assigns dedicated resources to network slices, their required performance levels are always met at the cost of preventing slices sharing resources. This leads to over provisioning, an undesired situation bearing in mind that the tenant has a finite set of assigned resources. This issue can be resolved permitting resource sharing, although this means that slices might not be completely decoupled in terms of performance. Thus, it is required to design adequate resource

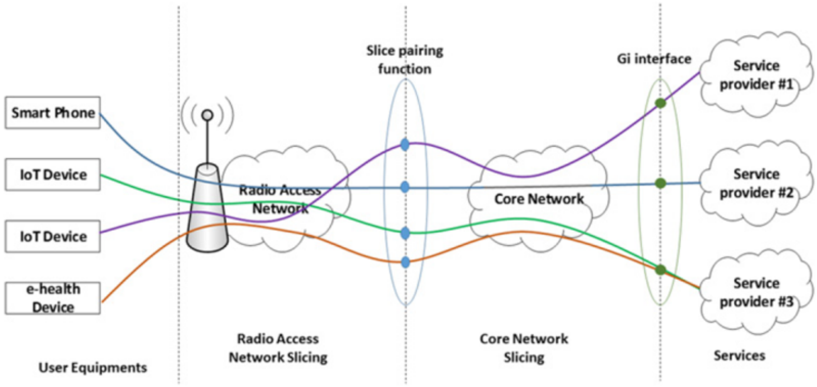


Figure 2.7: Slice chaining between different domains [KNS+18].

management mechanisms that enable resource sharing among slices when necessary without violating their required performance levels. To accomplish the sharing issue, Virtual Local Area Networks (LANs) are mostly used as means for traffic separation. Additionally, an original approach based on NFV and SDN driven queuing strategies is presented by Kurtz et al., in [KBDW18]. This novel proposal utilizes Hierarchical Token Buket mechanisms. In its general sense, a token bucket algorithm checks that data transmissions, in the form of packets, conform to defined limits on bandwidth and burstiness. It can also be used as a scheduling algorithm to determine a transmission timing compliant with the imposed limits. Therefore, this policy, in its hierarchical version to reflect the recursiveness of the infrastructure, can be exploited to assure performance requirement in a shared environment.

2.3.2 Dependability in a Slicing Architecture

Dependability is a crucial feature for 5G networks. In fact, in the 5G Public Private Partnership (5GPPP) vision, the requirement on the system availability must be of 99.999%. Detailed considerations for the design of dependability policies can be found in [GXG18]. In this paper, the authors present solutions to guarantee network dependability in the various network domains.

In the design of a robust wireless access network, the first fundamental aspect is the natural limitation posed on the maximum number of users attached to one antenna, and the data rates offered to each of them. These two variables should not be very high in order to provide ultra-reliable communications. Additionally, regarding the coverage, multi-homing techniques should be added to the traditional handover processes. Finally, cognitive radio has been proposed to implement efficient prevention and recovery mechanisms.

In order to automatically react to failures and to redirect the traffic from the failed paths to alternative ones, recovery mechanisms make use of redundancy in the network topology. In transport networks, at least two disjoint paths between any two nodes should be designed, and enough spare capacity to be used when needed should be reserved. The backup resources can be planned in advance or determined after the failure, and can be dedicated or shared among connections (e.g. tenants) depending on the implemented protection and restoration mechanisms. Finally, redundancy and fault tolerance features should be applied also to each VNF composing the core network.

In addition to what has been said about the dependability of network resources, the management and orchestration platform deserves special attention. It is required to regularly monitor it for detecting abnormal conditions and act with appropriate countermeasures. In some cases, an error can trigger more related alarms, due to the hierarchical nature of layers. Thus, an adequate isolation and segmentation of the monitoring tools must be implemented. Finally, it is important also to elaborate mechanisms that verify the correctness of the execution of the management and orchestration actions.

2.4 Isolation

Isolation is a decisive requirement that has to be met in a sliced environment. Nowadays, network slicing architectural aspects have been already discussed even in technical reports by standardization bodies, whilst the same is not true for the isolation aspect. It is well known that different slices should act as separated networks but how this property could be modelled and measured is still object of research.

2.4.1 Definition and Measurement of Isolation

Due to high diversity of network infrastructures, the mechanisms and techniques used to provide isolation are going to be different in each case. However, to guide the proper design for network slices isolation, a normalized approach should be developed.

Isolation in Network Domains

According to [GSSF12], the problem of isolation along an entire E2E slice should be solved in different ways in the various part of the mobile network: radio access, transport or datacenter (see Figure 2.8).

Depending on which network domain is considered, three kinds of isolation can be identified:

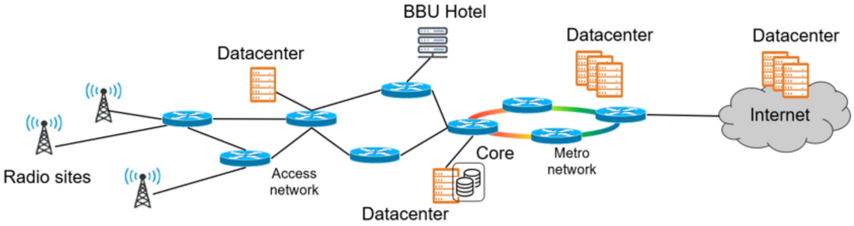


Figure 2.8: Radio access, transport and datacenter domains.

- Radio resource isolation. In the radio domain, all slices allocate some radio resources (e.g. bandwidth) and should not utilize any radio resource assigned to other slices.

- Traffic isolation. Network slicing should ensure that data flow of one slice does not move to another. This type of isolation is referred to the transport domain and it must be granted even if the underlying physical network (i.e. cables and optical fibers) is shared among slices.

- Control isolation. In the datacenter domain it may happen that all virtual slices use the same physical resources. Despite this, a proper processing of packet is required for each slice, which will be independent of all other slices. Furthermore, data related to a slice should be stored separately from data used by another slice.

Isolation Dimensions

Regardless the network structure and the E2E nature of slices, isolation can be seen under three dimensions, as clearly pointed in [GOLH$^+$19].

- Performance dimension. Isolation under performance point of view assures that the QoS on each slice is reached regardless of the traffic level on the others. In practice, performance can be quantified in terms of blocking probability, bit rate, latency, packet loss and jitter.

- Dependability dimension. Dependability of one slice can be measured in terms of availability (quantifying the alternation between delivery or not delivery of the service) and reliability (quantifying the continuous delivery of the service). Isolation under this dimension assures that dependability of different slices do not influence each others. An important threat to this type of isolation can be faults on a shared hardware element: in this case, the failure processes in different slices cannot be uncorrelated.

- Security dimension. It ensures that an attack directed to a certain slice does not have an impact on other slices. In a sliced environment, authentication, confidentiality, integrity and non-repudiation principles will not be easy and straightforward to be implemented. Slices exchange vital information through softwarized interfaces and, at the same time, services with very different security requirements may run on shared resources. An additional threat rises from the MANO entity: from there, since it gathers critical information about slices status, possible attacks may span to multiple slices.

Isolation Degrees

Isolation can be reached with different levels of strength, spanning from no isolation to complete isolation. In general, it holds that the more resources are shared among slices, the less isolation degree is achieved. Thus, since complete isolation is provided only assigning dedicated physical resources to each slice, it is necessary to introduce different levels of isolation. In each network segment, particular properties and parameters can be specified in the SLA to define proper isolation requirements. The necessity to introduce precise metrics is due to the need to measure isolation level and chose the proper technique to guarantee it. Kotulski et al., give some example of such metrics in [KNS$^+$18]. The authors define isolation strength as a n-tuple of properties and parameters: $I = (a_1, a_2, \ldots, a_n)$. This definition, allow to select proper technologies to measure and guarantee a certain isolation level in different areas of the network.

According to [GOLH$^+$19], there are three ways to specify isolation quality in the SLA. The most intuitive is to allow violation of isolation requirements for a certain percentage of time. Alternatively, the relaxation of the Service Level Objective (SLO) constrains due to a lack of isolation can be considered as metric. A last way to define isolation levels is to choose a precise isolation mechanism from a set of predefined ones, tailored for each network domain. In this case, the isolation strength offered by the tenant strictly depends on the type of the selected mechanism. The mechanisms used to provide isolation consist in resource splitting in the different network domains. In general terms, three main degrees of isolation can be identified:

- Hardware isolation. Complete isolation is achieved assigning different hardware entities to each slice. In the network infrastructure, examples of hardware resources can be: antennas in the radio access domain, cables and fibers in the transport domain, CPUs and storage systems in datacenter domain.
- Physical isolation. An intermediate level of isolation can be provided splitting the available physical resources. For instance, this can be performed assigning orthogonal radio resources to each slice in the radio channel, or using Wavelength Division Multiplexing (WDM) in optical fibers.

– Logical isolation. In this last case, isolation is guaranteed at the lowest level. Different slices share the same channel and the packet flows are managed with logical methods (e.g. packet tagging) so that traffic-class queuing can be provided.

Considering what has been said in the discussion performed so far, every particular methodology to enable isolation can be represented as a region in the 3D space whose axes are represented in Figure 2.9. For the sake of example, two techniques are represented by the blue regions. *Dedicated antennas* can be used to provide hardware isolation in the radio access domain. This solution is able to assure complete isolation and therefore to cover all the three dimensions (performance, dependability and security). In the transport domain, with reference to an optical fiber scenario, *WDM* can be employed to guarantee physical isolation. In fact, it is possible to assign different wavelengths in the available spectrum to each slice. Provided an ideal recovery of the signal at the receiver side, this technique can assure performance isolation. In the author's view, the use of such a 3D representation could help to compare various solutions and choose the most appropriate according to needs. In the next two sections, the impact of isolation issue on slice creation and management is analysed.

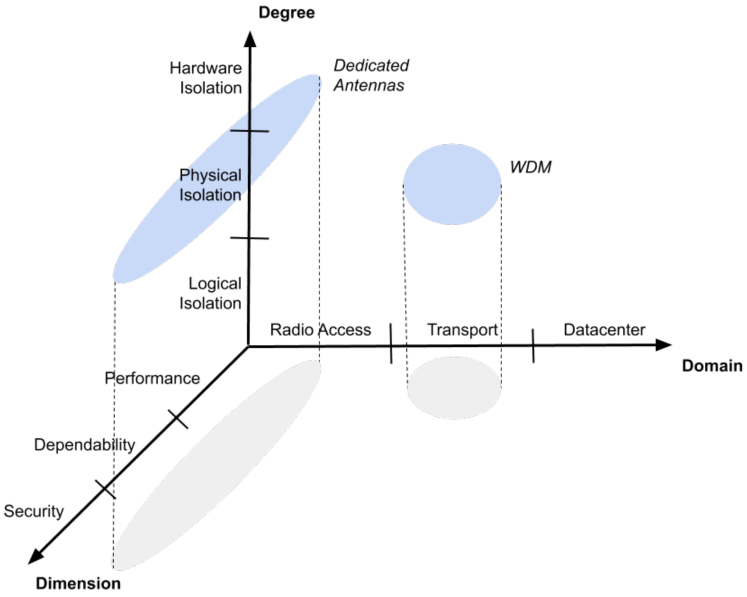


Figure 2.9: 3D space of isolation methodologies.

2.4.2 Isolated Slicing Creation

Isolation can be compromised when a new slice is created in addition to the already existing ones. In fact, reorganizing the allocated resources can create problems to all slices in terms of isolation violation and QoS degradation. Nowadays, this is still an open issue. A possible solution consists in defining a maximum number of slices that can coexist over a certain medium or resource (e.g. a wireless channel) under certain constraints of isolation and QoS, and consequently avoid to create a new slice instance if this number is overcome. In order to maintain performance isolation even in presence of more parallel services, a proper resource usage should be considered. For instance, in a frequency divided wireless channel proper guard band should be taken into account in defining the maximum number of parallel users.

2.4.3 Isolated Slicing Management and Orchestration

The ETSI MANO framework presented in Section 1.2 considers only isolation from the performance point of view. In order to assure also security in a multi-domain, multi-tenant isolated slicing scenario, a possible architecture is presented in [KNS$^+$18]. This solution is an extension of the ETSI MANO model, as shown in Figure 2.10. To deal with the large amount of variables characterizing the environment and requirements, a single MANO entity is decomposed into several cooperating MANO subsystems. Thus, dedicated entities are introduced to provide isolation at each stage of a NSI lifetime. In particular, the following task are performed: (*i*) secure isolated slice establishment during the instantiation phase, (*ii*) isolation checking during the run-time phase and (*iii*) secure critical data destruction during the decommissioning phase.

Furthermore, due to the E2E nature of slices, a service provider must ensure that the QoS and isolation requirement are met in every network domain. This brings additional complexity for slice access management. First of all, when a slice is created, it is essential to define inter-slice access functions to guarantee security isolation (*Inter-slice Orchestrator*). Second, mutual access between RAN and core network has to be controlled by the *Slice Chaining MANO* providing security and, at the same time, continuity of the slice. Last, in a multi-administrative domain environment interfaces between physical infrastructures owned by different network providers are regulate by the *Inter-domain MANO* entity.

2.5 Network Slicing Research Directions

So far, network slicing has been analysed under its main important aspects. Results obtained from a deep study of the literature have been presented: standardization bodies proposals have been analysed as well as the challenges raised by scientists and

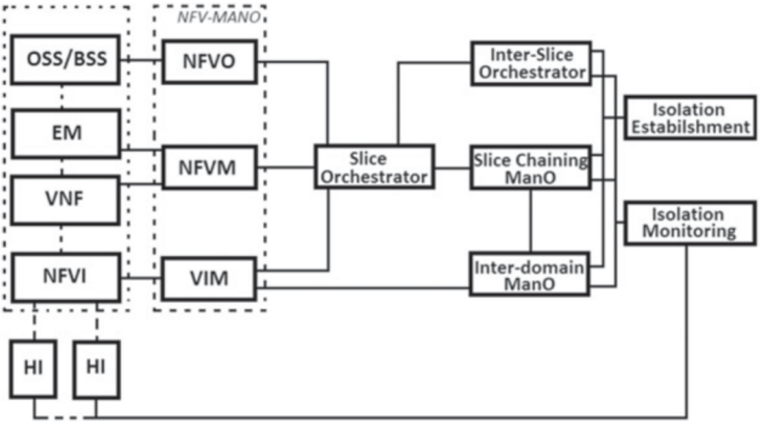


Figure 2.10: MANO architecture for isolated slicing [KNS+18].

experts. Network slicing brings forward a significant potential, but also introduces several technical and business challenges:

- **Techno-economical aspects**. The innovative partnerships between several players empowers promising business models. Charging models regarding network capabilities, dynamic slice usage, value-added services, and management and orchestration are still open.

- **Resource sharing and isolation**. Statically assigning resources to UEs allows ensuring traffic isolation on one hand but limits multiplexing gain on the other. Traffic prediction-based and policy-based resources allocation mechanisms may enhance resource utilization but at the cost of security and isolation. A univocal and unifying definition of these properties is necessary to specify the requirements that the network must meet and to reach, therefore, a fair compromise.

- **Security and privacy**. Since network slices use interaction interfaces to exchange vital information, these open programmable interfaces bring new potential attacks to softwarized networks. New solutions to ensure security in presence of multiple trust domains must be investigated.

- **Management and orchestration optimality**. Network slice resources must be dynamically scaled and orchestrated. Resource allocation algorithms which incorporate Pareto optimality techniques are not straightforward in such a complex environment. Thus, it is necessary to develop these algorithms, maybe using Machine Learning approaches to deal with the large amounts of data at stake.

- **UEs slicing**. 5G networks are expected to handle UEs based on their individual characteristics and usage-class types. So, every UE shall be connected to a customized slice or to multiple slices. How the OS of UEs must be modified to enable this connection is still object of future research.

Nowadays, and even more in the future, vertical industries are in need for a unique combination of network and computing resources to efficiently support their use cases. The solution is offered by network slicing: a concurrent deployment of multiple, logical, self-contained and independent, shared or partitioned networks on a common infrastructure platform. We investigated how network slicing could be implemented and how its state of the art is advancing. This comprised a study of the involved technologies across the different infrastructures and network segments. Many network slicing enablers already exist, and the techniques that will allow operators to fully exploit these enablers are being developed. Once all parts of the infrastructure can be managed by one platform, the advent of network slicing will greatly impact the evolving 5G technology.

In this study, we decided to focus our attention on Wi-Fi networks. Even though many solutions have been already proposed to implement network slicing in 5G, the Wi-Fi technology has not be deeply treated yet. Thus, our study wants to provide a methodology to create slices in the physical layer of these networks, that will accompany 5G in the coming years. A detailed description of the Wi-Fi standard (IEEE 802.11) is provided in next chapter.

Chapter 3
Wi-Fi Technology

To conclude the required background study, in this chapter we analyse the main aspects of the standard IEEE 802.11. The latest release of this standard can be entirely consulted on the dedicated IEEE document [80216]. The documentation is comprehensive and consists of more than 3500 pages. Here, we want to treat only the relevant properties that must be considered for the simulations that we are going to conduct. In the following chapters, we will discuss how these properties are implemented in the simulator we used, and which are its limitations.

3.1 Wi-Fi and IEEE 802.11

IEEE 802.11 is part of a broader family of LAN protocols, the IEEE 802, and specifies the group of Medium Access Control (MAC) and Physical (PHY) layer protocols for realizing Wireless LANs of devices.

In Figure 3.1, it is shown where this standard operates in the ISO/OSI model. At the PHY layer, it characterizes the Modulation and Coding Schemes (MCSs) that can be employed in wireless communications, and the spectrum usage in terms of available frequencies and bandwidths. At the MAC layer, it defines multiple access techniques and datagram structures.

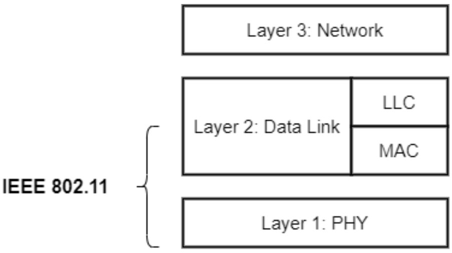


Figure 3.1: IEEE 802.11 standard in the ISO/OSI model.

Commonly, one refers to the technologies implementing the 802.11 standard as Wi-Fi but this is not strictly correct. Actually, Wi-Fi is a trademark of the non-profit organization Wi-Fi Alliance and it refers to a precise group of wireless networking technologies, based on the IEEE 802.11 standard. The main role of the Wi-Fi Alliance is to certify Wi-Fi products for conformity to certain standards of interoperability [wif]. Thus, only some versions of the 802.11 standard are certified by the Alliance and can be called Wi-Fi. In the following, we will focus only on those certified versions, since the non Wi-Fi ones find their place in niche applications or are thought to operate only in limited regions in the world.

3.1.1 Architecture

There are two types of components constituting a Wi-Fi network: the Access Points (APs) and the Stations (STAs). APs, also called Wi-Fi routers, are the base STAs of the network. They usually bridge the connection between wired (Ethernet) and wireless, and serve the clients within their range of coverage. The area they cover depends on the transmission power and on the type of utilized antenna. On the other hand, the STAs are the clients of the network. These devices, which require access to a wireless router, can be mobile (such as laptop and smartphones) or non-portable (such as desktop computers and printers).

In order to connect to a Wi-Fi network, a user needs the network name (i.e. the Service Set Identifier (SSID)), and the password, used to encrypt and decrypt the exchanged information and provide authentication.

A Service Set is the set of all the devices associated to a particular Wi-Fi network. Each Service Set has a corresponding 32-byte identifier, the SSID, which identifies the network. The SSID is configured in all the devices that are part of the network, which can include multiple APs and STAs. In general, devices in a Service Set are not connected on the same frequency. On the contrary, for large Service Sets, more channels can be allocated exploiting the concept of frequency reuse to decrease interference as in cellular networks.

A Basic Service Set (BSS) is a group of nodes that share the same channel, SSID, and other wireless settings. Usually, all the STAs in a BSS are connected to the same AP. A precise MAC address, called Basic Service Set Identifier (BSSID), identifies each BSS.

Operational principles

Wi-Fi nodes communicate by exchanging datagrams: i.e. blocks of data individually delivered over the radio channel. This is done by using modulation schemes defined by the different versions of Wi-Fi. Every wireless node comes with a globally unique

48-bit MAC address (usually printed on the device) so that each Wi-Fi node can be uniquely identified. The MAC addresses are used to specify both the destination and the source of each datagram. Wi-Fi communication is established by link-level connections, defined by these addresses. Upon reception of a transmission, the receiver checks the destination address to determine whether the transmission is relevant or can be ignored.

Uses

Wi-Fi technology is used mainly to provide Internet access to devices that are within the range of one or more routers connected to the Internet. The coverage of one or more interconnected APs can extend from a room-sized area to many square kilometres. Broadband Internet access can be provided, in fact, in citywide scale with the so called municipal wireless networks.

An additional popular use of Wi-Fi regards the geolocalization of mobile devices. In particular, Wi-Fi based positioning techniques can identify a STA's location through the position of the AP to which it is connected [KJBK15].

3.2 PHY Layer

The different versions of Wi-Fi are characterized by different PHY layer technologies determining the spectrum regions use, the allowed MCSs, the transmission power and, consequently, the bit rates that can be achieved.

3.2.1 Channels and Frequencies

Wi-Fi technology typically uses the 2.4 GHz and 5 GHz spectrum regions, which are Industrial Medical Scientific (ISM) radio bands. The only exception is the latest version drafted in 2019, the 802.11ax, which is supposed to operate in the whole ISM region from 1 to 6 GHz. More precisely, 802.11b and 802.11g utilize the 2.400 – 2.500 GHz spectrum, while 802.11a and 802.11ac use the 5.150 – 5.875 GHz band. The version 802.11n can work in both regions.

These bands are subdivided into multiple channels characterized by a center frequency and a bandwidth. More networks can share channels, but only one transmitter at time can locally transmit on a channel.

Spectrum management and imposed limitations on frequency and power are not consistent worldwide: countries apply their own regulations to the available channels, allowed users and maximum power levels within these frequency ranges. 802.11 specifications evolved in time to enable higher throughput, and the protocols have become much more efficient in exploiting the bandwidth. To gain more throughput where the bandwidth is available, a technique has been introduced in recent amendments: the so-called channel bonding, whose main idea is to aggregate (or “bond”) channels together. It will be treated with more detail in a following section.

Channels in the 2.4 GHz Band

The 2.4 GHz band is divided into 14 channels spaced 5 MHz from each other (except the 14^{th}), as shown in Figure 3.2. The last three channels have additional restrictions in many regions in the world. In particular, they are unavailable in America while only the 14^{th} is not available in Europe.

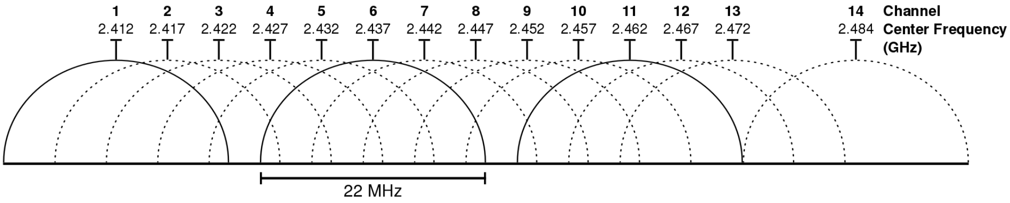

1 2 3 4 5 6 7 8 9 10 11 12 13 14 Channel
2.412 2.417 2.422 2.427 2.432 2.437 2.442 2.447 2.452 2.457 2.462 2.467 2.472 2.484 Center Frequency (GHz)
22 MHz

Figure 3.2: Wifi channel numbers with their center frequencies in the 2.4 GHz band. Adapted from [Fli07].

Originally, in the first Wi-Fi version, all the channels in the 2.4 GHz region had a bandwidth of 22 MHz. In recent standards, instead, it has been narrowed down to 20 MHz.

IEEE 802.11 specifies the channels’ center frequency, but also, in Clause 17, the spectral mask defining the allowed power distribution. The mask imposes the signal to be attenuated by a minimum of 20 dB from its peak amplitude at +/-11 MHz from the center frequency, in case the channel is 22 MHz wide. Thus, STAs can use only one channel every five without overlap.

In practice, given the restriction imposed by the spectral mask at the channel limits, it is usually assumed that the energy of the channel does not extend further that these borders. However, in reality, a separation between channels should be considered so that a signal on any channel is sufficiently attenuated to minimally affect the communication on any other channel. In fact, a transmitter can affect a receiver on a non-overlapping channel, if it is close to the victim receiver or if it is operating above the permitted power level. On the contrary, a transmitter far enough working on an overlapping channel can have no significant impact.

Channels in the 5 GHz Band

In the 5 GHz region, the organization of the channels is more complex and different channels are available in different regions of the world. In Table 3.1 the channels defined in Europe are presented.

