From WiFi Performance Evaluation to Controlled Mobility in Drone Networks

Rémy Grünblatt

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON

Opérée au sein de :

l'Université Claude Bernard Lyon 1

Ecole Doctorale 512
InfoMaths

Spécialité de doctorat : Informatique

Soutenue publiquement le 08/01/2021, par :

Rémy Grünblatt

From WiFi Performance Evaluation to Controlled Mobility in Drone Networks

Contents

List of Figures

List of Tables

Abstract

Mobility in telecommunication networks is often seen as a hassle that needs to be dealt with: a mobile wireless device has to adapt is transmission parameters in order to remain connected to its counterpart(s), as the channel evolves with the device's movements. Drones, which are unmanned aerial vehicles in the context of this thesis, are no exception. Because of their freedom of movement, their three-dimensional mobility in numerous and varied environments, their limited payload and their energy constraints, and because of the wide range of their real-world applications, drones represent new exciting study objects whose mobility is a challenge. Yet, mobility can also be a chance for drone networks, especially when we can control it. In this thesis, we explore how controlled mobility can be used to increase the performance of a drone network, with a focus on IEEE 802.11 networks and small multi-rotor drones. We first describe how mobility is dealt with in 802.11 networks, that is to say using rate adaptation mechanisms, and reverse engineer the rate adaptation algorithm used in the Wi-Fi chipset of the Intel Aero Drone. The study of this rate adaptation algorithm, both experimental and through simulation, through its implementation in the network simulator NS-3, allows its comparison against other well-known algorithms. This highlights how big the impact of such algorithms are for drone networks, with regard to their mobility, and how different the resulting behaviors of each node can be. Therefore, a controlled mobility solution aiming to improve network performances cannot assume much about the behavior of the rate adaptation algorithms. In addition to that, drone applications are diverse, and imposing mobility constraints without crippling a complete pan of these applications is difficult. We therefore propose a controlled mobility solution which leverages the antenna radiation pattern of the drones. This algorithm is evaluated thanks to a customized simulation framework for antenna and drone simulation, based on NS-3. This solution, which works with any rate adaptation algorithm, is distributed, and do not require a global coordination that would be costly. It also does not require a full and complete control of the drone mobility as existing controlled mobility solutions require, which makes this solution compatible with various applications.

Introduction | 1

Telecommunication networks make it possible, in a sense, to free oneself from distance. The electric charges in a conductor and electromagnetic waves both travel at speeds that few other physical objects can reach, all the more compared to things one actually handles, such as postal mail. The beginning of the 20th century saw a revolution in the field of telecommunications, namely wireless radio communications. With the miniaturization of the radio devices achieved over the past century, it has become possible to free oneself from distance while remaining mobile. Unsurprisingly, switching from a bounded transmission media such as copper wires to an unbounded one creates a few challenges, and mobility comes at a price. But it also enables a wide new range of applications and uses, which is at the heart of the development of the mobile Internet, and the *personal* Internet.

This year, 2020, has been a special year for telecommunication networks, both globally and locally. Worldwide, telecommunications networks in general, and Internet in particular, made it possible to communicate with family, friends and colleagues, all during a global pandemic and without many difficulties. Often described as de-sociabilizing, smartphones and the Internet became, for a few months, the center of our social life, at least with people with whom we do not share our home. Without reliable and resilient networks, it would have been impossible for hundreds of millions of people to work remotely. And, we are well-placed to know, it would have been impossible for tens of millions of students to be able to continue taking courses remotely. In this regard, working on networks seemed more meaningful and important to me during 2020 than during the previous year of my thesis.

At the same time, unprecedented movements of opposition to new telecommunications technologies, whether founded or not, have emerged. The next broadband cellular standard of the members of the 3rd Generation Partnership Project (3GPP), 5G, is being described by its opponents as an energy- and resource-wasting headlong rush, in a global context of growing environmental concerns. The new frequency bands that can be used by 5G also fueled many questions about its health impact, and many conspiracy theories about 5G were relayed online, on the web, in an ironic twist of fate as mobile traffic accounts for around 50% of the global web traffic. Starlink, a project whose goal is to provide internet access through a low earth orbit constellation of tens of thousands of satellites, drew criticisim from astronomer worldwide for the created light pollution and the increased number of space debris they would create once their lifespan has ended. As of 2020, a total of 12, 000 satellites have been authorized by the Federal Communications Commission (FCC) to be deployed over the spectrum, and filings have been submitted to the International Telecommunication Union (ITU) for 30, 000 additional satellites. More than 800 satellites have already been launched. In 2020, less than 60% of the world population has access to the Internet, but this number is steadily increasing every year.

Evidence suggest the human-caused global warning is increasing the number of extreme climate events, including storms, hurricanes, floods or large-scale wildfires. Large-scale natural disasters often destroy transport, communications and telecommunication infrastructures, which are nonetheless necessary for the organization of the disaster response. In the wake of storm Alex, which killed at least 8 people at the beginning of October 2020 in France, optical fiber networks were destroyed, including the one serving cellular antennas Figure 1.1. Disaster response included deploying Wi-Fi hotspots connected to the Internet using satellite links, and satellite phones [27]. The deployment of emergency communications' infrastructure is often a prerequisite for the organization of search and rescue operations and humanitarian responses. The importance of telecommunications networks is never more apparent than where they become off-line. Designing resilient networks which can be rapidly deployed is one of the envisioned application of drone networks.

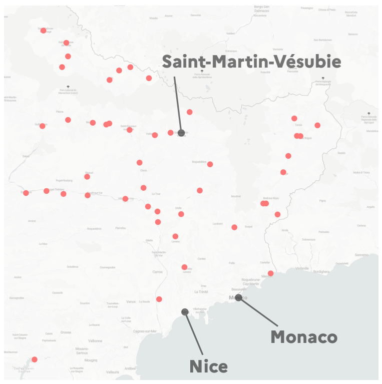


Figure 1.1: Out of order cellular sites of the Orange telecommunications operator, after the storm "Alex", on the 3 October 2020, as reported on the ARCEP website.

[27]: Fédération Française des Télécoms (2020), *Tempête Alex | Comment les opérateurs se sont mobilisés pour réparer les réseaux fixes et mobiles ?*

Drones

Drones, also known as Unmanned Aerial Vehicles (UAVs), are aircraft without humans on board. Previously mainly reserved for military use, the past decade have seen the development of many types of smaller drones [61], which have been used in civilian applications such as aerial photography, agriculture, surveillance, disaster management, network deployment, search-and-rescue missions, or transport. Embedded with several types of sensors and processing power, drones can be remote-controlled by a human operator, in which case (wireless) connectivity between the pilot and the drone is more than desirable, or they can be autonomous. The degrees of drone autonomy vary, ranging from following precomputed trajectories defined by waypoints, to having complete freedom of movement in order to carry out their missions. Most of the time, communication between the drone and the ground is necessary, be it for legal reasons or to exchange control, feedback or mission data.

[61]: Moschetta et al. (2017), 'Introduction to UAV Systems'

Fleets of drones which can coordinate themselves can achieve tasks that would be otherwise impossible to achieve by a single drone, or realize tasks more quickly, more efficiently or with more flexibility [18]. For this cooperation to be possible, connectivity between the drones is necessary, whether it is provided by a cellular network, in a mobile ad-hoc network scheme, or even by satellite. The resulting drone networks are subject to specific challenges, as their components are power and weight constrained, are highly mobile in three dimensions, and evolve in an environment that is in turns highly dynamic and mission dependent.

[18]: Chaumette (2017), 'Collaboration Between Autonomous Drones and Swarming'

Mobility in Networks

Mobility in networks is often seen as a hassle that needs to be dealt with. Indeed, a mobile wireless device, and a drone in particular, face many challenges. It has to adapt is transmission parameters in order to remain connected to its counterpart, as the channel evolves with the device's position. A non-autonomous drone that wanders outside the range of its remote controller is guaranteed to crash if no preventive measures are taken. Yet, mobility is often a key feature of network applications. Without mobility, it would be impossible to the participants of a low density network like the one envisioned by the Serval network, intended for resilient communications during crisis situations, to communicate [34]. Mobility of the first responders in the wake of a natural catastrophe is not an adjustement variable, it is a strong prerequisite to their missions.

[34]: Gardner-Stephen et al. (2013), *The serval mesh: A platform for resilient communications in disaster & crisis*

While mobility is always subject to external constraints one cannot avoid, such as the laws of physics, and therefore can never be completely controlled, we can still introduce the concept of controlled mobility. Controlled mobility, for a given entity, could be defined as any mobility on which there exists some degrees of freedom, for example in the acceleration, speed or position of the entity, mobility that can be *acted on* by this very same entity. Such *controlled* mobility can also be a chance for communication networks. A well-known trope of controlled mobility, as portrayed in many movies, is trying to move a cellphone to get a better signal. Mobility can be exploited in conjunction with store-and-forward mechanisms to deliver messages to otherwise isolated network segments. While an automated controlled mobility for devices like a smartphone or a laptop is more of a wishful thinking, when it comes to autonomous vehicles, robots, or drones, it is a reality.

Thesis Statement

Creating a drone networks is a matter of connecting drones together, which can be done through the use of different networking technologies. In this thesis, we chose to focus on Wi-Fi, defined in the IEEE 802.11 set of standards, and in turns on Wi-Fi-based drone networks. This choice can be mainly explained by the (apparent) simplicity of deploying Wi-Fi networks, which operate in licence-free bands and are supported by many end-user devices, including ours. Wi-Fi devices are also cheap, available, off-the-shelf, and can accommodate a wide range of application requirements, from the IOT low-frequency and lightweight requirements, to real-time video or Voice over IP, through more classic web browsing. For critical applications, being able to test such networks in real conditions before they are actually needed is also an advantage, which would be harder with for example cellular networks where the regulator's agreement is required to transmit.

The use of Wi-Fi to create drone networks is not a new idea, and has been studied, mainly in the past ten years, in many scientific works [7, 8, 42, 44, 69, 85, 86, 94, 95, 98]. In the experimental studies from this literature, Wi-Fi is described as under-performing, because of the peculiarity of the airborne channel on the one side, and the inability of Wi-Fi to cope with the specific mobility of the drones, both because of the use of unsuitable antennas and because of inefficiencies in Wi-Fi's *rate adaptation*. In the context of Wi-Fi networks, rate adaptation is the process of choosing suitable transmission parameters (Wi-Fi has many of them) to cope with changes in the communication channel and maintain a quality of service. Controlled mobility in the context of robot and drone networks is more confidential but a few solutions have also been proposed [19, 38, 53, 59, 73]. Still, the intersection on Wi-Fi networks, drone networks, and controlled mobility is limited.

The goal of this thesis is to study the underlying mechanisms of Wi-Fi networks in order to improve controlled mobility in the context of networks of autonomous drones. In particular, we focus on the impact of the Rate Adaptation Algorithms (RAAs) on such networks. Those algorithms are in charge of finding suitable transmission parameters for the nodes participating in Wi-Fi-based drone networks, and are thus closely linked to the network performances, whether good or bad. While, in our opinion, RAAs are at the heart of the performance evaluation of modern Wi-Fi networks, this subject has been mostly swept under the rug in the context of robotics networks, where using static and fixed transmission parameters is often the norm.

Thesis Organization

The rest of this manuscript is organized into six chapters (not counting the conclusion).

In Chapter 2, we present the general context of this thesis, with a focus on the IEEE 802.11 set of standards, drones and drone networks. Indeed, from its introduction more than two decades ago to the new amendment, 802.11ax, that will be adopted in the coming weeks, Wi-Fi evolved well beyond its original form and must be correctly introduced.

We then present in Chapter 3 a state of the art of experimental evaluation and controlled mobility of Wi-Fi based drone networks. We also present the tools, both simulators and testbeds, one can use to evaluate such networks.

In Chapter 4, we then describe and analyze the rate adaptation algorithm used by Intel Wi-Fi cards, which is the way Wi-Fi networks cope with mobility. Such analysis is necessary to understand the performances of mobile Wi-Fi networks, which we focus on in Chapter 5. In this chapter, we therefore compare the performances in simulation of multiple rate adaptation algorithms, including the Intel one, in the context of drone networks. The Intel RAA, while being used by tens of millions of device worldwide, had never been described before, nor implemented in a network simulator to study its performances differently than in experiments. The developed simulation implementation, for the ns–3 network simulator, is still at a prototype stage, but has been released as an open-source contribution. We hope to integrate it to the *upstream* ns–3 source code in the coming months.

Finally, in Chapter 6, we propose a controlled mobility solution relying on the antenna radiation patterns of drones, which allow to enhance the performances of a fleet of drones. Such a solution is interesting because it does not require a full control over the drone positions, which allows this solution to interface more easily with varied drone network applications. To study its performances, and to overcome the limits of the used network simulator, ns–3, a custom framework was developed to allow the simulation of the considered simulation scenarios.

Publications

- Rémy Grünblatt, Isabelle Guérin-Lassous, and Olivier Simonin. ‘Leveraging Antenna Orientation to Optimize Network Performance of Fleets of UAVs’. In: *MSWiM’20 - The 23rd International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. Alicante, Spain, Nov. 2020
- Rémy Grünblatt, Isabelle Guérin-Lassous, and Olivier Simonin. ‘Simulation and Performance Evaluation of the Intel Rate Adaptation Algorithm’. In: *MSWiM 2019 - 22nd ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. Miami Beach, United States: ACM, Nov. 2019
- Rémy Grünblatt, Isabelle Guérin-Lassous, and Olivier Simonin. ‘Study of the Intel WiFi Rate Adaptation Algorithm’. In: *CoRes 2019 - Rencontres Francophones sur la Conception de Protocoles, l’Évaluation de Performance et l’Expérimentation des Réseaux de Communication*. Saint-Laurent-de-la-Cabrerisse, France, June 2019

General Context of this Thesis 2

In this chapter, we present the general context of this thesis. We first focus on Wi-Fi networks and the evolution they have undergone since their introduction, 21 years ago. After a brief history of Wi-Fi networks, we present the basic principles behind Wi-Fi Networks, that is to say the Medium Access Control Layer and the Physical Layer of the protocols. Then, we focus on drones, their characteristics, their applications and their architectures. We first present drones and their history, then we focus on the applications of drones. Finally, we focus on drone networks, trying to illustrate their architecture, requirements, and we finish by looking at the adequacy of Wi-Fi for drone networks.

“ People move, networks don’t ”

– Mathew S. Gast

2.1 A Brief History of 802.11 Networks

The IEEE 802.11 set of standards, better known to the general public under the denomination “ Wi-Fi”, is a bit over 21 years old at the time of writing. While the general principle of the protocol, namely Carrier Sense, Multiple Access with Collision Avoidance (CSMA/CA), didn’t change much in that period, numerous additions and refinements transformed the original protocol, *802.11-1997*, which falls short of providing throughput of 2 Mbit/s, into its current state, which achieves throughput a thousand times higher. Both Wi-Fi and Bluetooth, another wireless protocol of a comparable age, were launched to operate at frequencies comprised between 2.4 GHz and 2.5 GHz, reserved for Industrial, Scientific and Medical applications, parts of the so-called ISM radio bands. In 1985, the Federal Communications Commission (FCC), agency of the United States government in charge of regulating telecommunications, allowed spread-spectrum communication systems whose power output didn’t exceed 1 watt on the ISM bands. No license are generally needed to operate on these frequencies, which had been used by microwave oven for decades (and still are), and thus probably considered at the time too polluted for any *serious* business. The absence of a licensing requirement also explains the success of WiFi networks.

While both using spread-spectrum techniques, and using packets (called *frames*) to transmit data, and even if they share the same band and some of their usages, the Bluetooth and Wi-Fi protocols grew apart, with Bluetooth being mainly used as a form of "wireless wire" [35], as a Wireless Personal Area Network (WPAN) connecting *personal* devices together, such as a smartphone and a headset, while Wi-Fi became mainly used as a Wireless Local Area Network (WLAN), connecting *any* local devices with each other or to a gateway to the internet, as an Ethernet network would do it. Wi-Fi became the de facto standard for WLAN, so much so that WLAN and Wi-Fi are synonyms in some languages such as German. However, Wi-Fi Networks are not limited to power WLAN, as (hopefully) illustrated by this thesis. When looking at the history of

[35]: Gast (2005), *802.11 Wireless Networks: The Definitive Guide, Second Edition*

Wi-Fi, two technologies often come up: ALOHAnet and Ethernet (also known as IEEE 802.3). While Wi-Fi was designed to be fully compatible with Ethernet networks, which are wired networks introduced in 1983 mainly used to power Local Area Network (LAN), it shares a lot with ALOHAnet, invented in 1971, precursor to Ethernet networks, although wireless and based on a slightly different medium access method.

The development of Wi-Fi networks can be linked to the development and democratization of the personal computer, and, in a sense, of the Internet. If someone asks you for the Wi-Fi password, they probably are not looking for a WLAN but for an Internet access. It is estimated by Cisco that globally, by 2022, the Wi-Fi traffic will represent more than half of the total IP traffic [32]. Wi-Fi networks are now in laptops, phones, oven, toothbrush, whether we like it or not, as it became a way to market products as being *smart*.

[32]: Forecast (2019), 'Cisco visual networking index: global mobile data traffic forecast update, 2017–2022'

Standard(s) and Naming Conventions

Countless amendments to the standard have been produced by the IEEE 802.11 Working Group throughout the years, either introducing new functionalities, security features [56], but only five versions of the standard have been published so far, as shown in Table 2.1.

[56]: McCann (2020), *Official IEEE 802.11 Working Group Project Timelines*

Table 2.1: Main 802.11 standards and amendments

Name	Year	Document
802.11ax	2020	Amendement †
802.11	2016	Standard
802.11ac	2013	Amendment *
802.11	2012	Standard *
802.11n	2009	Amendment *
802.11	2007	Standard *
802.11g	2003	Amendment *
802.11b	1999	Amendment *
802.11a	1999	Amendment *
802.11	1999	Standard *
802.11	1997	Standard *

†: Draft
* : Superseded

Only one version of the standard exists at a given time, as older versions are superseded by the latest published version. Marketed terms generally refer to specific amendments, such as 802.11a, 802.11b, or 802.11n, as they generally introduce better performances in terms of latency or throughput which are well understood by the general public. Multiple other amendments can be considered as important waypoint in the context of this thesis, such as IEEE 802.11p, published in 2010 and also known as WAVE, providing wireless access in vehicular environments, or 802.11s, published in 2011, which introduces mesh networking in the standard, but we can focus on the standards as these amendments end up being integrated into them. In October 2018, the Wi-Fi Alliance introduced a new numerical naming system for Wi-Fi amendments: devices that support the 802.11n, 802.11ac and 802.11ax amendments are to be respectively called Wi-Fi 4, Wi-Fi 5 and Wi-Fi 6 devices [28]. Previous amendments such as 802.11a, 802.11b or 802.11g didn't receive a new name. This further blurs the line between standards and amendments, but makes it easier for the general public to understand the evolution of Wi-Fi.

[28]: Wi-Fi Alliance (2018), *Wi-Fi Alliance® introduces Wi-Fi 6*

The 802.11ax amendment, the next "big" amendment for Wi-Fi networks, is expected to be approved and published by the end of the 2020 year. It introduces profound changes in the way Wi-Fi networks are operated, but is out of scope of this thesis for practical reasons such as the lack of readily available compatible hardware. In this thesis, we therefore focus on the 802.11-2016 standard, latest published standard of the 802.11 family, which will be referred to as *802.11ac* for simplicity, even if erroneous. The term *Wi-Fi* will also be used as a way to designate 802.11 networks in general, when the exact version of the standard is not important for the discussion.

2.2 Basic principles of Wi-Fi Networks

Wi-Fi networks are composed of two layers, the Physical Layer (PHY), dictating how data should be encoded over the spectrum, and the Medium Access Control Layer (MAC), defining how a participant in the network, hereafter called Station (STA), should access the shared electromagnetic spectrum. Both the Wi-Fi PHY and MAC, were widely transformed during their existence, with the PHY supporting more and more complex transmission techniques, and the MAC introducing new access mechanisms, in order to obtain higher throughout and increase robustness.

The Wi-Fi Medium Access Control Layer

As Ethernet, Wi-Fi is based on the Carrier Sense, Multiple Access (CSMA) mechanism, which means the carrier, here the electromagnetic spectrum, is sensed before transmissions to avoid creating collisions on the shared medium. Unlike Ethernet, which uses Carrier Sense, Multiple Access with Collision Detection (CSMA/CD) and detects collisions as they happen on the medium, which is made possible by the imposed bounds on the physical size of the networks and the fact one can sense the medium while sending, Wi-Fi uses Carrier Sense, Multiple Access with Collision Avoidance (CSMA/CA). The CSMA/CA mechanism relies on sensing the medium prior to transmitting, and waiting for it to be *idle* for at least a given amount of time, before the start of the transmission. If the medium is found to be busy, because another node is transmitting for example, a random gap period of backoff is used before attempting to transmit again, which avoids synchronized transmission attempts of multiple STAs when the medium becomes free again. By defining multiple values for the gap durations, known as Inter Frame Space (IFS), such as the Short Inter Frame Space (SIFS) or the Distributed Inter Frame Space (DIFS), "priority classes" for frames are created. As wireless transmissions are inherently lossy, some Quality Of Service (QOS) mechanism is needed. In Wi-Fi networks, as in ALOHAnet, it takes the form of short Acknowledgement (ACK) messages, which use the highest priority class, based on the smallest gap, the SIFS. Thus, ACK messages take precedence over any other messages, which makes it easy to detect whether they were received or not. The ACK messages are used by a receiver to acknowledge that data has been correctly received. In the case of a collision, or when reception fails, or if the acknowledgment sent from the receiver to the original transmitting node is not correctly received, the node retransmits the non-acknowledged frames under certain conditions [16], mainly to avoid infinite chains of retransmissions when the receiver is not able to receive frames anymore.

We focus on the DCF and not on the upper coordination functions such as the Point Coordination Function, optional and deprecated in the latest standard, or the Mesh Coordination Function.

[16]: Brunisholz et al. (2018), 'Anatomie des retransmissions dans les implémentations de 802.11'

In Wi-Fi Networks, STAs are organized in Basic Service Sets (BSSs), which can be thought of as groups in which communication over the PHY is possible. BSSs can be of different types, such as the Independant Basic Service Sets (IBSSs) (also known as *Ad-Hoc* mode), the Mesh Basic Service Sets (MBSSs) (also known as *Mesh* mode), or the more classical infrastructure BSS (also known as *infrastructure* mode). In infrastructure mode, special STAs called the Access Points (APs) are used to connect STAs together, even if they are not in range with each other. APs act as

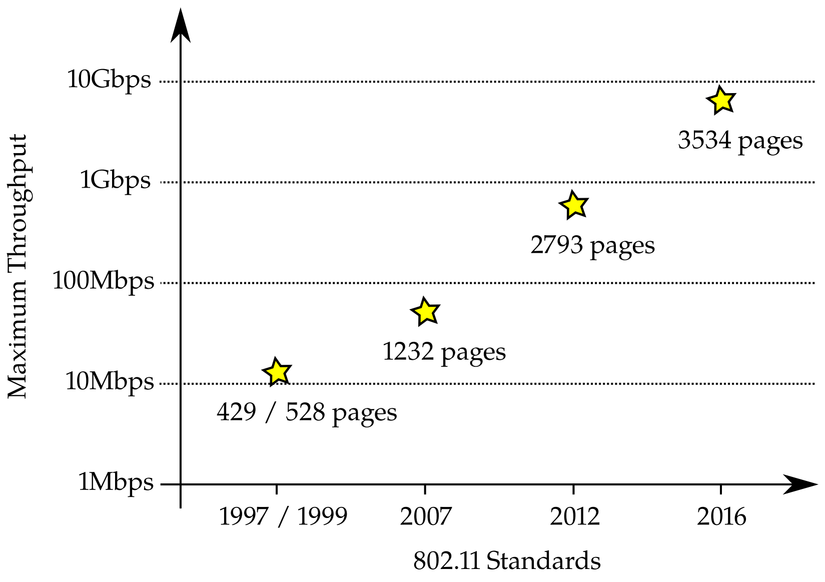


Figure 2.1: Evolution of the maximum physical transmission rates and number of pages of the different 802.11 standards.

centralizing entities, while in MBSSs or IBSSs, no such central entities exist. MBSSs were introduced in the 802.11-2012 standard, as a part of the 802.11s amendment, while IBSSs are present in the standard since 802.11-1997. Thus, the 802.11 mesh mode is more advanced than the ad-hoc mode, by integrating with Distribution System (DS) which allows it to operate in conjunction with the infrastructure mode, or by being equipped with a default mandatory routing protocol, Hybrid Wireless Mesh Protocol (HWMP).

[46]: (2016), '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'

The Wi-Fi Physical Layer

The 802.11-2016 [46] standard describes the Wi-Fi PHY as "fundamentally different" from the PHY used in wired media, such as Ethernet. In short, Wi-Fi networks use an unreliable medium, shared with other signals (Wi-Fi or not), medium which exhibits time-varying and asymmetric propagation properties. STAs can move, and the STAs connectivity graph, as well as the graph of interfering STAs, which are different, are neither complete nor static, are directed, and are not known in advance or easily measurable. While some problems that arise from these features can be handled in practice at the MAC layer, such as the hidden node problem, dealt with by the Request to Send, Clear to Send (RTS/CTS) mechanism, some are managed by the PHY layer of 802.11.

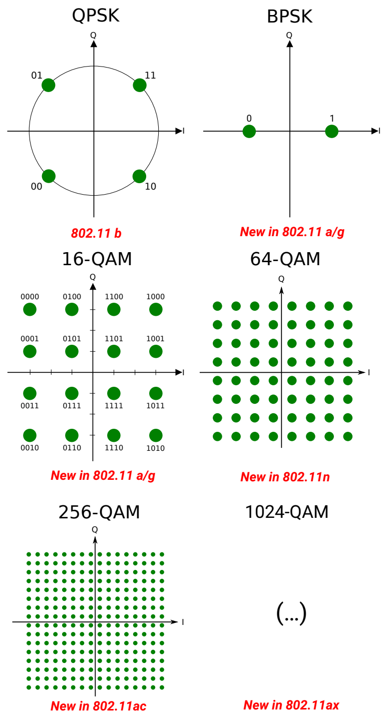


Figure 2.2: Constellations of the modulations used in modern version of the 802.11 standard.

From the start, Wi-Fi came with different methods to send and encode data over-the-air, which deals with the shortcomings of the PHY. The 802.11-1997 standard used Frequency-Hopping Spread Spectrum (FHSS), with Gaussian Frequency-Shift Keying (GFSK) with two and four levels, and Direct-Sequence Spread Spectrum (DSSS), with Differential Binary Phase-Shift Keying (DBPSK) and Differential Quadrature Phase-Shift Keying (DQPSK), to provide transmission rates of 1.0 and 2.0 Mbit/s. Barker code were used to transform single bits into fixed 11-bits long Pseudonoise (PN) code words, introducing redundancy. All four variants must fail to prevent communication between STAs, which introduces robustness against various radio environment. The 802.11b amendment introduced High-Rate Direct-Sequence Spread Spectrum (HR/DSSS) and transmission rates up to 11.0 Mbit/s using Complementary Code Keying (CCK), reducing the size of the PN code words to 8 bits, encoding groups of 4 and 8 bits with each code word. The new modulations didn't replace the older ones, as they required better channel conditions or better hardware, but they increased transmission rates, and introduced more transmission possibilities, and in turns, increased robustness.

The PHY was deeply changed by the 802.11a amendment which introduced a new modulation scheme, Orthogonal Frequency-Division Multiplexing (OFDM), a new operating frequency band located in the Unlicensed National Information Infrastructure (U-NII) bands (which overlaps with the 5 GHz ISM band), and new modulations based on Quadrature Amplitude Modulation (QAM). The OFDM modulation makes use of multiple orthogonal subcarriers to transmit the data stream concurrently, at a lower data rate than if it was sequentially transmitted, which reduces the relative impact of the channel multiple paths [64] especially present in indoor environments such as offices. The width of the OFDM signals used in 802.11a was chosen to be 20MHz, which allows for 25 non overlapping *channels* to coexist in the U-NII bands, while supporting transmission rates up to 54 Mbit/s when combined with the new 16-QAM and 64-QAM modulations.

[64]: Nee et al. (2000), *OFDM for Wireless Multimedia Communications*

The OFDM introduced new *knobs*, such as the Guard Interval (GI) size, a pause between subsequent transmissions preventing them to interfere with each other, either 800 ns or 400 ns long, or the coding rate, parameter specifying the number of bits of data transmitted per bit of code word of the forward error-correcting code used in OFDM, which trades throughput for robustness. Antenna diversity started being used at the sender and receiver to combat fast fading, by combining the signals with more than one antenna, for example using Space-Time Block Codes (STBCs) or Maximum Ratio Combining (MRC). OFDM was later extended to the 2.4 GHZ band in 802.11g, and Multiple Input, Multiple Output (MIMO) was introduced, which allowed multiple independent streams of the data to be transmitted at the same time. Instead of using one channel at a time, 802.11n introduces channel bonding, which allows using two adjacent channels at the same time, that is to say transmitting and receiving on 40MHz (later extended to 80 MHz and 160 MHz by 802.11ac). Overall, the maximum transmission rates increased with the complexity of the standards, as shown on Figure 2.1.

2.3 Drones

History

Depending on the definition of what a Drone is, e.g. an unmanned aerial vehicle, their invention can be traced back to Asia, thousands of years ago, under the form of kites which *are* unmanned aerial vehicles, although tethered. Noteworthy examples of drones include the Austrian incendiary balloons used to bomb Venice (or try to) in 1849, depicted on Figure 2.3, or Alphonse Pénaud's balloons, illustrated on Figure 2.4, which include a small airplane model with automatic rudder powered by a twisted rubber, popularized as a toy. The British "Queen Bee" radio-controlled plane, used as targets to train anti-aircraft gunners in the

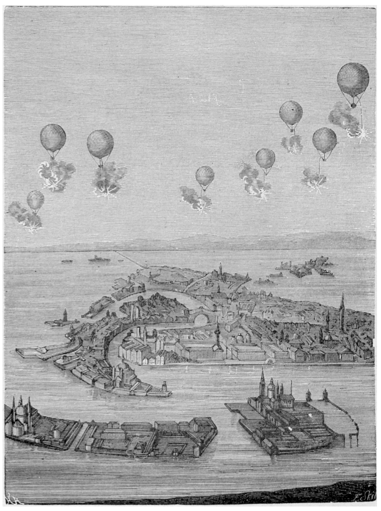


Figure 2.3: Artistic depiction of the bombing of Venice by Austrian Incendiary Balloons [13].