3.2.2 Radio Propagation

Compared to cellular network technology, Wi-Fi involves lower transmission powers. For the 2.4 GHz band, ETSI defines a limit for the mean Equivalent Isotropically Radiated Power (EIRP) equal to 20 dBm (i.e. 100 mW) [ETS12b]. The 5 GHz band, instead, is subdivided in sub-bands differently regulated. The limits on the EIRP imposed by ETSI are reported in Table 3.2 [ETS12a, ETS08]. Note that in the first two regions such limit varies depending on whether the Transmit Power Control (TPC) is used or not.

Wi-Fi's frequencies suffer from high absorption because of their short wavelength, and work best for Line-of-Sight (LoS) communications. Many common obstructions, such as furniture and walls, may, on one hand, dramatically reduce the coverage, but, on the other hand, can help to minimize interference between neighbor BSSs.

Interference can be caused by transmitters within the same channel, or can come from adjacent channels. To reduce the number of collisions a proper access channel technique is utilized by IEEE 802.11, which is described in the next section. Furthermore, another way to limit adjacent channel interference, is to assign to neighboring APs non-overlapping channels. For instance, within the 2.4 GHz band, only channels 1, 6 and 11 are allocated if three different channels are used at the same time in the same location.

3.3 MAC Layer

Wi-Fi channels are half duplex and can be shared in time by multiple networks. If more devices use the same channel, collisions are more likely to occur when two STAs try to transmit at the same time. Thus, transmitted data may be corrupted and need to be re-transmitted. As a consequence, the throughput degrades due to data losses and re-transmissions. To minimize the number of collisions, IEEE 802.11 technology implements a collision avoidance mechanism called Distributed Coordination Function (DCF).

3.3.1 Distributed Coordination Function

The DCF is the fundamental MAC technique of the IEEE 802.11-based Wireless LAN. For the channel access, DCF employs a Carrier Sensing Multiple Access with

20 MHz	40 MHz	80 MHz	160 MHz
36 (5180)	38 (5190)	42 (5210)	50 (5250)
40 (5200)			
44 (5220)	46 (5230)		
48 (5240)			
52 (5260)	54 (5270)	58 (5290)	
56 (5280)			
60 (5300)	62 (5310)		
64 (5320)			
100 (5500)	102 (5510)	106 (5530)	114 (5570)
104 (5520)			
108 (5540)	110 (5550)		
112 (5560)			
116 (5580)	118 (5590)	122 (5610)	
120 (5600)			
124 (5620)	126 (5630)		
128 (5640)			
132 (5660)	134 (5670)	138 (5690)	
136 (5680)			
140 (5700)	142 (5710)		
144 (5720)			
149 (5745)	151 (5755)	155 (5775)	
153 (5765)			
157 (5785)	159 (5795)		
161 (5805)			
165 (5825)			
169 (5845)			
173 (5865)			

Table 3.1: 5 GHz band Wi-Fi channel numbers with their center frequencies (in parentheses, expressed in MHz), for every channel bandwidth (from 20 MHz to 160 MHz).

Band	Range [MHz]	Mean EIRP limit [dBm]	
		with TPC	without TPC
1	5150 to 5350	23	20
2	5470 to 5725	30	27
3	5725 to 5875	36	

Table 3.2: ETSI limits on the mean EIRP for the 5 GHz band.

Collision Avoidance (CSMA/CA) with a binary exponential backoff algorithm.

According to the CSMA/CA protocol, a STA wishing to transmit is required to sense the channel status for a DCF Inter-Frame Spacing (DIFS) interval. If the channel is found busy during the DIFS interval, the STA delays its access. It means that other devices are active on the channel, or that significative noise is detected from adjacent channels or from non Wi-Fi sources. Conversely, if the channel is recognized to be free, the transmission can take place.

In a network where many STAs contend for the wireless medium, if multiple devices find the channel busy and postpone their transmission, they could simultaneously find the channel released and then they could try to access the channel at the same time. As a result, collisions may occur. In order to avoid that multiple waiting STAs transmit when the channel is released, DCF defines a random backoff time, which forces a device to defer its access to the channel for a random extra interval of time.

DCF has also an optional mechanism that exchanges short Request to Send (RTS) and Clear to Send (CTS) datagrams between the transmitter and the receiver. In particular, the source sends a RTS signal to the destination if, after the sensing period, it finds the channel free. Thus, the transmission of data will start only upon reception of a CTS message from the destination. The benefits of this signalling strategy will be experimentally analyzed in Chapter 5.

A positive acknowledge scheme is considered for every transmission, which means that if a datagram is successfully received by the destination, the receiver sends an Acknowledgement (ACK) to notify the source of the successful reception. Figure 3.3 represents how a node A can establish a communication with node B using CSMA/CA with RTS and CTS exchange.

3.3.2 IEEE 802.11 Frames

In the context of 802.11 standard, datagrams in the data link layer are called frames. 802.11 specifies the standardized fields which constitute a frame. As shown in

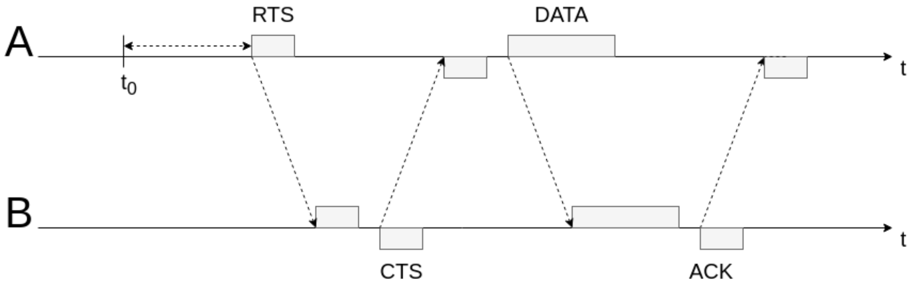
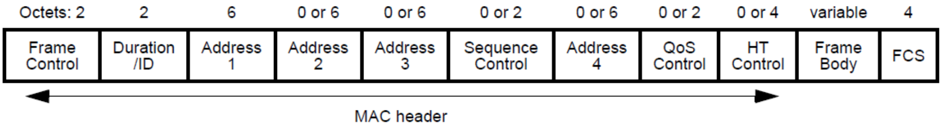


Figure 3.3: CSMA/CA with RTS and CTS.

Figure 3.4, each frame consists of a MAC header, a frame body (the payload), and a Frame Check Sequence (FCS). The header contains information on the frame type, on the source and on the destination of the message. The payload is the useful message transmitted, which is is variable in size, from 0 to 2304 bytes. Finally, the FCS is an error correcting code added at the end of the frame: it is used to check the correctness of the entire frame.



Octets: 2	2	6	0 or 6	0 or 6	0 or 2	0 or 6	0 or 2	0 or 4	variable	4
Frame Control	Duration /ID	Address 1	Address 2	Address 3	Sequence Control	Address 4	QoS Control	HT Control	Frame Body	FCS

Figure 3.4: Wi-Fi MAC frame format.

The IEEE 802.11 standard defines three types of frame, depending on their purpose:

- **Management Frames**: are used by the STAs to connect of disconnect to a BSS. They can be sent both by the device which is requiring to connect (for instance to authenticate and request the association), and by the AP, which spreads its beacon signal and replies to the association request.

- **Control Frames**: assist the transmission of data and management frames. They do not have a frame body since they do not carry information. Typical examples of control frames are: RTS, CTS and ACK messages.

- **Data Frames**: are the ones which include the useful information that the source wants to transmit to the destination.

3.4 Wi-Fi Generations

In the last 20 years, many versions of 802.11 standards has been developed by IEEE. So far, we analysed the common aspects characterizing the Wi-Fi technologies, with focus on those recognized and certified by the Wi-Fi Alliance. They are the ones present, in practice, in the devices we use in our daily life. In this section, we will treat all the versions separately and will examine how they evolved over the years. In Table 3.3, the 802.11 versions adopted by the Wi-Fi Alliance in certification programs are listed with their relative generational names [1].

Generation	IEEE Protocol	Year	Data rate [Mbit/s]	Freq. [GHz]	Bandwidth [MHz]	Mod.
Wi-Fi 6	802.11ax	2019	600 – 9608	1 – 6	20/40/80/160	OFDM
Wi-Fi 5	802.11ac	2014	433 – 6933	5	20/40/80/160	OFDM
Wi-Fi 4	802.11n	2009	72 – 600	2.4/5	20/40	OFDM
Wi-Fi 3	802.11g	2003	3 – 54	2.4	20	OFDM
Wi-Fi 2	802.11a	1999	1.5 – 54	5	20	OFDM
Wi-Fi 1	802.11b	1999	1 – 11	2.4	22	DSSS

Table 3.3: Wi-Fi generations.

3.4.1 802.11b

The IEEE 802.11b is an amendment to the original 802.11 version which aim to extend the maximum raw data rate to 11 Mbit/s. As the original specification, it uses CSMA/CA to schedule the channel access. The used modulation technique is an evolved version of the Direct Sequence Spread Spectrum (DSSS) technique employed in the original standard, that brought a significant increase in throughput. Since it operates in the 2.4 GHz band, it suffers interference from other products e.g. bluetooth devices, cordless telephones, baby monitors and microwave ovens. The APs communicate with the nodes within their range through an omnidirectional antenna.

3.4.2 802.11a

The 802.11a amendment introduced Orthogonal Frequency Division Multiplexing (OFDM) modulation in the PHY layer of Wi-Fi communications. It works in the 5 GHz band and the channel bandwidth is 20 MHz. The 52-subcarrier OFDM allows

[1] Wi-Fi 1, Wi-Fi 2 and Wi-Fi 3 generational names are assigned only unofficially. In 2018, when Wi-Fi Alliance designated Wi-Fi generations, 802.11g/a/b versions were already out of date.

to reach a maximum raw data rate of 54 Mbit/s. A graphical representation of the utilized OFDM modulation is shown in Figure 3.5.

Out of the 64 subcarriers constituting the OFDM symbol, 52 are used for data transmission, 4 are pilot subcarriers (i.e. used for channel estimation at the receiver), and the 8 at the border of the spectrum are unused to reduce the interference with the adjacent channels. The carrier separation is $\Delta f = 20\,\text{MHz}/64 = 312.5\,\text{kHz}$. Each of these subcarriers can be modulated with BPSK, QPSK, 16-QAM or 64-QAM. The symbol time is $T = 1/\Delta f = 3.2\,\mu\text{s}$, to which a Guard Interval (GI) of 0.8 µs is added.

In 5 GHz band, less interference from other non Wi-Fi technologies is present but the higher frequencies allow a shorter coverage range since they are absorbed more easily by the obstacles. On the other hand, OFDM has important propagation advantages in indoor environments. By dividing the entire channel into many narrow subcarriers, the modulation scheme is more robust against multipath.

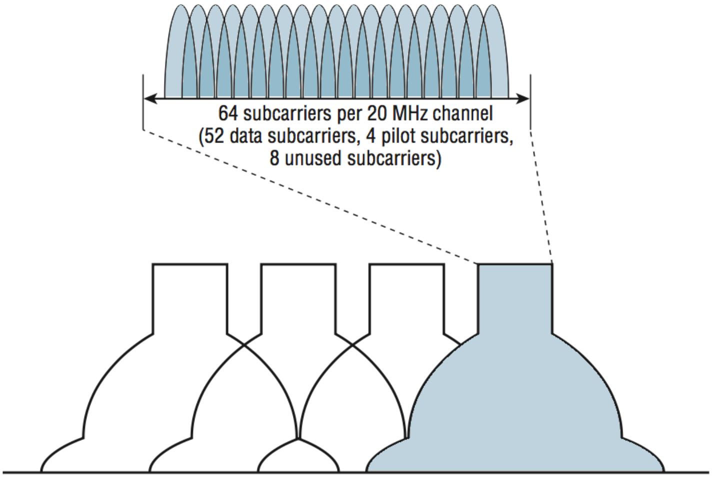

64 subcarriers per 20 MHz channel
(52 data subcarriers, 4 pilot subcarriers,
8 unused subcarriers)

Figure 3.5: Wi-Fi OFDM modulation for 802.11a/g/n/ac.

3.4.3 802.11g

This standard works in the 2.4 GHz band as 802.11b, so it also suffers the interference from other products operating in the same band. 802.11g is backwards compatible with 802.11b. Unlike the older 802.11b version, the throughput is extended to up to 54 Mbit/s using a 20 MHz bandwidth and the same OFDM transmission scheme as 802.11a.

3.4.4 Wi-Fi 4 – 802.11n

Wi-Fi 4 introduces many novelties with respect to the previous versions: it works in both 2.4 GHz and 5 GHz frequency bands and it is the first version of Wi-Fi to support Multiple-Input Multiple-Output (MIMO). It introduces the so-called channel bonding technique to use also 40 MHz channels (PHY layer), and frame aggregation to the MAC layer [80209]. The OFDM parameters are the ones described for 802.11a but the GI can be 0.8 µs or 0.4 µs.

MIMO technology, which exploits multiple antennas systems, can be used to enable Spatial Division Multiplexing (SDM). The 802.11n standard allows up to four transmit antennas, up to four receive antennas and up to four spatial streams. In a MIMO link transmitters and receivers perform precoding and postcoding, respectively. These processes enable two techniques:

- **Spatial beamforming**: allows to improve the received signal strength exploiting the antenna array directivity;
- **Spatial coding**: can be used to increase the throughput via spatial multiplexing or to increase the maximum transmission distance by exploiting the spatial diversity.

The objective of Wi-Fi 4 is to further increase the throughput with respect to 802.11g. The maximum raw data rate is 72 Mbit/s with a single spatial stream in a 20 MHz channel. However, up to 600 Mbit/s can be reached by using four spatial streams and a channel width of 40 MHz.

802.11n enhanced the PHY layer throughput by adopting the 40 MHz channel bonding technique. This methodology is deeply analysed by Daldoul et al. in [DMK17]. A 40 MHz channel consists of two adjacent 20 MHz channels: the primary channel and the secondary channel. The 802.11n STA checks if the primary and the secondary channels are idle for a DIFS with a Clear Channel Assessment (CCA) test. If both channels are free, the STA can transmit over a 40 MHz signal. However, if only the secondary channel was not idle during the sensing phase, the STA has two choices, depending on the network setting:

- In case of static 40 MHz channel access, the STA will reattempt to access the 40 MHz channel by sensing the channel again after a random backoff time.
- If dynamic 20/40 MHz channel access is selected, the STA can transmit a 20 MHz signal only on the primary channel.

Frame aggregation is a MAC technique which also provides a throughput improvement. Two types of aggregation are defined in 802.11n: aggregation of MAC Service Data Units (MSDUs), at the top of the MAC layer, and aggregation of MAC Protocol Data Units (MPDUs), at the bottom of the MAC. The process of frame aggregation consists in packing together more MSDUs or MPDUs to reduce the overheads. Headers can be used once over multiple frames, so that to increase the user level data rate.

3.4.5 Wi-Fi 5 – 802.11ac

Wi-Fi 5 includes wider channels with respect to Wi-Fi 4: it allows, in fact, 80 and 160 MHz channels in the 5 GHz band. The same channel bonding technique explained for Wi-Fi 4 is utilized, but extended also to wider channels. As shown in Figure 3.6, besides the 20 MHz primary channel, three secondary channels are considered. The access to these channels is ruled in a similar way as in 802.11n. Furthermore, Wi-Fi 5 supports more MIMO spatial streams (up to eight), and higher-order modulations can be used (up to 256-QAM).

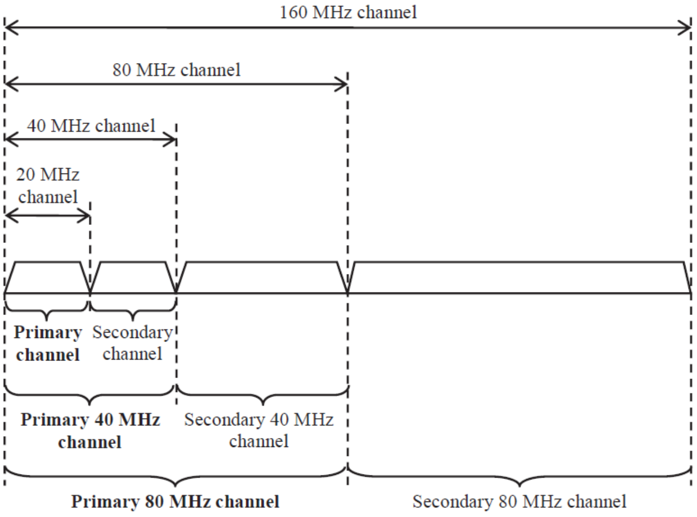


Figure 3.6: Channel bonding in 802.11ac [DMK17].

In this 802.11 version, the downlink Multi-User MIMO (MU-MIMO) is introduced: a spatial multiplexing technique for up to four clients. MU-MIMO allows the AP to form beams towards each STA, in order to transmit information simultaneously. As a result, the interference between STAs is reduced, and the overall throughput is increased, since multiple clients can receive data at the same time. Thanks to downlink MU-MIMO, a further multiple access technique is enabled: Spatial Division

Multiple Access (SDMA). It allows to use streams not separated in frequency, but spatially.

3.4.6 Wi-Fi 6 – 802.11ax

802.11ax has seen the end of its deployment in 2019 and is the last version certificated by Wi-Fi Alliance, also called Wi-Fi 6. It is thought to work in all the available ISM bands between 1 and 6 GHz in addition to the 2.4 GHz and 5 GHz dedicated bands. Since the IEEE draft of the standard is not yet available even with academic licence, our research was based on the study carried out by Khorov et al. [KKLB18].

In this Wi-Fi generation, Orthogonal Frequency Division Multiple Access (OFDMA) has been introduced as the multiple access technique. It is based on the division of the spectrum in time-frequency radio resources, called Resource Units (RUs). Thus, the AP assigns RUs to communicate with its associated STAs. Through a central scheduling of RUs, the overhead caused by the contention of the channel can be avoided, resulting in a better efficiency in dense scenarios.

MU-MIMO is now available both in downlink and in uplink directions. The GI has been extended to gain a better protection against the delay spread, which causes inter-symbol interference. Possible values of GI are 0.8 µs, 1.6 µs and 3.2 µs.

The allowed modulation formats in 802.11ax are reported in Table 3.4 with their relative cardinality M (i.e. the number of symbols in the constellation). For each MCS index, the utilized coding rate r is also presented. These two parameters, M and r, will be utilized in Chapter 5 to calculate the theoretical maximum throughput achievable in a Wi-Fi 6 channel.

802.11ax adds MCS with denser modulation schemes in order to reach higher throughput with respect to the previous versions. In fact, in 802.11n the MCS indexes available are limited to the range $0-7$, while 802.11ac can reach MCS number 9, still referring to Table 3.4.

MCS	Modulation	Cardinality M	Coding Rate r
0	BPSK	2	1/2
1	QPSK	4	1/2
2	QPSK	4	3/4
3	16-QAM	16	1/2
4	16-QAM	16	3/4
5	64-QAM	64	2/3
6	64-QAM	64	3/4
7	64-QAM	64	5/6
8	256-QAM	256	3/4
9	256-QAM	256	5/6
10	1024-QAM	1024	3/4
11	1024-QAM	1024	5/6

Table 3.4: MCS indexes with relative digital modulation scheme and coding rate in 802.11ax.

Chapter 4
Network Simulators

This chapter explains why the network simulator *ns-3* has been selected to carry out the experiments in this work. This decision was taken after a careful study of available network simulators.

4.1 Comparison of Recent Network Simulators

In order to decide which network simulator has the most suitable characteristics for our scope, a survey about the features offered by available network simulators was conducted. 15 network simulators were analysed exploring their web pages and their source code on GitHub, if available. Survey studies were critically consulted but they always turned out to be obsolete.

In 2009, Elias Weingartne et al. published their comparison of the emerging network simulators [WVLW09]. A performance comparison was carried out by simulating an identical scenario on five network simulators: JiST/SWANS, *ns-2*, *ns-3*, OMNET++ and SimPy[1]. Despite this study being dated, it highlighted promising features of two leading simulators still maintained today: *ns-3* and OMNET++. In 2012, a more detailed study on the same four network simulators (JiST/SWANS, *ns-2*, *ns-3* and OMNET++) was presented by Pal in [Pal12].

In 2012, Khan et al. compared the state-of-the-art of open source wireless network simulators, namely: *ns-2*, *ns-3*, OMNET++ and GloMoSim [KBO12]. The comparison was based on different parameters concerning the CPU and the memory usage, the computational time and the offered scalability.

A very similar study on wireless network simulators has been presented by Khan (a homonym of the previous one) et al. in [KHN14]. The considered simulators are: *ns-2*, *ns-3* and OMNET++, TOSSIM and J-SIM. Again, they were compared

[1] Actually, SimPy is a bare simulation API written in Python and not a network simulator.

according to the same performance metrics, and *ns-3* turned to outperform all the other software.

A more comprehensive study was published by Kabir et al., who analysed the strengths and the weaknesses of 14 network simulators [KIHH14]. However, unlike the previous works, they did not test the presented network simulators and did not measure any performance metric.

More recently, Aman et al. published a more updated comparison of *ns-2*, *ns-3*, OMNET++ and QualNet in terms of mobility support [AHA$^+$16]. A Proxy Mobile IPv6 network topology has been realized in the considered software, and *ns-3* was the selected one for its scalability and efficiency in CPU and memory utilization.

In conclusion, the available literature on network simulators was a good starting point which helped in identify the leading open source simulators. However, it turned to be out of date since the most recent study dates back to 2016. For our scope, the most recent Wi-Fi technologies need to be supported and continuously updated in order to reflect the latest standards. For this reason we performed our own survey on existing network simulators. The main properties of the considered simulators are reported in Table 4.1 and Table 4.2.

In particular, for all the network simulators (sorted alphabetically), the following characteristics are reported:

- The licence type, which can be open source or commercial;
- The programming language utilized to develop the simulator. For simulators without Graphical User Interface (GUI) available, this is the programming language in which simulation scripts must be written;
- The supported OS: Windows (W), Linux (L), MAC (M) or platform independent (Ind.) in case of Java-based software;
- If a GUI is present or not;
- If a visualization tool is present or not. In some cases, even if the software has no GUI, developers from the community created some visual platforms useful to represent the simulated scenarios. In this way, the involved nodes and the flows between them are visible at run-time for a better understanding of the results;
- The quality of the official documentation available: excellent, good, medium or poor;
- The type of simulation involved. Very often it turned to be Discrete-Event;

	License	**Language**	**OS**	**GUI**	**Visualization**	**Documentation**
DRMSim	Open Source	Java	LM	Yes	Yes	Good
GloMoSim	Open Source	C	WL	No	Yes	Poor
GrooveNet	Open Source	C++	L	No	Yes	Medium
J-SIM	Open Source	Java	Ind.	Yes	Yes	Medium
JIST/SWANS	Open Source	Java	Ind.	No	No	Medium
Komondor	Open Source	Python	L	No	No	Medium
NCTUns	Open Source	C++	L	Yes	Yes	Good
NetSim	Commercial	C, Java	WM	Yes	Yes	Excellent
ns-2	Open Source	C++	WLM	No	No	Excellent
ns-3	Open Source	C++, Python	WLM	No	Yes	Excellent
OMNeT++	Open Source	C++	WLM	Yes	Yes	Excellent
OPNET	Commercial	C, C++	WL	Yes	Yes	Good
QualNet	Commercial	C++	WLM	Yes	Yes	Excellent
SSFNet	Open Source	Java, C++	WL	No	Yes	Good
TOSSIM	Open Source	C++, Python	WL	No	Yes	Good

Table 4.1: Network simulators comparison, part 1.

	Simulation Event Type	**Available Modules**	**GitHub**	**Last Release**
DRMSim	Discrete-Event	Dynamic Routing Models	No	Mar. 2013
GloMoSim	Parallel Discrete-Event	Wired, Wireless	No	2000
GrooveNet	Hybrid Simulator	Vehicular Networks	Yes	Jul. 2013
J-SIM	Diverse Numerical Methods	Wired, Wireless	No	Jun. 2008
JIST/SWANS	Discrete-Event	Wireless, AdHoc	No	Mar. 2005
Komondor	Discrete-event	IEEE 802.11 ax	Yes	Mar. 2020
NCTUns	Discrete-Event	Wired, Wireless IP Networks	No	Dec. 2009
NetSim	Stochastic Discrete-event	Wired, Wireless, AdHoc, WSN	No	Aug. 2019
ns-2	Discrete-Event	Wired, Wireless, AdHoc, WSN	No	Nov. 2011
ns-3	Discrete-Event	Wired, Wireless, AdHoc, WSN	Yes	Sep. 2019
OMNeT++	Discrete-Event	Wired, Wireless, AdHoc, WSN	Yes	Jan. 2020
OPNET	Discrete-Event	Wired, Wireless, AdHoc, WSN	No	Feb. 2019
QualNet	Discrete-Event	Wired, Wireless, AdHoc, Tactical	No	Apr. 2020
SSFNet	Discrete-Event	Internet Protocols	No	Jan. 2004
TOSSIM	Bit Level Discrete-Event	WSN	Yes	Oct. 2016

Table 4.2: Network simulators comparison, part 2.

– The main available modules, that is which kind of networks the simulator can represent;

– If the source code is available on GitHub;

– The date of the last release. We have discarded from our decision the projects no longer maintained. For the ongoing projects, this is an indicator of how frequently the software is updated.

In the following paragraphs, each simulator is presented. The focus will be on strengths and weaknesses of each simulator, and on which aspects it is different from the others.

DRMSim (Dynamic Routing Model Simulator) allows to efficiently simulate routing protocols on large scale dynamic networks; in fact, up to 10,000 nodes can be simulated. Unluckily, it was created in 2011, and its last release dates back to 2013.

GloMoSim (Global Mobile Information System Simulator) is a network protocol simulation software that simulates both wired and wireless systems at the network layer. It is programmed in Parsec: a programming language which offers parallel discrete event simulation capabilities. For this reason it has good performance also with complex environments. Its last version was released in 2000 because a commercial derivate was launched in that year and GloMoSim was no longer maintained.

GrooveNet is unique for its capacity to analyze the performance and scalability of inter-vehicular communication protocols. It can model vehicular networks within a real street map-based topography and it facilitates both protocol design and in-vehicle deployment. It allows to study the transmission latency, and the coverage under various traffic conditions.

J-SIM (Java-Sim) is a component-based simulator. On top of the autonomous component architecture, a generalized packet switched network model is defined to simulate both wired and wireless network protocols. The model establishes the generic structure of a node (i.e. an end-host or a router) and the generic network components, which can then be used as interfaces to implement protocols across multiple layers.

JiST/SWANS (Java in Simulation Time / Scalable Wireless Ad-hoc Network Simulator) is a wireless network simulator built on top of the JiST platform. It was created to overcome some limitations of the already existing network simulators such as *ns-2* and GloMoSim. In particular, JiST/SWANS is able to simulate much larger networks, efficiently supporting higher simulation throughput.

Komondor is a simulator for Wi-Fi 6 networks deployed in 2018 [BMWSB19]. It was created to model the mechanisms of the standard IEEE 802.11ax. Furthermore, it is prepared for a simple integration with ML modules. Given its recent deployment and the offered functionalities, we decided to install this network simulator and to experiment its usage. The next section is entirely dedicated to it.

NCTUns (National Chiao Tung University network simulator) was developed to overcome the limitations of the Harvard network simulator. It integrates different protocol modules and can be used for both simulation and emulation. It supports remote and concurrent simulations which can be set by using the fully-integrated GUI environment. Unfortunately, the project was born in 2003 and it is no longer maintained since 2009. Thus, the most recent standards are not implemented.

NetSim is a leading simulation software for protocol modeling and simulation. It is developed in three commercial versions:

- NetSim Professional offers network capacity and growth studies through simulation and emulation to enterprise and defence;
- NetSim Standard allows to research and development in Universities to study and simulate their own protocols. To this goal, it offers special debug functionalities and interfaces with external software (e.g. MATLAB and Wireshark);
- NetSim Academic is designed for courses and experimentation. It can simulate all the common latest technologies and offers advanced visual tools for a better understanding.

ns-2 (Network Simulator 2) was born as a revision of its predecessor: *ns-1*. It is specially designed for protocol simulation and networking research. It provides support for simulation of Transmission Control Protocol (TCP), routing, and multicast protocols over wired and wireless networks. Today, it is no longer maintained since 2010.

ns-3 (Network Simulator 3) was created in 2006 as an improvement of *ns-2*. However, during its development, it was decided to abandon the backward compatibility with the previous version of the network simulator. *ns-3* is targeted primarily for research on internet systems and educational use. It is regularly updated and is based on a modular architecture which supports a vast variety of network elements and protocols. For this reason, it was installed and finally selected for carrying out our simulations. Later in this chapter, an entire section will be dedicated to *ns-3*.

OMNeT++ (Objective Modular Network Testbed in C++) is characterized by a modular extensible simulation library which allows simulations in different areas such as Internet protocols, LANs, vehicular networks, cloud computing and Wireless Sensor Networks (WSNs). The strength of this simulator is that the modules in its library are continuously extended and updated by the research community. For this reason, it offers a vast variety of models and tools. A model for Wi-Fi networks is provided within the INET Framework. However, it does not yet include 802.11ax features. At the PHY layer it support 802.11ac modes, while the MAC layer implementation is still entirely based on 802.11n.

OPNET (Optimized Network Engineering Tools) is a commercial solution which provides a powerful GUI to study communication networks, protocols, applications and devices. It allows to build the network topologies and define the entities from the PHY layer to the application layer. As a drawback, this approach limits researchers in creating their own protocols and customized scenarios.

QualNet was designed as a commercial derivate of GloMoSim. It is a platform for designing new protocols and analyzing their performance. The model library includes the IEEE 802.11 standard (the latest ax version is not supported), GSM, UMTS, LTE and military radio applications. In addition to the available models, it is unique for providing an environment to create new protocols, animating them through visual interfaces, and analyzing them through the packet tracer tool.

SSFNet (Scalable Simulation Framework Network Models) is an open source library which models different protocols (such as IP, TCP and User Datagram Protocol (UDP)) and network elements (such as hosts, routers and links). In addition, there are support classes for realistic multi-protocol, multi-domain Internet modeling and simulation.

TOSSIM (TinyOS Simulator) is specialized in simulating TinyOS applications. TinyOS a low power OS for wireless devices, often used in WSNs. Depending on the level required, simulation events can represent low-level systems such as hardware interrupts, or only higher-level events as a packet transmission. Since it is optimized for TinyOS's design, it can represent a network behaviour efficiently even in presence of thousands of nodes.

4.2 Komondor Network Simulator

Komondor is a event-based simulator thougth to reproduce the characteristics of the last version of the Wi-Fi standard: the IEEE 802.11ax. It is a recent work by Barrachina-Muñoz and Wilhelmi, open source and available on GitHub at the following link: https://github.com/wn-upf/Komondor.

4.2.1 Overview of Komondor

The Komondor simulation process follows the steps shown is Figure 4.1. Initial inputs, such as the simulation time and the seed, are passed to Komondor by console. Furthermore, three input files are required to describe the scenario to be simulated:

- The *System Input* file contains the global input parameters, for instance the information on the channels, the packet length, the pathloss model and the interference model;
- The *Nodes Input* file provides information about the APs and the STAs which constitute the network, such as the transmission powers used and their locations;
- The *Agent Input* file allows to define a learning mechanism able to modify parameters in the network and realize an intelligent resource allocation.

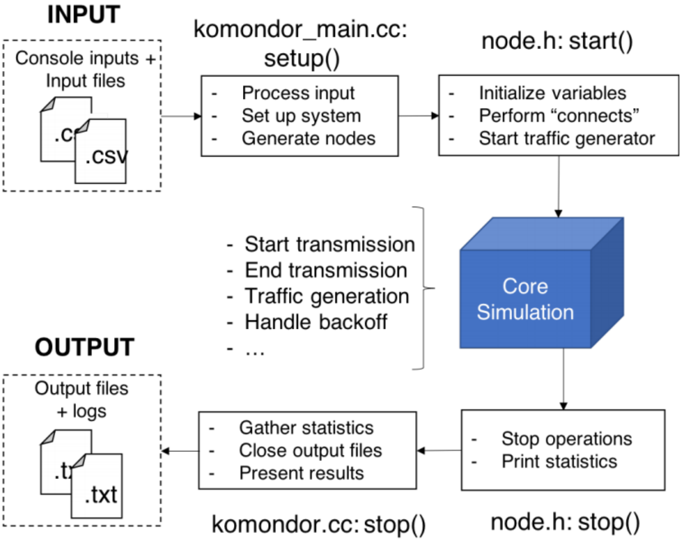


Figure 4.1: Komondor flowchart [BMWSB19].

These inputs are provided to the main module (*komondor_main.cc*) which creates the network environment and initializes the nodes present in the scenario. Each node is represented by a *node.h* component and act as an independent element which generates its own traffic. During the Core Simulation, the DCF manages the interactions between the nodes and the channel access.

Finally, when the simulation time is over, the simulation ends and the outputs are printed. By default, output logs are generated about the simulation activity, such as the achieved throughput, the number of transmitted packets and occurred collisions. In addition, log files relative to each node and agent can be generated upon request.

4.2.2 Komondor Limitations

Given the recent development and the integration of ML modules, Komondor immediately caught our attention. It is the only network simulator considering a native ML support, and it is updated regularly, almost every month. However it is still in a seminal state and many weaknesses are present.

The first problem is raised by the DCF. It manages the channel access by the STAs by utilizing CSMA/CA as access technique. Actually, the MAC Layer of the 802.11ax version includes OFDMA and not CSMA/CA as the previous versions. However, this inconsistency is present also in the *ns-3* simulator due to the fairly recent standardization of the Wi-Fi 6 amendment.

Mobility is not supported. In fact, the locations of the nodes are established only in the relative input file at the beginning of the simulation. This is a crucial weakness of Komondor since mobility is a fundamental property of today's wireless network.

Parameters of the network cannot be modified a run-time (i.e. while the simulation is running) in a customized manner. This implies that the number of nodes remains constant and it is not possible to act on the network changing parameters at run-time to enhance the performance. The only way to do it is by using the offered ML modules, which can be limiting.

The only ML technique implemented so far is the so-called Multi-Armed Bandit. In the user guide, the authors state that Reinforcement Learning (RL) techniques will be considered in future development, but, for the moment, the offered functionalities remain quite limited.

Using the available ML agent, it is possible to modify only few predefined parameters: the channel number, the CCA threshold, the transmission power and the channel bonding policy. This makes the current version of Komondor very restrictive. For the mentioned reasons, we decided not to use Komondor for our experiments. Nevertheless, it is a promising project which could become a excellent solution to study the ML integration in future Wi-Fi networks.

4.3 ns-3 Network Simulator

ns-3 is a discrete-event network simulator, open source and available on GitHub [ns3a]. The project is characterized by an open environment for researchers, which can contribute and share their software to extend the *ns-3* libraries. The large active community is the aspect that distinguishes this simulator from the other tools. Its set of libraries is continuously evolving and many recent standards are already supported (or partially supported). For instance, it is the only simulator including a module for millimeter waves communication, and the IEEE 802.11ax standard is implemented with only few limitation. In Chapter 5, these weaknesses will be largely examined.

ns-3 simulation scripts can be written in either C++ or Python programming languages. Since the entire *ns-3* library and the vast majority of the example scripts available online are written in C++, we adopted this language also to code our simulations.

In this section we will provide brief information on the *ns-3* GUIs and on how performance can be monitored. More details on the utilized modules can be found in Chapter 5, entirely dedicated on the model assumptions of this software.

4.3.1 ns-3 GUIs

ns-3 does not contain any GUI and there is no default graphical animation tool. However, there are currently two ways to provide animation, namely using *NetAnim* or *PyViz*. Thanks to these two methods, it is possible to get the simulated scenario in graphical format, where nodes interact with each other through visible links. In our work, we installed and used both graphical interfaces, which are described in this section. They have turned out to be out of date and not fully compatible with the most recent *ns-3* version (3.30). Nevertheless, they have been used to check the correct implementation of the mobility models.

NetAnim

NetAnim is a stand-alone program which uses the trace file generated by the *AnimationInterface* class to graphically display the simulation. The *NetAnim* GUI allows to start and stop the simulation, and to set the playback speed. When the simulation is running, nodes are shown and the messages exchanged between nodes are traced. This animator is fully documented in the official Wiki page https://www.nsnam.org/wiki/NetAnim. The last version is 3.108 and it was released in 2017. It does not support Wi-Fi standard 802.11ac/ax since it is not compatible with protocols using frame aggregation. This issue is still open, thus *NetAnim* cannot be employed to analyse the sequence of transmitted messages in the simulated scenario. A screenshot of our simulation seen in *NetAnim* is reported in Figure 4.2.

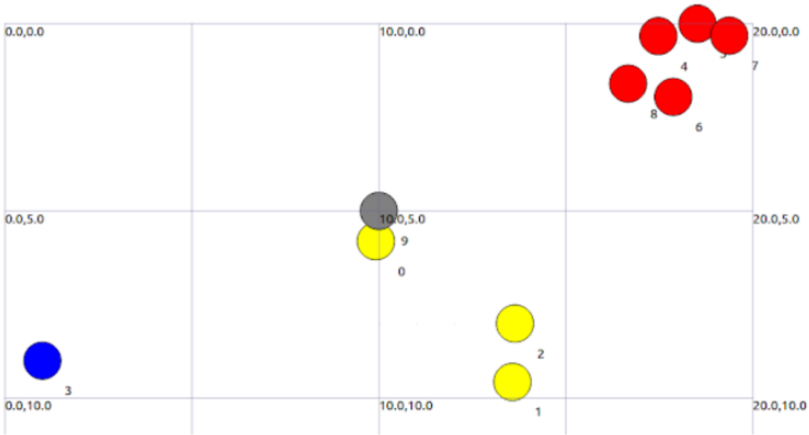


Figure 4.2: NetAnim screenshot.

PyViz

ns-3 PyViz is a live simulation visualizer, meaning that it uses no trace files. It is useful to figure out if mobility models are implemented as expected, and, in general, for debugging purposes. The documentation for this extension can be found at the link https://www.nsnam.org/wiki/PyViz. Some required python modules are no longer available for Ubuntu 18.04, thus it has been used with reduced functionalities in our machine. A screenshot is shown in Figure 4.3.

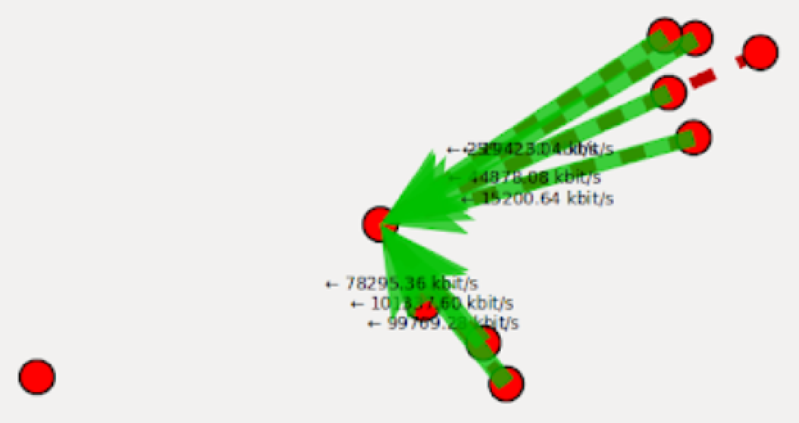


Figure 4.3: PyViz screenshot.

4.3.2 Performance Monitoring

In *ns-3*, the *FlowMonitor* module can be used to monitor the performance of the network. The goal of this module is to provide a flexible way to measure the performance metrics of the simulated networks. The module installs probes in network nodes, to track the flows of exchanged packets. Packets are divided according to the flow they belong to, and each flow is defined according to the probe's characteristics (e.g., for the IP protocol, the packets with the same {protocol, source (IP, port), destination (IP, port)} tuple belong to the same flow.

The statistics, collected for each flow, can be exported in a XML file or can be requested within the simulation script. In particular, for each flow we registered three parameters by using *FlowMonitor*:

- *txPackets*: the number of transmitted packets for a traffic flow. It is determined by the required throughput of a specific STA. Since we fixed the packet length, this metrics implies also the number of transmitted Bytes.
- *rxPackets*: the number of received packets for a specific flow. Given interference effects and congestion, some packets may be lost or not correctly received. Thus, the value of *rxPackets* gives an indication of the achieved throughput and of the packet error probability.
- *delaySum*: the sum of all E2E delays for all received packets of the flow. Dividing this value by the number of received packets we can obtain the average latency experienced by a packet for that precise flow.

In a realistic scenario, these values can be obtained in different ways without adding overhead to the traffic flow. For instance, the number of transmitted packets can be retrieved checking the packet sequence numbers and the latency can be given by the timestamp present in each packet, if time synchronization is provided. Otherwise, the parameters can be measured and reported periodically: some overhead would be added but would be limited if the report is done every few seconds.

These metrics were used for a twofold purpose. At run-time, they were considered to gain an understanding of the temporary state of the network. They were provided as input to our scheduling algorithms in order to take the most suitable decision according to the network condition. On the other hand, at the end of the simulation, these parameters were analysed to define the whole network performance and evaluate the efficiency of the utilized resource allocation technique.

Chapter 5 System Model

In this chapter we analyse the *ns-3* simulator, how it models the wireless communication, and how it has been used to carry out simulations. Particular attention is given to the assumptions and limitations characterizing the utilized *ns-3* models. Some experiments have been performed in order to test the functioning of the simulator and to better understand its behaviour. The observed results of these preliminary simulations are included in this chapter.

In the *ns-3* environment it is important to distinguish between modules and models:

- **Modules** are separated libraries which compose the entire software: individual *ns-3* programs can link the modules they need to build their scenarios and conduct their simulation;
- **Models** are abstract representations of real-world objects, protocols and devices. In general, a module consists of more models.

In our simulations, the main module utilized is the *Wi-Fi Module* documented in [ns3b] [ns3c]. The other used modules are: the *Buildings Module* to set up the indoor scenario, the *Propagation Module* to model the propagating signals' losses and delays, and the *Spectrum Module* for modeling the frequency-dependent aspects of communications. In the following sections, we provide a detailed description of the *Wi-Fi Module*, mentioning only the relevant properties of the other utilized modules.

5.1 ns-3 Wi-Fi Module

This library provides a set of 802.11 models which can be exploited to instantiate representations of Wi-Fi devices: the so called *WiFiNetDevices* objects. The six supported versions of the standard are described in Chapter 3: 802.11b, 802.11a, 802.11g, 802.11n (both 2.4 and 5 GHz bands), 802.11ac and 802.11ax (limited to 2.4 and 5 GHz bands). The implementation is modular and the available models characterize the different sublayers spanning from the communication channel to the high MAC layer. A representation of such architecture is shown in Figure 5.1.

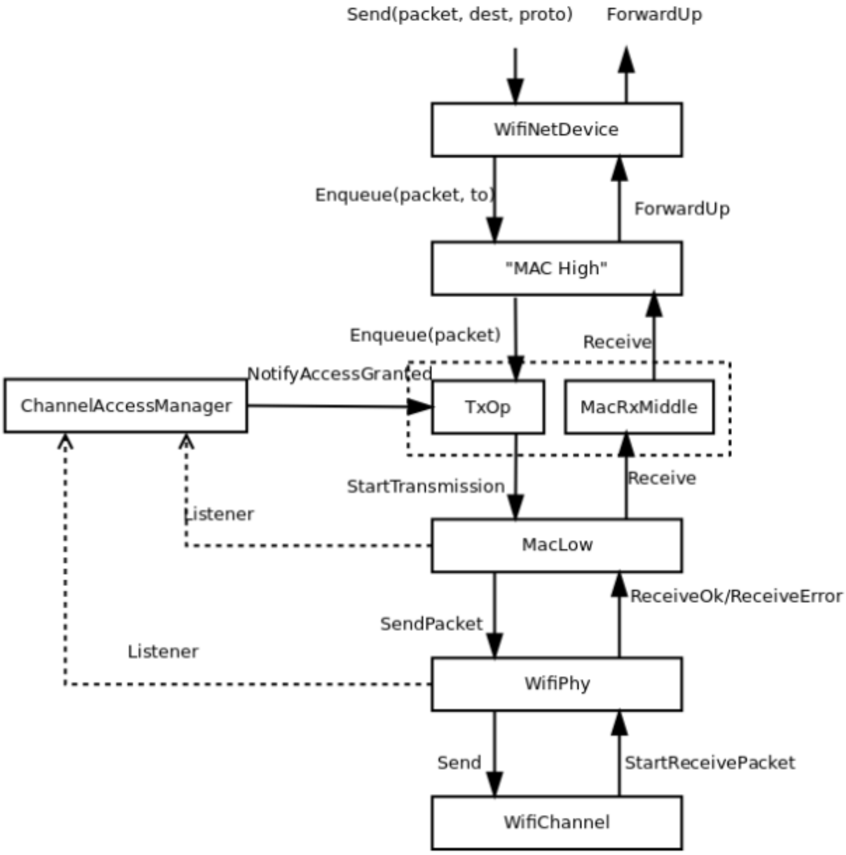


Figure 5.1: *WiFiNetDevice* architecture [ns3b].

ns-3 does not cover all aspects of the IEEE 802.11 standard, which is a quite large specification. There are two main limitations that we emphasize:

- 802.11ax OFDMA is not supported. The current implementation of this Wi-Fi version uses CSMA/CA access technique, as the previous standard realises.

- Directional antennas are not yet supported. As a consequence, 802.11ac/ax MU-MIMO is not supported and 802.11n/ac/ax beamforming is not supported.

In the following subsections, the main blocks of the *WiFiNetDevices* structure are described, with references to the most important C++ classes in the *ns-3* libraries.

5.1.1 Channel Models

The channel takes care of bringing the signal to all connected physical interfaces according to a propagation loss model and a propagation delay model. 15 propagation loss models are available and some have been tested during the various simulations carried out. When utilized, they will be specified case by case. In all the preliminary experiments presented in this chapter, the *LogDistancePropagationLossModel* has been employed with path loss exponent $n = 3$. This model calculates the reception power at distance d from the transmitter given the path loss:

$$L = L_0 + 10n \log_{10} \frac{d}{d_0} \tag{5.1}$$

where L_0 is path loss at reference distance d_0, set equal to 1 m by default. According to the Friis formula[1], L_0 is equal to ~47 dB in the 5 GHz band and ~40 dB when transmission takes place in the 2.4 GHz band.

The so called *ConstantSpeedPropagationDelayModel* has been always utilized in our scenarios as propagation delay model. According to this model, the propagation speed in the propagation medium is constant and set, by default, to the propagation speed of light in the vacuum ($c = 2.997\,92 \times 10^8$ m/s).

5.1.2 PHY Layer Models

The PHY layer is responsible for taking packets from the MAC and sending them onto the channel to which it is attached. It is also responsible for receiving packets from that channel, and for passing them up to the MAC in case of successful reception. There are two PHY layer models available in *ns-3*, configured with a common interface defined in the abstract class *WifiPhy*. *YansWifiPhy*, described in [LH06], is the suitable choice when the simulation does not involve frequency-dependent propagation effects and cross-channel interference is not relevant. On the other hand, *SpectrumWifiPhy*, based on [BM09], allows a fine-grained frequency representation of the wireless signals. Since in our scenarios interference between coexisting signals play a crucial role, we decided to employ the *SpectrumWifiPhy* class. The limitations of Yans model will be analyzed later in this chapter.

The PHY layer object works in conjunction with two other objects in order to evaluate the probability of error due to interference and noise, the *InterferenceHelper* and the *ErrorModel* respectively.

[1] The path loss according to the Friis formula is calculated as $L = 20 \log_{10} \frac{c}{4\pi d f_0}$ where c is the propagation speed of light in the vacuum, d is the distance between the transmitter and the receiver, and f_0 is the carrier frequency.

Interference Helper

The *InterferenceHelper* object tracks all incoming packets observed on the channel and calculates the error probability for packets being received given the amount of present interference. Furthermore, it evaluates whether energy on the channel rises above the CCA threshold. *ns-3* set this threshold to $-62\,\mathrm{dBm}$ by default, to reflect the standardized value.

Error Model

The error model calculates the probability of successful reception given the modulation and the standard version used. The current *ns-3* error rate models are only for Additive White Gaussian Noise (AWGN) channels; thus, fast fading effects are not taken into account. Three error models are present, implemented through the following classes:

- *DsssErrorRateModel* is dedicated to 802.11b Wi-Fi version, which utilizes the DSSS modulation technique;
- *YansErrorRateModel* characterizes also OFDM modulations and it is based on analytical results;
- *NistErrorRateModel* is an updated version of *YansErrorRateModel* since it is based on more recent studies. This is the default model for OFDM modes, and the selected one for this work.

5.1.3 MAC Layer Models

The MAC layer is further subdivided into two sublayers. The MAC-low models characterize functions such as DCF, RTS/CTS and ACK. On the other hand, the so-called MAC-high models implement the upper MAC functionalities: beacon generation, probing and association.

There are three MAC-high models, that provide for the three Wi-Fi topological elements:

- *ApWifiMac* implements an AP that generates periodic beacons, and that accepts attempts to associate.
- *StaWifiMac* implements a non-AP device (i.e. a STA). This element can be set to perform active probing or passive scanning of the channel in order to connect to an AP. An association state machine, represented in Figure 5.2, handles association and automatic re-association whenever too many beacons are missed.

- *AdhocWifiMac* is the third topological element used to implement a Wi-Fi MAC that, by default, does not perform any kind of beacon generation, probing, or association. It is not relevant for this work and has never been used in our simulations.