1920s, is thought to be the origin of the *Drone* term (the term “drone” refers to the male bee). Until recently, the main users of drones were either the military, either serious hobbyists engaged in aircraft modelling. In the last few decades, the use of drones has been trivialized. On the civilian side, multicopters, which are rotorcraft with more than two rotors have entered the entertainment landscape for the general public in 2010. Quadcopters, with their simple design and their capacity to fly in any direction, are particularly popular among amateur pilots. Around half a million units of the Parrot AR.Drone were sold in the three years following its launch in 2010. The AR.Drone, now discontinued, was probably the first quadcopter marketed to the general public, and was equipped with Wi-Fi connectivity used to control the drone from a smartphone and receive aerial pictures and videos from the two on-board cameras. On the military side, weaponized drones have been routinely used by the United States in the 2001 Afghanistan conflict, the 2003 Iraqi conflict, as well as in Pakistan in the so-called “drone war” [90]. In the Operation Barkhane, an ongoing counter-terrorism operation led by France and taking place in the Sahel region in Africa, “reaper” drones are regularly used to conduct airstrikes or provide intelligence [11]. This same model of drone, albeit unarmed, have been used by the Customs and Border Protection US agency in Minneapolis during the May 2020 George Floyd protests [51]. While military drones are used for civilian missions, civilian drones are also used in military conflicts. It has for example been reported that small commercial drones have been used by the terrorist organization ISIS in Iraq [78] to create improvised explosive devices. The same type of drones, quadcopters from the DJI company, have also been used by the Israel Defense Force to deploy tear gas around the Gaza strip [89].

[90]: Williams (2010), ‘The CIA’s covert Predator drone war in Pakistan, 2004–2010: the history of an assassination campaign’

[11]: (2020), *BARKHANE : Engagement de la composante aérienne dans une vaste opération*

[51]: Koebler et al. (2020), *Customs and Border Protection Is Flying a Predator Drone Over Minneapolis*

[78]: S. Schmidt et al. (2016), *Pentagon Confronts a New Threat From ISIS: Exploding Drones*

[89]: Waters (2018), *First ISIS, then Iraq, now Israel: IDF Use of Commercial Drones*

Flight Styles

Multiple classifications for drones exist. One way to classify drones is to look at how they fly. Most drones are either aircraft with fixed-wings, like modern days airplanes and gliders, or rotorcrafts, like helicopters. But more confidential drone designs also exist, such as flapping wings drones, also called ornithopter, lighter-than-air drones, like Google’s Project Loon (pictured in the chapter cover), or drones using combinations of designs, for example switching from rotorcrafts to fixed-wings after lift-off. These flight styles affect the freedom of movement of the drones, and their autonomy in terms of flight distance and duration. While a multicopter can hover at a specific position x, y and z, a fixed-wing drone needs thrust and thus a minimum flight speed in order to be able to compensate gravity with the lift and stay in the air, which is incompatible with maintaining a specific position. Yet, while all the energy of the multicopter is used to fight gravity by rotating propellers which push air downward, directly generating lift, a fixed-wing drone can glide. As such, the autonomy of fixed-wing drones greatly exceeds the autonomy of rotary-wings drones, which explains why most large drones, such as military drones, are fixed-wing, trading freedom of mobility for range.

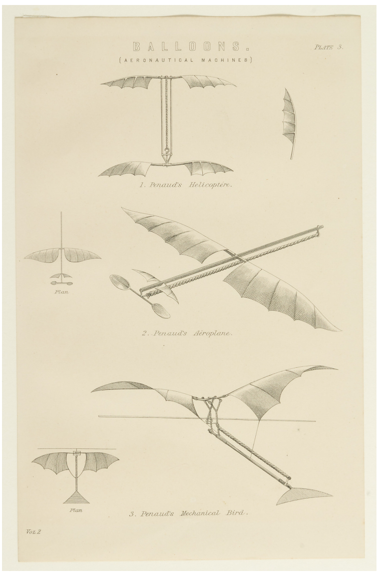
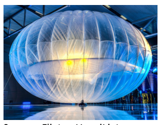


Figure 2.4: Penaud’s “Balloons”, displaying three different flight styles. From the Smithsonian National Air and Space Museum Collection, Gift of the Norfolk Charitable Trust.

Table 2.2: Comparison of different drones displaying different flight styles

	Source: U.S. Air Force Public Domain	 Source: Flicker User iLicker CC-BY-2.0	Source: Pixabay Pixabay License	Source: ZipLine Website	Source: Pixabay Pixabay License
Name:	MQ-9 Reaper	Loon	DJI Agras T16	Zipline	DJI Mavic Mini
First Introduced:	2001	2013	2019	2016	2019
Flight Style:	Fixed-Wing	Lighter-than-air balloon	Multirotor	Fixed-Wing	Multirotor
Max Payload:	1 700 kg	20 kg	16 kg	1.5 - 2 kg	30 g
Total Weight:	4 800 - 5 300 kg	?	42 kg	~20-22 kg	249 - 279g
Flight Duration:	30 - 40 hours	125 days (average)	10 - 18 minutes	90 minutes	30 minutes
Flight Range:	1850 km (radius)	-	3 - 5km	80 km (radius)	2km
Max Speed:	335 km/h	100 km/h	36 km/h	100 km/h	15-30 km/h
Applications:	Surveillance, Combat	Telecommunications	Agriculture	Delivery	Photo, Video, Hobby
Size:	20 m (wingspan)	15 m (diameter)	2.5m (width)	3.7 m (wingspan)	15 cm (width)
Power Source:	Fuel	Solar panels, gas, batteries	Batteries	Batteries	Batteries

Autonomy

While all the drones need a certain amount of automation to fly, for example to maintain engine speed in the presence of small control imperfections, not all drones are considered autonomous. A simple question to ask to decide whether a drone is autonomous or not could be "Can the drone carry out its mission independently of other entities, whether human or not ?". As for humans, the answer to this question is almost surely "no", which does not help in measuring the Level Of Autonomy (LOA) of drones.

According to ISO 8373:2012, which specifies the vocabulary used in relation with robots and robotic devices, autonomy is the "ability to perform intended tasks based on current state and sensing, without human intervention" [83].

Overall, drone autonomy is a continuum between no autonomy at all, or "operator does it all", and complete and full autonomy, or "drone does it all". With most drones being in the middle of the continuum, multiple authors have tried to split this continuum into sub-categories to ease reasoning about the drone autonomy. For example, three levels of increasing control autonomy are described in [31]. The first level, *sensory-motor autonomy*, describes drones which can "translate high-level human commands [. . .] into combinations of platform-dependent control signals", e.g. follow pre-programmed trajectory, but still require human supervision. The second level, *reactive autonomy*, describes drones which can react to their environment and external perturbations while maintaining their states, and require little human supervision. The third and last, *cognitive autonomy*, describes drones which *understand* their environment through the use of computer vision and perception techniques like SLAM, act accordingly, and require no human supervision.

[31]: Floreano et al. (2015), 'Science, technology and the future of small autonomous drones'

As the autonomous operation of drones cover many orthogonal aspects like obstacle and collision avoidance, energy autonomy or scene understanding, trying to quantify drones LOAs on a single scale seems futile. In the rest of this section, we focus on a few examples to illustrate different overall degree of autonomy, while retaining to give a general framework to classify the LOAs.

Energy As illustrated by Table 2.2, existing drones may use different source of power, with different energy density, which results in difference in their flight duration. Depending on its energy autonomy and energy source, a drone may need to pause its mission to renew its energy supplies, operation that can be manual or automated. Small to medium drones are most commonly powered by electricity, and use batteries as their power source. Such batteries need to be swapped out by an human operator when they are empty, resulting in a low autonomy regarding energy. Higher autonomy can be achieved by using automated charging pads, where the drones automatically land to be recharged by the main grid when they need to. This operation does not involve humans, but is often slower than just swapping the battery with a full one. Larger drones, especially Medium-Altitude Long Endurance (MALE) and High-Altitude Long Endurance (HALE), most commonly rely on fuel cell for their energy source. Such energy sources allow for longer period of flight than batteries, and automated aerial refueling has been demonstrated. High-Altitude Platform Station (HAPS) drones, which evolve at altitudes comprised between 20 to 50km, are probably the type of drone that display the bigger energy autonomy, as they rely on solar energy and evolve at altitude higher than the jet stream, allowing them to remain airborne for weeks without losing much energy fighting strong winds.

Flight Most drones can be remotely controlled by a human pilot, which takes full control in the position and movements of the device. This mode of operation is mandatory in many countries, including France, where the drone operator has to be able to directly control the drone at any time, for example to land immediately if a manned aircraft is in the drone vicinity. If the link between the pilot and the drone is severed, autonomy is needed to avoid a crash and safely land the drone; such a drone is therefore not considered as an autonomous drone as it does not operate independently of its operator. Greater flight autonomy can be reached by using global positioning systems such as Global Navigation Satellite Systems (GNSSs), or local ones based for example on the Ultra Wide Band (UWB) or 802.11mc technologies. Drones equipped with such positioning systems know their global or local positions, and can follow trajectories based on way-points or cover a certain area, autonomously. GNSSs are often included in the general public drones, as it allows drone manufacturers to enforce no-fly zones around sensitive areas such as airports, nuclear power plants or presidential palaces, and to comply with the regulations of the local authorities. It also enables functionalities like "return-to-home", allowing a drone to autonomously land at a specific position, most of the time set to its launch point, for example in case of failure of the link between the remote control and the drone.

Obstacle Avoidance Way-points and trajectories can be pre-computed offline, before the flight, transmitted to the drones while it flies, but they can also be computed on-the-fly, by the drones themselves. In this case, the drones needs to sense their environment, to avoid obstacles and to decide what will be their next way-point. This can be done through the use of techniques like Simultaneous Location And Mapping (SLAM), and the use of sensors like cameras, radar or lidar, or more generally any distance measuring sensor. Simple obstacle avoidance in multicopters based on ultrasound sensors has been there for nearly a decade (with "autonomous" landing being a simple obstacle detection located below the drones). As of 2020, more complex techniques have been integrated into general public drones since like the Skydio 2 [81] drone which

[81]: Skydio (2019), *Introducing Skydio 2*

can follow a target autonomously in an environment such as a forest, avoiding obstacles, thanks to its 6 camera and embedded GPU. Flying near large body of calm water is still a challenge for the Skydio 2, as "it can resemble a reflective, mirror-like surface that can confuse the drone's visual obstacle avoidance systems".

Applications

In this section, we try to give a general overview of drone applications, civil or military, and try to regroup them according to their main characteristics. Indeed, another way to classify drones is to look at their applications and fields of applications. Focusing only on the civilian applications or on the military applications seems futile, as the line between the two is blurry and the same platform can be used to do both. Many drone applications are mainly "sensing" applications, that is to say the drones' main purposes are to measure or detect changes and events in their environment, and report them back to the operator. Drones can also be used as actuators, with applications whose primary goal is to interact with the environment. These two modes of operations can be combined, as the drones becomes a sensor and an actuator.

3D Mapping Drones embarking localization sensors, cameras and distances sensors such as lidar can be used for mapping applications and 3D modelling, a sensing application. Fields of applications include archaeology, to search and discover new archaeological sites or model them, as well as cultural heritage preservation. Drones yield higher quality images at a cheaper price when compared to satellite images, and their maneuverability allows them to map otherwise unattainable areas. Such mapping can also be used for geological and mining surveys, post-disaster assessment, and military applications such as reconnaissance and surveillance [67]. As these applications require many overlapping passes, they are mostly automated with the drones following a pre-computed flight plan, and the generated data are stored on-board and uploaded from the drone after the flights.

[67]: Nex et al. (2014), 'UAV for 3D mapping applications: a review'

Aerial Videography and Photography Similar yet different fields of applications are the aerial videography and photography applications. They do not require localization sensors or distance sensors per se, but they might help for the flights. Drones, used as remote cameras, shoot videos and pictures for works of art, sports event, scientific applications, journalism or commercials. The mode of flying is most of the time manual, and requires a real-time video feedback from the drone to the operator in order for the video operator and flight operator (which sometimes are the same person) to correctly frame the images, and pilot the drone. Recent small commercial drones include tracking based on computer vision, allowing the drone to autonomously follow someone moving, e.g. during a sport activity.

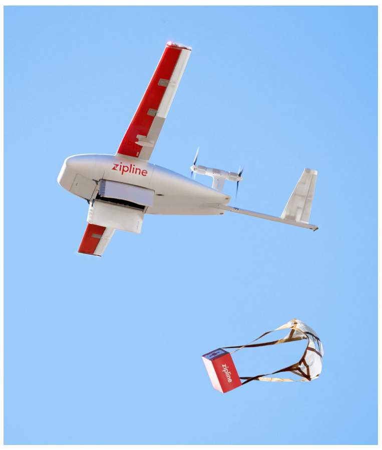


Figure 2.5: A ZipLine fixed-wing drone used to deliver medical materials and blood, using a parachuted package.

Real-time Surveillance, Detection and Data Collection As in the 3D mapping applications, the requirements are to cover areas or volumes and report the sensed data originating from the sensors, which can be

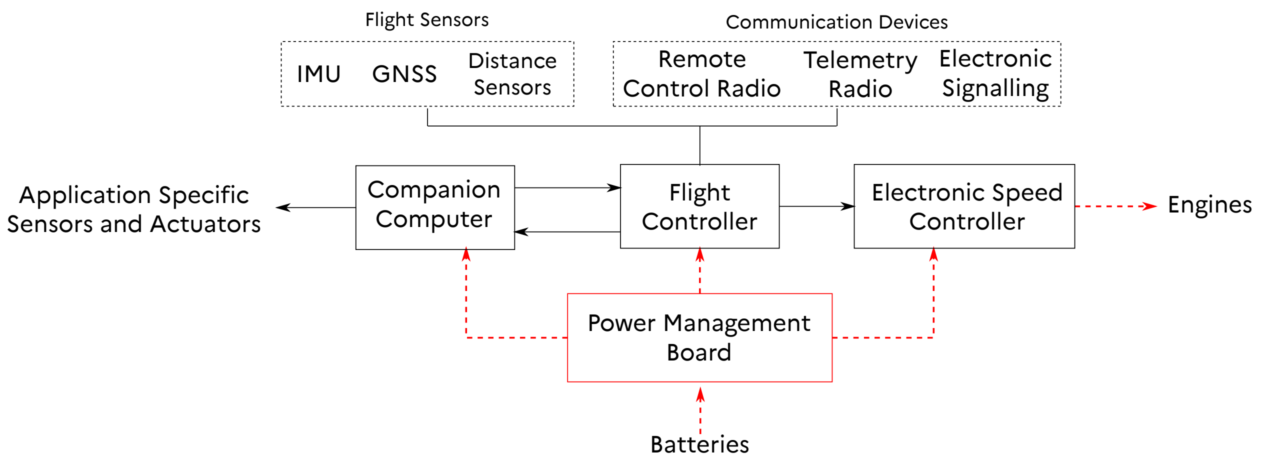


Figure 2.6: Generic architecture of a small multicopter drone.

cameras, chemical sensors or other physical sensors. Their operation is also mostly automated, but the data need to be available as it is collected, in real-time, which implies some network connection. Search and Rescue (SAR) applications fall within this category, as well as wildfire detection or border surveillance. In the law enforcement field, drones have for example been used to detect offenders and deliver officials instructions using embedded loudspeakers, in the city of Nice, France, during the 2020 COVID-19 related lock-down [76].

[76]: RFI (2020), *French police deploy drones to enforce coronavirus restrictions*

Payload Delivery This broad term covers applications where the drone acts as an actuator to move or deliver materials. This covers the delivery of goods, whether pizzas, parcel deliveries or for more serious matters like medical material deliveries or blood packs. Such deliveries are for example being experimented by ZipLine in Rwanda, a country whose road infrastructure does not permit efficient and fast delivery [1]. It can be also used in precision agriculture, where the payload can be pesticides to be sprayed over fields or grains to be planted. This also covers military combat drones, whose payload are ammunition or missiles, as well as law enforcement drones who have been used to deliver tear gas [89].

[1]: Ackerman et al. (2018), 'Medical delivery drones take flight in east africa'

[89]: Waters (2018), *First ISIS, then Iraq, now Israel: IDF Use of Commercial Drones*

Network Applications Drones, with their ability to move quickly in a predominantly barrier-free environment (given sufficient altitude) are of interest for network applications. They have been proposed to create relay networks for Wireless Sensor Networks (WSNs), combating sensor nodes isolation by creating paths of communication to a base station [22]. They also have been envisioned as aerial mobile base stations, whose position can be optimized to deploy communication networks without the need of a fixed infrastructure for a set of ground terminals [54].

[22]: De Freitas et al. (2010), 'UAV relay network to support WSN connectivity'

[54]: Lyu et al. (2017), 'Placement Optimization of UAV-Mounted Mobile Base Stations'

Architecture of a small multicopters

While each drone model is different, we describe the typical architecture of a small multicopter on Figure 2.6. The system energy is provided by batteries, and the power is distributed thanks to a power management board which also monitors the batteries' level. The drone flight capabilities are assured by the flight controller, in charge of computing which commands are sent to the electronic speed controller, itself in charge of directly controlling the engines. The flight controller most of the time runs a real-time operating system, which is required, given the low processing time requirements needed for the flight. The flight controllers directly receives orders from the remote control radio, and hosts the automation needed to maintain a global position using the GNSS and IMU sensors, as well as moving from one waypoint to another. The electronic speed controller is a low level component, the flight controller a medium level component, while the companion computer is a high-level component. Indeed, the companion computer is in charge of the intensive computing tasks, as well as the operation of mission dependent sensors and actuators that are not needed for the flight. Such tasks may include perception, network connectivity in the case of network applications, or managing video and photography surveys. Unlike the flight controller, the companion computer has no requirement for a real-time operating system. In Robot Operating System (ROS), a popular operating system for robots and drones, the position, speed, attitude or battery level information may be exchanged between the flight controller and the companion computer using software and hardware buses [71]. Such data can also be sent over the telemetry radio, the electronic signalling radio, and, when the footprint allows it, to the remote control radio.

[71]: Quigley et al. (2009), 'ROS: an open-source Robot Operating System'

2.4 Drone Networks

Many applications can profit from using multiple drones instead of one. Obviously, having multiple drones at hands allows them to carry out several unrelated tasks in parallel. But even for a single task or application, different drones can cooperate in order to speed up the achievement of the task, or simply to make it possible to carry out the task. For example, different drones can provide different angles of a scene at the same time, cooperating drones can lift more weight and thus deliver bigger payloads, and while the communication range of a network-providing drone might not be enough to serve as a relay between distant terminals, chaining multiple drones might allow to establish a communication.

In order to cooperate, drones in such applications create drone networks. Drone networks are not limited to drones cooperating on specific applications: even drones carrying out unrelated tasks might want to establish drone networks if they evolve in the same airspace, for example to exchange data making it easier to avoid collisions.

Network Requirements

The requirements of a drone network vary according to many parameters, starting with the considered applications. Such requirements for example differ according to the level of autonomy of the network participants: a *manual* drone needs to be able to receive control data at all time, while for a partially automated drone this control data is limited to emergency situations.

In [92], the authors give quantitative communication requirements for drone applications and classify these requirements according to the

[92]: Yanmaz et al. (2017), 'Aerial Wi-Fi Networks'

drone autonomy degree. The traffic is divided into two main families, traffic for the device autonomy and traffic for the mission autonomy. Device autonomy traffic relates to the control of the drones, while mission autonomy traffic relates to the coordination between the drones. Each of these families is itself divided into multiple categories:

1. **Control** traffic, which relates to the remote control data exchange needed to actually operate the drones;
2. **Telemetry** traffic, which includes IMU and GNSS data information and could be seen as monitoring traffic;
3. **Coordination** traffic, which is any data exchanged to coordinate the entities of the network;
4. **Sensed Data** traffic, which encompasses any data generated by the on-board drone sensors about their physical environment.

Control Traffic Standard remote control for drones may use different communication protocols, both analog and digital. The SBUS protocol, a popular serial-type digital protocol used to transmit data between the flight controller and the remote control radio, uses a baud-rate of 100 000 and a 8-E-2 configuration, which translates to a capacity for the throughput of at most 66.7 kb/s. It relies on 25 bytes long messages, allowing a remote to control 16 "servo" channels (each with a 11 bits resolution), and 2 binary digital channels. The messages are transmitted every 9 ms, putting the effective control throughput at 19.8 kb/s.

Such data is most of the time transmitted in the 433 MHz, the 915 MHz or the 2.4 GHz ISM band. Wi-Fi is not widespread (excepts for smartphone controlled toys), and simpler custom spread spectrum protocols are being used in its place. While Wi-Fi can accommodate the throughput of control traffic, the association, retransmissions and rate adaptation mechanisms of the Wi-Fi are typically not wanted features as they may introduce latency spikes and increase overall delay.

Telemetry Traffic While a standard GNSS module is expected to report its position at a 10Hz frequency, and an IMU is expected to report the linear acceleration and the rotational state at a frequency on the order of a hundred of Hz, telemetry data is often capped at a few Hz by sending filtered data. The standard baud-rate of the serial connection between the PX4 flight controller and its telemetry radio module is 57 600, which translates to a capacity of at most 46.1 kb/s of throughput in the standard 8-N-1 serial configuration, which allows to accommodate the 24 kb/s figure for the telemetry advanced in [92].

The world is divided into three International Telecommunication Union (ITU) regions: *Region 1* is composed of Europe, Africa, the former Soviet Union, and the Middle East without Iran, *Region 2* covers the Americas and some pacific islands, and *Region 3* is composed of the remaining countries, including most of Oceania.

Like control traffic, telemetry traffic is often transmitted in the 433 MHz (for the ITU Region 1) or 915 MHz ISM (for the ITU Region 2) bands, as they both are unlicensed bands with restrictions compatible with this application (for example, the 433 MHz has a duty cycle limit of 10%, which is sufficient to transmit telemetry data). Such data is also sometimes transmitted in the 2.4GHz ISM band, using Wi-Fi, as it allows ground station which already has a Wi-Fi WNIC, such as laptop or phones, to directly receive the data without the need of an additional radio.

Coordination Traffic Coordination traffic is loosely defined in [92] as any data which is used to coordinate the entities among the network. This can therefore overlap with telemetry data, or even sensed data.

In France, the law mandates that all drones weighing more than 800 grams are required to signal themselves using electronic or numeric signals, as well as light signals, if they're not tethered, not military or law enforcement drones, used outside or not used in special aero-modelling activity zones [58]. The signal format is defined in [57]: the messages are sent using 802.11 frames emitted on the channel 6 at least every 3 seconds or 30 meters apart, and contain the type of drone, the latitude, longitude, altitude, the horizontal speed, the lift-off position, as well as a unique registration number identifying the drone, which must be registered online on a government website. Each message containing 416 bits of payload, the minimum throughput needed by this system is 139 b/s (one message transmitted every three seconds). The Autonomous Dependent Surveillance-Broadcast (ADS-B) technology, transmitting at 1090 MHz using Pulse-Position Modulation (PPM) transmitted at 1Mb/s, which is already used in aircraft to broadcast their positions, identifier and speed, is also becoming widespread in drones. The drone manufacturer DJI announced in 2019 that all of its drones weighing more than 250 grams will receive an ADS-B receiver (but not a transmitter, to avoid "congesting the airwaves"), and transceivers are routinely used on bigger drones, which allows to detect them and follow their movements on specialized websites such as [3, 30].

[58]: Ministère de l'économie et des finances (2019), *Décret n° 2019-1114 du 30 octobre 2019 pris pour l'application de l'article L. 34-9-2 du code des postes et des communications électroniques*

[57]: Ministère de l'économie et des finances (2019), *Arrêté du 27 décembre 2019 définissant les caractéristiques techniques des dispositifs de signalement électronique et lumineux des aéronefs circulant sans personne à bord*

[30]: Flightradar24 AB (2020), *Live Flight Tracker*

[3]: ADSBexchange.com LLC (2020), *World's largest co-op of unfiltered flight data*

While such coordination traffic can be considered as *not* being *network* traffic, as it lacks routing, message, circuit or packet switching, such simple coordination traffic can enable flocking and formation flight. In [88], drones exchanging in a broadcast manner using the XBee protocol their ID, position, velocity, attitude and status info (which is essentially what the law mandates [58]) are creating a self-organized flock using a decentralized control algorithm analogous to Reynolds' Boids, a swarming control algorithm based on the use of virtual forces [75].

[88]: Vásárhelyi et al. (2014), 'Outdoor flocking and formation flight with autonomous aerial robots'

[75]: Reynolds (1987), 'Flocks, herds and schools: A distributed behavioral model'

Routing Algorithms In drone networks which rely on ad-hoc routing protocols, coordination traffic may include the maintenance data of the routing protocol. Whether proactive, reactive or hybrid, the overhead these protocols introduce in order to discover, establish and maintain routes is largely dependent on the number of participating nodes, the network topology and the application running in this network. In [66], three proactive mesh routing protocols are compared in emulation, in the context of a wireless community network, the BMX6, OLSR, and Babel protocols. For the three protocols, the network overhead increases with the number of participating nodes, and is comprised between 1.2 kb/s and 3.2 kb/s for 60 nodes. In [15], the overhead for an indoor 8 nodes multi-hop network with OLSR and AODV are compared. The OLSR overhead falls in 1.6 kb/s - 9.6 kb/s range, while the AODV overhead falls in the 1.6 kb/s - 3.2 kb/s range. In [62], the OLSR, the BATMAN L2 and L3 protocols, and the Babel mesh routing protocols are compared experimentally with N nodes. The overhead appears to widely exceed the previous values, as OLSR overhead is reported to be 86.528 kb/s, BATMAN L2 and L3 overheads are reported to be respectively 30.360

[66]: Neumann et al. (2015), 'Evaluation of mesh routing protocols for wireless community networks'

[15]: Borgia (2005), 'Experimental evaluation of ad hoc routing protocols'

[62]: Murray et al. (2010), 'An experimental comparison of routing protocols in multi hop ad hoc networks'

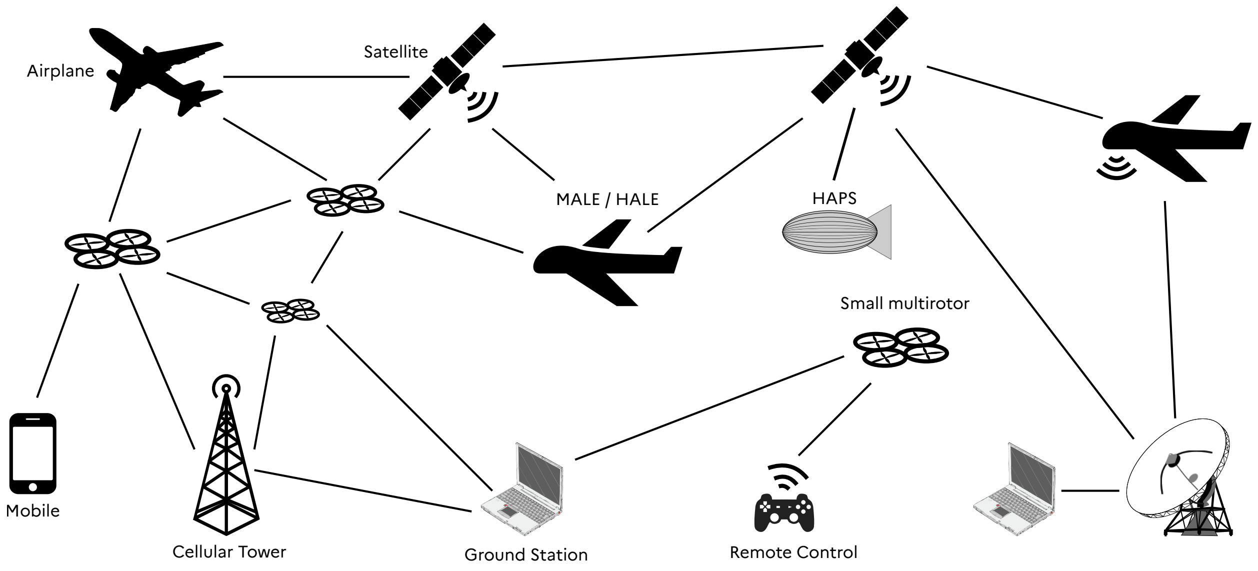


Figure 2.7: Synoptic view of the different entities that may compose a drone network, with lines representing a communication link between the entities. Communication link can be directional or bi-directional, and transport control, telemetry, coordination or application data.

kb/s and 31.616 kb/s, while Babel overhead is reported to be 3.576 kb/s.

Network Architecture

Drone Networks also vary according to the type of network architecture used, in particular its topology and its governance. Drones in a network may be connected to each other, but they also may be connected to non-drone entities. Such entities might be ground-based, for example a residential Wi-Fi access point, a cellular base station located on a communication tower or a vehicle, they can be air-based, like planes or helicopters or even space-based, like telecommunication satellites. In [33], the authors identify four main communication architectures which can be used for networks of small drone: satellite, cellular, direct link, and mesh networking. Overall, we can additionally classify communication links composing the networks into three categories:

[33]: Frew et al. (2008), ‘Airborne Communication Networks for Small Unmanned Aircraft Systems’

- Air-to-Air, e.g. between drones, drones and aircraft;
- Ground-to-Air and Air-to-Ground, e.g. between drones and ground stations, drones and vehicles, drones and fixed cellular infrastructure;
- Air-to-Space and Space-to-Air, e.g. between drones and satellites;

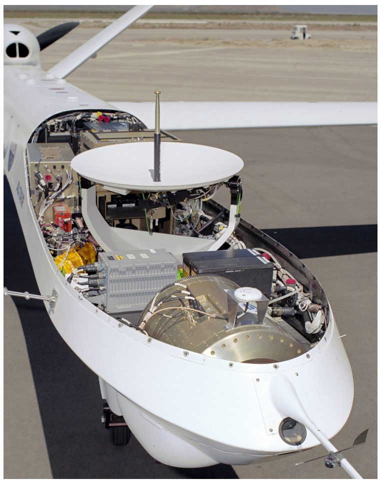


Figure 2.8: Close-up on the dish used for satellite communication in a NOAA-NASA MALE Reaper drone . The dish is mounted on a steerable mount, pointing upwards, allowing to aim specific satellites.

A synoptic overview of a drone network is shown on Figure 2.7. MALE, HALE and HAPS are mainly controlled through satellites, which can cover large area and cover the high speed mobility of these drones using a line-of-sight channel. Given the altitudes of such drones, obstacle avoidance is limited to other objects evolving in the air such as airplanes, which broadcast their positions, and does not need a low-latency connection as a drone evolving in an urban area would. Such drones also rely on satellites for their positioning, too, through the use of protocols such as Galileo, GPS, Beidou or Glonass. The membership in such networks is closed, with entities participating having very specific roles: most of the data flows between the ground station and the drone, with satellites serving as relays. This is illustrated by Figure 2.8, on which the steerable communication antenna is oriented upwards, pointing to satellites. The network architecture is therefore highly hierarchical.

Drones evolving at lower altitudes may connect to cellular networks such as LTE or 5G networks, which are also hierarchical networks. However, in [25], the authors point out that drones connected to LTE networks are expected to experience five handovers per minute when evolving at an altitude of 150 meters, compared to only one for a ground user. The higher rate of handover is being blamed on the high altitude (related to pedestrians) of drones which makes them prefer remote antennas as the drone evolve in their side lobes, whereas the main lobes of the antennas have been optimized for the ground. Handover between different technologies (namely 4G and 5G) are described in [63], where the authors observe more handovers as the height of the drone increases.