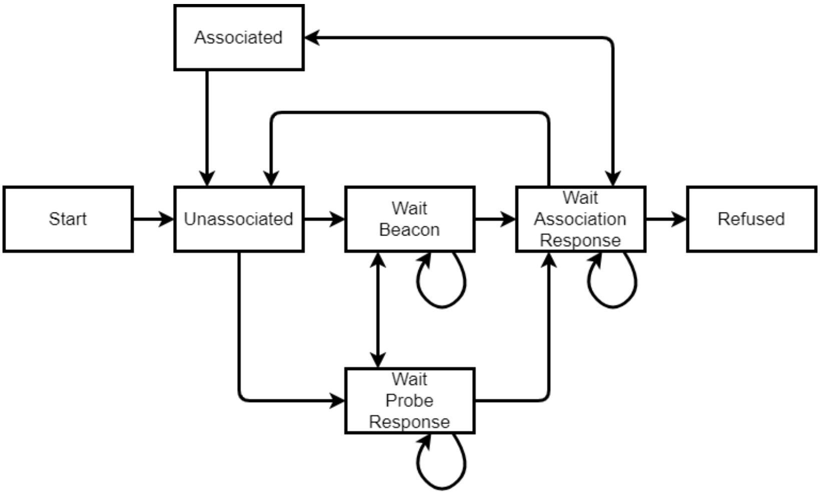


Figure 5.2: *StaWifiMac* state machine adapted from the *ns3::StaWifiMac* Class Reference in the Doxygen documentation.

Rate Control Algorithms

Among the MAC-high models we find also a set of rate control algorithms. *ns-3* implements a total of 12 algorithms to vary the MCS according to the channel condition. While some of them are used in real devices, others are only found in literature. In this work, two algorithms have been considered for being the only two compatible with the most recent versions 802.11ac/ax. These are:

- The *ConstantRateWifiManager* always uses the same transmission rate for every sent packet. The MCS index is fixed and adjusted at the beginning of the simulation. To date, this is the only rate control method available for 802.11ax devices.

- The *MinstrelHtWifiManager*, if 802.11ac standard is employed, is available in order to realize adaptive MCS in the network. It derives from the well known Minstrel algorithm. The rate is adapted based on statistics, collected by probing the environment, on the probability of successful transmission. The algorithm adapts the rate to the highest rate considered successful, and spends a fraction of time by trying other rates.

5.2 Transmitted Spectra

The spectra of the transmitted signals play a crucial role in our scenarios. Our final objective is to assign radio resources to users in an environment full of interfering signals, that must therefore be modelled in the frequencies domain with a sufficient level of realism. In order to understand how the Wi-Fi signals are represented in *ns-3*, a spectrum analyzer was implemented through the *SpectrumAnalyzer* class. This tool allowed to visualize the Power Spectrum Density (PSD) of the OFDM signals transmitted from a Wi-Fi station with different configurations. Two explanatory examples are shown in Figure 5.3.

In Figure 5.3a, the transmission occurs in the 5 GHz band on channel 36 (with corresponding center frequency of 5180 MHz) and with a channel bandwidth of 20 MHz. In Figure 5.3b the utilized channel is number 38 (center frequency equals 5190 MHz) and the channel bandwidth is 40 MHz.

It is clearly visible that spectrum mask imperfections are modelled as defined in the standard [80216]. In fact, the spectrum is attenuated by 20 dB at the boundaries of the channel (i.e. at ± 10 MHz with respect to the center frequencies in case of 20 MHz channel, and at ± 20 MHz when the channel bandwidth is 40 MHz). Another point of interest is the power density decrease of 40 dB. This happens 30 MHz away from the center frequency when the bandwidth is 20 MHz, and 60 MHz far in case of 40 MHz channel.

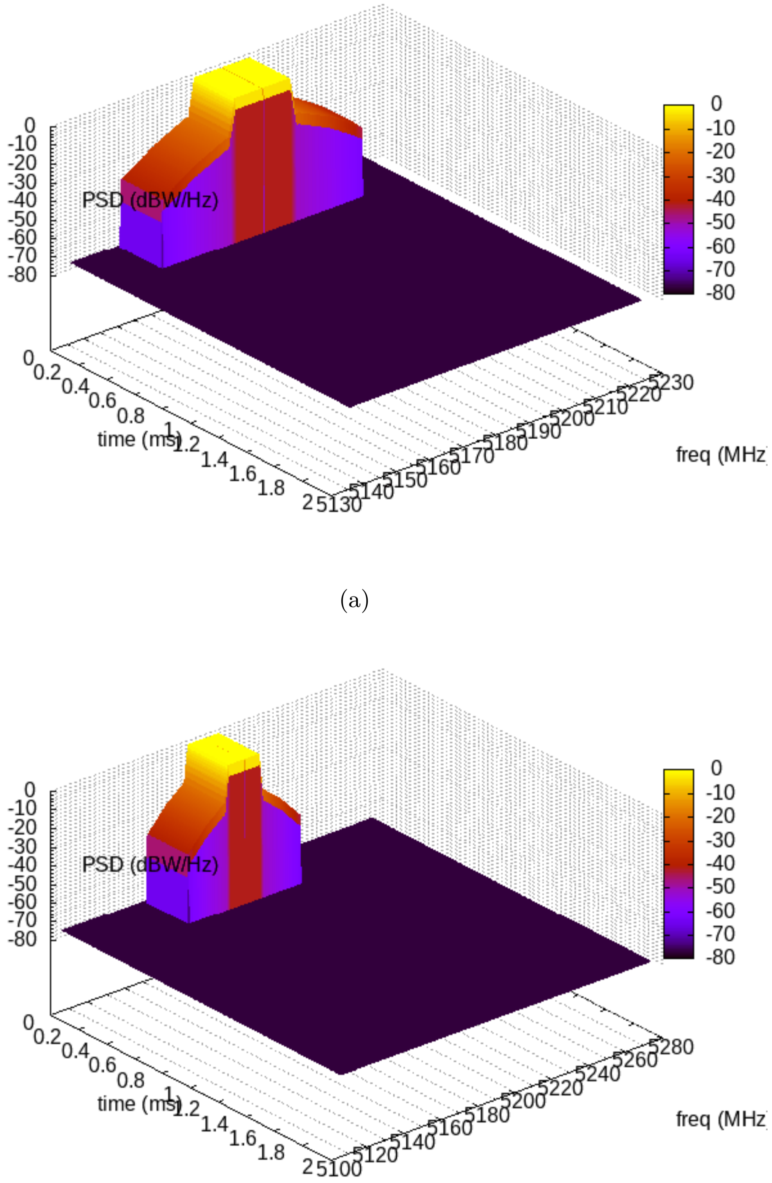
In conclusion, this model can be considered sufficiently realistic to perform experiments in which the main objective is to study the behaviour of interfering signals. Even if the single subcarriers composing the OFDM symbol are not modelled, the simulator reproduces the spectrum mask satisfactorily so that realistic cross-channel interference considerations can be made.

5.3 Single Channel Validation Experiments

Before analyzing the performance of two interfering Wi-Fi links, let us consider a scenario with only one link connecting two nodes: an AP and a STA. As a first validation experiment, we want to calculate the theoretical maximum bit rate achievable in a Wi-Fi channel using the standard 802.11ax, and compare it with the measured maximum throughput in simulations.

5.3.1 Theoretical Assessment

Version 802.11ax utilizes OFDM modulations as reported in the dedicated IEEE draft document. The spectrum of an OFDM signal, depicted in Figure 3.5, is a combination of subcarriers with a spacing of $\Delta f = 1/T$ from each other, where T is



(a)

(b)

Figure 5.3: 802.11ax transmitted spectra measured with a spectrum analyzer implemented in *ns-3*. The PSD trend is shown for two different channel bandwidths: 20 MHz in Figure 5.3a and 40 MHz in Figure 5.3b.

the symbol time interval. Let us now compute the bit rate for the OFDM scheme. Supposing there are N subcarriers in the channel, with bandwidth $B = N\Delta f$, not all of them are actually used. In fact, at the spectrum edges some of them are null to reduce adjacent channel interference. Let us denote as *active subcarriers* the non-null subcarriers, and their number N_a. Thus, the *subcarrier activation efficiency* is defined as:

$$\eta_a = \frac{N_a}{N}. \tag{5.2}$$

Another cause for reduced spectrum efficiency is the presence of the so called *pilot subcarriers*: they carry known symbols and are used for channel estimation at the receiver. Denoting their number N_p, we can introduce the so called *active subcarrier utilization efficiency* as:

$$\eta_p = \frac{N_a - N_p}{N_a}. \tag{5.3}$$

Now, assuming a coding rate r and a digital modulation scheme with cardinality M, the number of carried bits per OFDM symbol can be calculated:

$$N_b = r \log_2 M(N_a - N_p) = r \log_2 M \eta_a \eta_p N. \tag{5.4}$$

Since temporal dispersion can cause inter-OFDM symbol interference, to mitigate this problem, the GI T_g is introduced at the beginning of each OFDM symbol. Therefore, the total time assigned to each OFDM symbol is:

$$T_L = T + T_g \tag{5.5}$$

where T is the net time containing the OFDM information. We are now able to calculate the bit rate as:

$$R_b = \frac{N_b}{T_L} = r \log_2 M \eta_a \eta_p \frac{N}{T + T_g}. \tag{5.6}$$

The parameters r and M can be retrieved from Table 3.4 and depends exclusively on which MCS index is utilized. The value of N is also defined by the standard and depends on the channel bandwidth B. In particular, the standard defines the OFDM symbol duration $T = 12.8\,\mu\text{s}$ and from this value we can derive $N = TB$.

On the other hand, the values N_a and N_p can vary according to the network conditions. In fact, as analyzed by Khorov et al. [KKLB18], to enable OFDMA, the concept of RU is introduced in 802.11ax standard. As a result, depending on the number of users connected to the same AP, a single OFDM symbol can be divided in more RUs. Consequently, the number of *active subcarriers* and *pilot subcarriers* change with the number of RUs, that reflect the number of connected devices. The standardized N_a and N_p values for the best-case scenario (i.e. the one in which

B [MHz]	20	40	80	160
N	256	512	1024	2048
N_a	242	484	996	1992
N_p	8	16	16	32

Table 5.1: Values of N, N_a and N_p for the different channel bandwidths.

within a OFDM symbol there is only one RU, and it is assigned to the only present user) are reported in Table 5.1.

Using the Equation 5.6, we are now able to calculate the theoretical bit rate achievable for different configurations of MCS index, channel bandwidth and GI. Such results are reported in Table 5.3 (the top number in each cell).

5.3.2 Simulations and Comparison

In order to measure the achievable throughput when using *ns-3*, a simulation script has been created realizing the scenario shown in Figure 5.4. Only an AP and a STA connected through a Wi-Fi channel are present, and mobility is not yet taken into account, i.e. the devices have a fixed position.

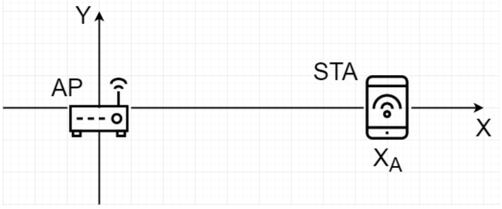


Figure 5.4: Graphical representation of the scenario created for the single channel validation experiments.

A UDP client is installed in the STA while an UDP server is placed in the AP. The client generates traffic towards the server with fixed data rate and packet size. The transmission power P_{TX} of the STA node and its distance x_A from the receiver have been chosen so that, even with the denser modulation schemes, the propagation loss does not degrade the performance. For the sake of reproducibility, all the parameters characterizing the Wi-Fi link are reported in Table 5.2. The throughput is returned taking into account the successfully received packets at the server side.

In order to fill the Table 5.3 with the experimental values of achievable throughput, a total of 96 settings have been simulated by varying the MCS index, channel bandwidth and GI. The obtained results are presented between parenthesis in Table 5.3.

Parameter	**Value**
Channel Number[2]	36, 38, 42 or 50
Data Rate[2]	150, 300, 600 or 1200 Mbit/s
Packet Size	1472 Bytes
PHY layer model	*SpectrumWifiPhy*
P_{TX}	20 dBm
RTS/CTS	Enabled
Seeds	1 – 5
Simulation Time	10 s
Standard – Band	802.11ax – 5 GHz
x_A	2 m

Table 5.2: Parameters used in the single channel validation experiment.

Each simulation has been run 5 times with different seeds and in all runs the same throughput (up to the third significant figure) has been measured. This means that, since the received power was significantly higher than the noise, the available channel has been fully exploited, without a notable occurrence of errors.

In conclusion, the observed values are perfectly aligned with the ones previously calculated. In Table 5.3, one can notice that, for each setting, the analytical bit rate is slightly higher than the measured throughput, and this difference increases for higher MCS indexes and larger bandwidths. However, this is expected due to the presence of network overhead. Indeed, the Equation 5.6 counts the number of PHY layer useful bits transmitted per second, while the throughput returned from the simulations considers the number of bits received per seconds at the UDP server application layer. The difference between the two values is justified by the presence of protocol headers added in the intermediate layers.

5.3.3 MCS index and Received Power

As a further experiment using the scenario represented in Figure 5.4, we want to understand which is the minimum received power P_{RX} that leads to a packet error probability below 0.001 for each MCS index. The importance of this experiment will be clear in Chapter 6, where these results will be used to setup a resource allocation algorithm.

The scenario consists of a STA connected to an AP and the parameters are set as reported in Table 5.4. For every MCS index, different settings have been simulated

[2] These parameters depend on the channel bandwidth value: 20, 40, 80 or 160 MHz respectively.

MCS	**Throughput** [Mbit/s]							
	20		**40**		**80**		**160** [MHz]	
	1600	**800**	**1600**	**800**	**1600**	**800**	**1600**	**800** [ns]
0	8.13	8.60	16.3	17.2	34.0	36.0	68.1	72.1
	(7.10)	(7.49)	(14.4)	(15.2)	(30.1)	(31.8)	(60.2)	(63.9)
1	16.3	17.2	32.5	34.4	68.1	72.1	136	144
	(14.5)	(15.3)	(29.0)	(30.7)	(60.6)	(64.3)	(119)	(125)
2	24.4	25.8	48.8	51.6	102	108	204	216
	(21.7)	(23.0)	(43.4)	(45.9)	(90.8)	(95.8)	(170)	(179)
3	32.5	34.4	65.0	68.8	136	144	272	288
	(29.0)	(30.8)	(58.1)	(61.6)	(119)	(125)	(220)	(231)
4	48.8	51.6	97.5	103	204	216	408	432
	(43.6)	(46.1)	(87.1)	(92.0)	(171)	(180)	(307)	(321)
5	65.0	68.8	130	138	272	288	544	576
	(58.1)	(61.6)	(114)	(120)	(220)	(231)	(382)	(399)
6	73.1	77.4	146	155	306	324	613	649
	(65.4)	(69.3)	(127)	(134)	(243)	(255)	(419)	(437)
7	81.2	86.0	162	172	340	360	681	721
	(72.6)	(76.9)	(140)	(147)	(266)	(279)	(453)	(471)
8	97.5	103	195	206	408	432	817	865
	(87.1)	(92.0)	(165)	(173)	(307)	(321)	(506)	(526)
9	108	115	217	229	454	480	907	961
	(96.2)	(101)	(180)	(190)	(333)	(349)	(547)	(567)
10	122	129	244	258	510	540	1021	1081
	(107)	(113)	(201)	(211)	(366)	(382)	(584)	(605)
11	135	143	271	287	567	600	1134	1201
	(118)	(125)	(218)	(229)	(395)	(412)	(627)	(648)

Table 5.3: Maximum theoretical bit rate achievable in a 802.11ax Wi-Fi link for different configurations, compared with the reached throughput in simulations (between parenthesis).

with increasing P_{TX}, and each of these has been run 5 times with 5 different seeds.

Parameter	**Value**
B	160 MHz
Channel Number	50
Data Rate	10 Mbit/s
Distance	10 m
GI	800 ns
MCS index	0 – 11
Packet Size	1472 Bytes
PHY layer model	*Spectrum WifiPhy*
P_{TX}	variable
Seeds	1 – 5
Simulation Time	10 s
Standard - Band	802.11ax - 5 GHz

Table 5.4: Parameters setting for the MCS experiments.

In Table 5.5, the thresholds are the minimum measured P_{RX} values with which a particular MCS index can be used. With those levels of P_{RX} (rounded up to the closest integer value) in all the 5 runs the measured packet error probability is below 0.001.

MCS	0	1	2	3	4	5	6	7	8	9	10	11
P_{RX} [dBm]	-69	-66	-63	-59	-56	-52	-50	-49	-44	-43	-39	-37

Table 5.5: Minimum P_{RX} values that allow to use the relative MCS indexes.

5.4 Interfering Channels Validation Experiments

Another *ns-3* script has been created in order to simulate the scenario illustrated by Figure 5.5. Three nodes are present: two STAs, named STA_A and STA_B, and one third node, named AP, containing two APs devices (AP_A and AP_B). The two APs have a different SSID and each of them is connected to its respective STA. The three nodes are positioned in the reference plane as following: the AP has coordinates $(0, 0)$, while STA_A and STA_B are placed in (x_A, y_A) and in (x_B, y_B), respectively. Mobility is not taken into account, this means that the nodes do not vary their position during a simulation. Similarly to the previous script, there are two UDP clients, installed in the two STAs, sending data to their respective servers situated in the APs.

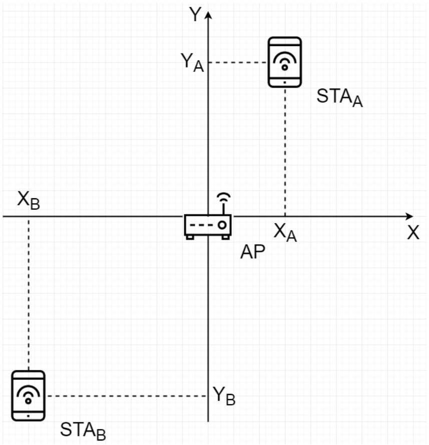


Figure 5.5: Graphical representation of the scenario created for the interference validation experiments.

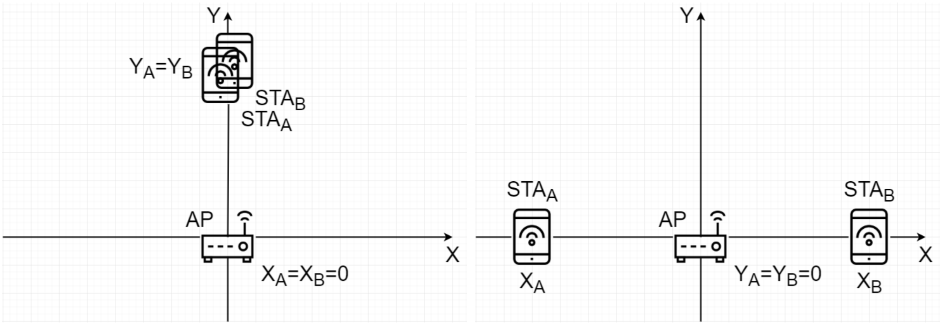
The main goal of the carried out simulations was to analyze how interference is modelled in *ns-3*. In particular, the achieved throughput in the two wireless links has been measured in order to understand how performance degrades in presence of another interfering signal.

Two opposite scenarios have been considered, as shown in Figure 5.6. In the first scenario (Figure 5.6a) the two STAs lie in the same point, distant $y_A = y_B = 10\,\text{m}$ from the AP. The interference listened by the two transmitters (i.e. the two STAs) during the channel sensing phase is, in this case, maximum.

On the other hand, the scenario in Figure 5.6b has been implemented to study how the network behaves in presence of the so called *hidden terminal problem.* This problem occurs when a node can communicate with the AP but is not in contact with another node that is communicating with that AP. Such a situation leads to difficulties in the MAC layer. In fact, multiple nodes can sense a clear channel and start sending data packets to the AP simultaneously, which creates interference resulting in neither packet getting correctly received. In this configuration, we set $x_A = -10\,\text{m}$ and $x_B = 10\,\text{m}$.

In the following subsections, six experiments are presented. The interference effects have been studied keeping the channel number 50 for link A, and varying from 40 to 60 the channel number of link B[3]. Each simulation has been run 15 times with different seeds, and the mean values are plotted in graphs 5.7 to 5.12. Unless

[3] Note that in the considered interval only even numbers are allowed as valid channel numbers according to the standard. This is also reflected in our *ns-3* implementation.



(a) Overlapping paths scenario.

(b) Hidden terminal scenario.

Figure 5.6: Graphical representation of the two studied scenarios with the interfering channels validation experiments.

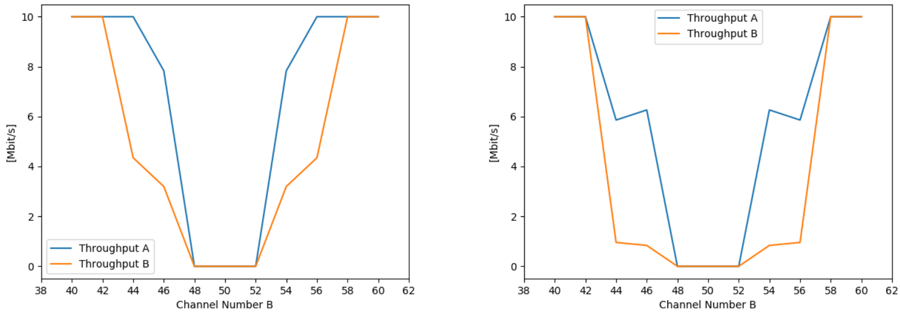
otherwise specified, each experiment has been carried out with a common setup, reported in Table 5.6.

Parameter	**Value**
B	20 MHz
Channel Number A	50
Channel Number B	40 to 60
Data Rate	10 Mbit/s
GI	800 ns
MCS index	5
Packet Size	1472 Bytes
PHY layer model	*SpectrumWifiPhy*
P_{TX}	16 dBm
Seeds	1 – 15
Simulation Time	15 s
Standard - Band	802.11ax - 5 GHz

Table 5.6: Parameters setting for the interfering channels validation experiments.

5.4.1 Experiment 1: A Naïve Approach

As mentioned in Chapter 3, in a Wi-Fi network, each AP shows its presence by sending beacon frames continuously in time. This happens, of course, also in our simulations to allow the STAs to identify and connect to their APs. The beacon interval has been set to 102.4 ms as recommended by the IEEE standard. In this experiment, we adopt a naïve approach in which the initial beacon starting time is always time 0 for all APs in the simulation. The employed multiple access technique is the required CSMA/CA, in its version without RTS/CTS signalling. This setting is the default setup suggested in the *ns-3* tutorial. The observed throughput are plotted in Figure 5.7a and Figure 5.7b for the two considered scenarios, respectively.



(a) Overlapping paths scenario.

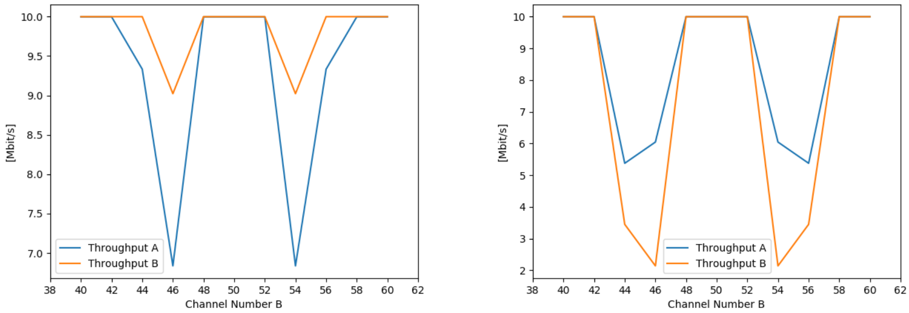
(b) Hidden terminal scenario.

Figure 5.7: Measured throughput for Experiment 1.

We can observe that the throughput in both scenarios is equal to the set data rate (10 Mbit/s) when the employed channels are far enough from each other, i.e. when the link B uses channels number 40, 42, 58 and 60. On the contrary, the throughput is null when the channels overlap, i.e. when the channel number B is 48, 50 or 52. In these cases, the beacon signals, having the same starting time, are always overlapping in time and in frequency and this collision makes the STAs not receive them properly. As a final remark, we can notice that also using the nearby non-overlapping channels, the performance is degraded due to the tails of the spectrum mask depicted in Figure 5.3. This degradation is stronger in the second scenario (Figure 5.7b) because RTS/CTS signalling is not employed, and the *hidden terminal problem* remains unsolved.

5.4.2 Experiment 2: Beacon Jitter

Considering the performance obtained in the previous simulations, let us now add the so called *beacon jitter* to our scenarios. *ns-3* allows to define, for each AP, the initial beacon starting time (after simulation time 0) with an uniform random variable distributed between 0 and the beacon interval. Also in this experiment, we do not use RTS/CTS frames to establish the communication. The measured throughput in AP_A and AP_B in these simulations are reported in Figure 5.8, again for both the considered scenarios.



(a) Overlapping paths scenario.

(b) Hidden terminal scenario.

Figure 5.8: Measured throughput for Experiment 2.