[25]: Fakhreddine et al. (2019), *Handover Challenges for Cellular-Connected Drones*

[63]: Muzaffar et al. (2020), 'First Experiments with a 5G-Connected Drone'

Low altitudes drones may also be directly controlled from the ground, using direct links. Such links may use specialized protocols, which is the case for most of the hobbyist remote control. For example, the FrSky Taranis Q X7, which was used during the thesis, the transmission protocol is a Frequency-Hopping Spread Spectrum (FHSS) protocol using a 2-FSK modulation with 47 channels, according to its FCC report. The direct links can also use WLAN protocols, such as Wi-Fi, Bluetooth, or Wi-Fi-like protocols: in the 2018 application for the Japan market of the "DJI Smart Controller" [12], we can note it embarks a 802.11ac compatible Wi-Fi chipset, a Bluetooth chipset, and a Software Defined Radio (SDR) using OFDM, as the recent versions of the 802.11 standard. In [95], the authors analyze the network performances of a Wi-Fi network between a ground station and a small quadcopter drone, as well as between two flying quadcopter drones. The drones are equipped with 802.11a/b/g/n compatible cards, and communicate with a 802.11a compatible Access Point (AP), which serves as a ground station. The use of the 5GHz frequency band for the Wi-Fi prevents interferences from the 2.4GHz operating remote control. Specific attention has to be put into the number of antennas used by the AP and drone, as well as their orientation: the authors introduce an antenna setup composed of three dipole antennas organized in a horizontal triangular manner, and compare its performances with a single vertical dipole antenna: the three-antenna setup can offer up to 15dB of gain compared to the single antenna setup, especially at elevation degree close to 90°, but this comes with a higher fluctuation in the elevation plane.

The Federal Communications Commission (FCC) and other regulatory organization require mandatory testing for new products that emits radio waves. Such reports are public and represent an easily accessible source of information for determining the characteristics of general public drone control methods.

[95]: Yanmaz et al. (2013), 'Achieving air-ground communications in 802.11 networks with three-dimensional aerial mobility'

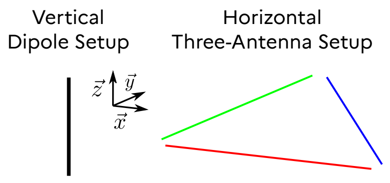


Figure 2.9: Illustration of the antenna setup used in [95].

For certain applications, for example enabling emergency communications in the wake of a natural disaster, low altitude drone mesh networks are envisioned [24]. Their self-organization, scalability and resiliency properties are of interest when dealing with entities such as drones that are highly mobile as well as highly energy constrained, which lead to network whose topology changes regularly, and whose membership is not hierarchical nor static: nodes may come and go according to their own constraints. As most of the drone mesh networks are based on Wi-Fi networks, this network architecture will be covered in the next section.

[24]: Erdelj et al. (2017), 'Help from the sky: Leveraging UAVs for disaster management'

Wi-Fi for Drone Networks

Wi-Fi has many characteristics that makes it a good candidate for small drone networks, and especially drone mesh networks.

Indeed, as of 2020, Wi-Fi hardware is available off-the-shelf, is inexpensive, and is mature. It is therefore easier for researchers to experiment with Wi-Fi for drone networks than experimenting with technologies like 5G, satellites networks, or even Long Term Evolution (LTE) networks, whose entry cost can be prohibitive.

No license is needed to operate a Wi-Fi access point or Wi-Fi devices, whereas deploying an LTE eNodeB can only be done in a controlled environment (e.g. in an anechoic chamber) or with the endorsement of the regulator and of the telecommunications' operator owning the concerned frequencies (at least, on the paper).

From a purely operational point of view, Wi-Fi hardware is available from many vendors, and any hardware should be interoperable with the other types of hardware to get the Wi-Fi Alliance Certification. This heterogeneity prevents vendor lock-in, allows not to rely on a single design, which, from a security and availability point of view, is important.

Wi-Fi is compatible with most of the end-user devices currently in use to access the Internet, which are smartphones and laptops, and its connection mechanism is well understood by the general public. Whether for direct control of the drones, or to use a network access provided by drones, Wi-Fi appears as a jack of all trades.

In addition to the different modes available such as infrastructure, ad-hoc or mesh, modern Wi-Fi networks support hundreds of different combinations of transmission parameters, leading to more than one hundred different attainable physical throughput, ranging from 6.5 Mb/s to 6933.3 Mb/s in the 2.4 and 5 GHz bands. All of these tweakable parameters allow Wi-Fi networks to operate in a wide range of conditions: if the channel is good, high throughputs are attainable, if the channel is degraded, Wi-Fi networks are still able to provide connectivity using more robust transmission parameters, although with lower throughputs.

State of the Art and Tools for Drone Networks Performance Evaluation 3

In this thesis, we focus on the use of Wi-Fi networks in drone networks, and our main goal is to find mechanisms that would increase the *performances* of Wi-Fi-based drone networks. In order to understand what are the challenges of such networks, we first give a state of the art concerning the experimental evaluation of Wi-Fi drone networks. Overall, this state-of-the-art underlines poor performances from Wi-Fi drone networks, which are in particular far from the expected performances of Wi-Fi networks in terms of attained throughput. Key parameters include the positioning and the general mobility of drones, which could be leveraged to improve the network performances. Therefore, we then focus on controlled mobility for drone networks, with a state on the art on controlled mobility. We finish with a panel describing the tools available for the performance evaluation of Wi-Fi-based drone networks, in particular simulators and testbeds, which allow reproducible and repeatable approaches in the development and evaluation of such solutions.

3.1 State of the Art of Experimental Evaluation of Wi-Fi-based Drone Networks

Deploying real drone networks is time-consuming and may have serious side-effects in case of engine or communication failure. We could argue experiments involving drone networks and Wi-Fi networks *only* reflect the reality in which they are carried out, that experiments are hard to design, set up correctly, and most of the time not fully controlled. But given the complexity of the considered systems, experiments remain, in my opinion, the best way to try to understand their behaviors and get things out of them. In this section, we present a state of the art on the experimental evaluation of drone networks and Wi-Fi drone networks, organized primarily in chronological order, with some exceptions for closely linked works.

In [20], the authors study the performance of "802.11a" air-to-ground wireless links, using a fixed-wing drone. The drone flies at heights of approximately 45 m at speeds of 65 km/h, over 4 ground nodes. Each drone is equipped with two wireless adapters with two antennas each, and ground nodes are equipped with two wireless adapters with one antenna each. Two types of antenna are used, one retail dipole antenna and one custom antenna. Both have omnidirectional radiation patterns in the E-plane, but the custom antenna has an overall much narrower beam in the H-plane. No association between the drone and the ground stations is needed, as the drone broadcasts frames, with the ground stations being simple listeners not acknowledging the received frames. The authors conclude that for the best performances in terms of throughput, the antenna should be horizontal both on the drone and on the ground stations. The authors study the relation between the Received Signal

[20]: Cheng et al. (2006), 'Performance Measurement of 802.11a Wireless Links from UAV to Ground Nodes with Various Antenna Orientations'

Strength (RSS) and distance, and, using a log-distance propagation loss model Equation 3.1, determine the path loss exponent to be $\alpha = 1.80$, resulting in better (in terms of range) performances than in free space. These results are believed to be due to the bias introduced by the inability to take into account frames with too little RSS. Indeed, frames with lower RSS cannot be decoded, and are not taken into account by the setup of the authors which only accounts for decoded frames, resulting in biased measurements and model. In a "flyover" scenario, the authors note the achieved throughput ranges from 11.1% to 42.1% of the maximum possible throughput, depending on the antenna configuration.

[94]: Yanmaz et al. (2011), 'Channel measurements over 802.11 a-based UAV-to-ground links'

[95]: Yanmaz et al. (2013), 'Achieving air-ground communications in 802.11 networks with three-dimensional aerial mobility'

In [94], the authors evaluate the channel between an airborne quadcopter drone and a ground station with two antennas, for 802.11a links, using RSS and throughput measurements, and using two different simple antenna orientations. In [95], the same authors look at similar metrics, also for a 802.11a link, but this time with a more complex antenna setup on the ground station which is equipped with three antennas. Fitting the data they obtain to a log-distance propagation loss model Equation 3.1, they obtain values for α of 2.01 and 2.03, which are very close to the value $\alpha_0 = 2.0$ which corresponds to a free-space path loss. When $\alpha > \alpha_0$, this corresponds to situations where the signal does not travel as easy as in a free-space such as vacuum or air, e.g. in a city, while $\alpha < \alpha_0$ corresponds to situations where the signal travels more easily, e.g. when travelling in waveguides. The obtained UDP throughput varies greatly depending on the yaw of the drone, and the antenna orientation, and drops are observed during the mobility phases of the drone.

$$PL(d) = PL(d_0) + 10\alpha \log_{10}(\frac{d}{d_0}) \tag{3.1}$$

with:

PL: Path Loss between the receiver and the transmitter, in dB
d: Distance between the receiver and the transmitter, in m
d_0: Reference distance, in m
$PL(d_0)$: Path Loss between the receiver and the transmitter at the reference distance $d = d_0$, in dB
α: Path Loss Exponent, dimensionless

[37]: Goddemeier et al. (2012), 'Experimental validation of RSS driven UAV mobility behaviors in IEEE 802.11s networks'

In [37], the authors evaluate the gap between simulation, software-in-the-loop, hardware-in-the-loop and experiments, for a single hop air-to-ground and ground-to-air 802.11n and 802.11s network between a base station and a single quadcopter drone. While the authors do not perform any modeling of the channel, they observe a match between the "general trend of the RSS" and a Friis channel model, that is to say a free-space propagation loss model, or a log-distance propagation loss model with $\alpha = 2.0$. By artificially reducing the transmission power of the Wi-Fi cards to simulate a Non Line Of Sight (NLOS) channel and compare it to a Line Of Sight (LOS) channel, the authors illustrate the use of the RSS as a metric to guide mobility, but do not infer much from the experiment, which serves as an example of their approach. No higher level metrics, such as throughput, are studied.

[8]: Asadpour et al. (2013), 'Characterizing 802.11n Aerial Communication'

[7]: Asadpour et al. (2013), 'From ground to aerial communication: Dissecting WLAN 802.11 n for the drones'

In [7, 8], the authors evaluate the performance of a 802.11n link between two airborne drones flying according to waypoints which allow to have a wide range of relative speeds between the drones. They study how application throughput is related with the distance between the drones, or their relative speeds, with different fixed Modulation and Coding Scheme (MCS), which are integers representing the type of modulation and coding rate used by the WiFi transmitter. They also look at the performances of the Rate Adaptation Algorithm (RAA), in charge of automatically setting physical layer parameters, like the MCS, for the transmissions. While they get a reference throughput of the order of 176 Mb/s in an indoor environment, they obtain throughput of the order of 19 Mb/s during their drone-to-drone experiment. Their conclusion is that the RAA of the Ralink 3572 chipset they use is not adapted to the high mobility of the drones, as setting the MCS to the value with the lowest associated throughput allows for a better throughput than the one obtained with the rate adaptation mechanism enabled. The lack of spatial diversity for the airborne channel is also advanced as a potential problem.

In [44], the authors evaluate the performance of 802.11n and 802.11ac links in single and two-hop air-to-air and air-to-ground tests using the AP and mesh network architecture. This work is an extension of the work in [93] which looked at 802.11a networks with the same approaches. Initial tests performed indoor show saturating UDP throughput (respectively TCP) can reach up to 350 Mb/s (respectively 260Mb/s) for the 802.11n link, and 480 Mb/s (respectively 345 Mb/s) for the 802.11ac link. For a single-hop static air-to-ground scenario, the reached throughput for 802.11n reaches 150 Mb/s for UDP, and 100Mb/s for TCP for small distances up to between 50m and 100m, but significant losses in the attained throughput are observed for bigger distances. Significant drops are also observed, with the throughput going from 80Mb/s at a 110m distance between the ground station and the UAV to 40Mb/s at a 120m distance. Drone mobility is observed to significantly decrease the obtained throughput. Outdoor scenarios do not underline a big advantage of 802.11ac over 802.11n (except, maybe, at short distances). The *ath10k* driver, used for 802.11ac, is blamed, as better performance are obtained in 802.11n (compared to 802.11ac) using the same hardware when using another driver, the *ath9k* driver.

[44]: Hayat et al. (2015), 'Experimental analysis of multipoint-to-point UAV communications with IEEE 802.11n and 802.11ac'

[93]: Yanmaz et al. (2014), 'Experimental performance analysis of two-hop aerial 802.11 networks'

In [98], the authors compare the performances of ZigBee and 802.11a in single hop air-to-ground and air-to-air scenarios, as well as in a two-hop scenario involving the two types of channel. Compared to ZigBee, the Wi-Fi link is characterized by a high latency of 230 ms for the air-to-air scenario, 380 ms for the air-to-ground scenario, and 840 ms for the two-hop scenario (whereas the ZigBee latency is respectively 25 ms, 42 ms and 106 ms). UDP throughput of 19 Mb/s, 13 Mb/s and 5 Mb/s are obtained for respectively the air-to-air, the air-to-ground and the two hop scenarios. The air-to-air link is observed to have a better throughput and latency than the air-to-ground link, which is explained by the authors to be due to the line of sight propagation.

[98]: Zhou et al. (2015), 'Multi-UAV-aided networks: Aerial-ground cooperative vehicular networking architecture'

In [69], the authors propose an emergency communication network based on multiple drones connected using 802.11n and 802.11ac Wi-Fi networks. Although they achieve promising performances in terms of throughput with regard to distance using both 802.11n and 802.11ac communication links, those results are not obtained with flying drones but only using WNIC located on the ground, in static positions.

[69]: Panda et al. (2019), 'Design and deployment of UAV-aided post-disaster emergency network'

[42]: Gu et al. (2015), 'Airborne WiFi networks through directional antennae: An experimental study'

In [42], the authors study the performance of a IEEE 802.11g airborne link between two drones equipped with directional antennas. Comparing ground experiments with airborne experiments using a 10m altitude, for three distances between the drones (150, 300 and 1000m), they observe a decrease in the throughput of up to 49% when hovering compared to the experiments done on the ground (UDP). The standard deviation of the obtained throughput also increase of up to 100% for the hovering drones. The obtained throughput IEEE 802.11g are however close to the limit of the standard, as it reach an average of 36.2 Mb/s for a maximum of 54 Mb/s for the 802.11g amendment.

[86]: Ullah et al. (2017), 'An unmanned aerial vehicle based wireless network for bridging communication'

[85]: Ullah et al. (2020), 'Connecting Disjoint Nodes Through a UAV-Based Wireless Network for Bridging Communication Using IEEE 802.11 Protocols'

In [86] and [85], the authors evaluate the performances of a two-hop 802.11n network composed of two ground nodes communicating through an airborne drone mounted AP. Even with small distances between the nodes, as the drone is located at altitudes of 10, 15 or 20m, with ground distances of less than 8m with the ground nodes, which caps the global distances at 22m, the obtained throughput range between 10 and 30 Mb/s while the maximum supported transmission rate is 144.4 Mb/s for the 20 MHz, 2 spatial streams setup they are using.

Overall, the performances for the 802.11 links for drone networks are a bit disappointing, even for a few nodes or a few hops. The antenna types and orientations need to be well-adapted for airborne communications, as the relative position of the drones are not comparable to those of more traditional, ground-based devices. While the drone-to-drone channel is mostly line of sight, performances are often worse than in a classical Wi-Fi office environment where the channels are often not line of sight. Some possible explanation for those poor performances is the lack of spatial diversity, preventing from taking advantage of multiple spatial streams. When looking at the global evolution of the performances with regard to distances between drone network entities, we can observe different trends which are not always easy to quantify. Assuming moving nodes farther will result in worse performances and moving nodes closer will result in better performances might be true on a large scale, but on a small scale this might not be. Less related to drones, but still interesting: the use of a newer standard does not guarantee overall better performances (even with the same hardware) as it implies changes in the software components part of the WNIC.

3.2 State of the Art of Controlled Mobility for Drone Networks

Controlled mobility for Wi-Fi drone networks could be used to increase the performances of such networks. For example, it could take advantage of the discontinuities in the evolution of the performances with regard to distance we mentioned in the previous section, or ensure that the antenna orientations between the different entities are maintained to "good" positions. In this section, we present a state of the art on the controlled mobility for Wi-Fi-based drone networks. We also present a few works that do not qualify completely as *controlled mobility*, but are still of interest for our study settings, as they have a lot in commons with Wi-Fi-based drone networks.

In [53], the authors describe a strategy to exploit multipath fading for patrolling robots networking based on the IEEE 802.15.4 communication technology. Such robots, which are expected to be able to precisely stop their movement, would be able to detect high SNR positions and pause periodically its movement to take advantage of those better than average positions, spending more time in them. The authors do not experiment the complete system, but using real SNR fluctuation traces, they simulate a simple scenario underlining some gain in the capacity of the obtained channel can be expected. Such mobility patterns can hardly be expected from drones that are subject to atmospheric disturbances, but this work was probably one of the first to introduce a notion of "mobility diversity". In [82], the authors use an SDR to generate a 802.11/OFDM-like waveform and study the evolution of fading with regard to the position of some ground robot mounted receiver. They arrive at the same conclusions as in [53], describing multiple strategies available to the robot to find hot spots by moving around its position.

[53]: Lindhé et al. (2009), 'Using robot mobility to exploit multipath fading'

[82]: Smith et al. (2009), 'RF-mobility gain: concept, measurement campaign, and exploitation'

In [38], the authors explore how *robotic wireless networks* based on Wi-Fi can be of interest because of their ability of the nodes to move. They do not experiment with drones, but with roomba-mounted laptops using 802.11n Intel WNIC, and the experiments are performed indoors. It is observed that many *high-gain* locations exist that could be used to improve the general performances of the network with an AP moving to such locations. They confirm mobility provides more diversity in the channel by observing an increase in the number of different throughputs they obtain, compared to a static AP. Their experiments underline that high gains (up to 65% for downlink and 90% for uplink) are attainable. Still, they rely on a centimeter scale mobility which drones maintaining their position based on GNSS measurements cannot easily achieve, and the applicability of their findings is unknown in an outdoor scenario. This work could be seen as a realistic test of the conclusions of [53] and [82].

[38]: Gowda et al. (2016), 'The case for robotic wireless networks'

In [43, 74], the authors describe a mobility solution based on virtual forces for the positioning of drones in a drone network. The communication between the drones is done using the 802.11b/g amendments, and it is assumed drones have access to their global position, for example using a Global Navigation Satellite System (GNSS) system, they exchange with neighboring nodes to be able to compute the virtual forces characteristics. Two types of force are considered, which encompasses three different behaviors. First, a force whose goal is to attract drones which are too far from each other, i.e. at a distance $d > d_1$, and to repel them when they are too close, i.e. for distances $d < d_0$, with $d_0 < d_1$. Then, another force, a *friction* force, whose goal is to slow down the movement of drones whose position is considered to be correct, i.e. for distances $d_0 <= d <= d_1$. Multiple scenarios, involving multi-hops communications, are considered and studied in simulation. The OLSR routing protocol is used. The transmission rate considered for the simulations is constant, at 11 Mb/s. The number of transmission rates to be considered would be small in any way, as the considered standards are 17 years old.

[74]: Reynaud et al. (2016), 'Design of a force-based controlled mobility on aerial vehicles for pest management'
[43]: Guérin-Lassous et al. (2017), 'Improving the Performance of Challenged Networks with Controlled Mobility'

In [19], the authors propose an antenna heading control system for drones equipped with directional antenna, and test the system using two drones. The system is mainly based on GNSS information (GPS), but also on the RSS when GPS data are not available. The AirMax proprietary protocol, a Time-division Multiple Access (TDMA) protocol, is used for

[19]: Chen et al. (2017), 'Long-Range and Broadband Aerial Communication Using Directional Antennas (ACDA): Design and Implementation'

the drone to drone communication, while Wi-Fi is used for air-to-ground communication, and synchronization between the drones is provided using the XBee communication protocol. The same authors study the same problem in [52] but use a reinforcement learning approach instead of the simpler algorithm presented in [19]. The main takeaways from those works are that it is possible to do online antenna orientation on drones to enable long-range communication links: the authors achieve an end-to-end capacity of 800 kb/s at a communication distance of 5 km, but this is mainly thanks to the AirMax protocol, and it is unclear if such results would hold using Wi-Fi. We will develop this approach for Wi-Fi networks and fleet of drones later in this thesis.

[52]: Li et al. (2019), 'Design and implementation of aerial communication using directional antennas: learning control in unknown communication environments'

In [59, 73], the authors propose a positioning algorithm to deploy a node (which is depicted as a ground robot) acting as a relay between a source and a sink in a two-hop 802.11b mesh network. The algorithm tries to equalize some metric for the first link (source to relay) and second link (relay to sink), metric which can be the RSS, the RTT, the transmission rate, or a hybrid metric. This algorithm is studied in the ns–2 simulator. While the movement of the relay depends on the used metric, the attained throughput seems to converge in all the different scenarios to 2Mb/s. In [60], ns–2 simulations involving more complex network topologies, with multiple relays or multiple sources, are studied. Depending on the channel model used, low level metrics such as the SNR or the RSS might need thresholds to avoid useless movements.

[59]: Miranda et al. (2012), 'Adaptive router deployment for multimedia services in mobile pervasive environments'
[73]: Razafimandimby et al. (2013), 'Fast and reliable robot deployment for substitution networks'

[60]: Miranda et al. (2015), 'On the Impact of Routers' Controlled Mobility in Self-Deployable Networks'

Controlled mobility has been explored both in simulation and in small scales experiments for robot networks, and in particular for drone networks. Yet, the precision needed by some mobility solutions is not compatible with the expected mobility of drone networks, which means what can be done on the ground is not directly applicable to airborne networks. Moreover, many solutions have not been evaluated in conjunction with Wi-Fi networks, but with other types of networks such as AirMax or SDR based networks. When Wi-Fi is used, it is only with older versions of the standards, relying on now deprecated simulators, whose behavior, while probably similar with the current state-of-the-art, is less relevant nowadays.

3.3 Tools for the performance evaluation of drone networks

In this section, we focus on the tools and methods available to conduct the performance evaluation of Wi-Fi-based drone networks, which include models, simulations, testbeds and experiments related to Wi-Fi networks and drone networks. This serve as a *state of the art* of performance evaluation of drone networks, and introduces tools that have been used throughout the thesis. We mainly focus on free and open-source software, when applicable, and freely usable testbeds. While we are not exhaustive, we try to cover the most important tools available.

Simulators

Broadly speaking, we can identify two types of simulators that are of interest for the study of Wi-Fi drone networks: network simulators and robot simulators. Network simulators focus on the simulation accuracy of networks: they re-implement networking stacks, from the physical layer up to the application layers, with different accuracy degrees for each layer. It is often possible to simulate robots in a network simulator, but the accuracy of their movement, their dynamics and the interactions of the drones with their environment are limited to basic functionalities. Robot simulators, at the opposite, focus on the simulation accuracy of robots, in particular their movement, dynamics and physics, their interactions with their environments and their sensors. Networks can also be simulated in certain robot simulators, but only using coarse models or simple approximations. Recent software development tried combining classical network simulators with classical robot simulators. Such simulators, which we will call *hybrid* simulators, act as glue between networking and robotic simulators. Unfortunately, they are often *one-shot* projects which end up unmaintained and unusable over the long term.

Drone and Robot Oriented Simulators

Figure 3.1: Two Intel Aero UAV and one Crazyflie drones simulated in Gazebo. Courtesy of Vincent Le Doze.

Gazebo Gazebo is a robot simulator written in C++ developed by *Open Robotics* (previously known as the *Open Source Robotics Foundation*). The same group also develops the Robot Operating System (ROS) middleware, which provides APIs abstracting robotic hardware and provides standardized communication interfaces between the different components that compose a robot such as a drone. Gazebo and ROS are compatible with each other, which means some implementation of a robot or drone controller in Gazebo can be with little or no modifications on a ROS-based system. Gazebo allows for a very-detailed simulation in a 3D environment of certain individual components of a drone such as sensors, engines or flight controllers, as long as one develops them. A library of common components and sensors such as a magnetometer, an altimeter, an IMU or a camera exists, but network components are, at best, lacking. Indeed, only two wireless network sensors exist, namely the *Wireless Transmitter* and the *Wireless Receiver*, and they only model a basic protocol which only broadcasts fixed-payload beacons. Beacons are transmitted at a specific rate and at a specific frequency, and can be received by any node in range if its receiving frequency range includes the transmitter frequency. To determine if two nodes are in range with each other, the simulator performs a link-budget calculation using a custom path log-distance loss model described in Equation 3.2, incorrectly labelled "Okumura–Hata model" in the source code of the simulator. If the received power P_r exceeds a sensitivity threshold of −90.0 dBm, the transmitted message is correctly received, else it is discarded.

$$P_r = P_t + G_t + G_r - 20\log_{10}(4\pi) + 20\log_{10}(\lambda) - 10n\log_{10}(d) - |X_\sigma| \quad (3.2)$$

with:

P_r**:** Received Power, in dBm

P_t: Transmitted Power, in dBm (default: 14.5 dBm)
G_t: Transmitter Antenna Gain, in dBi (default: 2.6 dBi)
G_r: Receiver Antenna Gain, in dBi (default: 2.5 dBi)
λ: Wavelength, in m (default frequency: 2442 MHz)
d: Distance between the receiver and transmitter, in m
X_σ: Normal Random Variable with standard deviation of σ, in dB (default for σ: 6.0 dB)
n: Path Loss Exponent, dimensionless (12.0 if there is at least one obstacle between the transmitter and the receiver, 6.0 otherwise)

The fixed transmission rate and payload and the fact multiple nodes can transmit at the same time on the same frequency without any contention make this sensor unsuitable for drone network simulation.

Fl-AIR Fl-AIR, which stands for *Framework Libre AIR* is a simulation framework mainly aimed at drone simulations, developed in the Heudiasyc Laboratory. It is written in C++, and relies on the Irrlicht 3D engine [36] for the physics simulation. The goal of Fl-AIR is to have simulations and real drones running the same codebase [79]. By running the exact same code as actual hardware, the simulator is expected to find implementations bugs without resorting to potentially costly and dangerous experiments, while lowering the costs of maintaining two different codebases. However, this increases the complexity of the simulation code that is expected to be able to drive real drones. A library of multiple sensors and actuators is provided in the project source code, but no network components are available in Fl-AIR, which makes it unsuitable to simulate drone networks "as is". The CUSCUS simulator, described in a few paragraphs, aims to fix this aspect. The community around Fl-AIR is limited to a few people which means its momentum is far smaller Gazebo's, for example, and its development is slower.

[36]: Gebhardt et al. (2002), 'Irrlicht Engine - A free open source 3d engine'

[79]: Sanahuja et al. (2016), *Fl-AIR: Framework Libre AIR*

Figure 3.2: Fl-AIR simulation environment screenshot, with a quadcopter drone (middle of the image) flying in an urban environment. Extracted from the project website.

AirSim AirSim, which stands for *Aerial Informatics and Robotics Simulation*, is an open-source robot simulator developed by Microsoft aimed at drones and autonomous vehicles research. Launched in 2017, it relies on the Unreal Engine 4 (source available) or Unity (experimental support, open-source) game engines for the physics simulation. It is compatible with ROS and supports hardware-in-the-loop controllers like the PX4 one [80]. AirSim focuses on the computer vision aspects of robot simulation, leveraging the Unreal engine to simulate rich and realist environments, supporting for example weather effects. It supports multiple sensors, such as lidar or camera sensors that can be used to develop algorithms such as SLAM algorithms: because of its "realism", AirSim is also being used to generate training data for deep learning applications. While the simulator has a few sensors and actuators, it does not have any built-in networking component or networking simulation (aside relying on the one from ROS). Compared to Fl-AIR, AirSim community is bigger, and the simulator seems more actively developed.

[80]: Shah et al. (2017), 'AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles'

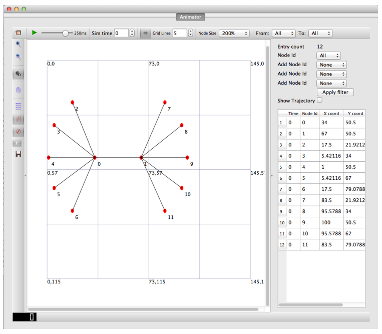
Figure 3.3: AirSim simulation environment screenshot, with a quadcopter drone (middle of the image) flying in an urban environment. Extracted from the project website.

Network Oriented Simulators

ns-3 The ns-2 simulator had been the simulation tool of choice for years and the de facto standard for academic research in networks when the ns-3 project was announced in 2006. Developed in C++ with Python bindings, while ns-2 was developed using a mix of Tcl and C++, ns-3 included parts of ns-2 in its code base (and still does). Overall, ns-3 is a monolithic network simulator organized in modules, and is to be used using its Command Line Interface (CLI), and even if there is a Graphical User Interface (GUI), its uses are limited to simple visualization purposes. Ns-3 has been created in order to “improve the realism of the models” used in network simulation, according to its authors [77], and it quickly took the place of ns-2 as the new standard for network simulation in research. Ns-3 is a discrete event simulator, which means the simulated system is modeled as a series of discrete events which change the state of the system, as opposed to a simulator that would change the system state smoothly and continuously with regard to time.

[77]: Riley et al. (2010), ‘The ns-3 network simulator’

Ns-3 abstracts physical devices such as smartphones, computers or in our case, drones, as network nodes. Such network nodes in turn host network devices representing the networking cards such as Wi-Fi, cellular or Ethernet networking cards. Each networking protocol comes with its own communication channel models, often based on the field literature. Focusing on Wi-Fi networks, the PHY and the MAC layers are modeled in ns-3, up to the 802.11ax amendment, including the infrastructure, the ad-hoc and the mesh modes. Higher layer protocols are also modeled, such as ARP, routing protocols like OLSR or the IPv4 and IPv6 protocols, as well as TCP. As Gazebo, link-budget calculations are made to determine the signal power levels at each node of a simulation, but ns-3 supports multiple standard path loss models, which are correctly labelled, while Gazebo supports only one. The ns-3 physical layer is also beyond comparison with Gazebo.



	Time	Node Id	X coord	Y coord
1	0	0	34	50.5
2	0	1	67	50.5
3	0	2	17.5	21.9212
4	0	3	5.42116	34
5	0	4	1	50.5
6	0	5	5.42116	67
7	0	6	17.5	79.0788
8	0	7	83.5	21.9212
9	0	8	95.5788	34
10	0	9	100	50.5
11	0	10	95.5788	67
12	0	11	83.5	79.0788

Figure 3.4: ns-3 NetAnim component, which can be used as an *offline* GUI for ns-3 simulations, replaying traces collected during simulation. Extracted from the project wiki.

Ns-3 supports different mobility models for its network nodes, which are massless point-like entities. Such models include simple waypoint models, random walk in two and three dimensions, constant position or acceleration models. As point-like objects, nodes have no size and no notion of orientation. Because the mass of the nodes is not modeled, and ns-3 has no notion of physical forces, node movements are limited to the ones where the acceleration is a piecewise constant function. Nodes can change direction instantly, without inertia, and no differential equation solver is needed in the simulation. From a physical point of view, this translates into a loss of realism when it comes to simulating drones or robots, and using ns-3 is therefore not recommended when you need a high physical accuracy (such as when developing a flight controller).

OMNeT++ OMNeT++ is more of a framework to create network simulators than a full featured simulator. Composed of multiple modules, it is written in C++ and has been available since 1997 [87]. As ns-3, it is a discrete event simulator but is less monolithic than ns-3: it is more a collection of libraries compatible with each other than a network simulator.

[87]: Varga et al. (2008), ‘An overview of the OMNeT++ simulation environment’

OMNeT++ comes with a dedicated model library called the *INET Framework* which can be used to simulate network protocols, agents and their mobility using OMNeT++. This library contains models for Wi-Fi networks up to the 802.11-2016 standard and higher level internet protocols including IPv4, IPv6, TCP, UDP, and routing protocols such as AODV. It

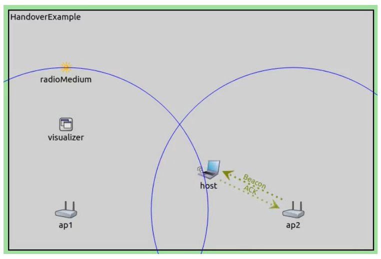


Figure 3.5: OMNeT++ and INET Framework GUI example, extracted from the Handover Example Animation from the OMNeT++ Website.

also contains node mobility models which can be stationary, deterministic, trace-based, stochastic or hybrid models.

The INET framework mobility models are similar to the one of ns-3, but nodes have an orientation in addition to a euclidean three-dimensional position, allowing the simulator to model phenomenons such as antenna orientations. Path loss models widely overlap with the path loss models available in ns-3, with the exception that obstacles are better supported by the INET framework. OMNeT++ also comes with a powerful GUI which can be used to follow the mobility of nodes, display obstacles, configure simulations graphically and debug them.

Hybrid Simulators

[55]: Marconato et al. (2017), 'Avens-a novel flying ad hoc network simulator with automatic code generation for unmanned aircraft system'

AVENS AVENS, which stands for *Aerial VEhicle Network Simulator*, is a hybrid simulator written in C++ based on OMNeT++ for the networking part, and on the proprietary X-Plane Flight Simulator for the flight simulation part [55]. The integration between OMNeT++ and X-Plane is realized through the development of two custom plugins, one for each, which exchange information during the simulation using an XML file. Released in 2017, the simulator is single platform (Windows 8) and has not been updated since.

[97]: Zema et al. (2017), 'CUSCUS: An integrated simulation architecture for distributed networked control systems'

[96]: Zema et al. (2018), 'The CUSCUS simulator for distributed networked control systems: Architecture and use-cases'

CUSCUS CUSCUS, which stands for *CommUnicationS-Control distribUted Simulator*, is a hybrid simulator written in C++ and based on ns-3 for its networking part, and Fl-AIR for its drone simulation and GUI part [97]. One of its peculiarity resides in its use of LXC containers and its use of ns-3 tap bridge network devices. Tap devices, which are virtual network devices in Linux, are used to connect Fl-AIR processes running in containers with simulated network devices in ns-3. Ns-3 is then in charge of simulating the rest of the network stack and the physical channels, connecting drones together. A module allows the simulator to use OpenStreetMap data to create environment and proposing a custom loss model taking into account buildings is presented in [96], but its source code has not been made available. Released in 2018, the simulator has not been updated since.

[6]: Ardupilot Dev Team (2016), *SITL Simulator (Software in the Loop)*

[9]: Baidya et al. (2018), 'FlyNetSim: An Open Source Synchronized UAV Network Simulator based on ns-3 and Ardupilot'

FlyNetSim FlyNetSim is a hybrid simulator written in python and C++ which relies on ns-3 for the networking part, and on SITL [6], a drone simulator based on the Ardupilot flight controller. The two simulators are connected through the use of a custom middleware using ZeroMq, a message-passing framework [9]. As SITL is a real-time simulator, while ns-3 is a discrete event-driven simulator, the clock synchronization between the two simulators is not possible. The real-time scheduler of ns-3 is used to reduce clock disparities between the two simulators, but heavy computational simulations for ns-3 may lead to situations where ns-3 "falls behind". To deal with such situations, the authors propose a pausing mechanism for SITL, but it has not been implemented. By relying on SITL, it is possible to use its emulation mode, which allows the Ardupilot controller running on a real drone to communicate with the simulator. Released in 2018, the simulator only supports ns-3 version 3.27, and has not been updated since except to add the option to run the simulator without using ns-3 or network simulation.

Testbeds

Testbeds can be used to conduct experiments with a twist: the experiments can be repeated, replicated (theoretically), and more easily parameterized that more classic field experiments. Looking at testbeds that can be of interest for drones and Wi-Fi networks, we first identify testbeds focusing on Wi-Fi networks. Drone testbeds exist, but they're not as easy to use as network testbeds. Indeed, because of the permanent human presence they require replacing batteries, move drones and because how easy it is to crash and break drones, they are mostly aimed at on-site experimentation and are not freely accessible.

R2lab R2lab is a wireless testbed located in INRIA Sophia-Antipolis, France [72]. It is part of the Fed4FIRE+ project, a federation of testbeds providing "open, accessible and reliable facilities" for scientists across Europe [26]. R2lab hosts 37 static computing nodes in an anechoic chamber, which can be equipped with USRP or LimeSDR Software Defined Radio (SDR), Bluetooth, BLE, or LTE network interface controllers. All the nodes are equipped with two Wi-Fi cards, each with 3 antennas, which are an Intel 5300 and an Atheros 93xx card. The nodes can be remotely reserved using a web interface, and controlled using ssh and a set of specific scripts which allow booting specific operating system images or reboot the nodes. The Wi-Fi amendments supported by the cards are the a/b/g versions for the Intel one, and the a/b/g/n versions for the Atheros one. In particular, the 802.11ac amendment, on which we focused on during the thesis, is not supported.

[72]: (2017), *R2lab: An open testbed for reproducible networking research*

[26]: (2017), *Fed4Fire+: Federation for Future Internet Research and Experimentation+*

CityLab CityLab is a "City of Things" testbed located in Antwerp, Belgium. It is also part of the Fed4FIRE+ project. Contrary to R2lab, CityLab is a non isolated outdoor testbeds, which hosts 32 static nodes in the Antwerp City in an area of approximately 500m by 500m located on the City Campus of the University of Antwerp [21]. Similarly to R2lab, each node can be reserved and remotely controlled. Nodes are equipped with two Wi-Fi cards, which can be Intel or Atheros cards, supporting either amendments up to 802.11n or 802.11ac, and with IEEE 802.15.4, Dash 7 or LoRa network interface controllers, which are IOT oriented protocols. Because of its proximity with the university campus, no traffic-generating Wi-Fi tests are allowed during the day, at least, when the impact can be significant, such as throughput tests. The CityLab nodes are based on PC Engines APU, which are single-board x86_64 computers aimed at network applications. The same kind of hardware was used throughout the thesis, in conjunction with the WalT software stack.

[21]: (2018), *CityLab: The City of Things Smart Cities FIRE Testbed*

WalT WalT is not a testbed, but a software collection that can be used to create and manage network experiments testbeds. Developed in the Drakkar team in Grenoble, France, WalT allows researchers to deploy customized operating system images on nodes which can be fully controlled remotely. The OS images are currently packaged and shared through Docker and the Docker hub, which allows different WalT based testbeds to execute the same network experiments, and, hopefully, to reproduce and replicate the results. Whereas testbeds such as R2Lab and CityLab are *repeatable* in the sense the same experiment executed twice on the same testbed should produce similar results, their highly specialized hardware management platform and costly deployments, e.g. in an anechoic chamber or in situ, makes it hard to *reproduce* the results elsewhere. WalT aims for cheaper deployments and *reproducibility* across real environments. WalT is organized around a client/server model, with the server hosting Linux images that will be booted using the network on the nodes, which can be x86_64, such as PC Engines APU or classical laptops and desktops computers, or ARM based machines, such as Raspberry Pi.

The Association for Computing Machinery (ACM) defines *repeatability* as one team reliably getting the same results from its experiments ("same team, same experimental setup"), *reproducibility* as a different team being able to obtain the same results using the same experimental setup ("different team, same experimental setup"), and *replicability* as a different team being able to obtain the same results using a different experimental setup ("different team, different experimental setup") [2]

AERPAW Aerial Experimentation and Research Platform for Advanced Wireless (AERPAW) is an ongoing project aimed at creating a testbed for drone networks research. AERPAW is expected to be ready to host first experiments by January 2021, and be completely operational by January 2023 [4]. Multiple networking technologies will be supported, with SDR, LoRa and 5G chipsets being deployed, and the possibility to bring your own devices is also planned. Perhaps the most interesting point about AERPAW is not the testbed in itself, but the eleven experimental scenarios that constitute the core of the project, 6 of which involve drones:

[4]: (2020), *AERPAW: Aerial Experimentation and Research Platform for Advanced Wireless*

- Scenario 3, or *Fixed to Aerial Mobile*: A single drone is connected to a fixed communication infrastructure, located on a tower. The drone is semi-static, and may change its 3D position in a limited manner. This scenario may be used to evaluate the 3D coverage of the fixed antenna, or produce propagation models.
- Scenario 7, or *Fixed to Aerial Mobile with Predefined Trajectory*: Similar to scenario 3 but the drone is mobile and follows waypoints in the flight area. Such waypoints can be set by the experimenter, or chose from a library of pre-configured trajectories. This scenario may be used to study data collection of a ground wireless sensor network using the drone as a sink, or to study mobility management mechanisms on the drone.
- Scenario 8, or *Fixed to Aerial Mobile with Autonomous Trajectory*: As in the scenario 7, but the waypoints are not predefined anymore but computed by an autonomous navigation algorithm, *on-the-fly*. The scenario is mainly a controlled mobility, with the drones "learning" the best trajectory to maximize network metrics such as connectivity or throughput.
- Scenario 9, or *Multiple Fixed Single Mobile*: Drones move in an area covered by multiple fixed nodes located on towers. This scenario is aimed at studying handover mechanisms for drones and drone tracking by the multiple antennas, and in the case of multiple drones, study resource allocation, interferences, or frequency reuse patterns.
- Scenario 10, or *Single-Fixed Multiple-Mobile*: Multiple drones evolve in an area covered by the same fixed node, located on a tower. This scenario can be used to study ad-hoc and mesh networks, as well as studying drones interferences.

- Scenario 11, or *Multiple-Fixed Multiple-Mobile*: Multiple drones evolve in an area covered by multiple fixed nodes located on towers. This scenario, the last of the AERPAW project, merges all the previous scenarios, and aims at studying multi-hop connectivity and long-distance millimeter-wave links.

While promising, this project cannot currently be used, but might be in the coming years. The experimental scenarios may constitute a good basis for common evaluation scenarios in the future.

Conclusion

Field studies of drone networks and Wi-Fi networks reflect the reality in which they are carried out, and expecting the same level of realism in a computer simulation would be foolish. Yet, in a non fully controlled experimental study, the result obtained are specific to the environment where the study was conducted, and hardware (which is especially true for Wi-Fi networks). Even if some parameters can be controlled, for example the distance between two communicating entities, parameters like the weather or the overall layout of the environment can hardly be freely controlled. Simulations, on the other hand, are more flexible and, in theory, can be fully controlled and reproduced. Most of the time, it's easier and faster to set up, launch, and get the results of a simulation than to set up, conduct and get the results of an experiment, and tweaking the parameters to iterate over some proposed algorithm or parameters is easier simulated than actually implemented. Of course, simulations may be only distantly related to the reality they ought to represent, and their accuracy heavily relies on the models and assumptions made in developing them. Testbeds, which are automated experimental platforms, represent a middle ground between simulation and field studies. They allow researchers to conduct parts of experimental campaigns with the ease of simulations, leading in greater reproducibility and flexibility for tests. Yet, the applicability to the real world of the results obtained when the testbeds are running in a controlled environment such as an anechoic chamber, is unclear. And when testbeds are not running in controlled environments, they are facing the same shortcomings as the experimental field studies.

Focusing on drone networks, available simulators do not provide the necessary precision, either for the simulation of the networks (for robot oriented simulators), either for the simulation of robots (for network oriented simulators). Glue projects trying to bridge both classes of simulators exist, but are not really maintained up-to-date. Wi-Fi testbeds are mostly running old hardware, and are most of the time limited to static experiments. In any case, they seem to be not well-suited for drone networks research requiring actually flying drones. But drone networks experiments require particular attention to safety and legal aspects, and are hard to set up, which is why we quickly focused on the simulation of the components of a drone networks, as we will present in the next chapter.

Study of the Intel Rate Adaptation Algorithm 4

One of the requirements of the IEEE 802.11 standard is the ability to handle portable and mobile stations, which are respectively stations that can move but are not communicating when moving, and stations that can move and communicate while moving. For this purpose, the IEEE 802.11 physical layer (*PHY*) defines many transmission features that can be chosen and combined in order to ensure the good receipt of data, notwithstanding changes in the transmission channel caused by mobility or propagation effects. In order to choose which transmission features to use, Rate Adaptation Algorithms (RAAs) are used. In this chapter, we focus on the "Intel" RAA (whose technical name is *iwl-mvm-rs*) because it is the RAA used on all the recent Intel chipsets at the beginning of this thesis. Nowadays, "recent" means Wi-Fi 6 and the general landscape has evolved, especially with the introduction of OFDMA. Focusing on the Intel hardware was not some random decision: the drones we had at our disposal were *Intel Aero Ready To Fly* drones, pictured in Figure 4.1, which are equipped with a computing board which, unsurprisingly, uses an Intel Wi-Fi controller. My own laptop contains an Intel chipset, and a look at the distribution of active Wi-Fi stations in the lab (excluding access points) gives us a percentage of 30% of Intel hardware, just after Apple which sits at 31% (for a total of 135 collected MAC addresses and assuming most of the devices exchanging data use their real MAC addresses). While highly skewed by the fact most of these machines are located in a university and therefore have been purchased through the same public procurement, we still believe understanding the inner working of these chipsets is of general interest.

Figure 4.1: Intel Aero Ready To Fly quadcopter

In this chapter, we first describe rate adaptation mechanisms available in Wi-Fi networks, and in particular which rate adaptation algorithms are implemented in the ns-3 and OMNeT++ / INET network simulators. We then focus on how rate adaptation algorithms are implemented in practice, especially in the context of the Linux operating system. General documentation about the organization of the Wi-Fi stack for Linux is scarce and dated, which is why we present an overview of this organization in this chapter. We then focus on the Intel Rate Adaptation algorithm, by explaining how it was reverse engineered and simulated into the network simulator ns–3. This process was first approached through experimentation, which then evolved to the observation of the behavior of the Linux kernel module in charge of this rate adaptation, through modification and instrumentation of its source code.

4.1 Rate Adaptation Algorithms in Wi-Fi networks

Since its launch, Wi-Fi supports different "transmission rates". Those transmission rates are the result of the combination of multiple physical

Infrared communication is covered by the 802.11-1997 and 802.11-1999 standards, but is described as obsolete since the 802.11-2007 standard, and therefore will not be covered. We generally focus on the *classical* PHY, compatible with the 2.4GHz and 5GHz bands, and we will not cover *exotic* PHY, for example the Directional Multi-Gigabit (DMG), also known as Very High Throughput (VHT) in the 60 GHz Band, or the Television Very High Throughput (TVHT) introduced in the 802.11-2016 standard.

layer transmission parameters, and they correspond to the rate at which some parts of the transmitted frames, containing upper layer data, are sent over the spectrum. Still, as this data is encapsulated in various headers and undergoes various operations such as padding or forward error correction operations, the transmission rate should only be seen as an unreachable upper bound on the actual rate at which *actual* data is transmitted over the PHY. The increase in the capacity (that is to say the maximum transmission rate) of Wi-Fi networks, illustrated on Figure 2.1, is mainly explained by the complexification of the physical layer of Wi-Fi, which is illustrated by the increasing number of pages in the versions of the standard.

While the first two versions of 802.11 supported three different PHYs, the number of supported PHY increased for each subsequent version, with the 802.11-2016 version supporting eight different PHY, as shown in Table 4.1. Of course, each different PHY supports different transmission parameters. For example, the 802.11-1997 and 802.11-1999 standards' FHSS and DSSS PHY each supports two different radio transmission rates, 1 Mb/s and 2 Mb/s, which can be obtained by using a two level or four level Gaussian Frequency-Shift Keying (GFSK) modulation. In total, four different combinations for two transmission rates are available for the radio transmissions of 802.11-1997 or 802.11-1999.

Limiting ourselves at PHYs which only concern the 2.4 GHz and 5 GHz bands, the OFDM, HR/DSSS and ERP PHY, introduced in 802.11-2007 (but modified since) respectively provide 24, 7 and 33[1] different combinations of parameters, resulting in 20 different transmission rates. For the HR/DSSS PHY, which operates in the 2.4 GHz band, the differences between the different transmission rates lie in the use of the DBPSK or the DQPSK modulation, with some optional short header. For the OFDM PHY, which operates in the 5GHz band, the differences between the different transmission rates lie across multiple parameters: first, the bandwidth, or channel spacing, which can be 5 MHz, 10 MHz or 20 MHz, second, the modulation, which can be the BPSK, the QPSK, the 16-QAM or the 64-QAM modulations, and third, the coding rate, which takes values in the $\{1/2, 2/3, 3/4\}$ set. The ERP PHY, finally, draws from the OFDM PHY and applies the same modulation technique, OFDM, but this time in the 2.4 GHz band. It also relies on the DSSS and HR/DSSS PHY.

1: This number of combinations can be obtained by adding the number of combination for the OFDM, the HR/DSSS and DSSS PHY, as it builds upon them: its "radio portion [[...]] implements all mandatory modes " of the aforementioned PHY, "except it uses the 2.4 GHz frequency band" and a specific "channelization plan".

The HT PHY, introduced in the 802.11-2012 standard, supports at least 306 combinations for 86 different transmission rates. It is based on the OFDM PHY, but supports up to four spatial streams (with at least 4 antennas), and operates in bandwidth of 20 and 40 MHz. In order to differentiate some parameter combinations, the PHY uses an integer named the VHT Modulation and Coding Scheme (MCS) index which encodes the used modulation, the coding rate, and the number of spatial streams. The VHT MCS index range from 0 to 76, with the values ranging from 0 to 32 encoding transmissions where the same modulation is used on all the spatial streams (32 being only available for single stream and 40 MHz operation), and values ranging from 33 to 76 encoding transmissions where different modulations are used for different spatial streams. In addition to the VHT MCS index, another parameter changes the transmission rate: the guard interval duration, which can be short (400 ns) or long (800 ns).

Table 4.1: Name of the PHY supported by each 802.11 standard.

Wi-Fi Standard	Modulation Classes (PHY)
1997	FHSS, DSSS, Infrared*
1999	FHSS, DSSS, Infrared*
2007	FHSS, DSSS, OFDM, HR/DSSS, ERP, Infrared* (deprecated)
2012	DSSS, HR/DSSS, OFDM, ERP, HT, Infrared* (deprecated), FHSS (deprecated)
2016	DSSS, HR/DSSS, OFDM, ERP, HT, DMG*, VHT, TVHT*

*: PHY not concerning the 2.4 or the 5 GHz bands.

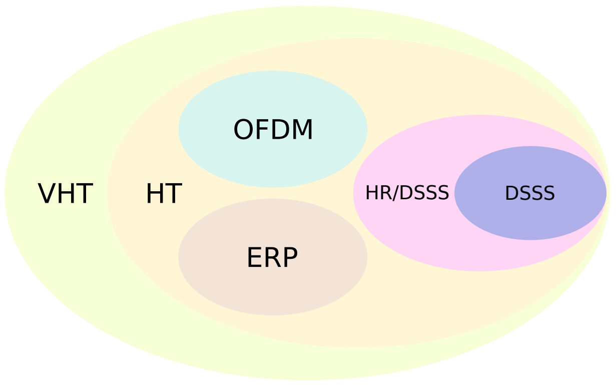


Figure 4.2: Overview of the different PHYs up to the 802.11-2016 standard and their relations with each other.

Finally, the VHT PHY, introduced in the 802.11-2016 standard, supports at least 620 combinations for 218 different transmission rates. As with HT PHY and its relation towards the OFDM PHY, the VHT PHY can itself be seen as an HT PHY on steroids. It may support 80 MHz, 160 MHz or 80 + 80 MHz bandwidth, a new modulation is supported (256-QAM) and the maximum number of spatial streams can go up to eight. A new MCS index definition is introduced: the VHT MCS, which this time only covers the modulation and coding rate, and not the number of spatial streams, as shown in Table 4.2. In the following, we will adopt the VHT MCS terminology (as opposed to the HT MCS one).

Let us note that newest PHYs do not mean the removal of the support for the older PHYs, as it is mandatory for newer PHYs to implement the older PHYs as illustrated on Figure 4.2.

While not all the combinations are mandatory for hardware vendors to implement, because for example it does not make much sense to require a smartphone to have 8 antennas to be able to support 8 spatial streams, at least a few hundred transmission parameters are supported by a classical laptop or smartphone Wi-Fi chipset, equipped with 2 or 3 antennas. In what follows, for the sake of simplicity, we will refer to choosing a *set of transmission parameters* as choosing a *transmission rate*, even if it is a little erroneous as multiple transmission parameters can result in the same transmission rate.

Choosing which transmission rate to use is important. Indeed, as a general rule, the higher the transmission rate, the quicker it will be to transmit data over the spectrum, so, the higher the application throughput. But the lower the MCS that is to say the lower the transmission rate, all other things being equal, the bigger the transmission range. To obtain the same Bit Error Rate (BER), and the same frame success rate,

Table 4.2: Correspondence between the VHT MCS index and the modulation and coding rate.

VHT MCS	Modulation	Coding Rate
0	BPSK	1/2
1	QPSK	1/2
2	QPSK	3/4
3	16-QAM	1/2
4	16-QAM	3/4
5	64-QAM	2/3
6	64-QAM	3/4
7	64-QAM	5/6
8	256-QAM	3/4
9	256-QAM	5/6

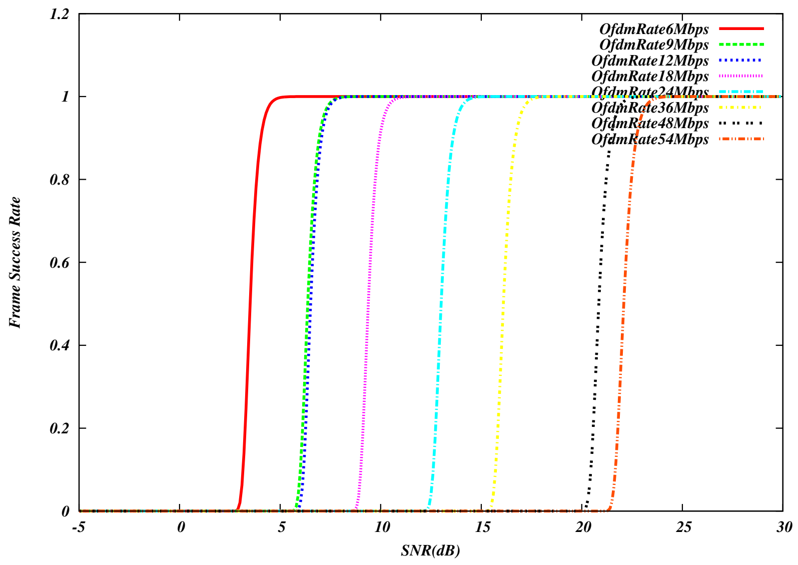


Figure 4.3: Frame Success Rate with regard to the SNR and the transmission rate using the NIST error model. Extracted from the ns–3 documentation.

higher MCS needs higher Signal-To-Noise (SNR) levels, as illustrated on Figure 4.3, which has been obtained using the ns–3 simulator and its "NistErrorRateModel" which models Additive White Gaussian Noise (AWGN) channels according to various theoretical and experimental works [70]. Because the IEEE 802.11 standard requires to support mobile and moving stations, and because the channel characteristics are dynamic, the transmission rate needs to be dynamic as well to be able to maintain connections in spite of mobility and channel evolutions. Of course, if one wishes to privilege the transmission range, one can fix the transmission rate to the lowest transmission rate supported by the considered PHY, but this means losing the ability to use Wi-Fi for higher application data rates, and not being able to scale to more than a few stations. Indeed, lower transmission rates mean longer transmission time, more contention on the wireless medium, more collisions, and worse performances [91]. Because Wi-Fi relies on CSMA/CA, which provides an "equal long term channel access probability" to all the STAs, choosing a lower than necessary transmission rate will also reduce the throughput of all the other STAs in a multiple STA network, as highlighted in [45]. And choosing a transmission rate that is too high will only result in the inability to efficiently communicate over the medium, as the transmitted frames will not be able to be correctly received by the STA we are communicating with.

[70]: Pei et al. (2011), *Validation of OFDM model in ns-3*

[91]: Wong et al. (2006), 'Robust rate adaptation for 802.11 wireless networks'

[45]: Heusse et al. (2003), 'Performance anomaly of 802.11b'

Transmission rate selection, called rate switching in the standard, is done by what is called a Rate Adaptation Algorithm (RAA), sometimes called rate selection or link adaptation algorithm. The IEEE 802.11 standard defines which combinations of transmission features are allowed and forbidden, but it does not enforce any behavior regarding how these features should be chosen, letting each Station (STA) (and ultimately, each hardware vendor) in charge of deciding its own transmission rates. The first RAA for Wi-Fi was probably the Automatic Rate Fallback (ARF) algorithm, designed for the 802.11-1997 compatible controller "WaveLAN–II" [49]. ARF must choose between two different rates, 1Mb/s and 2Mb/s, which both are using the DSSS technology. By default, the RAA transmits

"The algorithm for performing rate switching is beyond the scope of this standard, but in order to provide coexistence and interoperability on multirate-capable PHYs, this standard defines a set of rules to be followed by all STAs." [46]

[49]: Kamerman et al. (1997), 'WaveLAN II: a high-performance wireless LAN for the unlicensed band'

at 2Mb/s, and in case of the reception failure of the Acknowledgement (ACK), it first retransmits the frame using a 2Mb/s transmission rate. In case this transmission fails once again, ARF retransmits the frame at 1 Mb/s and uses a 1 Mb/s transmission rate for the subsequent frames. ARF maintains a timer and a counter to allow itself to start using the 2Mb/s transmission rate again, which is triggered after 10 consequent successful transmissions at 1 Mb/s, or a time-based counter whose duration is not specified.

With every new version of the standard, and new PHYs, new rate adaptation algorithms need to be developed to take advantage of the new features. Thus, nearly two decades of changes in the 802.11 standards have led to dozens of RAA proposals, sometimes generic, sometimes tackling specific aspects of 802.11 networks such as energy consumption or dealing with multiple antenna systems. In the Table 4.3, we list and compare some RAAs and their characteristics. In particular, all the RAAs implemented in the network simulators ns–3 and OMNeT++/INET are present. However, the majority of the algorithms implemented in these simulators are obsolete in the sense that they do not support the PHY from recent versions of the standards, such as the HT and VHT PHYs. While some listed algorithms have been implemented and are actually being used in real hardware, such as ARF [49], Minstrel [23] or Minstrel-Ht [29], most of these algorithms are either not used in commercial hardware, or used without us knowing so. Indeed, hardware vendors generally do not communicate on the RAA they use and leave little opportunity to change the behavior of their devices, which are not open hardware, and heavily rely on closed-source firmware[2] for their operation.

[49]: Kamerman et al. (1997), ‘WaveLAN II: a high-performance wireless LAN for the unlicensed band’

[23]: Derek (2005), *Minstrel*

[29]: Fietkau (2010), *Minstrel HT: New rate control module for 802.11n*

2: Firmware are pieces of software embedded in the controllers, running on their internal processing units. They can often be updated by the OS, but may need to be cryptographically signed by the hardware vendor, and reverse engineered as they are mostly distributed as binaries.

Reasons behind vendors refusing to provide open source firmwares and locking down their hardware, preventing modification, are various. One explanation is that the Federal Communications Commission (FCC) requires hardware vendors to secure their devices “against third party software modifications that would take [their devices] out of [their] RF compliance” [50], which can only be realistically met by locking firmware updates. Thus, trying to figure out what is the rate adaptation used in some commercial controller is hard, and coming with a new RAA algorithm and actually implementing it in real hardware is harder when you are not partnering with the vendor.

[50]: Knapp (2015), *Clearing the Air on Wi-Fi Software Updates*

Before going into details over the Intel Rate Adaptation Algorithm and its “reverse engineering”, we will try to explain why, in some cases, it’s still *reasonably* easy to determine what is the rate adaptation used in some commercial controller.

Table 4.3: Comparison of various Rate Adaptation Algorithm (RAA).

Paper Year	Algorithm Name	Abbreviation	Hardware Support	NS-3	OMNeT++ INET	802.11n	802.11ac
1997	Automatic Rate Fallback	ARF	?	✓	✓	✗	✗
2001	Receiver Based Auto Rate	RBAR	?	✗	✗	?	?
2003	Received Signal Strength Link Adaptation	RSSLA	?	✗	✗	?	?
2004	Adaptive Automatic Rate Fallback	AARF	?	✓	✓	✗	✗
2004	Adaptive Multi Rate Retry	AMRR	?	✓	✗	✗	✗
2005	Opportunistic Auto Rate	OAR	?	✗	✗	?	?
2005	Full Auto Rate	FAR	?	✗	✗	?	?
2005	Onoe	Onoe	?	✓	✓	✗	✗
2005	SampleRate	SampleRate	?	✗	✗	?	?
2005	Power-controlled Auto Rate Fallback	PARF	?	✓	✗	✗	✗
2005	Dynamic data rate and transmit power adjustment	APARF	?	✓	✗	✗	✗
2006	Robust Rate Adaptation Algorithms	RRAA	?	✓	✗	✗	✗
2006	Collision Aware Rate Adaptation	CARA	?	✓	✗	✗	✗
2007	Beacon Auto Rate Adaptation	BARA	?	✗	✗	?	?
2007	**Minstrel**	Minstrel	✓	✓	✗	✗	✗
2008	Collision Detection for Auto Rate Fallback Algorithm	AARF-CD	?	✓	✗	✗	✗
2009	**Minstrel-HT**	Minstrel-HT	✓	✓	✗	✓	✓
2011	Rate Adaptation for Multi Antenna Systems	RAMAS	?	✗	✗	?	?
2011	Rate Adaptation using Coherence Time	REACT	?	✗	✗	?	?
2013	Agile Rate Adaptation for MIMO Systems	ARAMIS	?	✗	✗	?	?
?	IdealWifi	IdealWifi	✗	✓	✗	✓	✓
?	ConstantRate	ConstantRate	✓	✓	✓	✓	✓
2019	**Intel Rate Adaptation Algorithm**	**IntelRate**	✓	✓	✗	✓	✓

4.2 Methods for in situ study of Rate Adaptation Algorithms

Given some Wi-Fi controllers for which we do not know the RAA, the first thing one can try to understand the algorithm is observing its behavior. To this end, we present multiple possible options:

- We can listen to the controller transmitted frames over-the-air, as a third-party monitor:
  - Using a Wi-Fi controller, with its interface set up in the *monitor* mode, which allows it to listen to all the frames one can receive, not only the frame for which we are the designated receiver. Such interface mode is not supported by all the controllers, and putting the card into a monitor mode means you are unable to use it to transmit. On Linux, this can be achieved using the "`iw dev <devname> set type monitor`" command.
  - Using a Software Defined Radio (SDR) in combination with a Wi-Fi software receiver, such as the Gnuradio based "gr-ieee802-11" [14] or the "openwifi" projects, which is based on Field Programmable Gate Arrays (FPGA) [48].

[14]: Bloessl et al. (2018), 'Performance Assessment of IEEE 802.11p with an Open Source SDR-Based Prototype'

[48]: Jiao et al. (2020), 'openwifi: a free and open-source IEEE802.11 SDR implementation on SoC'

- If we are the designated receiver of the frames, one we can listen to the received frames and their characteristics, which will illustrate how the RAA of the *sender* works. Indeed, transmis-

sion parameters are often available to tools such as `tcpdump` or `wireshark`, as long as one adds a *virtual* monitor interface to the controller, e.g. using the "`iw phy <phyname> interface add mon0 type monitor`" command on Linux. Again, this might not be supported by all the Wi-Fi controllers.
- If we are the sender, we might try to access the decisions of the RAA algorithm as they are made, instrumenting functions of the driver[3] or using debug interfaces of the controller.