Similar to the previous simulations, when the channel numbers 40, 42, 58 and 60 are utilized the throughput is equal to the employed data rate. However, when approaching channel 50, the achieved throughput decrease with worse performance observed in Figure 5.8b. When overlapping channels are utilized (48, 50 or 52), the measured throughput is 10 Mbit/s, differing from the previous case. This happens because we broke the synchrony between beacon signals by adding the *beacon jitter*. Furthermore, when the communications take place on two (partially) overlapping channels, the throughput is maximum because the CSMA/CA protocol is able to detect properly the presence of interference, and collisions are indeed avoided.

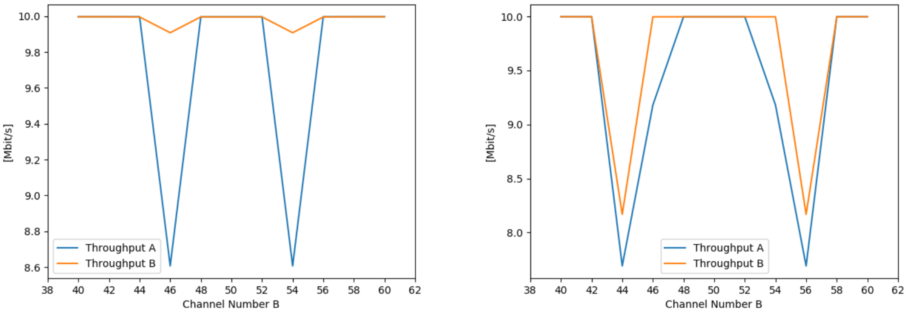
5.4.3 Experiment 3: RTS/CTS Handshake

In this third experiment, RTS/CTS communication was enabled in order to solve the *hidden terminal problem*. In addition, the *beacon jitter* was maintained since it brought a great performance improvement in the previous experiment. The outcome is shown in Figure 5.9.

The general trend of the throughput, depending on which channel has been employed by link B, reflects the one obtained in the previous experiment. However, as expected, a great improvement can be observed mainly in Figure 5.9b, representing the *hidden terminal problem* scenario.

5.4.4 Experiment 4: Saturated Channels

Let us now maintain the same setting as the previous experiment (*beacon jitter* and RTS/CTS signalling enabled) since it improved the reached performance. However, our new goal is to analyze the behaviours of the channels when they are saturated, i.e. the maximum throughput is reached. According to Table 5.3, for a 20 MHz channel,

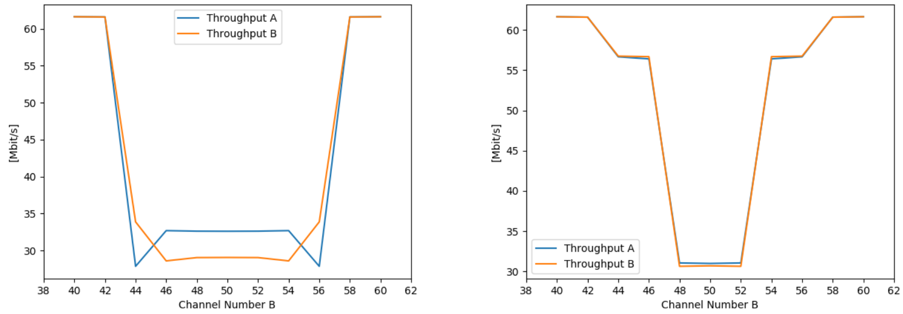


(a) Overlapping paths scenario.

(b) Hidden terminal scenario.

Figure 5.9: Measured throughput for Experiment 3.

using MCS index 5 and a GI equal to 800 ns, the achievable throughput is 61.6 Mbit/s. In order to reach this value, the data rate at which packets are generated by the client applications has been set to 100 Mbit/s. The obtained results are plotted for the two considered scenarios in Figure 5.10.



(a) Overlapping paths scenario.

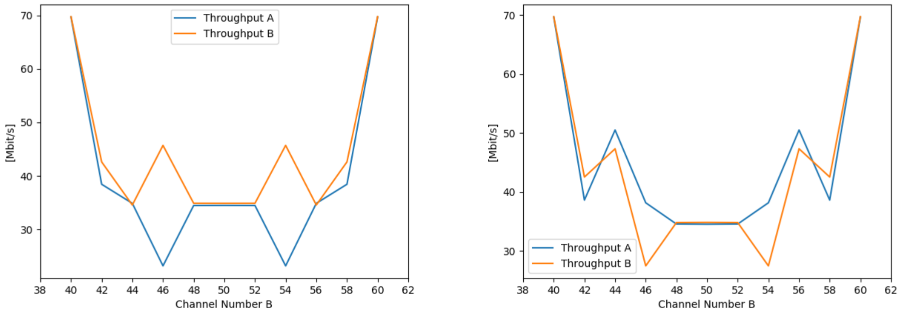
(b) Hidden terminal scenario.

Figure 5.10: Measured throughput for Experiment 4.

We observe that when the employed channel number B is far enough from channel 50, a throughput equal to 61.6 Mbit/s is reached in both the wireless links. Conversely, when interchannel interference is present, the channel capacity halves and ∼30 Mbit/s are achieved on each channel. This performance degradation is less significant in Figure 5.10b since the two transmitting STAs are spaced and less interference is perceived.

5.4.5 Experiment 5: 802.11ac with Adaptive MCS

In this experiment we want to analyze the behavior of the network when the MCS index is not constant, but is instead defined by an adaptation algorithm. Unfortunately, *ns-3* does not yet support adaptive MCS for the 802.11ax version. Thus, we use the 802.11ac standard, still in 5 GHz band, with its only available rate adaptation algorithm in *ns-3* environment, the *MinstrelHtWifiManager*. Once again, the data rate has been set to 100 Mbit/s in order to let the algorithm choose also from the highest MCS indexes, if necessary. Results are shown in Figure 5.11.



(a) Overlapping paths scenario.

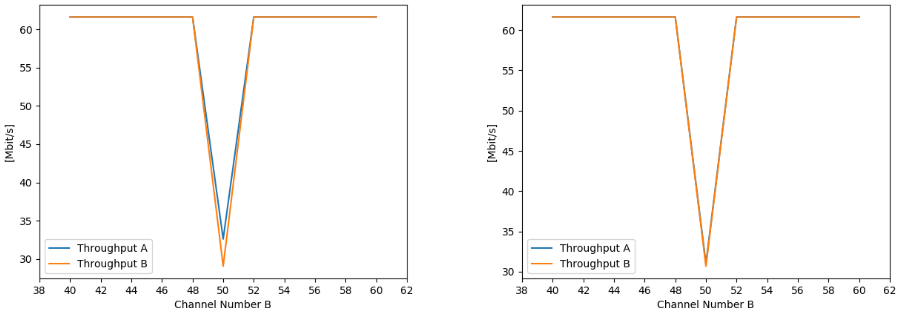
(b) Hidden terminal scenario.

Figure 5.11: Measured throughput for Experiment 5.

The coarse trend of the two graphs is similar to the one observed from the previous simulations. When channels 40 and 60 are employed for the communication on link B, a throughput equal to ∼70 Mbit/s is reached on both channels employing the highest MCS index. Instead, when the channels are overlapping, the performance halves reaching an average value of ∼35 Mbit/s per channel. Considering these values, we can deduce that the same MCS index is still used also when the channels are interfering. Thus, the only effect brought by the Minstrel algorithm was a lowering of the performance due to the “look around” actions in which it tries to use different rates. In the simulations that will follow, *MinstrelHtWifiManager* will be no longer used since it proved to be ineffective in managing interfering scenarios In fact, it takes into account only the noise superimposed to the signal and not the presence of other signals in the same spectrum.

5.4.6 Experiment 6: *YansWifiPhy* vs. *SpectrumWifiPhy*

This experiment is a repetition of Experiment 4 using *YansWifiPhy*, instead of the *SpectrumWifiPhy* model employed so far. *YansWifiPhy* is the default PHY layer model provided by *ns-3* for Wi-Fi communications and with this experiment we want to show how it is not realistic enough to model interfering scenarios. The observed maximum throughput is reported in Figure 5.12.



(a) Overlapping paths scenario.

(b) Hidden terminal scenario.

Figure 5.12: Measured throughput for Experiment 6.

From the graphs above we can notice that for both the scenarios the throughput is halved only when STA_B communicates on the channel number 50. Thus, it is clear that interference is considered only when the two interfering channels have the same channel number (50 in this case). This is unrealistic even with an ideally sharp spectrum mask which encloses all the spectrum in the 20 MHz band. In fact, considering a 20 MHz bandwidth, the two adjacent channels 48 and 52 are partially overlapped with the number 50. So, it is not physically possible to obtain the maximum throughput (61.6 Mbit/s) when these two channels are used.

Considering the unrealistic properties of *YansWifiPhy* model, the more complex *SpectrumWifiPhy* model will be used in this thesis for all the the simulations that will follow.

Chapter 6
Reference Scenario

In this chapter, we describe a reference scenario that is considered in our study. In this scenario, the E2E network slice problem is restricted to the radio access segment of the network. It consists of one AP which provides services to its associated STAs, in an indoor environment. Every STA requires a particular QoS and some slices have to be defined. Thus, our ultimate goal is to allocate radio resources to create different slices: in this way it will be possible to properly serve the connected STAs.

Only the uplink communication between the STAs and the AP is considered, since its scheduling is much more complicated than downlink. This differentiates our work from many other related studies, discussed in Section 6.3, which focus only on the downlink scheduling of resources. In a downlink scenario, the AP has the whole control and it can serve multiple users with well known broadcast techniques (e.g. space-time coding and beamforming if a multi antenna system is available in the AP). On the other hand, in uplink transmission the inputs are located in independent devices which do not have a broad view of the network.

6.1 Slices Definition

In today's Wi-Fi networks all the connected STAs communicate through only one channel offered by the AP. The IEEE 802.11e amendment introduces the Enhanced Distributed Channel Access (EDCA) to assign priority levels to different services. This channel access technique defines four traffic classes with increasing priority: AC_BK, AC_BE, AC_VI, AC_VO. This classification has some evident problems, as identified by Aleixedri et al., only four traffic classes are available, and this number is not scalable [ABCM19]. Furthermore, all the traffic is served by a single AP, and there is no guarantee of a certain percentage of airtime at every flow. In presence of the so-called rate anomaly problem, a traffic flow with high priority could consume too much resources, and optimality could be lost [TG04]. Thus, a different methodology is needed to serve prioritized flows.

As specified in Chapter 2, future generation networks are supposed to accommodate different type of services with diverging requirements. 3GPP, in the Technical Specification [3GP19], introduces the 5G QoS Identifiers (5QIs) to finely differentiate the various flows according to the required QoS. The 5QI is a scalar number used as a reference to a specific QoS forwarding behaviour to be provided to a traffic flow. Standardized 5QI values have one-to-one mapping to a standardized combination of 5G QoS characteristics as specified in Table 5.7.4-1 in [3GP19]. A simplified version of this table is reported here in Table 6.1.

Clause 5.7.4 of [3GP19] states that the standardized 5QI values are specified for services that are assumed to be frequently used and thus benefit from optimized signalling by using predetermined QoS characteristics. Dynamically assigned 5QI values require a signalling of QoS characteristics as part of the QoS profile.

In order to define QoS-based slices we considered these standardized classification methodologies and we adapted them to a 5G network scenario. The different types of 5G services can be grouped in three categories, as already mentioned in Chapter 2. Here, they are described more in detail:

- **Slice A – eMBB**: necessitates support for very high data rates and high data traffic volumes. The main issues taken into account are high user mobility and coverage. eMBB services are mapped in Table 6.1 with 5QI lower than 80. The standardized required throughput for this class of application is 100 Mbit/s in downlink, and 50 Mbit/s in uplink.

  In our scenario we will include between 2 and 6 STAs of this category. Mobility is taken into account moving the STAs at a pedestrian speed with variable direction. Indeed, this is the kind of mobility involved when a user is connected to a Wi-Fi network in an indoor scenario. For each device, the required uplink throughput is random, uniformly distributed between 80 and 100 Mbit/s.

- **Slice B – mMTC**: a large number of devices transmit low volumes of data that are not sensitive to delay and do not involve mobility. 3GPP does not map this class of services with standardized values of 5QI due to the low QoS needed. However, given the impact that this type of applications will have on future networks, we decided to implement it as a dedicated slice. The expected density is 1 000 000 devices/km^2 and the required throughput is in the interval 1 – 100 kbit/s. These values are reflected by the most popular Internet of Things (IoT) protocols. For example, LoRa features a raw data rate in the range 10^2 – 10^4 bit/s [AVTP$^+$17], while Narrowband IoT technology offers 10^2 – 10^5 bit/s [RMZ$^+$16].

  In the following simulations we consider 100 mMTC devices connected to the AP to reflect a density of about 1 device/m^2. Each of them transmits data at a

5QI Value	Packet Delay [ms]	Packet Error Rate	Example Services
1	100	10^{-2}	Conversational Voice
2	150	10^{-3}	Conversational Video
3	50	10^{-3}	Real-time Gaming, V2X
4	300	10^{-6}	Buffered Video Streaming
65	75	10^{-2}	Mission Critical Push-To-Talk
66	100	10^{-2}	Non-Mission Critical Push-To-Talk
67	100	10^{-3}	Mission Critical Video
71	150	10^{-6}	Live Uplink Streaming
72	300	10^{-4}	Live Uplink Streaming
73	300	10^{-8}	Live Uplink Streaming
74	500	10^{-8}	Live Uplink Streaming
76	500	10^{-4}	Live Uplink Streaming
5	100	10^{-6}	IMS Signalling
6	300	10^{-6}	Buffered Video Streaming
7	100	10^{-3}	Live Voice and Video Streaming
8	300	10^{-6}	Buffered Video Streaming
69	60	10^{-6}	Mission Critical Signalling
70	200	10^{-6}	Mission Critical Data
79	50	10^{-2}	V2X
80	10	10^{-6}	Augmented Reality
82	10	10^{-4}	Discrete Automation
83	10	10^{-4}	Discrete Automation, V2X
84	30	10^{-5}	Intelligent Transport Systems
85	5	10^{-5}	V2X
86	5	10^{-4}	V2X

Table 6.1: Standardized 5QI to QoS characteristics mapping adapted from Table 5.7.4-1 in [3GP19].

random speed in the range 30 – 50 kbit/s. Mobility is not considered for these devices, which maintain a fixed randomized position.

- **Slice C – URLLC**: is the class of services requiring ultra-low latency, high reliability and availability. In Table 6.1, URLLC applications are marked with the highest 5QIs: from 80 to 86. According to 3GPP release 15 the required latency for this class of applications is 1 ms and the reliability should be not lower than 99.9999%.

In our simulations there will be from 2 to 6 URLLC devices requiring a random throughput spanning from 20 to 40 Mbit/s. Furthermore, users of this slice move with a pedestrian trend.

6.2 Our Scenario

Figure 6.1 is a graphical representation of the scenario that we are going to study. All the STA devices are randomly placed in a rectangular room with dimensions 20 × 10 m, 1.5 m above the ground. The AP is situated at the center of the room at the ceiling level: 3 m high.

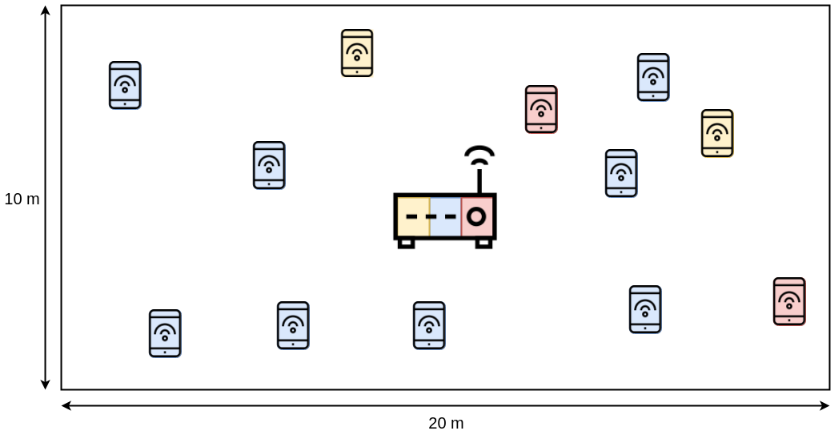


Figure 6.1: Graphical representation of the studied scenario. The slices A, B and C are depicted with different colours (for eMBB, mMTC and URLLC respectively).

Network slicing is realized by instantiating three radio channels and assigning each of them to one slice. The parameters characterizing these channels are adjusted based on the number of connected users and on the condition of the environment.

In the *ns-3* environment, it is possible to simulate the presence of buildings and rooms with the *Buildings Module*. Using this model we created a single room and located devices inside that. *HybridBuildingsPropagationLossModel* has been used as propagation loss model and *RandomWalk2dMobilityModel* was used to add mobility to nodes in slices A and C.

6.2.1 *HybridBuildingsPropagationLossModel*

This model is obtained through a combination of several well known pathloss models in order to mimic different environmental scenarios in urban and rural areas. Furthermore, it takes into account both outdoor and indoor communication since APs might be installed either within buildings or outside. For indoor communication, the model determines wall penetration losses and power loss coefficient according to the type of building.

In case of indoor-to-indoor communications, the ones we are interested in, the *HybridBuildingsPropagationLossModel* calculates the propagation loss considering two contributions: the loss according to the ITU P.1238 model and the internal walls loss. In case of residential building, the analytical expression of the loss given by the ITU P.1238 model is:

$$L_{P.1238} = 20 \log_{10} f + 28 \log_{10} d + 4n - 28 \tag{6.1}$$

where f is the frequency [MHz], d is the distance between the STA and the AP[1] and n is number of interposed floors. The total loss is calculated assuming that each single internal wall has a constant penetration loss $L_{siw} = 5\,\text{dB}$, and approximating the number of walls that are penetrated with the Manhattan distance (in number of rooms) between the transmitter and the receiver. In detail, let x_1, y_1, x_2, y_2 denote the room number along the x and y axis respectively for user 1 and 2; the loss L_{IWL} is calculated as:

$$L_{IWL} = L_{siw}(|x_1 - x_2| + |y_1 - y_2|) \tag{6.2}$$

In conclusion, as anticipated, the pathloss in case of indoor-to-indoor scenarios is given by $L = L_{P.1238} + L_{IWL}$.

In addition to the pathloss, the shadowing is modeled according to a log-normal distribution with variable standard deviation as function of the relative position (indoor or outdoor) of the devices involved. The mean of the shadowing μ_I is always $0\,\text{dB}$. In case of indoor-to-indoor communications, the default value of the standard deviation σ_I is 8. Thus, the shadowing effect can be represented as $X_I \sim N(\mu_I, \sigma_I^2)$.

Notice that in the simulations presented in Chapter 5 a different loss model was employed: i.e. the *LogDistancePropagationLossModel*. Since the objective of those preliminary simulations was to validate the simulator, a simpler model could be used. However, to simulate our scenario, the most realistic propagation model available in *ns-3* was chosen.

6.2.2 *RandomWalk2dMobilityModel*

Mobility is a crucial characteristic of today's network. We decided to consider its effect in our simulations in order to achieve more realistic results. As stated above, mMTC devices have a fixed position since this type of application is not likely to involve mobility. On the other hand, eMBB and URLLC services must deal with

[1] In this expression, it always holds $d > 1\,\text{m}$.

mobility. We decided to implement a pedestrian movement for these devices using the *RandomWalk2dMobilityModel*.

Each device moves with a speed and direction chosen with provided random variables until either a fixed distance has been walked or until a fixed amount of time has elapsed. If it hits one of the boundaries (specified by the 20 × 10 m rectangle), it rebounds on the boundary with a reflexive angle and speed. More precisely, in our implementation, every second the direction is chosen uniformly random in the range $[0,2\pi]$ radians and the speed in the interval 2 – 4 m/s.

6.3 Related Works

To date, only few studies have been performed on possible network slicing implementations in Wi-Fi contexts. Most of them are based only on a time scheduled resource allocation, which is not innovative and proved to be inefficient for resource utilization.

Aleixendri et al. present a scheduling algorithm to allocate airtime to different users, depending on their SLAs [ABCM19]. In their approach, each AP is equipped with a local scheduler which performs network slicing in time domain. A global scheduler is also present to manage a group of local schedulers and make adjustments in case some SLA is violated. The global scheduler is needed since it is able to manage interference with a broader view of the network. This algorithm separates resources in orthogonal pools, dividing them in time. Thus, it is able to ensure a good performance isolation between slices assigning them a minimum of airtime.

However, the main limitation of the study is that resource utilization will have worsened and network slicing is considered only along the time dimension. In this way, tailored resources are not assigned to each slice but only a certain percentage of airtime is dedicated. Furthermore, the work is limited only to the downlink transmissions. While it will be feasible to schedule time resources for downlink connections (since the AP has knowledge about the outgoing flows), it will be very complex to schedule the uplink traffic in an efficient and practical way. This second, more interesting, situation is not considered by the authors of the paper.

Similar models, which suffer of the same issues, have already been presented in previous studies such as the one presented by Richart et al. [RBS+17]. Again their objective was to allocate the airtime resource to different slices. In their approach, when a slice requests a percentage of the total airtime to an AP, if enough resources are available, the airtime is allocated to the slice.

Another similar methodology based on airtime slicing was studied by Zehl et al. in [ZZW17]. Their methodology provides complete traffic separation in downlink between the home network and the so-called guest network. Again, the main problem is that traffic is scheduled only for the downlink connection and the time division of resources does not provide any advantage from the resource utilization point of view.

De Bast et al. present a Deep Learning (DL) algorithm to enable network slicing [DBTDC$^+$19]. They consider a radio access segment of an IEEE 802.11ac network and aim at assigning the proper resources for each user depending on its required throughput. The resources managed by the algorithm are: the MCS index, the CCA threshold, and the transmission power. The main limitation of this approach is that all the STAs are connected to the same channel and will interfere each other. In this way, a good performance isolation will never be reached. Furthermore, the DL technique employed requires an extensive pre-training and a long time to converge on the optimal solution. This time is not available in a realistic scenario in which decisions must be taken at run-time by the scheduling algorithm.

Makhlouf et al. propose a totally new access scheme to realize network slicing [MH14]. They observe that the main reason for the inefficiency of Wi-Fi standards is that the MAC Layer randomly allocates the whole channel available to only one user as a single resource. For this reason they divide the entire channel in different sub-channels assigning them to the users according to their requirements and conditions. The problem of this approach is that it is not standard compliant, instead, another MAC technique is presented.

Finally, Gu et al. want to realize a multi-tenant architecture on a single AP by installing different SSIDs on it. The objective is to create coexisting separated network and increase the overall network throughput. Network slicing is not realized because each tenant is served with an opportunistic approach and users are not differentiated by their requirements.

Our solution wants to address the main problems present in the previously mentioned approaches. First of all, our solution is standard compliant, i.e. we discuss a general approach to realize network slicing which can be applied to many different technologies with their own access scheme. Our approach will not modify the PHY and MAC layer functionalities of the Wi-Fi standard. Furthermore, the overall resource utilization will be of paramount importance and we will analyze the performance of our method also under this point of view. To this end, we will not consider the downlink connections as in previous studies, which can be simply scheduled by the AP (i.e. the only transmitter). On the contrary, we will focus on the uplink traffic, often ignored due to its behaviour, unknown at the AP. As a final remark, our algorithm will act at run-time, adapting the slices according to the requirements without needing a pre-training.

6.4 Simulations

For the sake of reproducibility, in Table 6.2 are reported the parameters that are fixed for every carried out experiment. Three settings are considered with different number of STAs per slice:

1. **Setting 2-100-6**: `nStaA` = 2, `nStaB` = 100, `nStaC` = 6;
2. **Setting 4-100-4**: `nStaA` = 4, `nStaB` = 100, `nStaC` = 4;
3. **Setting 6-100-2**: `nStaA` = 6, `nStaB` = 100, `nStaC` = 2.

From now on, the writings `nStaA`, `nStaB` and `nStaC` denote the number of devices in slice A (eMBB), B (mMTC) and C (URLLC) respectively. The goal of varying these numbers is to consider different proportions among the number of users in the different slices. In this way, the study of the proposed algorithms will be more robust. Each setting is simulated 20 times with a different seed. The used seeds correspond to the integer numbers from 1 to 20, with some exceptions depending on the considered algorithm (introduced below). These exceptions are reported in the Appendix A. The required throughput and the initial position of each user is randomized every run.

The simulation time has been set to 15 s for every run. This value has been decided by the actual time needed to simulate the studied scenario in our machine: 15 s of simulation time, in which 108 STAs transmit data continuously, corresponded to about 2 h of running simulation. In this way, we had the possibility to run all the scenarios multiple times (with varying seeds) in a reasonable time. In addition, increasing the simulation time did not significantly influence the results because flows are defined at the start of each simulation. Furthermore, even at the lowest speed setting (2 m/s), a node could move across the entire room in 15 s, which should be representative of different radio conditions.