3: Drivers are software components which interface the hardware to the rest of the OS, in particular to the Linux kernel. They abstract hardware differences by offering a generic API to the rest of the Linux kernel.

Each method has weak and strong points. Listening frames over the air as a "monitor" means one is subject to the communication channel and its imperfections: one might not be able to receive or decode all the frames, especially since the RAA is not trying to adapt its transmission parameters for *us* but for the receiver. This could mean an incomplete or wrong picture when looking at the RAA. Using a Wi-Fi controller to do so is cheaper than using a SDR, but this means one are bound to the supported features of the controller, whereas we could theoretically implement the needed feature in software in a SDR. But using an SDR is more expensive, and requires way more work on the protocol, as no fully featured receiver exists for VHT or even HT PHYs. Listening to transmitted frames as the designated receiver means we only get to see the good decisions from the RAA, or the conservative decisions: one will only see the transmission parameters that one can decode, and not the *too complex to be received* transmission parameters. Accessing the decisions of the RAA as they are decided by the controller is probably the jack-of-all-trade, but this means we trust the controller's radio to actually use what the controller decides, and accessing the interfaces of the controller might not be an option for all the devices, as it assumes having some control over the operating system. In any way, just listening to the transmission parameters of a controller does not explain why it made the decision to use these transmission parameters. This may, however, help to identify patterns that are already known in the literature.

Although incomplete, we still present some experimental tests that led to the study of the Intel Rate Adaptation algorithm. We started with a simple question: how are throughput and distance related, for a single STA connected to an AP ? To do so, we used the setup shown in Figure 4.4, with a laptop (Dell Precision 5520) using an Intel Wireless-AC 8260 WNIC running ArchLinux (Linux kernel version 4.20.7) acting as the STA and a TP-Link TL-WR802Nv4 using a Mediatek MT7603 WNIC running OpenWrt (Linux kernel version 4.9.73). The choice of the TL-WR802Nv4 as an access point was made because it was available (as in "one can buy it online") and its small form factor made it possible to embed it in a drone. Yet, its default OS was not open enough one could install network tools such as iPerf3 on it, which is why we had to first port the OpenWrt operating system on the device, which involved a bit of soldering and was made possible thanks to the precious help of the OpenWrt community through its online forum. Another drawback of this hardware is that it only supports amendments up to the 802.11n one, which is why we then stopped using it. A 20 MHz channel width was used, in the 2.4 GHz channel band (channel 11), with an HT PHY, in a setup which allowed up to two spatial streams to be used.

Using the iPerf3 traffic generator, we saturated the link from the laptop to the AP using UDP, and observed the relationship between distance,

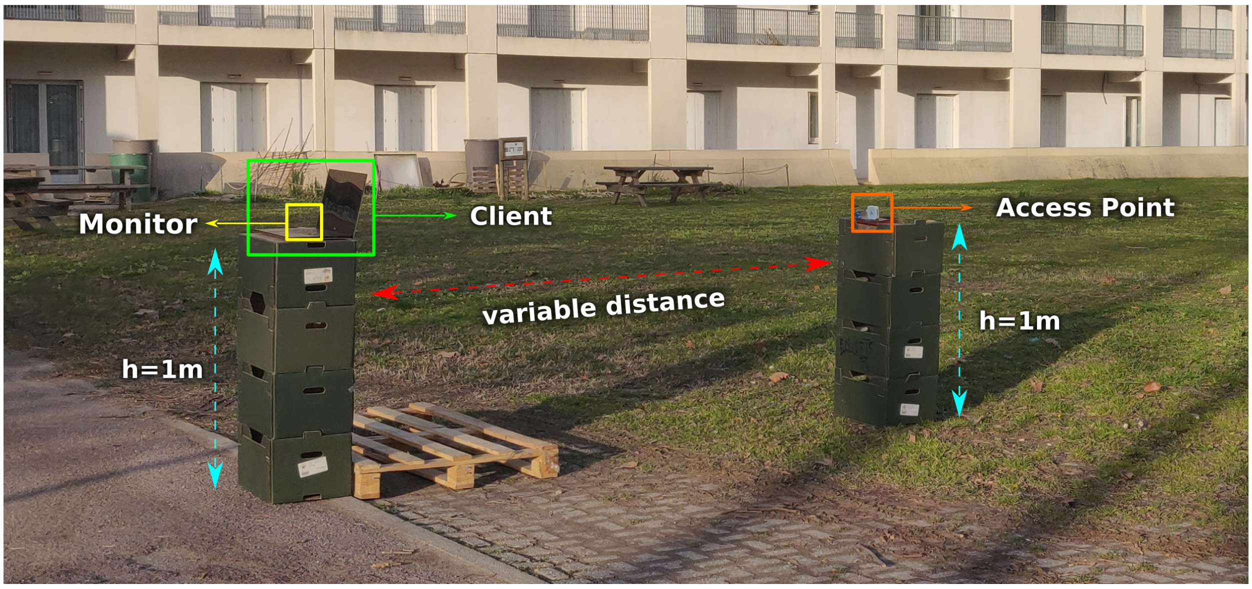


Figure 4.4: Setup of the test

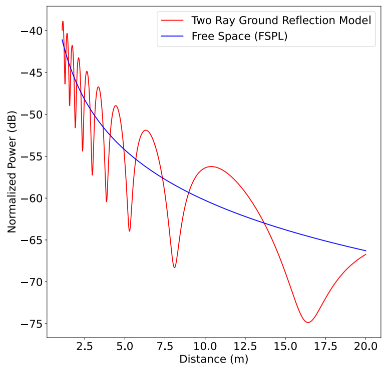


Figure 4.5: Path loss for the two ray interference and free space models. The reflection coefficient is an arbitrary $R = -0.70$, the frequency is $f = 2462$MHz and the elevation of the two antennas are $h_t = h_r =$ 1m, mimicking the experiment in Figure 4.4.

average transmission rate, average RSS and application throughput. The distance was changed from 1m to 20m, in steps of 50 cm below 10 m, and 1 m above. For each distance, 5 measurements of 30 seconds were conducted, each measurement being separated by 5 seconds. To record the transmission rate, we used a CSL AC1200 USB WNIC, based on the Realtek RTL8812AU chipset, plugged into the laptop: because the relative position of the monitoring node and the laptop do not change, the considered RSS is therefore the one on the link from the AP to the STA, while data is going from the STA to the AP. Given the principle of reversibility of light, we still believe it is a good indicator of the path loss; it also has this good property that acknowledgments are sent with overall constant transmission rates, which makes it possible not to have to take into account modulation dependent transmission power and sensitivity thresholds of the hardware.

The results in terms of RSS and throughput with regard to distance are shown in Figure 4.6. We can identify a general trend, which is the greater the distance, the less the RSS and the obtained throughput, which is expected. We can observe some wells and peaks in the RSS which can be explained by the fact the direct ground reflection between the emitter and the transmitter will interfere constructively and destructively with the direct, line of sight ray. In Figure 4.5, we illustrate the resulting power due to a two ray ground reflection model with reflection coefficient chosen arbitrarily of $R = 0.7$ and other parameters mimicking the experiment. We can observe some similarities with the model and the experiment, but the comparison was not pursued further.

Looking at the throughput, the same general trend can be observed, but the relation between the throughput and the RSS, plotted on the left of Figure 4.7, is not as clear as we could expect. For example, for average RSS values around −74dBm, the average throughput can vary between around 20 Mb/s and 50 Mb/s. As the throughput is related to the transmission rate, the relation between the average throughput and average transmission rate is plotted on the right of Figure 4.7, as

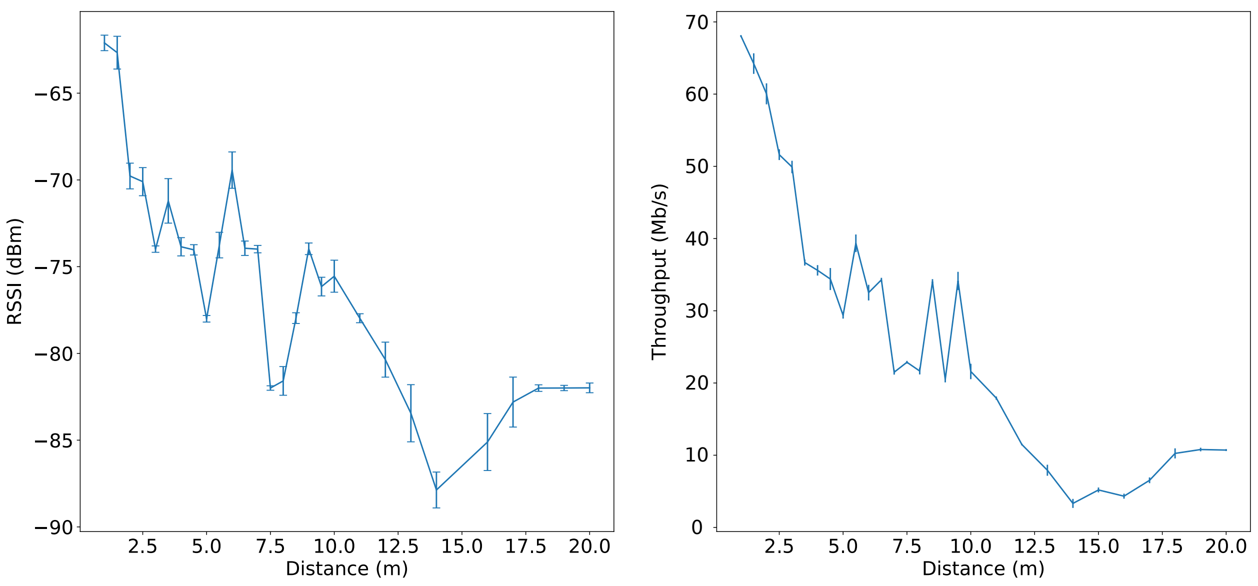


Figure 4.6: Evolution of the average RSS and the iPerf3 UDP reported throughput according to distance for the experiment described in Figure 4.4.

well as the $y = x$ line. Each point represents a single measurement of 30s and its average throughput and average transmission rate. Most of the experiment are above the $y = x$ line, which means they have a higher transmission rate than throughput, which is normal. For high throughputs, above 55Mb/s, the points are closer to this line, and for one experiment, the average transmission rate is lower than the throughput. This is due to the fact the monitoring device is missing some frames with a high transmission rate (which are harder to decode), which creates some bias in the observed transmission rate, whereas the throughput is always reported correctly by iPerf3. Nevertheless, the average transmission rate looks like a good predictor of the performances of this single hop communication link, or, at least, seems more suitable than the reported RSS.

As both the laptop and the AP supported up to two spatial streams, we expected better performances in this simple case: even at short distances, the average transmission rate is well below the theoretical maximum for such hardware, which is 144.4Mb/s for a 20 MHz wide channel. Looking at the distribution of the used MCS, which play a big role in the transmission rate value, we observe not a single distance use HT MCS bigger than 10, and most of the frames are sent using a HT MCS smaller than 7, which means the use of a single spatial stream. This explains the *poor* performances we observe, as using a single spatial stream effectively caps the maximum transmission rate to 72.2 Mb/s. We illustrate those distributions in Figure 4.8 for three distances. We can observe that only a few MCS values are used to transmit most of the frames during the tests, compared to the set of 16 different possible HT MCS values. This leads us to wonder all the more about how these values, and ultimately the transmission rates, are chosen.

After trying to pursue the "over-the-air" options for the Intel hardware we had at hand, both using Wi-Fi cards and SDR, we ultimately started to the study of the RAA by looking at the decision of the controller as they are made. To explain how this was done, a bit of context on the

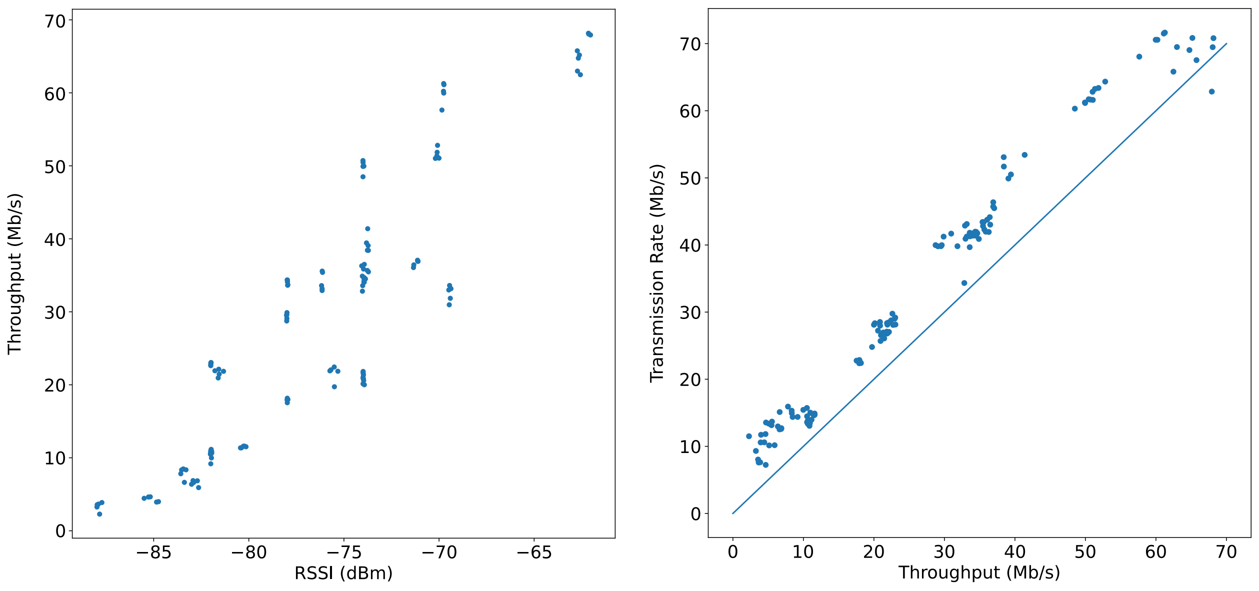


Figure 4.7: Evolution of the average RSS and the iPerf3 UDP reported throughput according to distance for the experiment described in Figure 4.4.

architecture of Wi-Fi controllers and their interactions with the Linux operating system is needed.

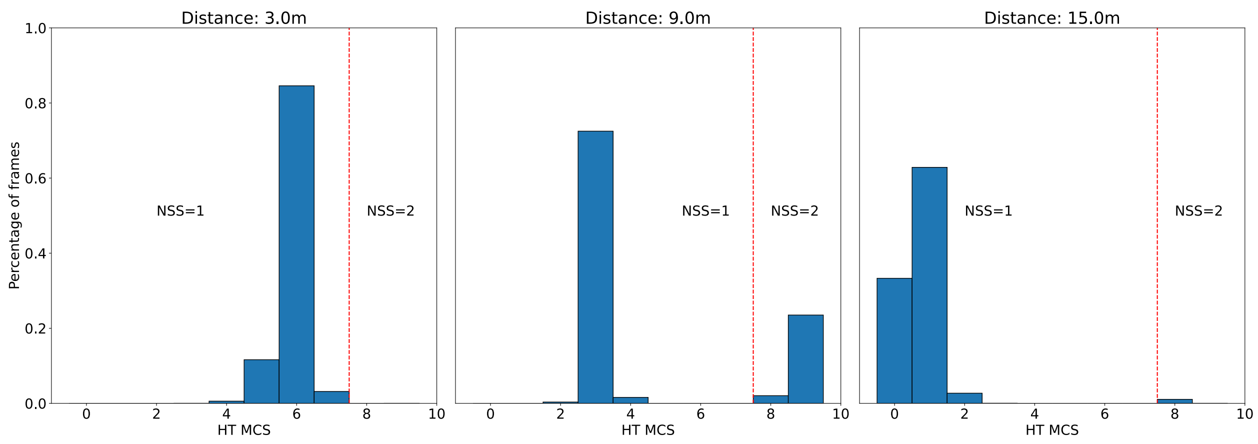


Figure 4.8: Distribution of the used HT MCS for three different distances (3.0, 9.0 and 15.0m) for the experiment described in Figure 4.4.

4.3 Architecture of Wi-Fi Networks Controllers

Wi-Fi controllers can be considered as small computers of their own. In [5], the authors reverse engineer Broadcom wireless controllers from the bcm43 family, and underline their internal structures: they contain ARM Cortex M3 or M4 chipsets, which are RISC processors, as well as Read Only Memory (ROM) and Random Access Memory (RAM) chipsets. Some specialized processors for signal processing and time critical operations, such as the D11 core, are also included. Given that some operations are time critical in Wi-Fi networks, it's expected to have specialized cores in charge of most of the PHY covering much of the PHY Sublayer Management Entity (PLME). For example, in the Distributed Coordination Function (DCF) procedure, the acknowledgment must be sent after a Short Inter Frame Space (SIFS) whose duration is 16 μs in the VHT PHY, which is not compatible with a non realtime operating system like Linux [14, 68]. But the 802.11 standards also cover less time-critical functions, like the functions residing in the MAC Sublayer Management Entity (MLME), which include the association, de-association, re-association and authentication of the STAs, and beaconing, or the functions residing in the Station Management Entity (SME), entity in charge of controlling and interacting with both the MLME and the PLME. For financial reasons (from the hardware vendor point of view), it makes sense to cover these functions in the operating system as this means smaller needed RAM and ROM in the Wi-Fi controller, and, ultimately, smaller costs.

[5]: Anguelkov (2019), *Reverse-engineering Broadcom wireless chipsets*

[68]: Nychis et al. (2009), 'Enabling MAC Protocol Implementations on Software-Defined Radios.'

[14]: Bloessl et al. (2018), 'Performance Assessment of IEEE 802.11p with an Open Source SDR-Based Prototype'

Thus, it is possible to differentiate two families for the architecture of Wi-Fi controllers: *SoftMAC* and *FullMAC* (also known as *HardMAC*) controllers. FullMAC controllers implement the MLME and SME in the controllers, providing high level APIs to the OS, while SoftMAC controllers let the OS implement these functions, providing lower level APIs. This also allows a single implementation, the OS one, to drive multiple controllers from various hardware vendors, which often brings more flexibility: it is for example easier to update Linux source code than to update the firmware of each the controllers supported by Linux. SoftMAC devices, on the downside, are probably less energy efficient and some of their functions have a higher latency, as they imply a communication between the controller and the operating system.

To complicate the situation, hybrid SoftMAC and FullMAC controllers

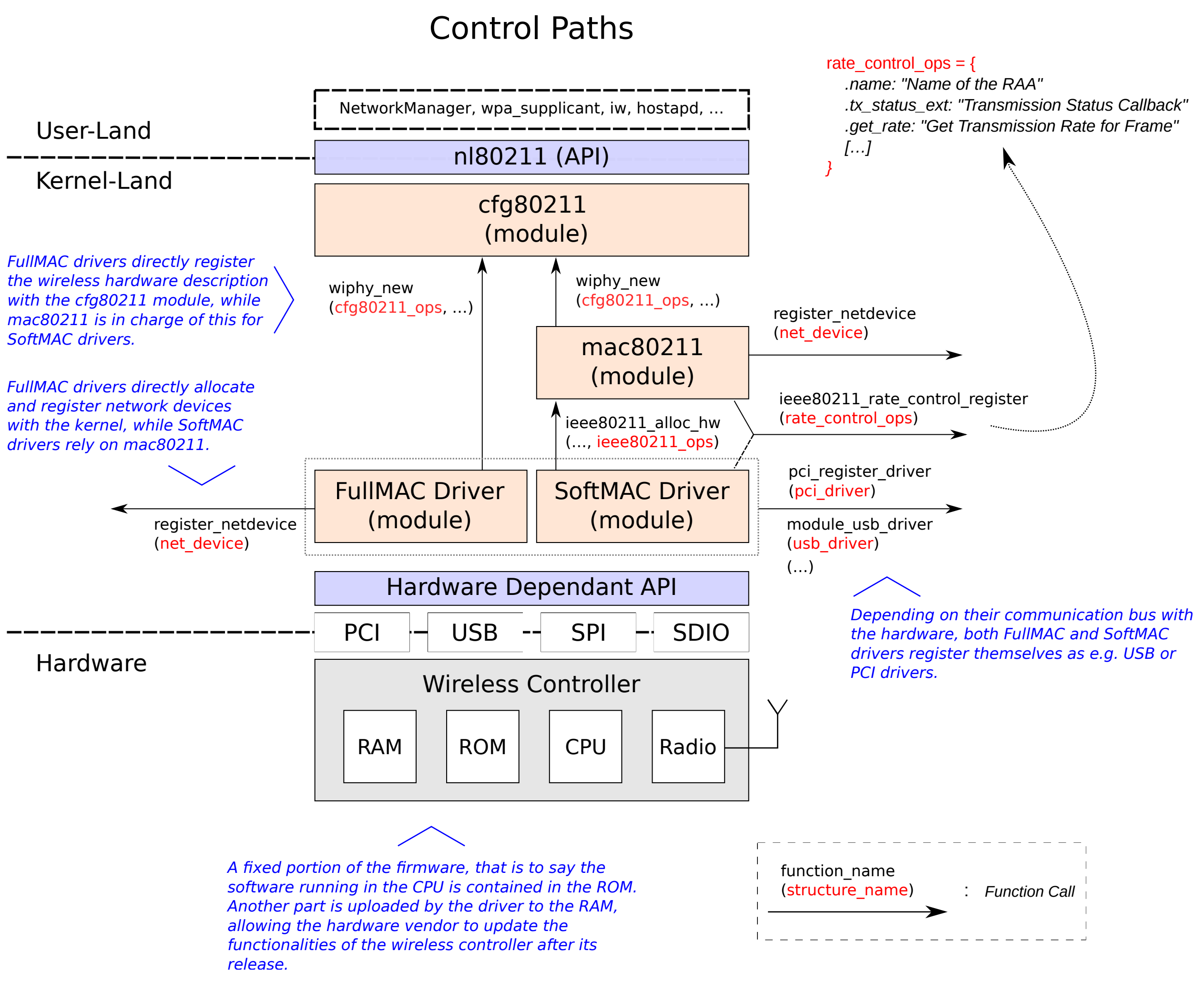

Control Paths
User-Land
Kernel-Land
Hardware
NetworkManager, wpa_supplicant, iw, hostapd, ...
nl80211 (API)
cfg80211 (module)
mac80211 (module)
FullMAC Driver (module)
SoftMAC Driver (module)
Hardware Dependant API
PCI
USB
SPI
SDIO
Wireless Controller
RAM
ROM
CPU
Radio
rate_control_ops = {
.name: "Name of the RAA"
.tx_status_ext: "Transmission Status Callback"
.get_rate: "Get Transmission Rate for Frame"
[...]
}
wiphy_new (cfg80211_ops, ...)
wiphy_new (cfg80211_ops, ...)
register_netdevice (net_device)
ieee80211_rate_control_register (rate_control_ops)
ieee80211_alloc_hw (..., ieee80211_ops)
pci_register_driver (pci_driver)
module_usb_driver (usb_driver)
(...)
register_netdevice (net_device)
FullMAC drivers directly register the wireless hardware description with the cfg80211 module, while mac80211 is in charge of this for SoftMAC drivers.
FullMAC drivers directly allocate and register network devices with the kernel, while SoftMAC drivers rely on mac80211.
Depending on their communication bus with the hardware, both FullMAC and SoftMAC drivers register themselves as e.g. USB or PCI drivers.
A fixed portion of the firmware, that is to say the software running in the CPU is contained in the ROM. Another part is uploaded by the driver to the RAM, allowing the hardware vendor to update the functionalities of the wireless controller after its release.
function_name (structure_name) : Function Call

Figure 4.9: Overview of the organisation of the Wi-Fi Linux stack for the control path.

exist, relying for some functions on the OS and for some other functions on their own implementations, and some controllers can work both as SoftMAC controllers or FullMAC controllers, depending on the used driver or firmware. The general organization of the Wi-Fi Linux stack is shown on Figure 4.9, with the nl80211 public API at the top, available to userspace tools, kernel modules running in kernelland below, and the specific controller API at the bottom.

The MAC implementation of the Linux kernel for SoftMAC devices is called “mac80211”, and provides two RAA implementation which can be used by SoftMAC controllers: Minstrel and Minstrel-Ht. There source codes are available in the `./net/mac80211/rc80211_{minstrel,minstrel_ht}.c` files. To determine if a Wi-Fi controller compatible with Linux uses Minstrel or Minstrel-Ht, we must first determine if the controller is SoftMAC, and if it is, determine if it actually uses the provided RAA (which it might decide to not to). To do so, it is possible to read the controller driver source codes, available, unlike firmware source codes, in the `./drivers/net/wireless/` path in the Linux source tree. Such drivers take the form of Linux modules which can be compiled into the Linux kernel, or dynamically loaded when the corresponding hardware is detected.

4.4 Reverse Engineering of the Intel Rate Adaptation Algorithm

Looking at the Intel hardware drivers, available in the Linux source tree in the path `./drivers/net/wireless/intel/`, we can observe that the corresponding drivers, namely *ipw2x00*, *iwlegacy* and *iwlwifi*, are all SoftMAC drivers. Focusing on IwlWiFi, which powers recent Intel Wi-Fi controllers, one can observe that it does not rely on the mac80211 RAAs, but instead comes with its own algorithms: Iwl-Agn-Rs and Iwl-Mvm-Rs, the former being un-maintained, and limited to HT PHY, and the latter being maintained and used with the VHT PHY "Mvm" hardware. Fortunately, the RAAs are implemented in the driver, which means one can read their source codes and try to understand them.

In the task of understanding the inner workings of the Iwl-Mvm-Rs, two files and their headers are of interest[4]:

4: Most of the Linux kernel is written in C and assembly, but partial support for a new programming languages, *Rust*, might be added in late 2020.

- The `./mvm/rs.c` and `./mvm/rs.h` files, which contain the actual implementation of the RAA (or "rate scaling", hence the filenames). According to the *SLOCCount* tools, these files contain around 3500 source lines of code. They also implement debug functions, which can be used through the Linux *debugfs*, a special RAM based file system usually mounted at the `/sys/kernel/debug/` mountpoint which allows users to interact with low level functions of the Linux kernel.
- The `./iwl-debug.h` header, which lists the debug options one can use when loading the `iwlwifi` module, which can for example cover the tracing of some parameters in the system log.

Accessing the RAA decisions

In order to access the RAA decisions as they are made, one can load the `iwlwifi` module with the debug option `IWL_DL_RATE`, corresponding to a debug mask of 0x00100000 according to the aforementioned file, which enables the debug output in the system log, which can be accessed using the `dmesg` command:

```
# modprobe iwlwifi debug=0x00100000
# dmesg | tail
[...] I __iwl_mvm_rs_tx_status Tx idle for too long. reinit rs
[...] I rs_rate_scale_clear_tbl_windows Clearing up window stats
[...] I rs_drv_rate_init LQ: *** rate scale station global init
    for station 0 ***
[...] I rs_drv_rate_init LEGACY=FFF SISO=1FD0 MIMO2=1FD0 VHT=0
    LDPC=0 STBC=0 BFER=0
[...] I rs_drv_rate_init MAX RATE: LEGACY=11 SISO=12 MIMO2=12
[...] I rs_get_initial_rate Best ANT: A Best RSSI: -47
```

Reading the `./mvm/rs.c` file should theoretically be enough to understand how the RAA works, but no high level description or documentation on the RAA existed. Thanks to a combination of tracing, in part by modifying the source code of the driver, and over-the-air monitoring, and *reading* the source code, we managed to understand the Iwl-Mvm-Rs RAA, which is now presented.

Transmission parameters managed by Iwl-Mvm-Rs

The Iwl-Mvm-Rs Rate Adaptation Algorithm (RAA) takes care of managing multiple transmission parameters. It is aimed at hardware supporting up to 2 antennas, supporting up to 2 spatial streams, but we believe the same techniques could be applied for hardware with more antennas or more spatial streams.

Transmission Mode First, Iwl-Mvm-Rs decides which transmission "mode" to use. These modes represent the combination of a specific PHY family (legacy or not legacy) and the number of supported spatial streams when applicable[5]:

5: The driver source code mentions the HE_SISO and HE_MIMO mode for HE PHY (introduced in the 802.11ax amendment), but the RAA for this mode is actually done in the firmware

- the LEGACY mode, which corresponds to a OFDM PHY (when operating in the 5GHz band) or a ERP PHY (when operating in the 2.4 GHz band);
- the SISO mode, which corresponds to a single spatial stream HT or VHT PHY;
- the MIMO2 mode, which corresponds to a two spatial stream HT or VHT PHY;

More generally, Iwl-Mvm-Rs has to decide which PHY to use for its transmissions, which can be the ERP or the HT PHY in the 2.4 GHz, and the OFDM, the HT or the VHT PHY in the 5 GHz band. In practice, Iwl-Mvm-Rs chooses the best PHY available: if the VHT PHY is supported by the STA, it uses it, else, it uses the HT PHY. If both the HT and VHT PHY are unsupported, it uses the ERP PHY or the OFDM PHY depending on the band.

Transmission Rate or MCS index For legacy modes, Iwl-Mvm-Rs chooses which transmission rate to use, which differ depending on the used band:

- For the 5 GHz band, it supports the 6, 9, 12, 18, 24, 36, 48 and 54 Mb/s transmission rates (OFDM PHY);
- For the 2.4 GHz band, it additionally supports the 1, 2, 5.5, and 11 Mb/s (ERP-DSSS and ERP-CCK PHY);

For non-legacy modes, Iwl-Mvm-Rs chooses which MCS index to use for the transmissions (we use the VHT MCS definition, as the driver does, which means they are comprised between 0 and 9).

Antenna Configuration For single spatial stream modes, Iwl-Mvm-Rs chooses which antenna to use, identified by the letters A, B as it is suited for hardware using up to two antennas. For two spatial stream modes, it does not choose for any antenna configuration as it implies using the two available antennas.

Channel Width Iwl-Mvm-Rs chooses which channel width, or bandwidth, to use when communicating, which is 20 MHz for the legacy modes, or 20, 40, 80 or 160 MHz for the non legacy modes. When transmitting in a non legacy mode, it uses the maximum bandwidth supported by the recipient.

Guard Interval Duration Iwl-Mvm-Rs chooses whether to use an Short Guard Interval (SGI) or an Long Guard Interval (LGI) when the mode is not a legacy mode.

LDPC and STBC Iwl-Mvm-Rs enables Low-Density Parity-Check Codes (LDPC) and Space-Time Block Code (STBC) if the recipient STA support them, which assumes it is a HT or VHT STA.

Transmission Power Control In some very specific cases, Iwl-Mvm-Rs performs transmission power control, or adaptation, trying to reduce its default transmission power.

A-MSDU and A-MPDU Iwl-Mvm-Rs decides whether to enable Aggregation of Service Data Unit (A-MSDU) and Aggregation of MPDU (A-MPDU).

Algorithm Description

Iwl-Mvm-Rs has two main components:

- **MCS Scaling** which changes the MCS index, trying to maximize the "throughput" (or changing the Transmission Rate, in case of legacy modes).
- **Column Scaling**, which changes the *column*, which is a combination of *transmission mode*, *guard interval duration* when applicable[6], and *antenna configuration* parameters.

6: For legacy transmission modes, the guard interval duration is fixed.

The Iwl-Mvm-Rs RAA interleaves MCS Scaling phases and Column Scaling phases, forming what is called a "search cycle", as illustrated on the left part of Figure 4.10. The algorithm starts with the lowest transmission rate, which has the best reliability, which corresponds to a 1 Mb/s rate in both the 2.4 GHz and 5 GHz bands, but with different PHY.

No data is generated in order to perform rate adaptation: only frames that are received from the upper layers are transmitted, which means the rate adaptation algorithm makes progress only when data is transmitted. When a frame needs to be transmitted, it is transmitted according to the set of transmission parameters currently chose by Iwl-Mvm-Rs, but Iwl-Mvm-Rs is not actively involved in the transmission. Rather, Iwl-Mvm-Rs receives the results of the transmissions from the lower layer, the transmission *status*, e.g. "transmission with the set of parameters S succeeded, after N retries" or "transmission with the set of parameters S' failed", and makes its decision based on those statuses.

Thus, the MCS scaling and column scaling are only executed when some transmission status has been received.

Column Scaling starts when the MCS Scaling phase chooses not to change MCS (or transmission rate) after the reception of a transmission status. The alternation of MCS Scaling and Column Scaling continues until all the columns have been tried, which means the end of the search cycle

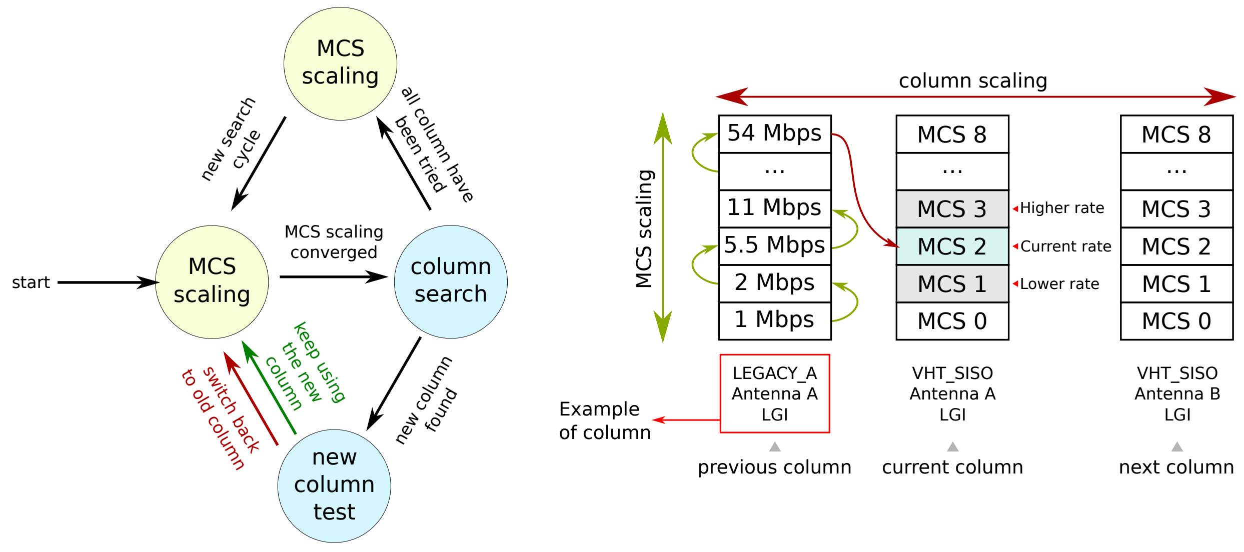


Figure 4.10: Flowchart of the different states of Iwl-Mvm-Rs (left) and example of sequence of decisions made by the Iwl-Mvm-Rs (right).

and the MCS scaling phase runs until the beginning of a new search cycle.

Depending on the band (2.4 or 5 GHz) used for the communications, which is not something the RAA chooses (for example, some APs are only using a single band), not all the PHY can be used. If the band is 2.4 GHz, then the VHT PHY cannot be used, and if the band is 5 GHz, then the ERP PHY cannot either. As the guard interval duration is not tunable for the legacy transmission modes, the number of valid columns is further reduced.

MCS Scaling The MCS index (or transmission rate, in the case of a legacy mode) is the only parameter that can be changed by the MCS Scaling component. Given a specific column c which is currently in use for the transmissions, the MCS scaling component can take one of the following decisions: lowering the MCS index, raising the MCS index, or keep using the current MCS index. The decision is made in a deterministic manner, according to the MCS scaling internal data structures values, which are illustrated in Table 4.4.

The MCS Scaling internal data structure include, for each MCS index i (in column c):

- $T_{\max}[i,c]$, the maximum theoretical throughput for MCS index i in column c;
- $N_{\text{success}}[i,c]$, the number of frames successfully transmitted at the MCS index i in column c;
- $N_{\text{failure}}[i,c]$, the number of frames unsuccessfully transmitted at the MCS index i in column c;
- $SR[i,c]$, the success ratio of the transmitted frames for the MCS index i in column c, defined by:

$$SR[i,c] = \frac{N_{\text{success}}[i,c]}{N_{\text{success}}[i,c] + N_{\text{failure}}[i,c]}$$

- $T_{\text{measured}}[i,c]$, the "measured throughput" for the MCS index i in column c, defined by:

$$T_{\text{measured}}[i,c] = SR[i,c] * T_{\max}[i,c]$$

Those data structures are updated whenever a new transmission status is received. Multiple cases can be distinguished, depending on whether the frame was aggregated or not:

- If the frame was not aggregated: the N_{success} of the successful transmissions and the N_{failure} of the unsuccessful transmissions are updated with the exact number of times they were used for the transmission.
- If the frame was aggregated using A-MPDU:
  - If a block acknowledgment was received, increase the N_{success} by the number of correctly received MPDU and increase N_{failure} by the number of incorrectly received MPDU;
  - If no block acknowledgment was received, increase N_{failure} only by one, to avoid penalizing too much the MCS for a single missed block acknowledgment.

The values for the maximum theoretical throughput for the MCS indexes, $T_{\max}$, are hard-coded into tables for each MCS index i, each bandwidth bw (20, 40, 80 or 160 MHz), each number of spatial streams (1 or 2), for the two possible guard interval durations (SGI or LGI) and for the two possible state of A-MPDU aggregation (A-MPDU enabled or disabled).

The "measured" throughput $T_{\text{measured}}[i,c]$ for a MCS index i is computed by multiplying the success ratio $SR[i,c]$ of up to the last 62 frame transmissions at the MCS index i, with the theoretical throughput $T_{\max}[i,c]$ for this MCS index. At least 8 successful transmissions or 3 failed transmissions are required to compute the success ratio: if either $N_{\text{failure}}[i,c] < 3$ or $N_{\text{success}}[i,c] < 8$, $SR[i]$ and $T_{\text{measured}}[i,c]$ are undefined (None).

The definition of T_{measured} means it is not exactly a *measured throughput*, as it does not take into account the size of the transmitted frames, but more of an equivalent, in terms of throughput, of a frame success ratio. In the implementation of Iwl-Mvm-Rs, all the computations are done in *fixed point*: throughput are computed using integers between 0 and 12800, as well as percentage. The null (0 Mb/s) throughput is mapped onto 0, while the maximum supported throughput is mapped onto 12800. The correspondence is not a one-to-one correspondence with the values present in the standard, and we observed multiple commits (in the Linux source tree) updating some hard-coded $T_{\max}$ values. This means the $T_{\max}$ values might actually represent the real performances of Intel hardware, and not the values from the standard.

If we assume the current used MCS index is i, in column c, the decisions made by MCS Scaling are the following:

1. if $SR[i,c] < 15\%$, or $T_{\text{measured}}[i,c] = 0$, then $i \leftarrow i-1$
2. else:
   a) if $T_{\text{measured}}[i-1,c] = \text{None}$ and $T_{\text{measured}}[i+1,c] = \text{None}$
   b) or if $T_{\text{measured}}[i-1,c] \leq T_{\text{measured}}[i,c]$ and $T_{\text{measured}}[i+1,c] = \text{None}$

Table 4.4: Illustration of the internal data structure of the MCS Scaling algorithm ($c = 2$ or $c = 3$, bw = 20MHz, LGI, $i = 4$)

MCS index i	$T_{\text{max}}[i, c]$	$N_{\text{failure}}[i, c]$	$N_{\text{success}}[i, c]$	$SR[i]$	$T_{\text{measured}}[i, c]$
8	216	0	0	None	None
7	202	0	0	None	None
6	193	0	0	None	None
5	183	0	0	None	None
4	159	0	6	None	None
3	124	3	7	70%	87
2	102	0	8	100%	102
1	76	1	8	88.8%	67
0	42	0	8	100%	42

   c) or if $T_{\text{measured}}[i+1, c] \geq T_{\text{measured}}[i, c]$

   then $i \leftarrow i + 1$

3. else, if $T_{\text{measured}}[i-1, c] \leq T_{\text{measured}}[i, c]$ and $T_{\text{measured}}[i+1, c] \leq T_{\text{measured}}[i, c]$, then $i \leftarrow i$
4. else, if $SR[i, c] \leq 85\%$ and $T_{\text{max}}[i-1, c] \geq T_{\text{measured}}[i, c]$ and:

   a) $T_{\text{measured}}[i-1, c] \geq T_{\text{measured}}[i, c]$
   b) or $T_{\text{measured}}[i-1, c] = $ None

   then $i \leftarrow i - 1$
5. else, $i \leftarrow i$

which can be reformulated as:

1. if the success ratio is too small ($< 15\%$) or the measured throughput is zero, **decrease** the MCS index;
2. else:

   a) if the measured throughputs with the lower and higher adjacent MCS indexes are unknown;
   b) or the measured throughput with the lower adjacent MCS index is worse and the measured throughput with the higher adjacent MCS index is unknown;
   c) or the measured throughput with the higher adjacent MCS index is better;

   increase the MCS index;
3. else, if the measured throughputs with the lower and higher adjacent MCS indexes are worse, **maintain** the MCS index;
4. else, if the success ratio is lower than 85% and the lower adjacent MCS index throughput can theoretically beat the current measured throughput, and:

   a) if the measured throughput with the lower adjacent MCS index is better;
   b) or the measured throughput with the lower adjacent MCS index is unknown;

   decrease the MCS index;
5. else, **maintain** the MCS index.

Column Scaling The column scaling component is in charge of managing the column, which is a combination of transmission mode, guard interval duration, and antenna configuration parameters. Available columns are listed in the Table 4.5. During each search cycle, the column scaling

Table 4.5: Columns used by the column scaling component in the Intel Iwl-Mvm-Rs.

Column Index c	Column Name	Antenna Configuration	Guard Interval	Next Columns $col_{\text{next}}(c)$
0	LEGACY_ANT_A	A	LGI	[1, 2, 6]
1	LEGACY_ANT_B	B	LGI	[0, 3, 6]
2	SISO_ANT_A	A	LGI	[3, 6, 4, 0, 1]
3	SISO_ANT_B	B	LGI	[2, 6, 5, 0, 1]
4	SISO_ANT_A_SGI	A	SGI	[5, 7, 2, 0, 1]
5	SISO_ANT_B_SGI	B	SGI	[4, 7, 3, 0, 1]
6	MIMO2	AB	LGI	[2, 7, 0, 1]
7	MIMO2_SGI	AB	SGI	[4, 6, 0, 1]

component will try to find the best column, and will maintain a set col_{visited} of the visited column indices.

Iwl-Mvm-Rs starts in the column with the lower possible column index, which is most of the time the LEGACY_ANT_A column, i.e. $c = 0$ (but legacy modes might be disabled by some STAs), and starts executing the MCS Scaling phase. At this point, $col_{\text{visited}} = \{0\}$ (or the index of the starting column). When the MCS Scaling phase converges, that is to say the MCS index i is kept constant, the column scaling starts searching for a new column.

To know which columns to try, each column c has a list of "next columns" $col_{\text{next}}(c)$ that will be tried in the order of the list. For example, the next columns for the column $c = 0$ are the columns 1, 2 and 6, as shown on Table 4.5.

A column $c' \in col_{\text{next}}(c)$ is tried when:

- All the columns $c'' \in col_{\text{next}}(c)$ such that $c'' < c'$ have already been tried;
- c' has not been visited during the current search cycle:

$$c' \notin col_{\text{visited}}$$

- c' can theoretically beat the current measured throughput $T_{\text{measured}}[i, c]$ (otherwise it is skipped and added to col_{visited}):

$$\max_{i'}(T_{\text{max}}[i', c']) > T_{\text{measured}}[i, c]$$

Trying a new column c' means switching to this column and trying a specific MCS index in the column, and adding c' to the col_{visited} set. The initial starting MCS i' index in the new column c' is chosen according to the success ratio:

- if it is high enough, i.e. $SR[i, c] \geq 85\%$, i' is the smaller MCS index whose theoretical throughput $T_{\text{max}}[i', c']$ is higher than the current theoretical throughput $T_{\text{max}}[i, c]$,
- otherwise, i' is the smaller MCS index whose theoretical throughput $T_{\text{max}}[i', c']$ is higher than the current measured throughput $T_{\text{measured}}[i, c]$.

If $T_{\text{measured}}[i', c'] \geq T_{\text{measured}}[i, c]$, the column scaling algorithm stops (which means c' keeps being used) and the MCS scaling component starts again in the new column, using the data already gathered, which is illustrated by the green arrow on Figure 4.10. Otherwise, the column scaling component switches back to the old column c and executes the MCS scaling component again, as illustrated by the red arrow on Figure 4.10. In any case, the statistics of the MCS scaling of the column that was not chosen are reset, that is to say the statistics of the old column if the tried column is better, or the statistics of the tried column if the old column is better.

New Search Cycle At some point, all the columns have been tried and one "final" column has been found, which marks the end of the search cycle. MCS Scaling runs in this final column until the start of a new search cycle, at which point the set of visited columns col_{visited} is re-initialized, marking only this final column as visited.

A new search cycle is triggered when:

- too many frames have failed (160 in legacy, 400 otherwise) since the beginning of the previous cycle;
- too many frames have succeeded (400 in legacy, 4500 otherwise) since the beginning of the previous cycle;
- too much time has been spent after the end of the previous search cycle. The maximum time between two consecutive cycles is set to 5 seconds.

Aggregation A-MSDU, which wraps multiple MSDUs in a single MPDU and therefore in a single Wi-Fi frame, is only available for non-legacy modes. It is enabled or disabled by the MCS scaling algorithm when it makes a decision about the MCS index. To be enabled, the MCS index has to be greater or equal to 5, and the MCS scaling decision has to be either maintaining the MCS index, or increasing the MCS index. In all the other cases, A-MSDU is disabled. A-MPDU, which wraps multiple MPDUs in a single PPDU and therefore in a single Wi-Fi frame, is enabled on a per-hardware queue basis, when the number of frame per second exceeds 10 and depending on the traffic identifier of the data: real-time data, such as voice over IP, will not be aggregated using A-MPDU if its traffic identifier identifies it correctly.

Retry Chain When frames are not acknowledged, and not using A-MPDU, they are retransmitted. Depending on the original set of transmission parameters, Iwl-Mvm-Rs tries to retransmit frames failing to be acknowledged with different transmission rates in order to increase the delivery probability. The "retry chain" allows frames to be retransmitted up to 14 times, for a total of at most 15 transmission tries for a single frame. The first re-transmission uses the same transmissions parameters as the original frame, the next 4 re-transmissions use the lower two MCS indexes in the same column, and the ones after change the used column and use decreasing MCS indexes and alternating antennas.

Conclusion Iwl-Mvm-Rs tackles rate adaptation by moving continuously over the MCS ladder, which is in contrast with the behavior of e.g. Minstrel which samples arbitrary rates. While other RAA may rely on randomness for their decisions, the Intel RAA makes deterministic choices based on a few thresholds. Its behavior can be easily predicted, within the limits of what can be predicted about the channel, whereas performances for RAA relying on randomness are more difficult to predict. Many thresholds or counters seems to be chosen arbitrarily. Indeed, their values seem too *round* to have been chosen in another way. Given the difficulty to completely change one RAA for another, in modern hardware, there is room for improvement in the choice of such constants which could be tinkered without much effort, and studied in situ or in simulation. Given the complexity of such algorithm, our first reflex is to try to study it in simulation, and observe its behaviors in a controlled environment.

4.5 Simulation of the Intel Rate Adaptation Algorithm

The algorithm has been implemented as a Very High Throughput (VHT) low-latency WifiManager in the ns–3 simulator. The choice of the ns–3 simulator for this work is driven by multiple reasons:

- ns–3 is currently the network simulator supporting the most RAA, as illustrated by the comparison in Table 4.3.
- ns–3 is the de facto standard for the simulation of Wi-Fi networks: implementing this RAA in ns–3 means this work can be reused by many (hopefully).
- The use of a network simulator as ns–3 allows us to better illustrate the relations between the whole protocol stack, including TCP/IP, and the RAA, which might not be easily done when developing an *ad-hoc* simulator in for example Matlab or Python.

As the language used to implement the manager algorithms in ns–3 is C++, one could have blindly translated the rate adaptation algorithm of the driver code from C to C++ and simulate the behavior of Iwl-Mvm-Rs. Still, many parts of the driver code have only housekeeping functions for the underlying hardware, such as catching unexpected bugs or resynchronizing the state of the rate adaptation algorithm that runs in the kernel with the state of the WNIC hardware.

As these behaviors should not happen in a simulator, the RAA in the simulator has been re-implemented using the skeleton of the RAA in the Intel driver, but is not a one-to-one correspondence. The number of source lines of code of the simulated RAA is thus cut by more than two-thirds with regard to the driver code: according to the *sloccount* utility, the number of line of code of our implementation is 834, to compare to the 3204 lines of the original.

The simulation covers most of the algorithm but ignores some part of the original RAA for the sake of simplicity. First, it blindly enables A-MPDU aggregation when possible, as we assume no real-time traffic will be sent, and that the number of frames per second exceeds 10. Then, the retry chain does not use decreasing transmission rates but instead uses the current transmission rate. As re-transmissions of lost MPDUs in a A-MPDU frame are not made using this retry chain because missing MPDUs are resent as a part of the next A-MPDU, and as the first retransmission rate used in the retry chain is the original transmission rate, we believe this simplification introduces no major changes in the results of our simulations.

The algorithm is also only suited for WNIC with at most two antennas that support at most two spatial streams, as this is the only hardware supported by the Intel IwlWifi/Mvm driver. Thus, integrating this RAA *as is* to the ns–3 upstream repository is not currently desirable, as it would not be usable in all simulations. From this point of view, proposing a more generic algorithm based on the same principles could be useful, but it remains to be done.

Validation In order to validate the behavior of the simulated code, we have compared the decisions of the rate adaptation algorithm with the decisions of a real piece of hardware, the Intel Corporation Wireless 8260 WNIC, in different situations. To get easy access to the decisions made by the RAA on a Linux system, one can load the IwlWifi Linux kernel module with the option 'debug=0x00100000', which enables debug messages about the rate adaptation process inside the kernel log (accessible using the 'dmesg' command). Multiple patterns observed over-the-air are correctly reproduced in the simulator.

On Figure 4.11, we compare two ns-3 simulator traces with two Over-The-Air (OTA) captures made using a device in monitoring mode. The left graph is constructed by tracing the transmission rates at the start of a simulation, and by extracting the transmission rates of a capture made after an inactivity period longer than 5 seconds. It presents a specific pattern of the Iwl-Mvm-Rs algorithm, which is to transmit frames in legacy rates in steps of approximately 8 frames: 6Mb/s, then 9Mb/s, 12Mb/s, etc. To create the right graph, the computer equipped with the Intel WNIC saturated the card with a UDP stream generated by the iPerf3 tool, and a UDP generator was used in NS-3. One can observe regular patterns that are spaced by more than 4500 frames, which is the number of successful frames needed by the algorithm to trigger a new search cycle. In these situations, the search is not successful as the algorithms (both in simulation and in the hardware) finally come back to the previous transmission rate. The differences in the radio environment leads Iwl-Mvm-Rs to an apparent different behavior when compared to the simulator trace, but this is only due to the simplifying assumptions made to construct the simulation environment. While these simple comparisons are by no mean a proof that the algorithm implemented in ns-3 is the exact same algorithm than the one implemented in the Linux kernel, it allows us to underline its realism.

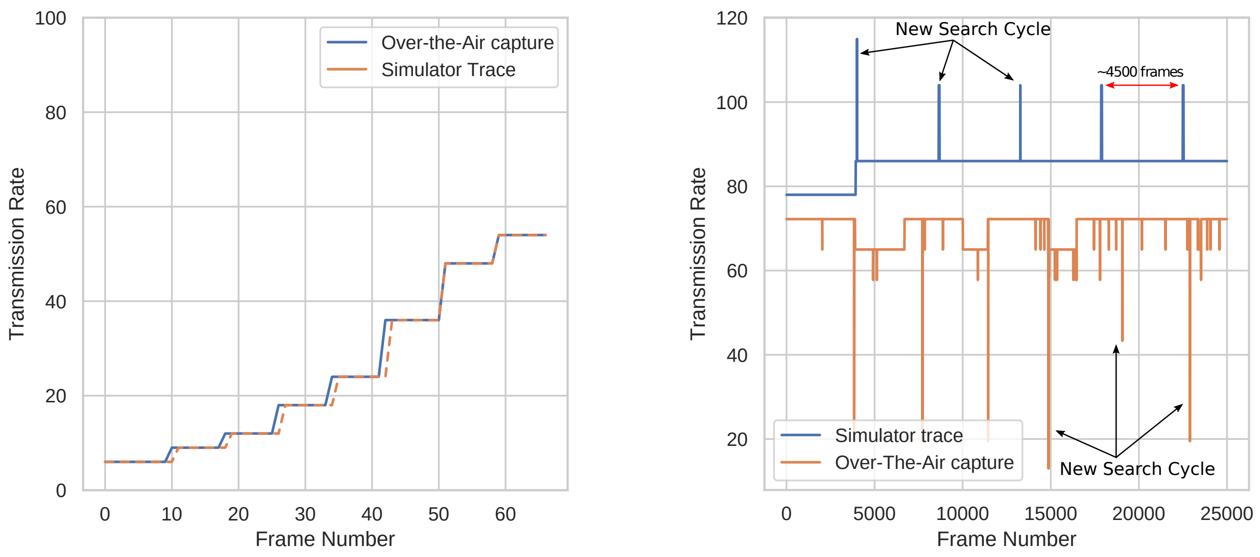


Figure 4.11: Comparison of the behavior of the simulated RAA and the RAA running on real hardware. Left graph is taken at the start of the algorithm, while right graph is taken when saturating the link.

4.6 Conclusion

In this chapter, we explained how we reverse engineered the RAA used in most Intel devices, and in particular in Intel Aero drones and presented the algorithm for the first time. If we were to argue to correctly model or simulate Wi-Fi networks in general, you need to correctly model or simulate each of their components, then describing this algorithm in a high level language is a step in the good direction. Stepping back, we can make a few comments about Iwl-Mvm-Rs.

First, and the reading of the previous parts should have convinced you, Rate Adaptation Algorithm are complex pieces of software that inherit of the complexity of the 802.11 standards. New features are regularly added to the 802.11 standard, but few to none old or unused features are removed. The upside is that it is still possible to use a fifteen-year-old Wi-Fi device in 2020, the downside is that this complexity has to be managed, and represents, in my opinion, a technical debt that could be ditched.

Having tens of parameters taking a few values each means the search space for the best transmission parameters cannot be covered exhaustively in a reasonable amount of time, so smarter solutions are needed. While the Iwl-Mvm-Rs RAA seems to be reasonably efficient at this task – after all, it's used by at least tens of millions of devices worldwide – its global structure and inner working looks a bit simplistic. Many thresholds seem to be chosen arbitrarily, e.g. success ratios of 15% or 85%, and only a few minor changes were made to the algorithm in the past years, while there is probably room for improvement.

Before our work of the algorithm, Iwl-Mvm-Rs was not described in the scientific literature, while still being used in real hardware.

In the next chapter, we will describe two well known rate adaptation algorithms, Minstrel-Ht and IdealWifi, and compare their performances in simulation in the context of drone networks.

Performance Comparison of Rate Adaptation Algorithms 5

Having described the Intel RAA, Iwl-Mvm-Rs, we wanted to study its performances in scenarios related to drone networks. Would it be suitable for drone-to-drone airborne communications ? Would it be fast enough to react to changes in the channel and maintain connectivity ? Having implemented the RAA in the ns–3 simulator, we can easily compare it to other RAAs in simulation. In this chapter, we therefore use the network simulator ns–3 in order to evaluate and compare the performances of three different rate adaptation algorithms, namely Iwl-Mvm-Rs, Minstrel-HT and IdealWifi. After describing the IdealWifi and Minstrel-HT algorithms, we compare the performances of the three algorithms using three different simulation scenarios which illustrate the behavior of such algorithms. In particular, to measure the adaptability and the responsiveness of these algorithms we design one simulation scenario involving node mobility or sudden changes in the communication channel characteristics, one scenario where the transmission conditions do not change, acting as a baseline simulation, and one scenario involving a two hop communication link.

5.1 Minstrel-HT and IdealWifi

The network simulator ns–3 supports multiple RAAs, as already stated in Chapter 4 and illustrated by Table 4.3, but only a few are compatible with the latest published standard and support VHT PHY features, for example. They are ConstantRate, which uses constant transmission parameters, IdealWifi and Minstrel-HT, which we will describe in a few paragraphs.

While ConstantRate and IdealWifi can be understood pretty quickly, for example by reading the source code of the implementation in the ns–3 source code, literature refers to Minstrel-HT by linking to the original commit [29] introducing it in the Linux Kernel source code for HT PHY, as it is one of the two algorithm (with Minstrel) implemented in the mac80211 component. This is problematic, as the algorithm has since evolved to support VHT PHY in the Linux kernel, as well as in ns–3, which means this citation is not really up-to-date with neither simulation nor the state of the actual algorithm. While it is true Minstrel (which has been extensively studied) and Minstrel-HT share a lot, they still behave differently and Minstrel-HT deserves to be described. Originally disjoint algorithms, more and more functions of Minstrel and Minstrel-HT have been mutualized in the previous decade, mainly to simplify code maintenance. As a result, even Minstrel behavior has been changed.

[29]: Fietkau (2010), *Minstrel HT: New rate control module for 802.11n*

Table 5.1: Organization of the different transmission groups used by Minstrel-HT. The groups are sorted by bandwidth, then by the guard interval duration, then by the number of streams.

Group Index	PHY	Bandwidth	Guard Interval	# Spatial Streams
0	HT	20 MHz	LGI	1
1				2
2				3
3				4
4			SGI	1
5				2
6				. . .
7		40 MHz		
. . .				
15	HT	40 MHz	SGI	4
16	CCK			
17	VHT	20 MHz	LGI	1
18				2
. . .				
40	VHT	80 MHz	SGI	4

Minstrel-HT

For stations not supporting the HT or VHT PHYs, Minstrel-HT falls back to the classical Minstrel algorithm. In any other case, Minstrel-HT has its own behavior (when compared to Minstrel), but some data structures and functions are still shared with Minstrel, at least in the Linux source code. For the description of Minstrel-HT in this chapter, we focus on hardware supporting multi-rate retries, which means a failed frame can be sent with other rates, but the Minstrel-HT RAA also supports hardware not supporting multi-rate retries.

In Minstrel-HT, the transmission parameters are organized into groups which are a combination of number of streams, bandwidth, and guard interval duration (SGI or LGI), as depicted in Table 5.1, with each group containing multiple VHT MCS values. In order to choose which transmission parameters to use, Minstrel-HT maintains a statistics tables which contain data about the number of transmission attempts, and transmission successes for each VHT MCS in each group. Those statistics are updated for every new transmission status coming from the lower layer, that is to say every time an acknowledgment has been received correctly, or *not* received in a timely manner.

The RAA also maintains other statistics, which are not recomputed or updated for every transmission status received from the lower layer, but only when more than 50 ms have elapsed since the previous update. One of the reason behind this is that the computations involve divisions, which is too computationally expensive to be done for every received transmission status. Such statistics are the average number of MPDU in the transmitted A-MPDU, named AVG_AMPDU_LEN, and the success probabilities for each VHT MCS in each group. Whenever 50 ms have elapsed since the last transmission status reception, a special function, called UPDATE_STATS, is in charge of updating those statistics.

In the literature, we can read that the duration between two calls of UPDATE_STATS is 100 ms (as it was effectively the case for Minstrel), and the behavior of ns–3 has also been to use a 100 ms duration, consistently, since the introduction of the Minstrel-HT RAA in its source code. Yet, this

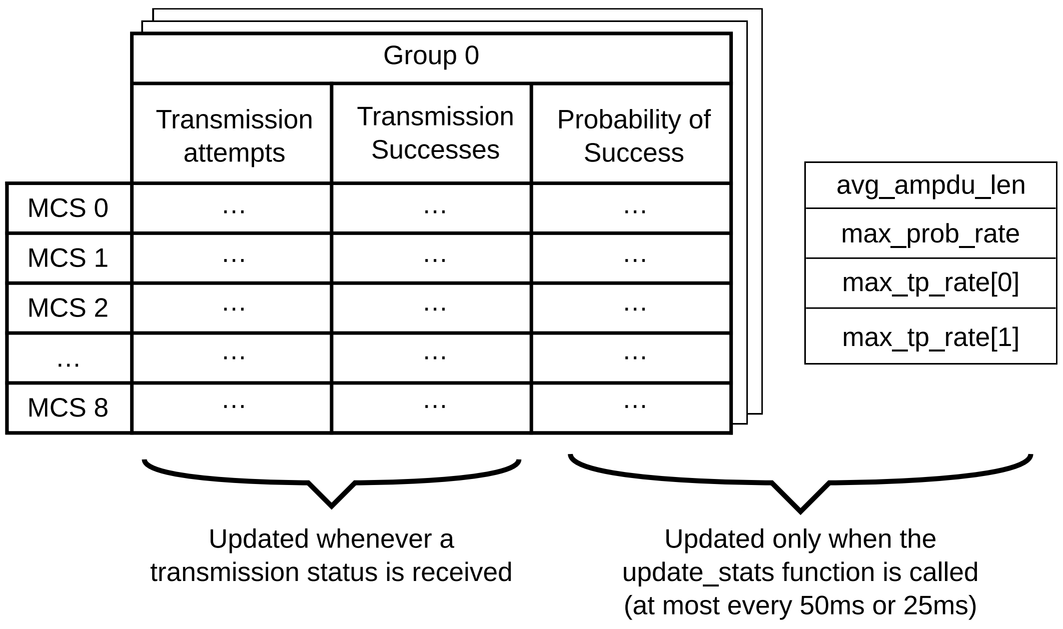


Figure 5.1: Overview of statistics maintained by the Minstrel-Ht RAA.

is in contradiction with what is actually implemented in the Linux kernel, and therefore used in real hardware. We discovered this behavior during the writing of this thesis, and it has now been corrected in the upstream source code of ns–3.* This duration is also only a minimum on the duration between the update of the statistics: indeed, statistics are only updated when a transmission status has been received from the lower layers, which means they are only recomputed when a transmission has been made. If no transmissions are made, the statistics are not updated. An overview of the stats maintained by the RAA is shown on Figure 5.1.