Running on the above mentioned settings, we analyzed three algorithms which perform different resource allocation to create slices:

- **One Channel Approach**: network slicing is not considered the STAs are connected to only one available channel;
- **Static network slicing**: resources are allocated at the beginning of the simulation and the slices are static;
- **Dynamic network slicing**: at every time interval T, the algorithm reallocates resources to adjust the slices based on the achieved performance and the network conditions.

Parameter	**Value**
Data Rate eMBB	$\sim U[80, 100]$ Mbit/s
Data Rate mMTC	$\sim U[30, 50]$ Kbit/s
Data Rate URLLC	$\sim U[20, 40]$ Mbit/s
Packet Size	1472 Bytes
PHY layer model	*SpectrumWifiPhy*
Positions (x, y)	$X \sim U[0, 20]$ m, $Y \sim U[0, 10]$ m
Seeds	1 – 20
Simulation Time	15 s
Standard - Band	802.11ax - 5 GHz

Table 6.2: Parameters setting for all the simulations.

The objective of the last two methodologies is to define the three channels, assigning them a bandwidth and a center frequency. Furthermore, the GI, the MCS index and the P_{TX} are set for each channel. The following chapter explains in detail the functioning of the above mentioned algorithms.

Chapter 7
Proposed solution

Our network slicing proposal is presented in this chapter. Two slicing algorithms are proposed: one static, naïve, and one dynamic, which takes elaborated decisions. They are compared with the one channel approach, representing the traditional access methodology implemented in today's Wi-Fi networks.

7.1 One Channel Approach

All STAs are connected to one unique channel, as in today's network in absence of network slicing. The channel should be as wide as possible, so the bandwidth is set to 160 MHz. Also the transmission power is the highest allowed in Europe by the IEEE standard: 20 dBm. This allows us to obtain the best Signal-to-Noise Ratio (SNR) but could create problems from the interference point of view.

To reach a trade-off between coverage and maximum achievable throughput, the MCS is set to 5. In fact, an higher MCS would not have allowed distant devices to properly communicate due to an insufficient SNR. On the other hand a lower MCS would have decreased the maximum throughput supported by the channel, as we can see from Table 5.3. The GI is kept high to decrease the inter-symbol interference. The values of the parameters used are summarized in Table 7.1.

Parameter	**Value**
B	160 MHz
Channel Number	50
GI	1600 ns
MCS index	5
P_{TX}	20 dBm

Table 7.1: Parameters setting for the one channel approach.

7.2 Static Algorithm

Now, three separated channels are created, one for each slice. These channels are placed in the 5 GHz region in order to have the maximum free distance between each other. More precisely, the channel number for Slice A is as low as possible; Slice B is allocated in the center of the 5 GHz, with channel number 100; and the channel of Slice C has the highest channel number. In this way, inter-channel interference is minimized. The channel bandwidth is defined considering the sum of the required throughput in each slice for a given number of STAs. The GIs, the MCS indexes and the transmission powers have the same fixed values as in the one channel case.

The pseudo-code listings reported in Algorithms 7.1, 7.2 and 7.3 show how resources are allocated for each slice at the beginning of the simulation. The variable `dataRateSumX` represents the total throughput required in the slice `X`, where `X` $\in$ {`A`,`B`,`C`}. Note that this number is compared with Table 5.3 to obtain the needed channel bandwidth[1]. Given the low data volumes required in slice B, the bandwidth is fixed to 20 MHz for this slice.

Algorithm 7.1 Static Algorithm Slice A (eMBB).

```
giA = 1600; // [ns]
mcsA = 5;
txPowerA = 20; // [dBm]
  if (dataRateSumA < Tab5.3[mcsA][0])
    channelWidthA = 20; channelNumberA = 36;
  else if (dataRateSumA < Tab5.3[mcsA][2])
    channelWidthA = 40; channelNumberA = 38;
  else if (dataRateSumA < Tab5.3[mcsA][4])
    channelWidthA = 80; channelNumberA = 42;
  else
    channelWidthA = 160; channelNumberA = 50;
```

Algorithm 7.2 Static Algorithm Slice B (mMTC).

```
giB = 1600; // [ns]
mcsB = 5;
txPowerB = 20; // [dBm]
channelWidthB = 20; // [MHz]
channelNumberB = 100;
```

[1] In Table 5.3, values between parenthesis have been considered.

Algorithm 7.3 Static Algorithm Slice C (URLLC).

```
giC = 1600; // [ns]
mcsC = 5;
txPowerC = 20; // [dBm]
if (dataRateSumC < Tab5.3[mcsC][0])
  channelWidthC = 20; channelNumberC = 161;
else if (dataRateSumC < Tab5.3[mcsC][2])
  channelWidthC = 40; channelNumberC = 159;
else if (dataRateSumC < Tab5.3[mcsC][4])
  channelWidthC = 80; channelNumberC = 155;
else
  channelWidthC = 160; channelNumberC = 144;
```

7.3 Dynamic Algorithm

Also in this approach three slices are instantiated by assigning one channel per slice. However, after the initialization of the channels, they are continuously updated at run-time every time interval T. We set $T = 1\,\text{s}$ in our simulations. The updating algorithm exploits the network conditions and the reached performance to adapt the slices to the needs of the network. Three specialized algorithms have been created for each slice type because of their distinct needs.

7.3.1 Algorithm Slice A (eMBB)

Slice A is initialized according to the Algorithm 7.4. The P_{TX} is set to 20 dBm, the maximum value, in order to use higher MCS indexes and to enable high data rates. For the same reason, the GI is set to the minimum value: 800 ns.

Before the transmissions start, by using signalling messages, the AP obtains the received power for every connected device. Thus, it finds the minimum value `rxPowerA_min` corresponding to the farthest STA. This value is compared with the ones reported in Table 5.5, and the maximum achievable MCS, called `mcsA_max`, is obtained. Now, the minimum channel bandwidth that can accommodate a throughput equal to `dataRateSumA`, with GI = 800 and MCS = `mcsA_max`, is obtained from Table 5.3. Finally, fixed the channel bandwidth, we will use the minimum MCS that still supports the required throughput, again looking at Table 5.3.

After the initialization phase, the Algorithm 7.5 is called to update the slice according to the network conditions every interval of time T. First of all, the packet error probability experienced in each flow during the last interval T is calculated and

Algorithm 7.4 Dynamic Algorithm Slice A (eMBB) – Initialization.

```
txPowerA = 20; // Maximum Ptx [dBm]
giA = 800; // Minimum GI [ns]

// max MCS according to Tab5.5 and rxPowerA_min
for (int i = 0; i < 12; i++)
{
  if (Tab5.5[i] > rxPowerA_min)
  {
    mcsA_max = i - 2;
    break;
  }
  mcsA_max = 11;
}

// min channelWidth according to Tab5.3 and dataRateSumA
if (dataRateSumA < Tab5.3[mcsA_max][1])
  channelWidthA = 20; channelNumberA = 36;
else if (dataRateSumA < Tab5.3[mcsA_max][3])
  channelWidthA = 40; channelNumberA = 38;
else if (dataRateSumA < Tab5.3[mcsA_max][5])
  channelWidthA = 80; channelNumberA = 42;
else
  channelWidthA = 160; channelNumberA = 50;

// min MCS allowed by channelWidthA from Tab5.3
for (int i = 0; i < 12; i++)
{
  if (dataRateSumA < Tab5.3[i][2*log2(channelWidthA/10)-1])
  {
    mcsA_min = i;
    break;
  }
  mcsA_min = 11;
}

// min MCS
if (mcsA_max >= mcsA_min + 1) mcsA = mcsA_min + 1;
else mcsA = mcsA_max;
```

compared with a threshold:

$$P_e = \frac{txPackets - rxPackets}{txPackets} \overset{?}{\leq} 0.02. \tag{7.1}$$

The threshold has been set equal to 0.02 after preliminary experiments based on the one channel approach. Only about a tenth of the STAs reached an error probability lower than 0.02. Thus, we used this value to define our SLA, and based on the result obtained in Equation 7.1, the algorithm takes a decision on the required bandwidth for next interval of time. We can study the behaviour of the algorithm by looking at the state machine represented in Figure 7.1.

Every T, the needed bandwidth is recomputed as in Algorithm 7.4, and it is multiplied by the value `channelWidthMulA` which can be `1` (no need to duplicate the bandwidth), or `2` (the bandwidth must be duplicated). A transition from state `1` to state `2` occurs if the Equation 7.1 does not hold for at least one STA, and the total error probability in the last T is worse than in the previous interval. On the other hand, a transition from state `2` to state `1` occurs if Equation 7.1 holds for every STA and the total error probability in the last T is better than in the previous interval. Once the bandwidth has been fixed, the MCS which will be used is determined as in Algorithm 7.4.

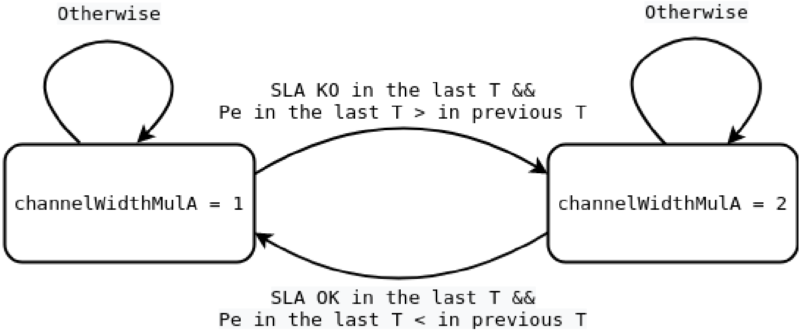


Figure 7.1: State machine implemented by the dynamic algorithm for resource allocation in slice A (eMBB).

7.3.2 Algorithm Slice B (mMTC)

Slice B is initialized according to Algorithm 7.6. The objective of this algorithm is to minimize the transmission power since energy saving is the crucial problem in IoT applications and mMTC communications.

Since a low throughput is always required, the channel bandwidth for this slice has been fixed to 20 MHz, the smallest possible in 802.11ax. The channel number is

Algorithm 7.5 Dynamic Algorithm Slice A (eMBB).

```
if (SLA KO in the last T && Pe in the last T > in previous T)
  channelWidthMulA = 2;
if (SLA OK in the last T && Pe in the last T < in previous T)
  channelWidthMulA = 1;

// max MCS according to Tab5.5 and rxPowerA_min
for (int i = 0; i < 12; i++)
{
  if (Tab5.5[i] > rxPowerA_min)
  {
    mcsA_max = i - 2;
    break;
  }
  mcsA_max = 11;
}

// min channelWidthA according to Tab5.3 and dataRateSumA
if (dataRateSumA < Tab5.3[mcsA_max][1])
  channelWidthA = 20*channelWidthMulA;
else if (dataRateSumA < Tab5.3[mcsA_max][3])
  channelWidthA = 40*channelWidthMulA;
else if (dataRateSumA < Tab5.3[mcsA_max][5])
  channelWidthA = 80*channelWidthMulA;
else
  channelWidthA = 160;
if (channelWidthA == 20) channelNumberA = 36;
else if (channelWidthA == 40) channelNumberA = 38;
else if (channelWidthA == 80) channelNumberA = 42;
else channelNumberA = 50;

// min MCS allowed by channelWidthA from Tab5.3
for (int i = 0; i < 12; i++)
{
  if (dataRateSumA < Tab5.3[i][2*log2(channelWidthA/10)-1])
  {
    mcsA_min = i;
    break;
  }
  mcsA_min = 11;
}

// min MCS
if (mcsA_max >= mcsA_min + 1) mcsA = mcsA_min + 1;
else mcsA = mcsA_max;
```

100, at the center of the 5 GHz region far from the other two slices. Also the GI is fixed and can be relaxed to 1600 ns since there are not strict requirements on latency. The lowest MCS is selected according to Table 5.3, with a margin `mcsAddB` initialized to 1. Once the MCS is fixed, Table 5.5 is used to find the minimum required received power. From this value, and the losses experienced by the users, the minimum usable transmission power is found[2].

The transmission power is set to accommodate the requirements of 90% of the users. We assume that having re-transmissions for a tenth of the users in this slice is not an issue. The data volumes are small and there are not stringent low latency needs. So, in the vector of the increasingly sorted losses (`lossB` in the algorithm), the 90^{th} is considered to calculate the required P_{TX} for slice B. We can see from the algorithm that a margin `txPowerAddB` (initialized to 3) is added to the minimum power needed. As explained below, this parameter is modified at run-time to adjust the transmission power based on the obtained QoS.

Algorithm 7.6 Dynamic Algorithm Slice B (mMTC) – Initialization.

```
mcsAddB = 1; txPowerAddB = 3;
channelWidthB = 20; // Minimum bandwidth [MHz]
channelNumberB = 100;
giB = 1600; // Maximum GI [ns]

// min MCS according to Tab5.3 + mcsAddB
for (int i = 0; i < 12; i++)
{
  if (Tab5.3[i][0] > dataRateSumB/1000)
  {
    mcsB = i + mcsAddB;
    break;
  }
  mcsB = 11;
}

// min txPower according to Tab5.5 + txPowerAddB
txPowerB = lossB[nStaB-nStaB/10-1] + Tab5.5[mcsB] + txPowerAddB;
```

Algorithm 7.7 is called to update the slice according to the network conditions, every interval of time T. The error probability is calculated for each flow as in Equation 7.1 and compared with the threshold 0.2.

[2] The losses are obtained by the AP by fixing the transmission powers of the STAs for the signalling messages. After having measured the received powers, we will have: $loss = P_{TX} - P_{RX}$ $[dB]$.

If Equation 7.1 does not hold in at least 90% of the STAs, and the average error probability is getting worse with respect to the previous interval T, the algorithm reacts by increasing the margin txPowerAddB. When it reaches its maximum value, the mcsAddB margin is also increased. In this way, the time on air required to each transmission will decrease and the number of collisions due to interference will be reduced. Of course, this will require a higher transmission power.

On the other hand, when Equation 7.1 is satisfied in 90% of the STAs, and the overall error probability is lower that in the previous interval T, the transmission power can be decreased. In a dual way, the algorithm acts on the parameters txPowerAddB and mcsAddB to relax the assigned resources.

Algorithm 7.7 Dynamic Algorithm Slice B (mMTC).

```
if (SLA KO in the last T && Pe in the last T > in previous T)
{
  if (txPowerAddB < 6) txPowerAddB++;
  else if (mcsAddB < 4) {mcsAddB++; txPowerAddB = 3;}
}

if (SLA OK in the last T && Pe in the last T < in previous T)
{
  if (txPowerAddB > 1) txPowerAddB--;
  else if (mcsAddB > 1) {mcsAddB--; txPowerAddB = 3;}
}

// min MCS according to Tab5.3 + mcsAddB
for (int i = 0; i < 12; i++)
{
  if (Tab5.3[i][0] > dataRateSumB/1000)
  {
    mcsB = i + mcsAddB;
    break;
  }
  mcsB = 11;
}

// min txPower according to Tab5.5 + txPowerAddB
txPowerB = lossB[nStaB-nStaB/10-1] + Tab5.5[mcsB] + txPowerAddB;
```

7.3.3 Algorithm Slice C (URLLC)

Slice C is initialized according to Algorithm 7.8. This mechanism is similar to the one presented for Slice A, with two differences:

- The channel numbers selected for Slice A are the lowest, whilst in Slice C are the highest. This because we want to allocate the three channels as distant as possible from each other. So, Slice A and Slice C are situated at the extremes of the 5 GHz band, and Slice B exactly in the middle.
- The MCS is obtained in the last operation of the algorithm in a different way. In Slice A, the minimum MCS able to support the required throughput is chosen, here we select the highest one usable according to the received powers. This difference is due to the fact that an high MCS should decrease the latency, an important aspect of the URLLC slice. At the same time, it can be reason of an increasing error probability, as we will learn from the conducted simulations.

At every interval of time T, Algorithm 7.9 is called to update the slice. The objective of this algorithm is to always recompute the maximum MCS that can be used with the maximum transmission power. Experimentally it has been found that a duplication in the bandwidth in case the requirements are not fulfilled, do not bring any positive effects. In fact, the modulation with the highest cardinality should be always used to allow low latency, and adding more bandwidth only resulted in a worsening of the resource utilization.

Algorithm 7.8 Dynamic Algorithm Slice C (URLLC) – Initialization.

```
txPowerC = 20; // Maximum Ptx [dBm]
giC = 800; // Minimum GI [ns]

// max MCS according to Tab5.5 and rxPowerC_min
for (int i = 0; i < 12; i++)
{
  if (Tab5.5[i] > rxPowerA_min)
  {
    mcsA_max = i - 2;
    break;
  }
  mcsA_max = 11;
}

// min channelWidth according to Tab5.3 and dataRateSumC
if (dataRateSumC < Tab5.3[mcsC_max][1])
  channelWidthC = 20; channelNumberC = 161;
else if (dataRateSumC < Tab5.3[mcsC_max][3])
  channelWidthC = 40; channelNumberC = 159;
else if (dataRateSumC < Tab5.3[mcsC_max][5])
  channelWidthC = 80; channelNumberC = 155;
else
  channelWidthC = 160; channelNumberC = 144;

// min MCS allowed by channelWidthC from Tab5.3
for (int i = 0; i < 12; i++)
{
  if (dataRateSumC < Tab5.3[i][2*log2(channelWidthC/10)-1])
  {
    mcsC_min = i;
    break;
  }
  mcsC_min = 11;
}

// max MCS
if (mcsC_max >= mcsC_min + 1) mcsC = mcsC_max;
else mcsC = mcsC_min + 1;
```

Algorithm 7.9 Dynamic Algorithm Slice C (URLLC).

```
// max MCS according to Tab5.5 and rxPowerC_min
for (int i = 0; i < 12; i++)
{
  if (Tab5.5[i] > rxPowerC_min)
  {
    mcsC_max = i - 2;
    break;
  }
  mcsC_max = 11;
}

// min channelWidthC according to Tab5.3
if (dataRateSumC < Tab5.3[mcsC_max][1])
  channelWidthC = 20; channelNumberC = 161;
else if (dataRateSumC < Tab5.3[mcsC_max][3])
  channelWidthC = 40; channelNumberC = 159;
else if (dataRateSumC < Tab5.3[mcsC_max][5])
  channelWidthC = 80; channelNumberC = 155;
else
  channelWidthC = 160; channelNumberC = 144;

// min MCS allowed by channelWidthC from Tab5.3
for (int i = 0; i < 12; i++)
{
  if (dataRateSumC < Tab5.3[i][2*log2(channelWidthC/10)-1])
  {
    mcsC_min = i;
    break;
  }
  mcsC_min = 11;
}

// max MCS
if (mcsC_max >= mcsC_min + 1) mcsC = mcsC_max;
else mcsC = mcsC_min + 1;
```

Chapter 8

Performance Evaluation and Results

In this chapter, we present and discuss the results obtained by simulating the resource allocation algorithms described in the previous chapter. We consider the three algorithms running on the three different settings, as discussed in Section 6.4. Firstly, we will analyze these configurations from the performance point of view, looking at the packet error probability and the latency.

For each flow, the packet error probability is calculated as:

$$P_e = \frac{txPackets - rxPackets}{txPackets} \tag{8.1}$$

where *txPackets* and *rxPackets* are the total number of transmitted and received packets within the flow, respectively.

In the *ns-3* terminology, which we adopt, a packet is considered transmitted when it is generated by the source, i.e. by the client application running on the STA. So, the value *txPackets* for a traffic flow depends solely on the data rate for that flow, which is randomly initialized according to Table 6.2.

The latency is obtained from the *delaySum* value retrieved using the *flowMonitor* module, i.e. the sum of all E2E delays for all the received packets of the flow. Thus, the average latency experienced by each packet in a flow is determined as:

$$L = \frac{delaySum}{rxPackets}. \tag{8.2}$$

Finally, in Section 8.3, the resource utilization is discussed. Where we verify that possible performance improvements are not caused by an over-provisioning of resources, such as bandwidth and transmission power.

8.1 Elements of a Box Plot

The results are presented in the form of box plots: a graphical method used in descriptive statistics to depict groups of numerical data. A box plot, shown in Figure 8.1, contains five numbers representative of the whole set of values:

- *min*: the lowest value, excluding any outliers;
- Q_1: the lower quartile, i.e. the median of the lower half of the dataset;
- *median*: the median of the dataset;
- Q_3: the upper quartile, i.e. the median of the upper half of the dataset;
- *max*: the highest value, excluding any outliers.

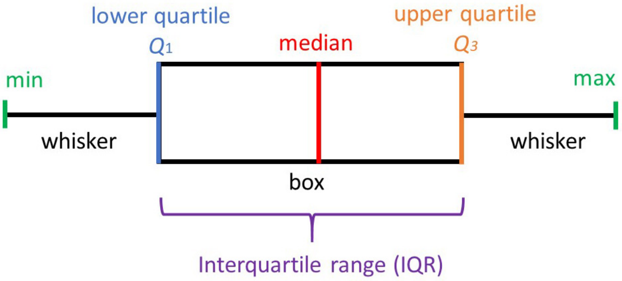


Figure 8.1: Box plot elements.

The function *.boxplot* of the Python library *matplotlib* has been used to create such diagrams. We operated with its default configuration, in which the *min* value is defined as the lowest data above $Q_1 - 1.5IQR$, where $IQR = Q_3 - Q_1$ is the interquartile range. This reflects the Tukey's original definition of box plots. On the other hand, the *max* value is defined as the highest data below $Q_3 + 1.5IQR$. Results below the *min* value, or above the *max*, are considered as outliers.

8.2 Performance Analysis

The diagrams in this section are intended to evaluate the performance of our algorithms studying the packet error probability and the latency obtained for all traffic flows. In the following figures, six box plots are shown representing the values achieved using: *(1)* the one channel approach, *(2)* the static network slicing algorithm and *(3)* the dynamic network slicing one.

For each configuration, we report results both with and without mobility. However, in both cases the position of the AP does not change. In the static scenarios the positions of the STAs are randomized at the beginning of each run and remain constant for the entire simulation. When mobility is considered, after an initial random positioning, eMBB and URLLC devices move according to the pedestrian model explained in Chapter 6.

8.2.1 Packet Error Probability

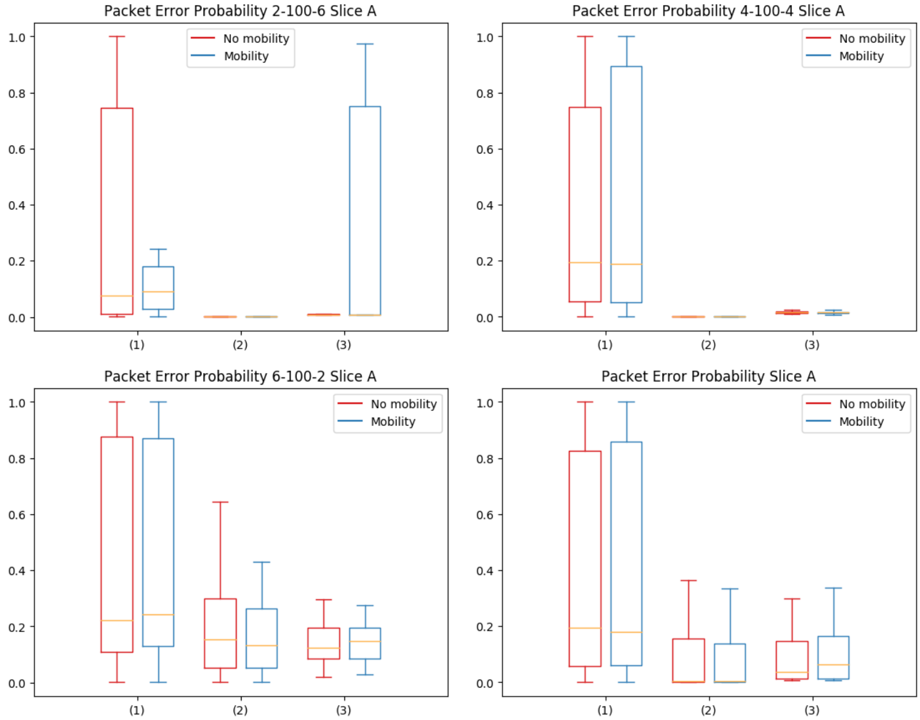


Figure 8.2: Packet error probability for slice A (eMBB): (1) one channel approach, (2) static network slicing, (3) dynamic network slicing.

In Figure 8.2, the packet error probabilities experienced by eMBB devices (slice A) are reported. The first three diagrams show the results for the three settings introduced in Section 6.4 (i.e. 2-100-6, 4-100-4, 6-100-2), where the number of eMBB and URLLC users varies from one setting to another. The fourth diagram shows the overall results obtained by merging data from all the settings.

First of all, when network slicing is not implemented, and only one channel is utilized, the network is highly congested. All the devices transmit on the same channel and send their RTS messages to the AP. The AP can serve one user at a time, and it will not reply with a CTS signal if it is busy. The presence of many RTS messages creates interference which degrades the error probability, which can be as high as 100% for some flows.