As Minstrel, Minstrel-HT uses a sampling approach, using the best transmission parameters found yet for most of the frames (using normal frames), according to its statistics table, while sampling other transmission parameters regularly to try to find better transmission parameters (using sampling frames). Let us note than both normal and sampling frames are data frames that would need to be transmitted; no specific transmission is made by the RAA for its operation. The best transmission parameters are updated by the UPDATE_STATS function, executed whenever a transmission status is received from the lower layers *and* more than 50 ms have elapsed since its last call. The *best* transmission rates are of two types:

- MAX_PROB_RATE is the transmission rate providing the highest probability of transmission success;
- MAX_TP_RATE[0] and MAX_TP_RATE[1] are the best and second-best transmission rates in terms of throughput.

As Minstrel, Minstrel-HT also uses retry chains to send frames: the frames are initially sent using their initial expected transmission parameters, but in case they are not acknowledged, they are sent with different transmission parameters which increases the probability of reception. Both normal frames and sampling frames have the same retry chain, and

* https://gitlab.com/nsnam/ns--3-dev/-/merge_requests/462

Table 5.2: Organization of the transmission rates in the Minstrel-HT retry chains.

	Normal Frame	**Sample Frame**
Transmission	Best Throughput (max_tp_rate[0])	Random Rate
First Retransmission	Second to Best Throughput (max_tp_rate[1])	Second-best Throughput (max_tp_rate[1])
Second Retransmission	Best Success Probability (max_prob_rate)	Best Success Probability (max_prob_rate)

only differ by the use of different initial transmission parameters. Whereas Minstrel records the top four transmission parameters offering the highest throughput, Minstrel-HT only uses the top two transmission parameters, MAX_TP_RATE[0] (best) and MAX_TP_RATE[1] (second-best). Another special transmission parameter, MAX_PROB_RATE, is the transmission parameter using a single transmission stream that offers the highest probability of delivery according to the statistics table. Normal frames, are opposed to sampling frames, are sent using MAX_TP_RATE[0]. The retry chain, common to normal and sampling frames, is composed of MAX_TP_RATE[1] and MAX_PROB_RATE.

When a frame needs to be sent, we can therefore distinguish two cases: either it will be a sampling frame, either it will be a normal frame. When frames are sent, Minstrel-HT decides if they should be sampling frames or not based on three counters, SAMPLE_WAIT, SAMPLE_TRIES and SAMPLE_COUNT. The first counter, SAMPLE_WAIT, is a decreasing counter allowing sample frames to be sent when it is equal to zero, which is also its initial value. It is used to avoid sampling too much, by preventing sampling. The second counter, SAMPLE_TRIES, is the number of frames that can be sampled consecutively, and its initial value is 4, but will always be set at 1 after the first sampling. Finally, SAMPLE_COUNT is the number of frames that can be sampled between each statistics update (which are separated by 50 ms), and its initial value is 16. At the start of the algorithm, as SAMPLE_WAIT is equal 0, then 16 (SAMPLE_COUNT) frames can be used for sampling before the next update of the statistics, with at most 4 (SAMPLE_TRIES) consecutive frames used for sampling.

Whenever a sampling frame is actually sent, SAMPLE_TRIES is decreased by one, and this counter ultimately reaches zero. At this point, both SAMPLE_WAIT and SAMPLE_TRIES are equal to zero, which means it is now only possible to send normal frames (as opposed to a sampling frame). The values for the counters are reset whenever a transmission status is received, at the same time transmission attempts and successes counters are updated. When a transmission status is received, if SAMPLE_COUNT is bigger than zero, which means the quota of frame sampling for a single statistics update duration has not been completely used, then SAMPLE_-COUNT is decreased by one, SAMPLE_TRIES is set to 1, and SAMPLE_WAIT is set to $16 + 2 *$ AVG_AMPDU_LEN with AVG_AMPDU_LEN representing the average number of MPDU in a A-MPDU. This means at least $16 + 2 *$ AVG_AMPDU_LEN normal frames must be sent before a single sampling frame will be able to be sent. When the statistics are updated, in the UPDATE_STATS function, the SAMPLE_COUNT counter is reset at a value covering all the possible transmission parameters, that is to say the number of groups multiplied by the 8 possible MCS values. This behavior is one key difference with Minstrel, as Minstrel uses 10% of the sent

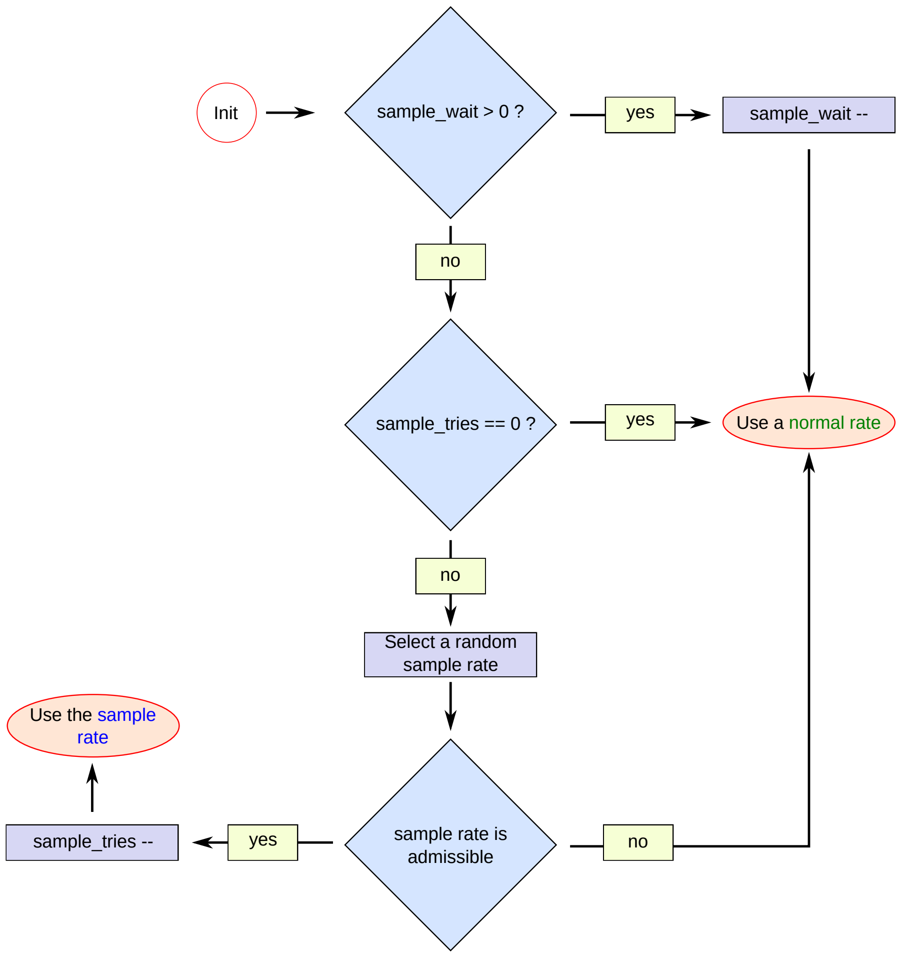


Figure 5.2: Overview of the transmission process for the Minstrel-Ht RAA, executed whenever a frame needs to be transmitted.

frames for sampling purposes, while this means Minstrel-HT uses at most around 5.3% of the sent frames for sampling purposes (assuming AVG_AMPDU_LEN = 1).

If the frame to send should be a sampling frame, then Minstrel-HT will try to find an admissible set of parameters to use for sampling. Yet, it might not find one, in which case it will not send a sampling frame but a normal frame. To do so, Minstrel-HT randomly selects a MCS in the sample group, and checks if it is appropriate to use as sampling parameters. Initially, the sample group is the group used for the first transmissions, but for every sampling attempt, it is increased in the order described in Table 5.1, in a round-robin manner. If we denote by TP_RATE1 and TP_RATE2 the rates in {MAX_TP_RATE[0], MAX_TP_RATE[1]} with respectively the smaller and bigger transmission durations, then the transmission parameter is deemed inappropriate if at least one of the following condition is true:

- it is not supported by the remote station;
- it is MAX_TP_RATE[0] or MAX_TP_RATE[1];
- the current average success probability is higher than 95%;
- its transmission duration is bigger than 3 times the one from MAX_PROB_RATE;
- its transmission duration is bigger than the one from TP_RATE1 and it has already been tried once since the last update of the statistics;

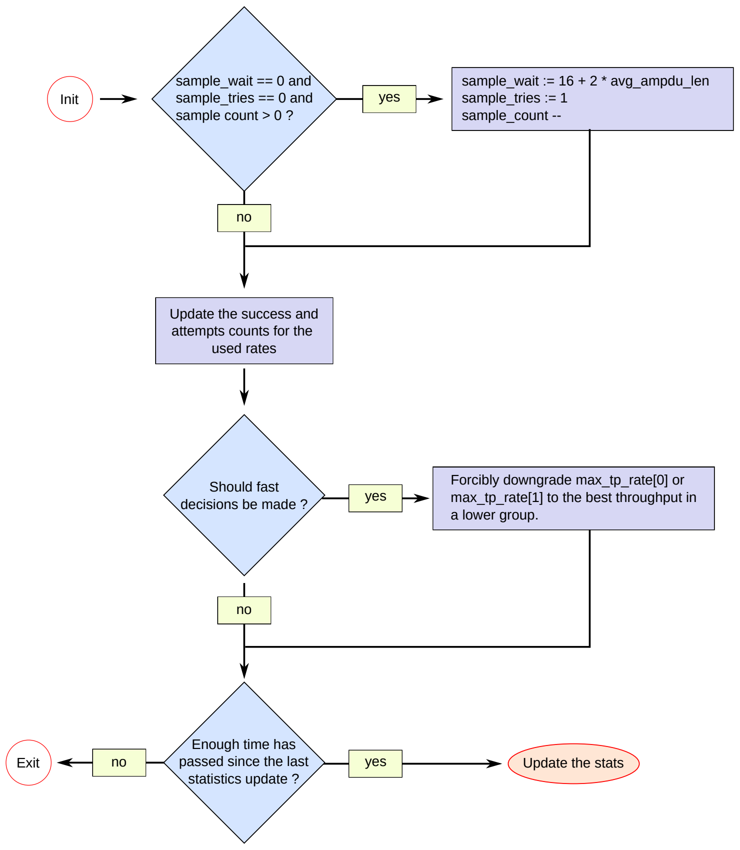

Init
sample_wait == 0 and sample_tries == 0 and sample count > 0 ?
yes
sample_wait := 16 + 2 * avg_ampdu_len
sample_tries := 1
sample_count --
no
Update the success and attempts counts for the used rates
Should fast decisions be made ?
yes
Forcibly downgrade max_tp_rate[0] or max_tp_rate[1] to the best throughput in a lower group.
no
Exit
no
Enough time has passed since the last statistics update ?
yes
Update the stats

Figure 5.3: Overview of the process happening whenever Minstrel-HT receives a transmission status.

- its transmission duration is bigger than the one from TP_RATE2 and either:
  - its number of spatial streams is strictly smaller than the number of spatial streams used for the TP_RATE1;
  - its transmission duration is bigger than the one from MAX_-PROB_RATE;

  and it has not been skipped for 20 statistics update duration already (2 seconds) or it has already been tried twice since the last update of the statistics.

An overview of what happens when a frame needs to be transmitted is depicted on Figure 5.2.

The average number of MPDUs in the A-MPDUs and the probability of success for each VHT MCS in each group were originally computed using an Exponential Weight Moving Average (EWMA) whenever UPDATE_STATS was called, every 50 ms, as described in Equation 5.1. The emphasis is being put on the previous values as they account for 75% of the new value. In the equation, $y[n]$ represents the new computed smoothed value, $y[n-1]$ represents the previous smoothed value, and $x[n]$ represents the new observed value since the previous update. The use of such a moving average is an attempt to smooth the rate success probability and the AVG_AMPDU_LEN and avoid reacting to every small temporary change in the channel, such as noise.

$$y[n] = 0.75 * y[n-1] + 0.25x[n] \tag{5.1}$$

Yet, the use of 50ms windows and an EWMA introduce some lags when actual, important changes occur in the channel. A first mechanism to deal with such situations is executed whenever transmissions status are received, and transmissions counters are updated, which allows for a faster response than updating the statistics only every 50 ms. This mechanism, we call the *fast decision* mechanism, checks whether the current probability of success with MAX_TP_RATE[0] and MAX_TP_RATE[1] is less than 25% if at least 30 transmissions attempts have been made with such transmission rates. If so, the best throughput in a group whose index is smaller than the group of MAX_TP_RATE[0] (respectively MAX_TP_RATE[1]) is chosen, effectively making the transmissions more robust.

An overview of what happens when a transmission status is received by Minstrel-HT is depicted on Figure 5.3.

Another changes have been made in the Linux Minstrel-HT implementation in October 2019 to fight the latency introduced by the EWMA and a 50ms window. First, a new *smoothing* low-pass filter was introduced for the computation of the probability of success for the transmission parameters, in place of the EWMA. This filter appears to have originated from the Figure 14-8 from [17], and its expression is given in Equation 5.2. For Minstrel-HT, $Q = 1$ and $F = \frac{\sqrt{2}}{16}$. As the filter is used to compute a percentage, values below 0.0 and above 1.0 are respectively set to 0.0 and 1.0. Then, a smaller statistics update periods of 25 ms is used, in place of the previous 50ms period, always modulo the inaccuracy of the Linux software clock (which is not a real time operating system). A trace of the duration between the statistics update periods for a Archer T2U, equipped with a MT7610U chipset, using Minstrel-Ht, is shown on figure Figure 5.4. It has been obtained by patching the mac80211 module to print a message in *dmesg* for each update of the statistic table and then flooding the WNIC using a ping flood. The timers have then been extracted from the *dmesg* output for this plot. For this specific scenario, the minimum time between two statistics update is 25.97 ms, the maximum time between two call is 80.74 ms, and the average is 44.17 ms.

[17]: Chamberlin (1985), *Musical Applications of Microprocessor*

$$
\begin{aligned}
y[n] &= Ay[n-1] - By[n-1] + Cx[n] \text{ with} \\
A &= 2\cos(2\pi F)\exp\left(\frac{-\pi F}{Q}\right), \\
B &= \exp\left(\frac{-2\pi F}{Q}\right), \\
\text{and } C &= 1 - A + B
\end{aligned}
\tag{5.2}
$$

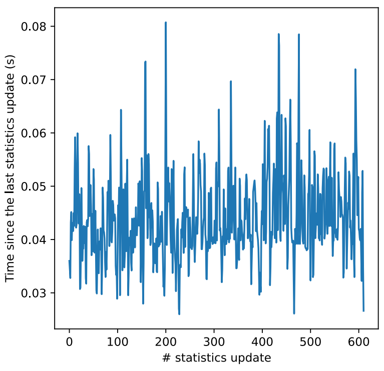
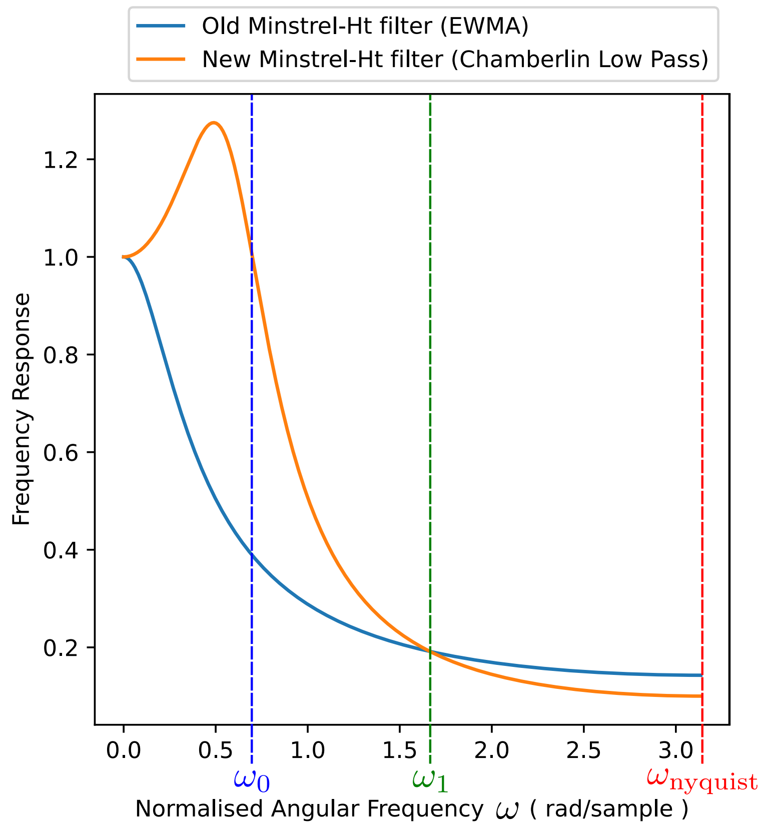


Figure 5.4: Duration between each statistic update call of Minstrel-Ht in a ping flood scenario.

A comparison of the frequency responses of the EWMA and the new filter used in Minstrel-HT is given in Figure 5.5

It is unclear how those two changes are changing the overall behavior of Minstrel-HT, as the used filter is little-used. Moreover, those two changes have not been implemented in the ns–3 implementation of Minstrel-HT. As a result, the ns–3 behavior is not on par with the Linux kernel behavior for kernel versions newer than the introduction of those changes.

Of course, what we presented is also a simplification of the actual implemented Minstrel-HT RAA, but most of the behavior of the algorithm



The Chamberlin Low Pass filter increases the importance of frequencies below ω_0 and decreases the importance of frequencies above ω_0.

Compared to the old Minstrel-Ht filter, EWMA based, frequencies between ω_0 and ω_1 are less filtered while frequencies above ω_1 are more filtered.

Frequencies above the nyquist frequency ω_{nyquist} are not considered.

Figure 5.5: Comparison of the frequency response of the old and new method for success probability averaging in Minstrel-HT.

has been described here, which to the best of our knowledge is the first detailed description.

IdealWifi

The ns–3 IdealWifi RAA is a theoretical RAA in that it relies on a perfect out-of-band mechanism to transmit the Signal-To-Noise (SNR) of each frame, as seen by the receiver, back to the sender. This algorithm is said to be based on Receiver-Based AutoRate (RBAR), in which the receiver is in charge of choosing the transmission rate of the sender. In IdealWifi, the sender uses the SNR to then choose the transmission parameters which provide a Bit Error Rate (BER) below 10^{-5} and which maximize throughput.

IdealWifi, despite its name, is far from being ideal, and does not represent an optimum. The first reason is that it uses the SNR of the previous frames to choose its transmission rate: if we assume some kind of magical channel whose capacity alternates between low and high for every frame, then IdealWifi will try to use a high transmission rate when the capacity of the channel is low, resulting in a transmission failure, and will then use a low transmission rate while the channel capacity is high, resulting in poor performances. For such a channel, transmitting with any fixed set of transmission parameters would be better. The second reason behind the fact IdealWifi is not ideal is that it relies on maintaining the Bit Error Rate (BER) below a certain threshold, while maximizing throughput.

This threshold is fixed, which means some transmission rates that would ultimately provide higher throughput are not used because their BER is too high.

Simulation Environment and Parameters

The simulations we will describe use the ns–3 simulator in its 3.29 version. All the devices use 2 antennas supporting 2 spatial streams in transmission and reception (so-called "2x2:2" devices) using a 20MHz bandwidth, in the channel 42 (5210MHz). Simulations #1 and #2 involve two stations (STA), both running the same RAA, one acting as UDP traffic generator and the other one acting as a sink, and the simulation #3 introduces a relay STA. At $t = 1s$, the traffic generator sends UDP datagrams of size 1420 bytes to the sink (receiver), until the end of the simulation, either at the specified application data rate, either in saturation, which means an application data rate bigger than the maximum theoretical transmission rate achieved by the device in its configuration. The simulations parameters are summarized in Table 5.3.

Table 5.3: ns–3 simulation parameters used in the three scenarios of Chapter 5.

Simulation Parameter		**Value**
ns–3 Version		3.29
Simulation Duration	Scenario #1:	30s
	Scenario #2:	variable
	Scenario #3:	60s
Wi-Fi Standard		802.11 ac
Wi-Fi MAC type		Sta (source) / Ap (relay and sink)
Rate adaptation algorithm		Minstrel-HT, Ideal or Intel
Spatial Streams		2x2:2
Channel Width	Scenario #1:	20MHz
	Scenario #2:	20MHz
	Scenario #3:	20MHz
Propagation Loss Model		*Log Distance* and optionally *Nakagami*
Routing		Static
Application		OnOff (constant bitrate, UDP)

In the following, we look a three different simulation scenarios illustrating the performances of the RAA in a context of drone networks. The first scenario acts as a base, or reference scenario, to study the performances of the three different RAAs. We expect the three RAA to have the same overall behavior, as the characteristics of the channel, including the characteristics of the fast-fading when it is used, do not change during the simulation. Failure to behave *nicely* over this simulation scenario would be the sign of a broken RAA. The second scenario involves mobility of one node and a changing channel. The goal of the second scenario is to check how RAAs react to sudden changes in the channel, as we can expect to find in mobile and urban scenarios for drone networks. The third and last simulation scenario involves a two-hop communication link. Multi-hop links are of interest in the context of drone networks because of their prominence in mesh networks, but RAAs are often not designed with such applications in mind, but for more classical one-hop Client/Access Point architectures.

5.2 Scenario #1 - Fixed distance

In this scenario, the two STAs are static and are separated by a fixed distance. Simulations last 30 seconds and each result is the mean of 5 simulations. The distance is increased with a step of 1 m. The simulation uses the log-distance path loss model, described in Equation (5.3), which models the path loss Pl as a function of the distance d and of γ, an environment-dependent constant called the loss exponent, and the power Pl_0 at distance $d = d_0$.

$$Pl(d) = Pl_0 + 10\gamma \log_{10}(\frac{d}{d_0}) \tag{5.3}$$

We optionally add a Nakagami-m fast fading loss model to account for the changes in power due to the presence of multiple paths. The expression of the added loss, denoted Pn, at distance d and when incoming power is equal to P is given in Equation. (5.4).

$$Pn(d, P) = X(m, P/m) \tag{5.4}$$

with X a realization of the $\mathcal{X}$ Erlang random variable whose density function is:

$$f(x; k, \mu) = \frac{x^{k-1}e^{-\frac{x}{\mu}}}{\mu^k (k-1)!} \quad \text{for } x, \mu \geq 0$$

The parameter m is chosen to be 1.5 for distances smaller than 80 m and 0.75 for distances bigger than 80 m, which are the default parameters used in the ns–3 model.

The mean throughput (measured at the application level of the sink) with regard to distance is depicted on Figure 5.6, without fast-fading (left) and with fast-fading (right). Without fast-fading, as no time-varying fading is present, the reception power of the frames remains constant during each simulation. While Minstrel-HT and Iwl-Mvm-Rs have overall comparable performances, IdealWifi performs significantly worse at distance larger than 15 meters because it does not use transmission rates that would result in a BER bigger than 10^{-5}. By doing so, it does not use transmission parameters that would result in a better throughput. With Nakagami-m fading, overall, Minstrel-HT performs best, and Iwl-Mvm-Rs performs worst. For example, at distance $d = 45$ m, the throughput is divided by a factor 2.2 for Iwl-Mvm-Rs compared to the case without fast-fading, while it is only divided by a factor 1.7 for Minstrel-HT and a factor 1.1 for IdealWifi.

We now comment on the reasons behind the loss of throughput for the Iwl-Mvm-Rs algorithm. The Nakagami-m fading is a time varying fading. It means that the reception power for each frame may vary a lot while the position or the mobility of the STAs remains the same, with periods of destructive fading and constructive fading. Looking at the transmission rates used by Iwl-Mvm-Rs, we can observe that overall, the average transmission rate of Iwl-Mvm-Rs is lower than the ones of Minstrel-HT and IdealWifi. The reasons behind this behavior are present in the two phases of the Iwl-Mvm-Rs algorithm, the MCS scaling phase and the

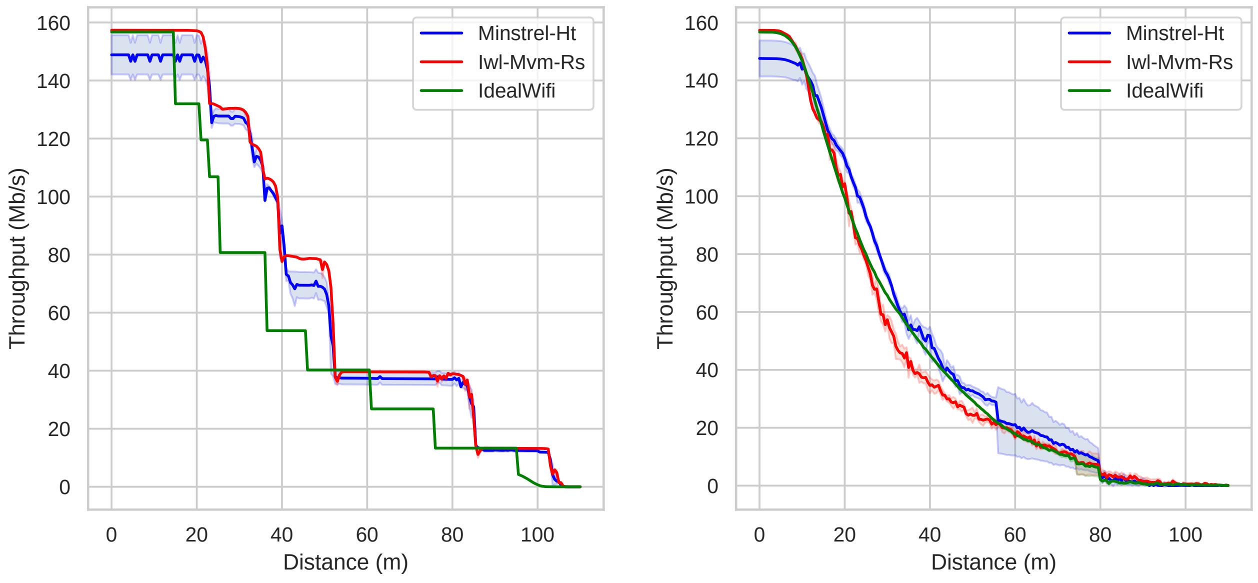


Figure 5.6: Scenario #1: Throughput as a function of the distance between the source and the sink in the first scenario, without fast-fading (left) and with fast-fading (right). Shaded regions represent the standard deviation.

Column scaling phase. First, in the MCS scaling phase, the algorithm will make its decisions based on at least 3 failed transmissions or 8 successful transmissions. The randomness introduced by the fading may result, locally, in bad performances when these decisions are made, leading the algorithm to stop its exploration and choose a smaller transmission rate instead of climbing the MCS ladder and finding a transmission rate that would result in a better throughput over the long term. At the heart of this behavior is the asymmetry between the number of failed transmissions and the number of successful transmissions needed to take a decision, as well as the low number of frames required to take a decision. A bigger test window would result in more robust estimations of the potential throughput associated with a given MCS. Then, in the column scaling phase, only a single MCS (corresponding to the smaller MCS index whose theoretical throughput is higher than the current theoretical throughput or the current measured throughput, depending on the success ratio) is tested to decide whether a more in-depth exploration (*i.e.* am MCS scaling phase) should be done in the tested column. As previously, this single test can be very short and coincide with a period of destructive fading, resulting in the whole column being wrongly marked as unsuitable.

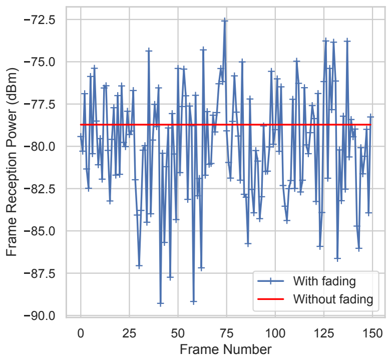


Figure 5.7: Comparison of the reception power of 150 frames when a Nakagami-m fading is absent or present, without any other change in the channel.

Figure 5.8 confirms the results from Figure 5.6. It represents, for each manager, the distribution on the transmission rates used by the source to send its frames when the distance between the source and the destination is 45 m. Without fading, IdealWifi uses a lower transmission rate for numerous frames compared to Iwl-Mvm-Rs and Minstrel-HT. These latter two mainly use the same transmission rate, but Iwl-Mvm-Rs sends more frames and sometimes use a higher transmission rate. Table 5.4 gives the associated mean transmission rate and the success ratio. It shows that without fading Iwl-Mvm-Rs achieves a good trade-off between the mean transmission rate and the success ratio, leading to the highest throughput with this distance. Conversely, Figure 5.8 shows that, with fading, Iwl-Mvm-Rs mainly uses smaller transmission rates than the other solutions. Minstrel-HT uses the higher transmission rates for most

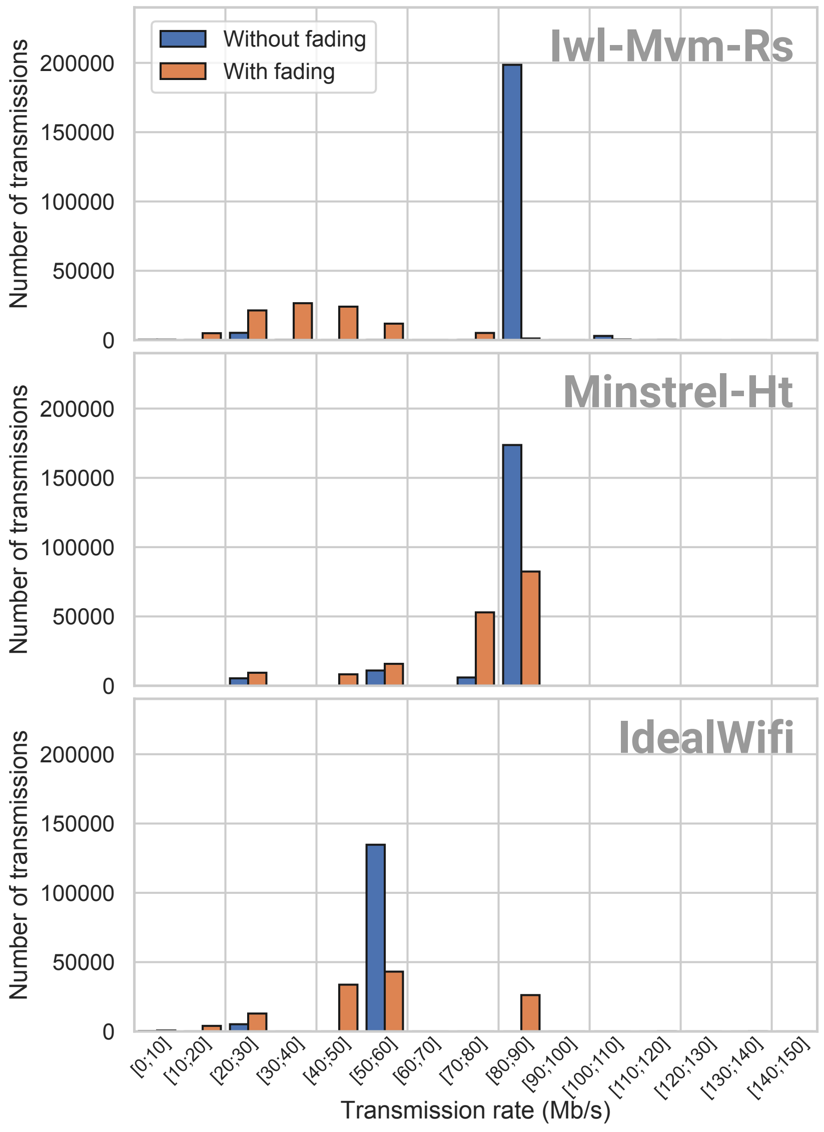


Figure 5.8: Scenario #1: Distribution of the used transmission rates for the frame sent by the source, without or with fading and for a distance of 45 m. Top to bottom: Iwl-Mvm-Rs, Minstrel-HT, IdealWifi.

of the sent frames and thus has the higher mean transmission rate. Table 5.4 shows that the transmission rates used by Minstrel-HT also lead to frames losses since its achieved success ratio is around 71%. However, the trade-off achieved by this manager is good enough to transmit data with the highest throughput.