In the setting 2-100-6, when mobility is involved, the performance for the one channel approach is significantly better than in the other settings: the maximum packet error probability is ~ 0.2. This could be caused by lucky positions reached by the only 2 eMBB STAs in the scenario thanks to mobility. For instance, if the devices approach each other, the hidden terminal problem could disappear and interference due to signalling could decrease (since the RTS message is no more sent when the channel is busy).

The static network slicing approach decreases the error probability significantly. However, when 6 eMBB STAs are present, the packet error probability can reach 0.7 in some flows. It means that interference from the other stations still degrades the communication when many devices are present in this slice.

On the other hand, in the setting 6-100-2, the dynamic slicing algorithm slightly outperforms the static one. In fact, in this case a better performance can be reached enlarging the channel bandwidth. With more resources assigned, we can avoid to saturate the channel with 6 transmitting devices. The dynamic algorithm does not perform equally well when 2 STAs are present in the slice A and mobility is involved. However, the median packet error probability is very low. Thus, we can assume that an high upper quartile is caused by an unlucky positioning of the STAs occurred in few runs.

By using network slicing techniques, the performance definitely improves: in Table 8.1 the median error probabilities are quantified. From these results, we can observe that, in general, by doubling the allocated bandwidth with the dynamic algorithm does not help in decreasing the error probability. However, as we will see, it will greatly reduce the latency for this slice.

The same results are plotted in Figure 8.3 for slice B, i.e. the mMTC devices. The number of STAs belonging to slice B is fixed, and equal to 100, in every setting. Thus, we can notice that the box plots values vary slightly from a setting to another when one channel is considered. In particular, when this approach is applied, the error probability is almost always lower than 0.2, even though long whiskers are present between the upper quartile and the *max* value.

The packet error probability in slice B is always zero when the static network slicing algorithm is considered. The number of transmitted packets is very low and interference does not affect the correct reception of information. Thus, in all the runs, we had no losses.

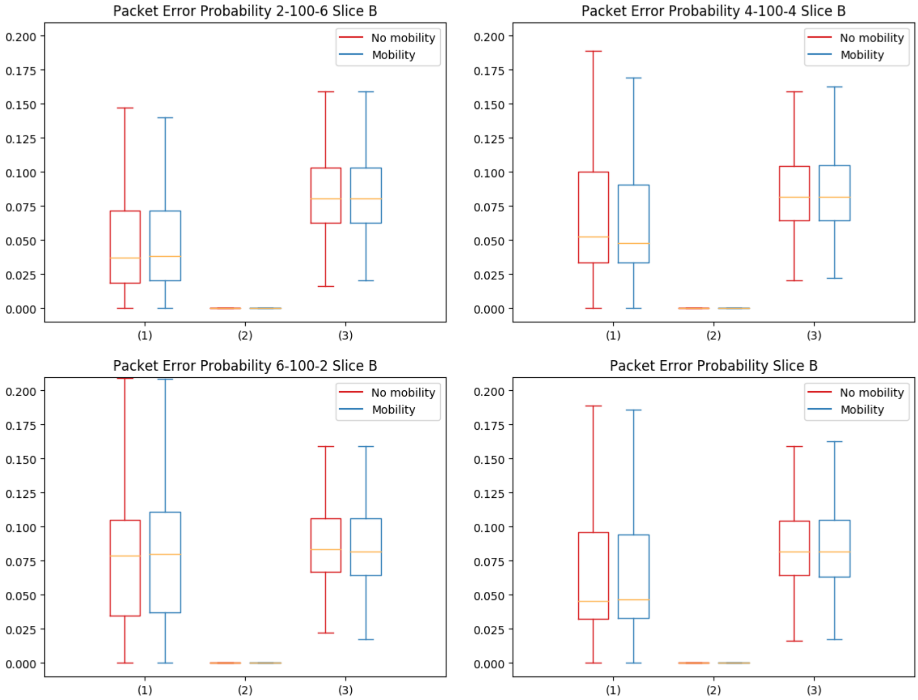


Figure 8.3: Packet error probability for slice B (mMTC): (1) one channel approach, (2) static network slicing, (3) dynamic network slicing.

With respect to the one channel approach, when dynamic network slicing is considered, the packet delivery performance worsens slightly. This is due to the adaptive transmission power scheme, which aims to save energy. A lower transmission power results in a greater SNR, which makes the error rate worse than the static approach. Later in this chapter, we will analyse how much energy is saved by the dynamic network slicing algorithm, and if this justifies the resulting performance deterioration. Also for this slice, the median packet error probabilities obtained with the three algorithms are reported in Table 8.1.

Finally, let us consider the packet error probability experienced by the URLLC devices, reported in Figure 8.4. Again, when all the devices communicate through only one channel, as it is in today's Wi-Fi networks, we get the worst error probability. As we can see from the box plots, it reaches the value 1 in some flows.

On the other hand, network slicing techniques outperform by far the one channel approach. Among the two slicing approaches proposed, the dynamic one achieves the worst performance. This algorithm employs an higher MCS index to decrease the latency in the URLLC slice. As a result, when the modulation cardinality increases

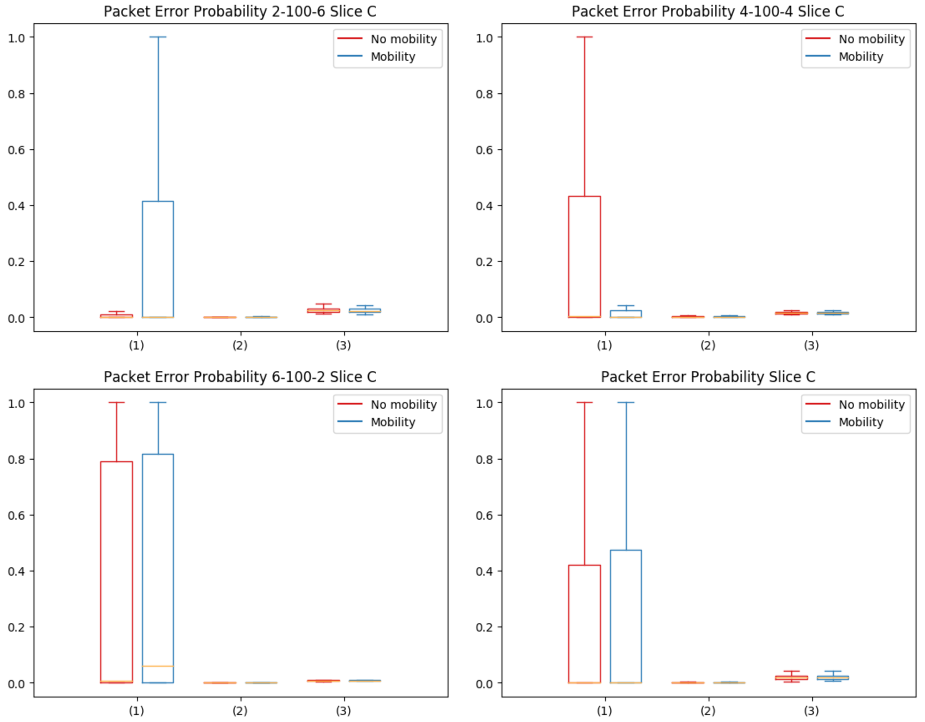


Figure 8.4: Packet error probability for slice C (URLLC): (1) one channel approach, (2) static network slicing, (3) dynamic network slicing.

and the SNR remains constant, the packet error probability worsens. However, in the following section, we will see that the benefit of increasing the MCS consists in a much lower latency. The median values of packet error probability for the slice C are reported in Table 8.1.

	(1) One Channel		**(2) Static**		**(3) Dynamic**	
	No mob.	**Mob.**	**No mob.**	**Mob.**	**No mob.**	**Mob.**
Slice A	1.96×10^{-1}	1.81×10^{-1}	3.91×10^{-3}	3.82×10^{-3}	3.59×10^{-2}	6.34×10^{-2}
Slice B	4.55×10^{-2}	4.65×10^{-2}	0	0	8.16×10^{-2}	8.16×10^{-2}
Slice C	2.56×10^{-3}	2.71×10^{-3}	8.81×10^{-4}	9.40×10^{-4}	1.82×10^{-2}	1.86×10^{-2}

Table 8.1: Median packet error probability in the three slices: (1) one channel approach, (2) static network slicing, (3) dynamic network slicing.

If we evaluate our algorithms based on the achieved error probability, the slicing algorithms proved to be the most performing. In particular, slightly better results can be obtained with the static approach. Let us now analyse the performance of the algorithms considering the E2E latency experienced by the received packets in each slice.

8.2.2 Latency

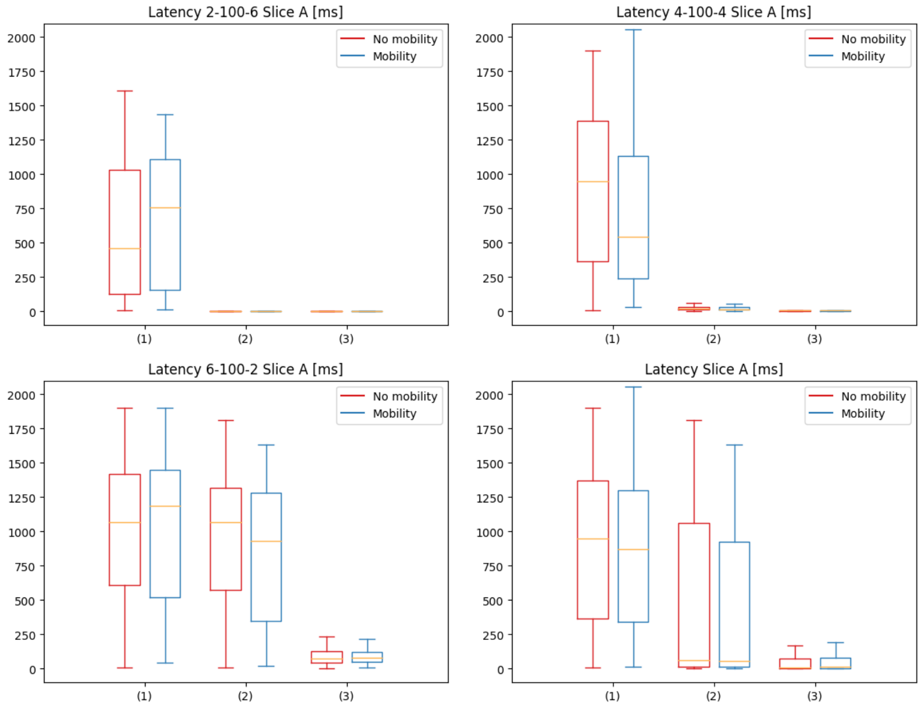


Figure 8.5: Latency for slice A (eMBB): (1) one channel approach, (2) static network slicing, (3) dynamic network slicing.

In Figure 8.5 the latency for the eMBB devices is shown. When all the devices communicate over the same channel, the experienced latency is very poor in every setting (2-100-6, 4-100-4, 6-100-2). It is difficult for the channel to accommodate the required total throughput and packets need to be queued before being transmitted. In fact, they are sent only upon reception of the CTS signal from the AP. As a consequence, the average latency is raised up to almost 2 s in some flows.

Employing the static network slicing technique, the latency is significantly decreased in the first two settings: where the number of eMBB devices is 2 and 4. However, when there are 6 STAs in this slice, the latency is only slightly improved. 6 devices are likely to saturate the channel, which still is forced to keep packets waiting before being transmitted.

As we can see, the problem of having an high latency in the setting 6-100-2 is completely solved by the dynamic approach which offers a flexible bandwidth. By doubling the bandwidth, in fact, the channel capacity increases allowing an higher flow of packets. For greater clarity, the median values of latency for all the techniques considered are reported in Table 8.2.

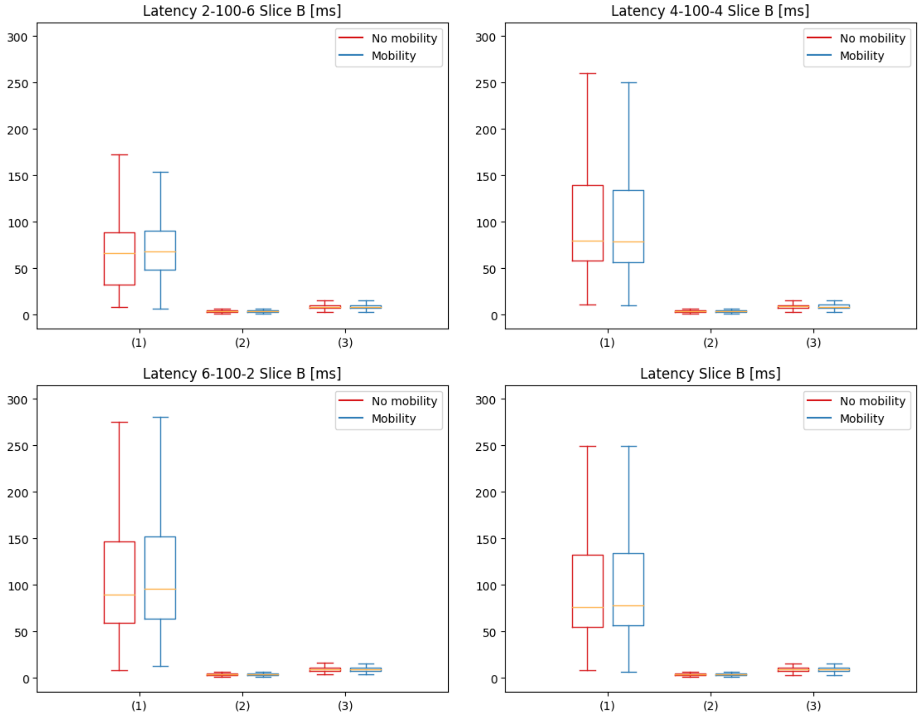


Figure 8.6: Latency for slice B (mMTC): (1) one channel approach, (2) static network slicing, (3) dynamic network slicing.

The latency experienced in slice B (i.e. by the mMTC devices) is reported in Figure 8.6. Since the data volumes are lower, shorter queues are created and latency in this slice never reaches 300 ms. Again, slicing techniques outperform the single channel method as we can see from Table 8.2, which reports the median latency for slice B depending on the applied algorithm.

We notice that latency results for mMTC users using the dynamic algorithm are slightly worse with respect to the ones obtained with static slicing. This is caused by the used MCS index. In fact, it is fixed to 5 in case of the static network slicing algorithm. Conversely, the objective of the dynamic algorithm is to use the lowest possible MCS and decrease the transmission power consequently. In practice, in our simulations it is always strictly lower than 5. The drawback of having a lower MCS is an increasing of the required time to perform a transmission. However, for mMTC devices, low latency is not a crucial requirement as stated in Section 6.1. Thus the latency improvement achieved with the dynamic algorithm with respect to the one channel approach has to be considered significant.

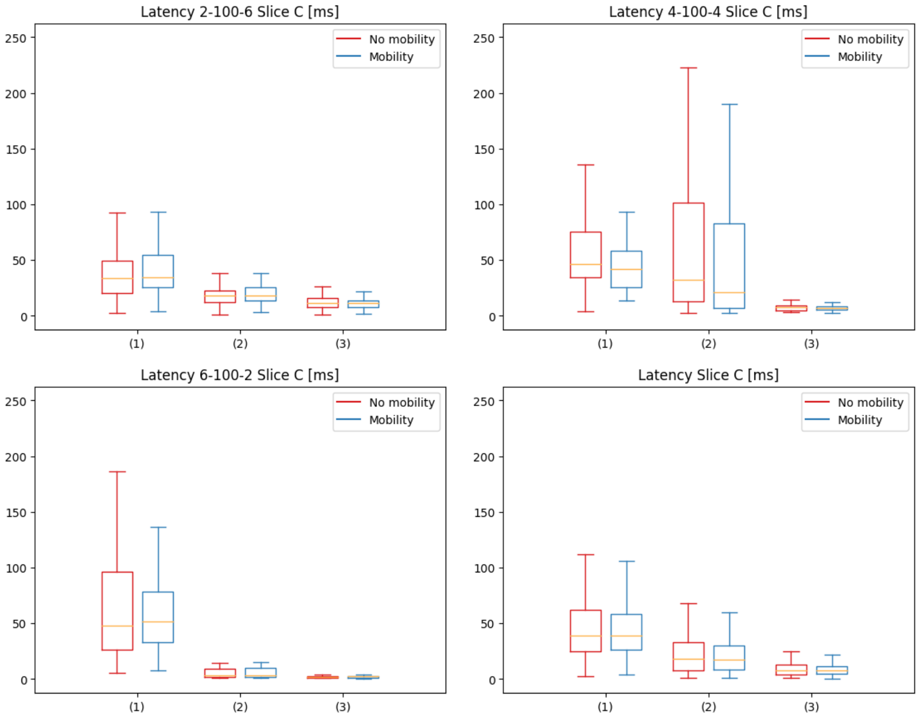


Figure 8.7: Latency for slice C (URLLC): (1) one channel approach, (2) static network slicing, (3) dynamic network slicing.

Finally, the last slice, characterized by URLLC services, is analysed in Figure 8.7. Overall, having one unique channel which serves all the connected STAs turned to be the worst choice. In Table 8.2, the median latencies obtained with each technique are summarized.

The dynamic algorithm clearly outperforms the other two thanks to the higher MCS index used. In fact, it is designed to select the highest possible MCS according to the bandwidth and the received powers. This allows to transmit more information in a shorter time.

In conclusion, the dynamic network slicing algorithm is the one that guarantees the lowest latency in every scenario to eMBB and URLLC devices. For the above mentioned reasons, the mMTC slice experiences better latency with the static network

	(1) One Channel		**(2) Static**		**(3) Dynamic**	
	No mob.	**Mob.**	**No mob.**	**Mob.**	**No mob.**	**Mob.**
Slice A	948	875	61.7	55.8	11.0	14.2
Slice B	76.3	77.8	3.89	3.90	8.97	8.92
Slice C	39.1	39.1	18.0	17.4	8.07	7.88

Table 8.2: Latency (expressed in [ms]) for the three slices: (1) one channel approach, (2) static network slicing, (3) dynamic network slicing.

slicing technique. However, the dynamic approach is still the preferred one since mMTC services do not have strict requirements on the delay; rather, they aim to save energy. In the following section, we will assess our algorithms regarding resource utilization.

8.3 Resource Utilization Analysis

The efficiency of our algorithms in using energy and bandwidth is now studied. The fundamental question we want to address is the following: "Is the network over-provisioned when network slicing is realized?" We can observe that the performance, both in terms of error probability and latency is greatly enhanced when slicing techniques are applied. However, we want to investigate if this improvement is only possible because more resources are allocated, or if better efficiency can be achieved.

8.3.1 Transmission Power

For slice B, which includes mMTC devices, energy saving is the main concern, as stated in Section 6.1. For this reason, with the dynamic network slicing algorithm, we aim to decrease the transmission power for devices in this slice.

As shown in Figure 8.8, on average, the transmission power is lowered from 20 dBm to 5 dBm by using this method. This result is more impressive than it might seem if we consider the conversion from dB to linear units. The two conversion formulas which link a gain of a_{dB} dB with a multiplicative factor a_{lin} are:

$$a_{dB} = 10 \cdot log_{10}(a_{lin}) \tag{8.3}$$

$$a_{lin} = 10^{\frac{a_{dB}}{10}} \tag{8.4}$$

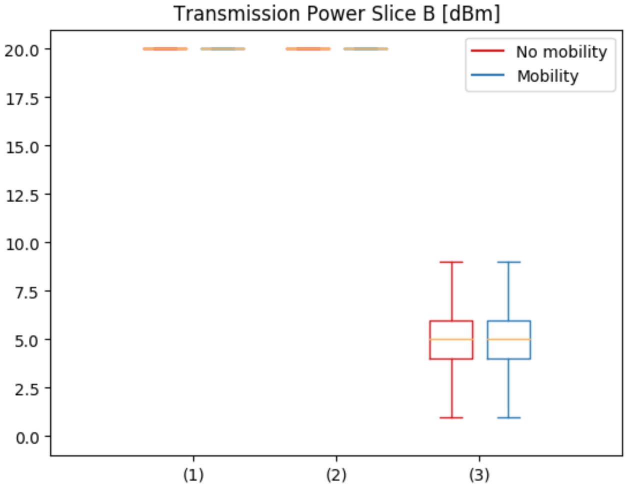


Figure 8.8: Transmission power in slice B (mMTC): (1) one channel approach, (2) static network slicing, (3) dynamic network slicing.

Thus, according to Equation 8.4, a difference of 15 dB means that the transmission power is decreased by about 32 times in linear units[1].

If the IoT devices served by the mMTC slice were battery powered and their consumption depended only on the transmission of data, the dynamic network slicing approach would extend the battery life by 32 times.

8.3.2 Spectrum Efficiency

Now, let us consider the most precious resource when dealing with wireless networks: the bandwidth. To quantify how much bandwidth has been allocated by the algorithms we introduce the notion of spectrum efficiency. This metric refers to the data rate that can be transmitted over a limited spectrum region.

First of all, we consider the total throughput achieved in each run, defined as the sum of the throughput of all the STAs. From the number of received packets on the i-th link, $rxPackets_i$, the total throughput can be obtained as:

$$Th_{sum} = \sum_{i=1}^{nSta} Th_i = \sum_{i=1}^{nSta} \frac{rxPackets_i \times packetSize \times 8}{simulationTime} \tag{8.5}$$

where the parameters $packetSize$ and $simulationTime$ are given by Table 6.2. $nSta = nStaA + nStaB + nStaC$ is the number of devices connected to the AP, and Th_i is the throughput of the i-th device.

[1] The conversion formula from dB to linear units gives us: $10^{15/10} = 31.6$.

After having defined the achieved throughput, we introduce the used bandwidth B_w in each run. This metric needs to be determined with three different expressions depending on which allocation algorithm is employed. In particular, B_w is defined as:

$$B_w = \begin{cases} 160, & \text{if one channel approach;} \\ B_{wA} + B_{wB} + B_{wC}, & \text{if static network slicing;} \\ \frac{1}{N}\sum_{i=1}^{N} B_{wAi} + B_{wBi} + B_{wCi}, & \text{if dynamic network slicing.} \end{cases} \tag{8.6}$$

In the above equation, three cases are considered:

- When only one channel is allocated, its bandwidth is fixed. Thus we have $B_w = 160\,\text{MHz}$ in each run, as stated in Table 7.1;
- If the static network slicing algorithm is used, at the beginning of the run three channels are defined depending on the required throughput. These channels have a fixed bandwidth B_{wA}, B_{wB} and B_{wC} for the three slices, respectively. Thus, summing these values we obtain the total amount of spectrum utilized in that specific run.
- Finally, in case of dynamic network slicing, every interval T new bandwidths are calculated and allocated. Let us suppose that a run is NT long, and that at the i-th interval the bandwidths B_{wAi}, B_{wBi} and B_{wCi} are assigned[2]. Then, the average bandwidth used in a run can be calculated as reported in Equation 8.6.

Finally, the spectrum efficiency in each run is the ratio:

$$\mu = \frac{Th_{sum}}{B_w}. \tag{8.7}$$

In Figure 8.9 the spectrum efficiency of the network is reported for each algorithm employed. As in the previous images, the fourth diagram is obtained by merging the data of the three settings individually analysed: thus, it represents the overall performance.

The data considered are no longer “per flow” as it was the case for the packet error probability and the latency. Instead, we consider the data “per run”, since the spectrum efficiency is calculated for each run. Thus, the cardinality of the plotted datasets is lower, which results in longer whiskers in the box plots. More precisely, in the three diagrams representing the three settings, the cardinality of the datasets

[2] In our case, the simulation time is 15 s and T is 1 s. Thus, $N = 15$.

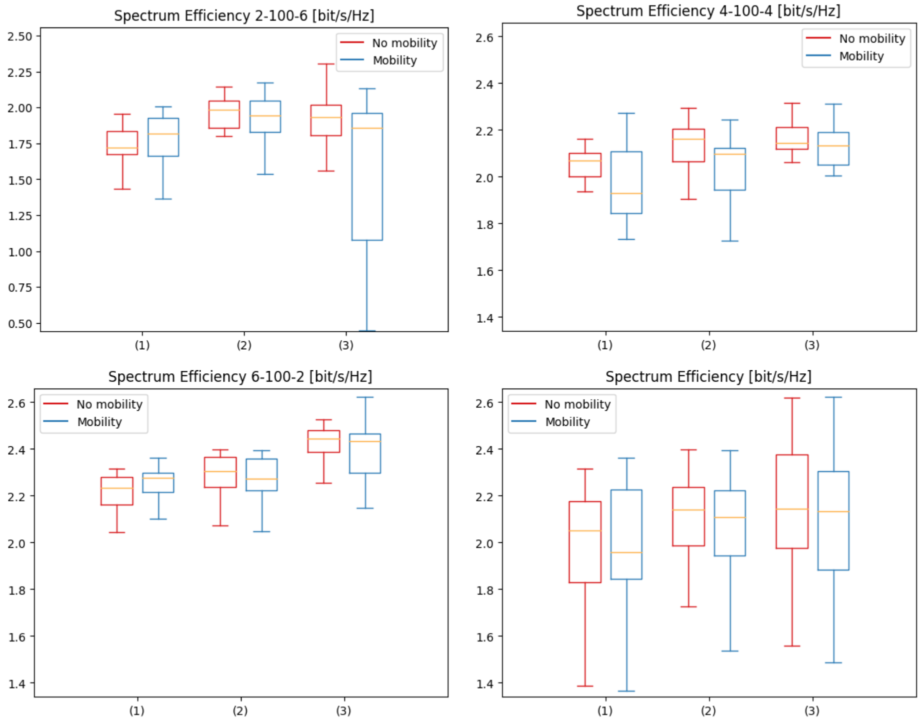


Figure 8.9: Spectrum efficiency: (1) one channel approach, (2) static network slicing, (3) dynamic network slicing.

showed by the box plots is only 20 (as the number of seeds employed). In the forth diagram such cardinality is 60.

An abnormal behavior is visible when the dynamic slicing algorithm is considered in the setting 2-100-6, and mobility is involved. In fact, this algorithm achieves very low spectrum efficiency in some runs, as low as 0.5 bit/s/Hz (note that the scale changes for the setting 2-100-6 in Figure 8.9). The reason of this low resource utilization is the high error probability experienced in the same scenario, for slice A, as reported in Figure 8.2.

However, the two slicing approaches provide, in general, the highest spectrum efficiency. The dynamic algorithm, described in the previous chapter, is also the one that is likely to assign more bandwidth. In particular, in the setting where 6 eMBB devices are present, this algorithm allocates, when necessary, a large amount of bandwidth for slice A, since it has the possibility to double the required bandwidth. Nevertheless, in the third diagram we can see that it has the best spectrum efficiency over the two other approaches, reaching up to 2.6 bit/s/Hz. Thanks to these plots, we can state that allocating more bandwidth is not a waste of resources since the spectrum efficiency increases. In Table 8.3 the median values of spectrum efficiency achieved with the three algorithms are reported.

	(1) One Channel		**(2) Static**		**(3) Dynamic**	
	No mob.	**Mob.**	**No mob.**	**Mob.**	**No mob.**	**Mob.**
μ [bit/s/Hz]	2.05	1.96	2.14	2.10	2.14	2.13

Table 8.3: Median spectrum efficiency: (1) one channel approach, (2) static network slicing, (3) dynamic network slicing.

8.4 Discussion

The one channel approach is clearly outperformed by network slicing under every point of view. In general, the dynamic algorithm is preferred, but still some degrees of flexibility can be tweaked in order to reach optimality and overcome the limitations of the static approach under every metric. In fact, despite the rational approach followed to design the dynamic algorithm, in some cases it does not perform better than the static one.

In particular, the dynamic algorithm for slice A (eMBB) allocates more bandwidth (if needed), and consequently is able to use a lower MCS to accommodate the required throughput. This choice has been made because the error probability should decrease using a lower MCS when the SNR remains constant. However, after the validation results reported in Figure 8.2, we notice that the packet error probability is slightly worsened with respect to the static network slicing algorithm. A possible reason could be that a too low MCS increases the transmission time of signalling messages such as RTS[3]. This could cause more interference and collisions are more likely to occur.

When the dynamic algorithm is applied to slice B (mMTC), errors are caused by the low transmission power, necessary to save energy in low power devices. The power saved with our approach is very significant. Thus, a compromise could be considered to enhance the resulting error probability.

In slice C (URLLC), the dynamic network slicing approach aims to use an high MCS to decrease the latency. As a drawback, the error probability is slightly worsened with respect to the static slicing algorithm. Using an MCS lower than the maximum

[3] In *ns-3* the control messages are modulated with the same modulation used for the data, i.e. specified by the chosen MCS. This could be different from real methods used when an adaptive MCS is employed. Usually, in fact, control messages are typically always sent with the lowest MCS.

allowed by the received powers could help in decrease the error probability, however with a worsening in the experienced latency.

In conclusion, we can state that the proposed slicing technique, consisting in assigning different radio channels to the different slices, performs much better than the today's method both in terms of offered QoS and resource utilization. Thanks to these promising results, this study may pave the way for a future implementation of network slicing in Wi-Fi networks. Additional discussion about future developments is carried out in the next chapter.

Chapter 9
Concluding Remarks

This report studied how network slicing can be implemented in Wi-Fi networks. We decided to allocate three radio channels to serve three different types of services and we proposed two slicing approaches. One that simply allocates three channels according to the required throughput, the other, smarter, that adapts at run-time the assigned resources according to the network needs. To validate our algorithms, the *ns-3* network simulator was selected after a careful comparison of the available network simulators. Thus, extensive simulations were conducted to test the slicing algorithms.

Under every point of view, our slicing approach outperforms the today's access scheme in which an AP serves all the connected users with one unique wireless channel. They enable lower packet error probabilities and lower latencies. Furthermore, slicing is able to reduce the energy needed by low-power devices and increase the spectrum efficiency.

9.1 Reached Isolation

Let us evaluate the level of isolation which has been achieved with our approach. To this end, we use the reference 3D space defined in Section 2.4. We remark that a well defined notion of isolation is still missing to date. For this reason, we developed this graphical approach to represent and compare isolation techniques according to their properties. In Figure 9.1, the isolation reached by our network slicing approach is shown by the blue region in the 3D space.

A fundamental property of network slices is that they are E2E communication link. In particular, they span from the radio access to the datacenter, passing through the transport domain, as introduced in Chapter 2. However, in this work, our interest is on the radio access domain. In fact, the vast majority of available studies on network slicing treat the traffic separation only at the networking level within the context of the Core Network: i.e. transport and datacenter domains.

Our algorithms aim to isolate the slices under the performance point of view. It means that the QoS in the different slices are independent from each other. When hardware is shared between slices, dependability isolation cannot be provided. In fact, if a fault occurs in the AP, the failures affecting the different slices will be correlated. In Section 9.2 we discuss how this isolation dimension can be reached in future studies based on our developed architecture. Security isolation is the third dimension under which isolation can be seen. However, it has not been deeply considered by standardization bodies yet. Thus, it is also omitted in our work.

Finally, the isolation degree is reached at the physical level since different radio resources are allocated to different slices. In Section 9.2, we present possible directions to achieve also hardware isolation, essential to guarantee dependability isolation.

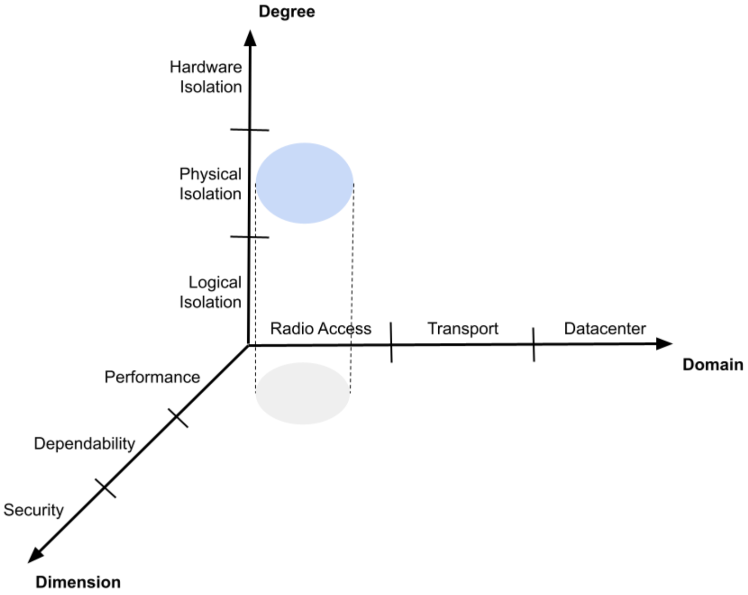


Figure 9.1: 3D graphical representation of the reached isolation through our approach.

9.2 Future Developments

This study can be the basis for a more extensive research on network slicing for Wi-Fi. Our *ns-3* simulated network can be easily reused to run and test any resource allocation algorithm. It uses the most realistic *ns-3* modules which allow to simulate an indoor environment and it is completely customizable through command line instructions.

Our simulation script provides a vast variety of run-time data such as the number of transmitted and received packets per flow, the experienced latency and the position of the STAs if mobility is involved. This information can serve as input to an adaptable network slicing algorithm which can be called every interval of time T. Every type of algorithm is supported, both based on ML or traditional.

The parameters, which can be modified at run-time, used by by the considered algorithms are the radio resources that completely describe a Wi-Fi radio channel: the channel number (i.e. the center frequency), the channel bandwidth, the MCS, the GI and the transmission power. Thus, maximum flexibility is offered. In addition, it could be extended to support RUs management in 802.11ax version, as soon as this functionality will be implemented in *ns-3*.

Re-transmissions are not considered in our validation experiments, since a UDP protocol has been employed. The effect of this choice consists in a lower global latency, but in a worse throughput (i.e. higher packet error probability). Further study could analyse the same scenario where TCP protocol, or link-layer retransmission, is used. The goal could be to see how latency is worsened in favour of an higher throughput and find for which kind of applications this change is optimal.

As a final remark, our simulated system is suitable to be part of the ITU-T ML architectural framework introduced in [IT19]. In particular, it can be used as a sandbox in which the ML algorithm is trained and tested before being deployed in the real underlay scenario.

9.2.1 Multiple AP Scenarios

A possible development of our study could involve the presence of multiple interfering APs, which constitute the entire Service Set. Two developments are identified.

In the first, the concept of frequency reuse can be considered, often applied to cellular networks [YAL+17]. Let us suppose to work in the 5 GHz band, and to apply reuse-3 to our network. In this context, every AP can allocate only one third of the total bandwidth available in the considered spectrum region. From Table 3.1, we can assume that one third of the total bandwidth corresponds to a region 160 MHz wide. It means that every access point can instantiate, to enable network slicing, at most two 80 MHz channel with a third 20 MHz channel overlapping to one of them. Or, three non overlapping channel with bandwidths: 80 MHz, 40 MHz and 20 MHz. A similar dynamic algorithm forced to respect this limitation could be developed, and frequency reuse utilized to offer Wi-Fi coverage in large areas.

Another possible future development could involve multiple APs with different roles. We do not have to deploy all slices in every AP, instead an adaptive algorithm

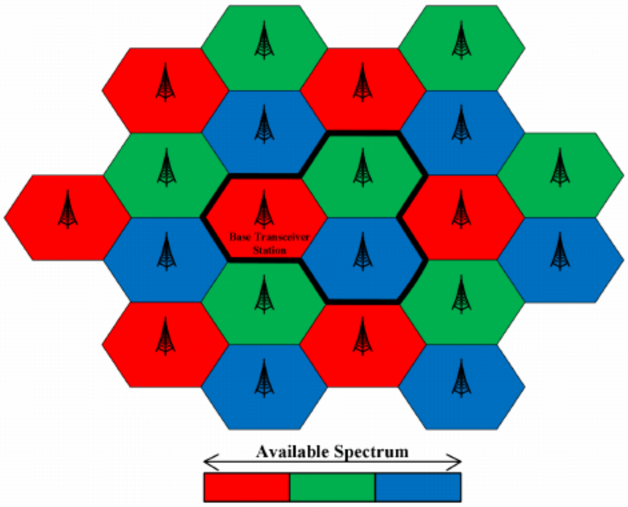


Figure 9.2: Frequency reuse-3 model, adapted from [YAL+17].

can decide where to instantiate a precise channel corresponding to a slice. In this way, different APs could be dedicated to different slices. This approach would help to improve slice isolation also towards the dependability dimension. In fact, when a failure happens in one AP supplying one or more slices, the algorithm could re-manage the available APs such that the failure has a minimum effect on the slices not affected. In the extreme case where APs are dedicated to only one slice, hardware isolation could be provided between different slices.

9.2.2 ML Integration

In order to further enhance our dynamic solution, some additive margins can be added to slightly modify the assigned radio resources such as the MCS and the transmission power. In this way, for instance, the selected MCS for slice C would not be the maximum allowed by the received powers, instead, would be that maximum minus a margin `mcsSubC` (similarly to what is done for slice B). Thus, optimality in the trade-offs introduced in Section 8.4 can be reached according to the SLO in the different slices.

This functionality could be implemented using an ML mechanism. The proposed dynamic network slicing algorithm could decide for the resource allocation, while a ML algorithm could tweak its parameters to reach the desired compromises in performance and resource utilization.

References

[3GP16] 3GPP. Study on architecture for next generation system (Release 14). Technical Report TR 23.799, Dec. 2016.

[3GP18] 3GPP. Study on management and orchestration of network slicing for next generation network (Release 14). Technical Report TR 28.801, Jan. 2018.

[3GP19] 3GPP. System architecture for the 5G System (5GS) (Release 16). Technical Report TS 23.501, Dec. 2019.

[80209] IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC)and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput. *IEEE Std 802.11n-2009 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, and IEEE Std 802.11w-2009)*, pages 1–565, Oct 2009.

[80216] IEEE Standard for Information technology—Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, Dec 2016.

[ABCM19] Joan Josep Aleixendri, August Betzler, and Daniel Camps-Mur. A practical approach to slicing wi-fi rans in future 5g networks. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2019.

[AHA+16] Azana Hafizah Mohd Aman, Aisha-Hassan A Hashim, Azween Abdullah, Huda Adibah Mohd Ramli, and Shayla Islam. Network simulators parametric comprasion for network mobility management. *International Journal of Future Generation Communication and Networking*, pages 17–28, 2016.

[ATS+18] Ibrahim Afolabi, Tarik Taleb, Konstantinos Samdanis, Adlen Ksentini, and Hannu Flinck. Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys & Tutorials*, 20(3):2429–2453, 2018.

[AVTP+17] Ferran Adelantado, Xavier Vilajosana, Pere Tuset-Peiro, Borja Martinez, Joan Melia-Segui, and Thomas Watteyne. Understanding the limits of lorawan. *IEEE Communications magazine*, 55(9):34–40, 2017.

[BM09] Nicola Baldo and Marco Miozzo. Spectrum-aware Channel and PHY layer modeling for ns3. In *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, pages 1–8, 2009.

[BMWSB19] Sergio Barrachina-Muñoz, Francesc Wilhelmi, Ioannis Selinis, and Boris Bellalta. komondor: a wireless network simulator for next-generation high-density WLANs. In *2019 Wireless Days (WD)*, pages 1–8. IEEE, 2019.

[DBTDC+19] Sibren De Bast, Rodolfo Torrea-Duran, Alessandro Chiumento, Sofie Pollin, and Haris Gacanin. Deep reinforcement learning for dynamic network slicing in ieee 802.11 networks. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 264–269. IEEE, 2019.

[DMK17] Yousri Daldoul, Djamal-Eddine Meddour, and Adlen Ksentini. Ieee 802.11 ac: Effect of channel bonding on spectrum utilization in dense environments. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2017.

[ETS08] ETSI. Broadband Radio Access Networks (BRAN); 5,8 GHz fixed broadband data transmitting systems; Harmonized EN covering the essential requirements of article 3.2 of the R&TTE Directive. Technical Report ETSI EN 302 502, Jul. 2008. V1.2.1.

[ETS12a] ETSI. Broadband Radio Access Networks (BRAN); 5 GHz high performance RLAN; Harmonized EN covering the essential requirements of article 3.2 of the R&TTE Directive. Technical Report ETSI EN 301 893, Jun. 2012. V1.7.1.

[ETS12b] ETSI. Electromagnetic compatibility and Radio spectrum Matters (ERM); Wideband transmission systems; Data transmission equipment operating in the 2,4 GHz ISM band and using wide band modulation techniques; Harmonized EN covering the essential requirements of article 3.2 of the R&TTE Directive. Technical Report ETSI EN 300 328, Jun. 2012. V1.8.1.

[ETS14] ETSI. Network Functions Virtualization (NFV); Architectural Framework. Technical Report ETSI GS NFV 002, Dec. 2014. V1.1.1.

[ETS15] ETSI. Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework. Technical Report ETSI GS NFV-EVE 005, Dec. 2015. V1.1.1.

[ETS17] ETSI. Network Functions Virtualisation (NFV) Release 3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework. Technical Report Group Report ETSI GR NFV-EVE 012, Dec. 2017. V3.1.1.

[Fli07] Rob Flickenger. *Wireless Networking in the Developing World: A practical guide to planning and building low-cost telecommunications infrastructure.* Hacker Friendly LLC, Seattle, WA, US, 2007.

[GM18] Alex Galis and Kiran Makhijani. Network slicing landscape: A holistic architectural approach, orchestration and management with applicability in mobile and fixed networks and clouds. In *IEEE Network Softwarization (NetSoft) 2018*, 2018.

[GOLH$^+$19] Andres J Gonzalez, Jose Ordonez-Lucena, Bjarne E Helvik, Gianfranco Nencioni, Min Xie, Diego R Lopez, and Pål Grønsund. The isolation concept in the 5g network slicing. *Unpublished*, 2019.

[GSSF12] Stephen Gutz, Alec Story, Cole Schlesinger, and Nate Foster. Splendid isolation: A slice abstraction for software-defined networks. In *Proceedings of the first workshop on Hot topics in software defined networks*, pages 79–84. ACM, 2012.

[GXG18] Andres J Gonzalez, Min Xie, and Pål Grønsund. Network slicing architecture and dependability. In *International Conference on Mobile, Secure, and Programmable Networking*, pages 207–223. Springer, 2018.

[IET17] IETF. Network Slicing Architecture draft-geng-netslices-architecture-01. Technical report, Network Working Group, Jun. 2017.

[IR15] ITU-R. IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and beyond. ITU-R Recommendation M.2083-0, Sep. 2015.

[IT12] ITU-T. Framework of network virtualization for future networks, next generation network - Future networks. ITU-T Recommendation Y.3011, Jan. 2012.

[IT19] ITU-T. Framework of network virtualization for future networks, next generation network - Future networks. ITU-T Recommendation Y.3172, Jun. 2019.

[KBDW18] Fabian Kurtz, Caner Bektas, Nils Dorsch, and Christian Wietfeld. Network slicing for critical communications in shared 5g infrastructures-an empirical evaluation. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 393–399. IEEE, 2018.

[KBO12] Atta R Khan, Sardar M Bilal, and Mazliza Othman. A performance comparison of open source network simulators for wireless networks. In *2012 IEEE international conference on control system, computing and engineering*, pages 34–38. IEEE, 2012.

[KHN14] Muhammad Amir Khan, Halabi Hasbullah, and Babar Nazir. Recent open source wireless sensor network supporting simulators: A performance comparison. In *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*, pages 324–328. IEEE, 2014.

[KIHH14] Mohammed Humayun Kabir, Syful Islam, Md Javed Hossain, and Sazzad Hossain. Detail comparison of network simulators. *International Journal of Scientific & Engineering Research*, 5(10):203–218, 2014.

[KJBK15] Manikanta Kotaru, Kiran Joshi, Dinesh Bharadia, and Sachin Katti. Spotfi: Decimeter level localization using wi-fi. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 269–282, 2015.

[KKLB18] Evgeny Khorov, Anton Kiryanov, Andrey Lyakhov, and Giuseppe Bianchi. A tutorial on IEEE 802.11 ax high efficiency WLANs. *IEEE Communications Surveys & Tutorials*, 21(1):197–216, 2018.

[KNS+18] Zbigniew Kotulski, Tomasz Wojciech Nowak, Mariusz Sepczuk, Marcin Tunia, Rafal Artych, Krzysztof Bocianiak, Tomasz Osko, and Jean-Philippe Wary. Towards constructive approach to end-to-end slice isolation in 5g networks. *EURASIP Journal on Information Security*, 2018(1):2, 2018.

[LH06] Mathieu Lacage and Thomas R Henderson. Yet another network simulator. In *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, pages 12–es, 2006.

[MH14] Arafet Ben Makhlouf and Mounir Hamdi. Dynamic multi-user access scheme for ieee 802.11 wlan channels. In *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, pages 18–23. IEEE, 2014.

[Ner19] Matteo Nerini. 5G Network Slicing. Final report in TTM4517, Department of Information Security and Communication Technology, NTNU – Norwegian University of Science and Technology, Nov. 2019.

[NGM15] NGMN. NGMN 5G White Paper. Technical Report Frankfurt, Germany, NGMN Alliance, Feb. 2015.

[NGM16] NGMN. Description of Network Slicing Concept, NGMN 5G P1 Requirements and Architecture, Work Stream End-to-End Architecture. Technical Report Frankfurt, Germany, NGMN Alliance, Jan. 2016. Version 1.0.

[ns3a] ns-3 on GitHub. https://github.com/nsnam/ns-3-dev-git. Accessed: May 2020.

[ns3b] ns-3 Wi-Fi Module, Design Documentation. https://www.nsnam.org/docs/models/html/wifi-design.html. Accessed: Jan 2020.

[ns3c] ns-3 Wi-Fi Module, User Documentation. https://www.nsnam.org/docs/models/html/wifi-user.html. Accessed: Jan 2020.

[OLAL+17] Jose Ordonez-Lucena, Pablo Ameigeiras, Diego Lopez, Juan J Ramos-Munoz, Javier Lorca, and Jesus Folgueira. Network slicing for 5g with sdn/nfv: Concepts, architectures, and challenges. *IEEE Communications Magazine*, 55(5):80–87, 2017.

[ONF16a] ONF. Applying SDN Architecture to 5G Slicing. Technical Report TR-526, Apr. 2016.

[ONF16b] ONF. SDN Architecture. Technical Report TR-521, Feb. 2016.

[Pal12] Debajyoti Pal. A comparative analysis of modern day network simulators. In *Advances in Computer Science, Engineering & Applications*, pages 489–498. Springer, 2012.

[RBS$^+$17] Matias Richart, Javier Baliosian, Joan Serrati, Juan-Luis Gorricho, Ramón Agüero, and Nazim Agoulmine. Resource allocation for network slicing in wifi access points. In *2017 13th International conference on network and service management (CNSM)*, pages 1–4. IEEE, 2017.

[RMZ$^+$16] Rapeepat Ratasuk, Nitin Mangalvedhe, Yanji Zhang, Michel Robert, and Jussi-Pekka Koskinen. Overview of narrowband iot in lte rel-13. In *2016 IEEE conference on standards for communications and networking (CSCN)*, pages 1–7. IEEE, 2016.

[SBT$^+$17] Thomas Soenen, Ratul Banerjee, Wouter Tavernier, Didier Colle, and Mario Pickavet. Demystifying network slicing: From theory to practice. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 1115–1120. IEEE, 2017.

[TG04] Godfrey Tan and John V Guttag. Time-based fairness improves performance in multi-rate wlans. In *USENIX annual technical conference, general track*, pages 269–282, 2004.

[Wie14] Roel J Wieringa. *Design science methodology for information systems and software engineering.* Springer, 2014.

[wif] Wi-Fi Alliance. https://www.wi-fi.org/. Accessed: Jan 2020.

[WVLW09] Elias Weingartner, Hendrik Vom Lehn, and Klaus Wehrle. A performance comparison of recent network simulators. In *2009 IEEE International Conference on Communications*, pages 1–5. IEEE, 2009.

[YAL$^+$17] Mohamad Yassin, Mohamed A AboulHassan, Samer Lahoud, Marc Ibrahim, Dany Mezher, Bernard Cousin, and Essam A Sourour. Survey of icic techniques in lte networks under various mobile environment parameters. *Wireless Networks*, 23(2):403–418, 2017.

[ZZW17] Sven Zehl, Anatolij Zubow, and Adam Wolisz. Hotspot slicer: Slicing virtualized home wi-fi networks for air-time guarantee and traffic isolation. In *2017 IEEE 18th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–3. IEEE, 2017.

Appendix A

Appendix

This appendix clarifies which seeds were used to run multiple times the simulations described in Chapter 6. When the two slicing approaches (static and dynamic) were considered, the 20 seeds corresponding to the first integer numbers from 1 to 20 were employed.

However, the following error message from *ns-3* occurred using some precise seeds when the one channel approach was implemented: `msg="no free association id available!", file=../src/wifi/model/ap-wifi-mac.cc, line=1625`. When all the STAs are connected to the same channel, a significant amount of traffic is generated directed to the same device, and this effect could be source of errors in the *ns-3* software. After an analysis of *ns-3* source code, this seems to be an implementation issue with the dissociation procedure. As a result, for the one channel approach, not all the integer numbers from 1 to 20 could be used. The actual seeds used in this scenario are reported in Table A.1.

Setting	**Seeds**
2-100-6	1 2 3 4 5 6 7 8 9 11 12 14 15 16 17 18 19 20 22 23
4-100-4	1 2 3 4 5 6 7 8 9 11 12 14 16 17 18 19 20 22 23 24
6-100-2	1 2 3 4 5 6 7 8 9 11 12 14 15 16 17 18 19 20 22 23

Table A.1: Seeds employed to simulate the one channel approach.