Solution	**Without fading**		**With fading**	
	Mean transmission rate (Mb/s)	Success ratio	Mean transmission rate (Mb/s)	Success ratio
Iwl-Mvm-Rs	86.1	97.5%	38.3	88.5%
Minstrel-HT	80.7	98.1%	74.0	71.5%
IdealWifi	57.0	99.9%	56.4	83.2%

Table 5.4: Scenario #1: Mean transmission rate and success ratio for a distance of 45 m.

5.3 Scenario #2 - Circular Alternating Walls

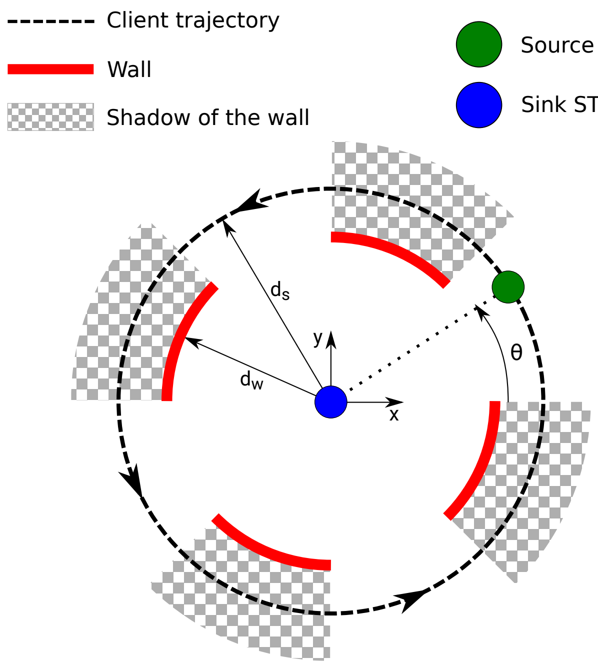


Figure 5.9: Scenario #2: Top view of the "Circular Alternating Walls" simulation. Walls create shadows leading to sudden changes in the channel between the source and the sink.

In this scenario, the source moves in circle around the static sink, at constant speed. The overall distance $d_s = 30m$ between the two STAs does not change during the simulation, but walls are present at distance $d_w = 15m$ for angles θ ranging in $[\pi/4; \pi/2]$, $[3\pi/4; \pi]$, $[-3\pi/4; -\pi/2]$ and $[-\pi/4; 0]$, as shown on Figure 5.9. In the "shadows" of the walls, a fixed loss of 5 dBm is added to account for the presence of an obstacle. A log-distance path loss model is used, as well as an optional fast Nakagami-m fading. Each simulation lasts five laps and each result is the mean of 5 simulations.

The goal of the *Circular Alternating Walls* scenario is to test the responsiveness of the RAAs. By making sudden changes in the channel, we can illustrate how well a RAA can react to the presence of a wall or a building, which is important in an urban context. The quicker the RAAs adapt, the best, as they can use the full capacity of the channel. This scenario was designed by imagining a drone flying in an urban environment, where various obstacles, such as buildings, can cause the quality of the channel to vary very quickly. Figure 5.10 presents the evolution of the transmission rate in this scenario, for a source throughput of 125 Mb/s and a speed of 5 m/s. Without fading, one can observes that the Minstrel-HT algorithm is not switching rate when walls are present while Iwl-Mvm-Rs and IdealWifi react to the presence of walls, which

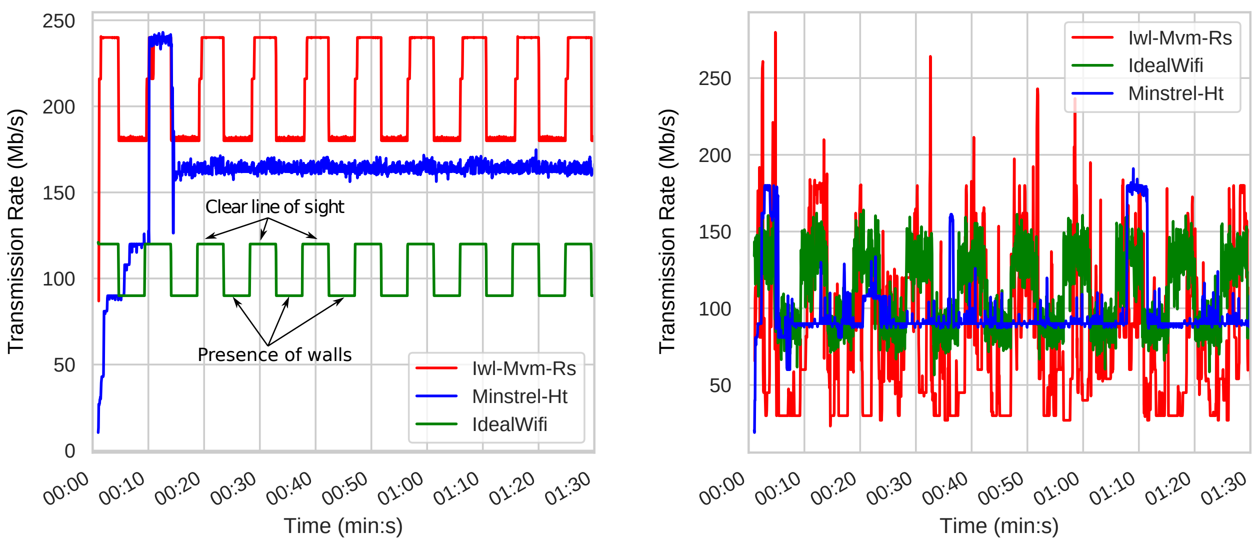


Figure 5.10: Scenario #2: Evolution of the transmission rate in the Alternating Wall Scenario for a source throughput of 125 Mb/s and a speed of 5 m/s, and a bandwidth of 40MHz (only used in this example). Throughput are averaged over periods of 100 ms. Left is without Nakagami-m fading, right is with Nakagami-m fading.

illustrates the fact that a RAA might not adapt at all to the presence of walls. When Nakagami-m fading is added, Iwl-Mvm-Rs still adapts to the channel, but its rate of transmission rate changes is higher than Minstrel-Ht's one. Overall, Minstrel-Ht maintains a transmission rate close to 90Mb/s, which remains stable, while Iwl-Mvm-Rs oscillates between 27 Mb/s (the lowest transmission rate in a 40MHz channel for a 2x2:2 device, at MCS0), and sometimes reach 270Mb/s (MCS7). As in the scenario #1, Nakagami-m fading is more detrimental to Iwl-Mvm-Rs than to Minstrel-Ht, and even leads the Intel RAA to try transmission rates it would not try when no Nakagami-m fading is present. In this scenario, IdealWiFi is probably the RAA performing the best, as it adapts to presence of walls, which both Minstrel-Ht and Iwl-Mvm-Rs fails to.

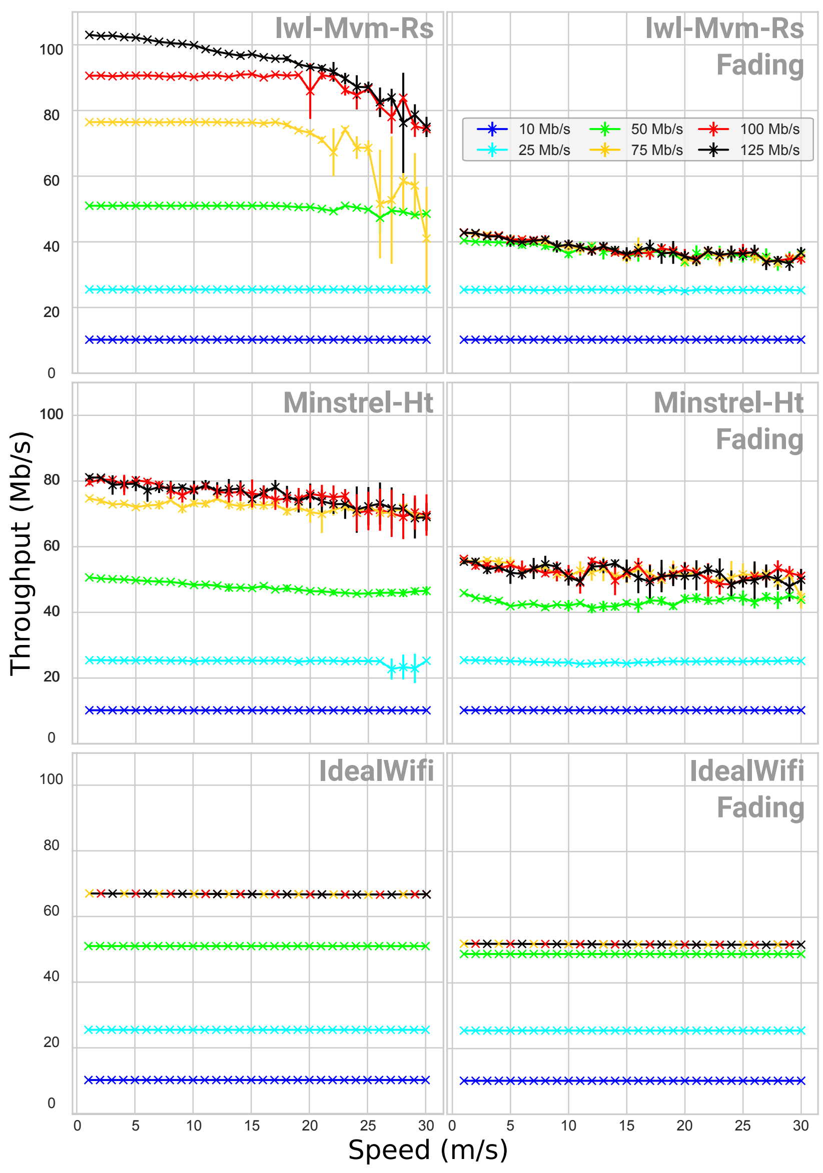


Figure 5.11: Scenario #2: Evolution of the throughput with regard to the speed and the throughput of the source. Left: without fading, Right: with Nakagami-m fading. The vertical bars represent the standard deviation.

Figure 5.11 shows the evolution of the throughput with regard to speed for the three managers, with and without fading. One observes that above a certain source throughput, the sink throughput measured at the application level decreases as the speed of the moving source increases for both Minstrel-HT and Iwl-Mvm-Rs. For source throughput of 10Mb/s and 20Mb/s, the impact of the source speed is limited as the same transmission rate is used whether the source is in the shadows or not, because it can accommodate the needs in throughput. At higher source throughput, such as 75Mb/s and above, a small but steady decrease of the throughput with the speed of the source can be observed for Minstrel-HT. The big differences in performances between Minstrel-HT and Iwl-Mvm-Rs are explained by the fact the former does not do any adaptation (as shown on Fig. 5.10) while the latter does.

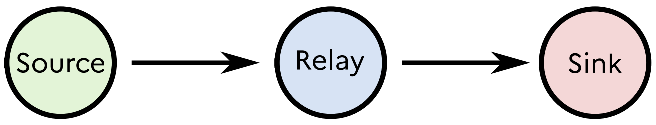


Figure 5.12: Two Hop Scenario in Chapter 5. The sink and the source are fixed at $x = 0$ and $x = 130$ m respectively, and the relay position can be changed.

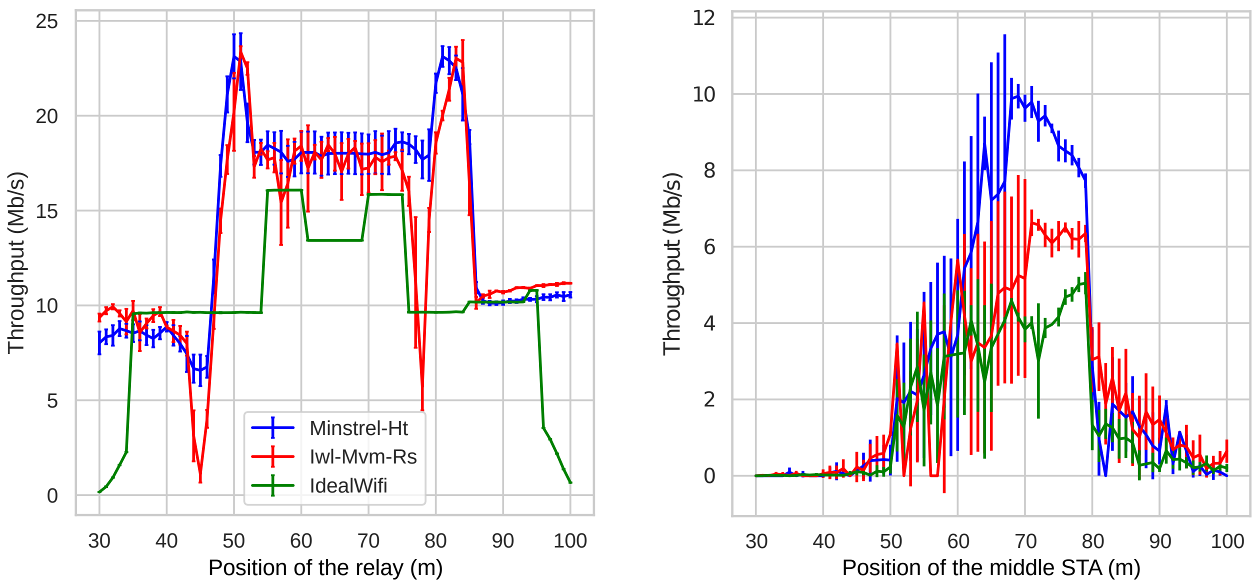


Figure 5.13: Scenario #3: Evolution of the end-to-end throughput with regard to the relay position. The source is located at d=0 m and the sink is located at d=130 m. Left is without Nakagami-m fading, right is with Nakagami-m fading.

5.4 Scenario #3 - Two-hop Flow with Relay

In this scenario, one considers three static STAs communicating in a two-hop fashion, as illustrated on Figure 5.12. The source sends packets to the sink (receiver), packets which are forwarded by the relay. The source is positioned at $x = 0$ m and the sink is positioned at $x = 130$ m, while the relay position changes depending on the scenario. As in the *Fixed distance* scenario (Scenario #1), the source saturates its communication link with UDP datagrams at $t = 1$s until the end of the simulation that lasts 30s. The results in terms of throughput (measured at the application level of the sink) depending on the position of the relay are displayed on Figure 5.13. We observe a central symmetry in the throughput results when no fast fading is present, centered on $x = 65$ m, which is expected due to the topology of the simulation. When the relay is far from one of the endpoint (source or sink), then the obtained throughput is low since one of the links is of bad quality. As a result, the sink throughput is capped by the throughput of the low quality link.

We could think the best throughput would be obtained when the relay is equidistant from the source and the sink. Indeed, the throughput is capped by the throughput of the worst link, so having the relay in the middle ensure both links have similar capacity and should ensure the maximum throughput. Still, this is not the case. With all three RAAs, we get the best performance when the relay is not equidistant from the source and the sink. This is explained by threshold effects, as the quality of the channel does not change linearly with the distance, but in steps. Reusing the left figure from Figure 5.6, we explain in Figure 5.14 this behavior for the IdealWifi RAA. Because the position $d = 65$ m (vertical red dashed line) is not strictly located in the middle of a *step* where the

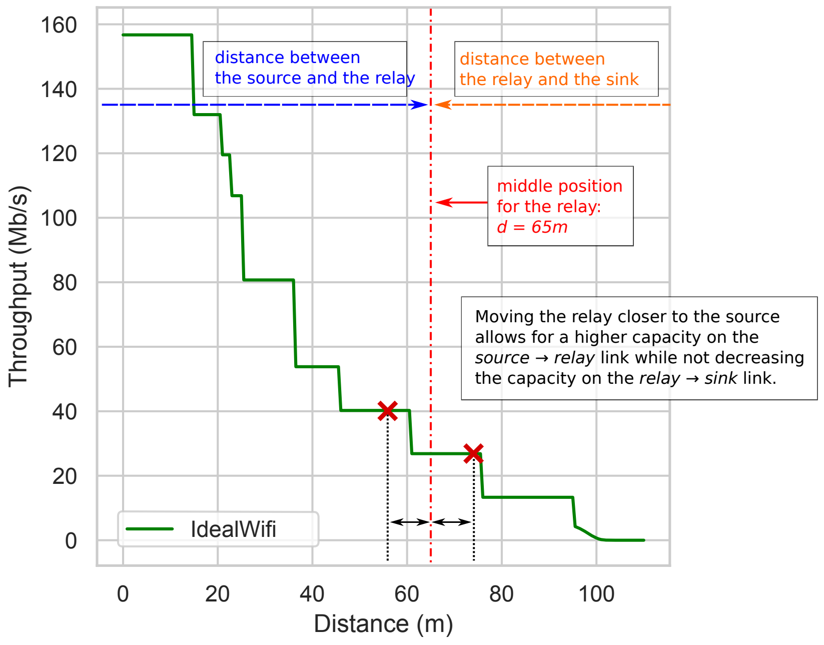


Figure 5.14: Scenario #3: Focus on the effect of moving the relay for the IdealWifi RAA on the link capacity.

capacity is constant, a capacity change can be observed by decreasing the x-axis by a certain value (left red cross marker), while no capacity change in the link happens when increasing the x-axis by this same value (right red cross marker). In particular, for IdealWifi, at $x = 65$ m, the capacity of both links (source ⟶ relay and relay ⟶ sink) are equal, around 28 Mb/s. For $x = 58$ m, the source ⟶ relay link has an increased capacity of 40 Mb/s, whereas the relay ⟶ sink link, which has now a length of $130 - 58 = 72$ m, still has the same capacity of 28 Mb/s. What is gained on some link is not necessarily lost on the other link, and this remains true for Minstrel-Ht and Iwl-Mvm-Rs, even if the curves are shifted due to the internal workings of the algorithms.

Focusing on Iwl-Mvm-Rs, and without fast fading, when the relay is located at $x = 45$ m and $x = 78m$ between the source and the relay, translating to distances of respectively 85 m and 52 m between the relay and the sink, we can observe wells in the throughput for the Iwl-Mvm-Rs RAA. Moving the relay left or right result in better throughput. To explain this behavior, we focus on one simulation with the Iwl-Mvm-Rs RAA. Looking at the average transmission rate of the relay ⟶ sink link, when $x = 78$ m we observe an average transmission rate of 59.9 Mb/s, for a resulting throughput of less than 5 Mb/s, whereas when $x = 70m$ we observe an average transmission rate of 43.1 Mb/s, for a resulting throughput of 17 Mb/s. For $x = 45$ m, we observe an average transmission rate of 35.5 Mb/s, for a resulting throughput of around 1 Mb/s, whereas for $x = 40$ m, we observe an average transmission rate of 14.0 Mb/s, for a resulting throughput of close to 9 Mb/s. For $x = 45$ m (respectively $x = 78$ m), the average transmission rate used for the relay ⟶ sink link is therefore higher than for $x = 60$ m (respectively $x = 90$ m), whereas the average transmission rate used for the source ⟶ relay link remains the same. Looking at frames emitted by the source, we can observe they are mostly correctly received by the relay, and the frames emitted by the relay are mostly correctly received by the sink, which means the RAA decisions are correct. We identify the main problem: at $x = 78$ m (for example), the relay is only transmitting during 6 seconds,

Table 5.5: Scenario #3: Mean transmission rate and success ratio.

RAA	d = 65 m		d = 82 m	
	Mean transmission rate (Mb/s)	Success ratio	Mean transmission rate (Mb/s)	Success ratio
Iwl-Mvm-Rs				
source - relay	42.6	94.5%	42.7	94.5%
relay - sink	43	99.5%	64.3	97.5%
Minstrel-HT				
source - relay	43.7	92.1%	41.5	91.9%
relay - sink	38.9	92.1%	85.6	98.6%

that is to say for 10% of the simulation duration, whereas at $x = 70$ m, it transmits for 26 seconds, accounting for 43% of the simulation duration. By tracing the decisions made by each node, we find the reason for this behavior: the sink (receiver) is sending acknowledgment (under the form of block acknowledgment) using a wrong transmission rate, which means they are not correctly received by the relay. According to the standard, block acknowledgments frames are to be sent using "*the highest rate in the BSSBasicRateSet parameter that is less than or equal to the rate [. . .] of the previous frame.*", which in our case means using a transmission rate of 24 Mb/s, the maximum rate in the Set of Basic Rate of the BSS, as data frames are sent using a VHT MCS of 4, which translates to a transmission rate of 78 Mb/s (long guard interval, 2 spatial streams). So why are frames sent at 24 Mb/s lost but not the data frames sent at 78 Mb/s ? This is due to the fact the block acknowledgments are transmitted using only one antenna, whereas data is sent using two antennas which results in a better SNR for the data than for the acknowledgments, as calculated by the ns–3 3.29 InterferenceHelper model. The ns–3 3.31 version has changed the way gains due to the antenna diversity or MIMO are computed, which might change or remove this behavior.

With fast fading, we observe an asymmetry on Figure 5.13 regarding the position of the relay. Indeed, attained throughput where the relay is located at $x = 55$ m are less than half of the attained throughput when the relay is located at $x = 75$ m. This behavior can be explained by the fact that the source $\longrightarrow$ relay link and the relay $\longrightarrow$ sink link do not have the same importance with regard to throughput. Indeed, the performance hit due to the loss of a frame on the source $\longrightarrow$ relay link is smaller than the performance hit due to the loss of a frame on the relay $\longrightarrow$ source link, as for any frame transmitted on the latter link, the channel has already been used at least once for the transmission on the former link. Therefore, it is better to have the relay closer to the sink to ensure frame for which some transmission cost has already been paid between the source and the relay are correctly delivered. This asymmetry can also be observed when no fast fading is present, but is more subtle.

In Table 5.5, one compares the mean transmission rate and the success ratio for the source-relay and the relay-sink links, when the relay is located at 65 m and 82 m from the source, and for the Minstrel-HT and Iwl-Mvm-Rs RAA which exhibit the same overall behavior with regard to distance.

One observes that when the distance between the source and the relay is 82 m, the transmission rate and the success ratio are similar to the ones obtained when the distance is 65 m. However, the second link has a better quality which results in higher transmission rates. This increase of transmission rate has also an impact on the medium occupancy. Indeed, even if the number of sent frames on the second link is higher with a distance of 82 m than with 65 m, the radio medium is less used by the relay when $d = 82$. This results in more radio accesses for the source that can send more frames when $d = 82$ compared to $d = 65$ although the radio conditions are similar on the first link. For instance, the source with Iwl-Mvm-Rs sends 61900 frames when $d = 82$ whereas it sends 53535 frames when $d = 65$. As a result, the end-to-end throughput at the sink is higher when $d = 82$ m.

5.5 Conclusion

Before looking at the performance of the Iwl-Mvm-Rs rate adaptation algorithm in ns–3, we expected the results to be strongly in favor of Minstrel-Ht. Indeed, the empiricism behind much of Iwl-Mvm-Rs inner working lead us to believe Minstrel-Ht, as the venerable successor of Minstrel, which had been studied extensively and been the subject of many papers, would always work better than Iwl-Mvm-Rs. Surprisingly, this is not the case, at least when comparing our implementation of Iwl-Mvm-Rs and the ns–3 implementation of Minstrel-Ht.

Without fading and node mobility, Minstrel-HT and Iwl-Mvm-Rs perform similarly. Indeed, the asymmetry in the number of lost frames or frames in success taken into account in the test window of Iwl-Mvm-Rs does not really disadvantage the RAA. Yet, the rate adaptation mechanisms are different, and they do not lead to the same used transmission rates and success ratio. IdealWifi obtains limited performance due to its conservative and rigid behavior.

Without fading and with mobility, Iwl-Mvm-Rs shows good results, compared to Minstrel-HT and IdealWifi, specifically when the throughput of the source is high. Iwl-Mvm-Rs is able to quickly adapt its transmission rate to the change in the channels, which looks not to be the case for Minstrel-Ht.

With fading and whatever the mobility, the use of Iwl-Mvm-Rs gives lower performance than with Minstrel-HT and IdealWifi. The algorithm has difficulties to deal with the randomness introduced by the fading, due in parts to the small test windows on which it bases its decisions, as well as the asymmetry in the number of lost frames or successful frames needed to make its decisions. A few "bad" frames, in terms of transmission power, will have a far greater influence than the same number of "good" frames.

Looking at the Linux kernel Minstrel-Ht implementation, we discovered a few important differences with the ns–3 implementation, leading to some disparities between the behavior of the simulator and the behavior of real hardware. As our simulations were conducted before those changes were taken into account into ns–3, our results should be taken with caution if one was to try to apply them directly to a non-simulated environment. But looking seriously at Minstrel-Ht, we can confidently say it is also far from perfect. As with Iwl-Mvm-Rs, many operational thresholds look arbitrarily chosen, and could probably be set more intelligently. In

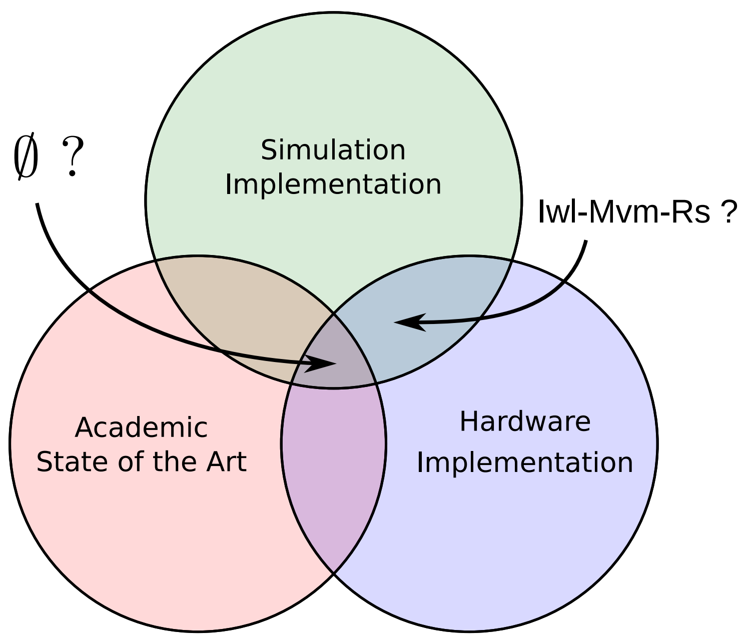


Figure 5.15: Venn diagram representing the articulation of the academic state of the art, simulation implementations and hardware implementations regarding rate adaptation algorithms.

particular, recent changes for the computation of the success probability in Minstrel-Ht have been made without much justification, and the science behind this change seems very thin. Those observations lead us to wonder whether the intersection of the rate adaptation algorithms implemented in network simulators, with the rate adaptation algorithms used in real hardware and the rate adaptation algorithms described in the academic literature is an empty set or not. While the Linux kernel evolves, papers describing the rate adaptation algorithms are somehow frozen, and the network simulators often base their implementation on those papers. Our description of the Intel rate adaptation algorithm, Iwl-Mvm-Rs, and its implementation, although imperfect, are a step in the right direction to fill this intersection Figure 5.15.

Now that we better understand the relationship between those three rate adaptation algorithms, mobility, and the performance one can get, we focus the next chapter on how using mobility to try to obtain better network performances, namely, controlled mobility.

Controlled Mobility: Antenna Orientation 6

We just illustrated in the previous chapter how important Rate Adaptation Algorithms (RAAs) are when considering the performances of Wi-Fi networks. The close relationship between nodes positioning, and node mobility in particular, and the different performances shapes one can expect with different RAA, leads us to search a RAA-agnostic mobility solution for improving Wi-Fi drone network performances. Relying on a specific RAA and modifying it might lead to better overall performances, but this means sticking to specific hardware vendors, and this requires being actually able to access and modify the RAA for testing purposes. This, unfortunately, seems not to be the current fashion as RAA are slowly leaving the operating system realm to relocate in chipsets silicon, which is of course not helped by the complexification of the Wi-Fi PHY.

Beside being independent of the used RAA, we didn't want the proposed controlled mobility solution to be linked to a specific application. While the distance between nodes of a drone network is an important parameter for the performances, being able to assume control over the position of drones means to interface with the considered application. Some middle ground between changing the position of the drone and not being to control the mobility at all, is to focus on the attitude of the drone, specifically its orientation. Indeed, when considering small multicopters, their orientation can be set independently of their position and their path, which is not the case for fixed-wings drones. For patrolling and coverage applications in particular, the orientation of the drone can be decoupled from the rest of the application as most sensors are either omnidirectional, e.g. chemical sensors, sound sensors, or are usually already mounted on gimbals, as this is the case for photography, video and imaging purposes for example.

In this chapter, we design a controlled mobility solution for Wi-Fi based drone networks relying solely on the orientation of the drones. The solution does not require any a priori knowledge on the antenna radiation patterns, and is completely distributed and decentralized. To study the solution, a custom framework based on ns–3 was developed to study the effects of non-isotropic antennas. We first describe the considered problem and its modeling, then we detail our proposed solution. We then present the evaluation framework we developed, and study our algorithm in different simulation setups, allowing us to underline its performances.

6.1 Controlled Antenna Orientation and Antenna Effects

In this section, we first introduce the studied problem, and we then describe the proposed solution for the antenna orientation. We consider a set of UAVs (also named as agents or nodes hereafter), each equipped with a wireless network interface controller using Wi-Fi and a directional antenna whose radiation pattern (also named the antenna gain pattern) is unknown. All the agents use the same Wi-Fi channel to communicate. The studied problem is the following: given a UAV fleet configuration, can each agent optimize its local antenna orientation in order to enhance the communication performance, such as throughput? We focus on multi-rotor UAVs because their three-dimensional positions and orientations can be fully controlled and maintained through time by the flight controller, while, for example, fixed-wing UAVs cannot hover at a given position. We also limit this study to UAVs whose 3D positions are static, but whose orientations in their horizontal plane, around the normal axis, named yaw, can be changed. Indeed, as the 3D UAVs positions are often application dependent, we focus on parameters that can be modified without interactions with the applications, for the sake of generality. These requirements cover, in particular, the class of coverage applications, such as surveillance, continuous monitoring or network coverage. As changing the roll (orientation along the longitudinal axis) or the pitch (orientation along the transverse axis) of a multi-rotor changes its 3D position when it is not subject to external forces apart from gravity, we assume those two quantities are also fixed.

Problem Modeling

Let $G = \{V, E\}$ be an undirected graph representing a set of N networked agents, where $V = \{A_1, A_2, \dots, A_N\}$ and $E \subset V \times V$ denote respectively the set of vertices and the set of edges. We denote by $Ad = (a_{i,j})_{(i,j) \in N \times N}$ the adjacency matrix of the graph: $a_{i,j} = 1$ if $(i, j) \in E$ meaning that agents A_i wants to communicate with agent A_j, and $a_{i,j} = 0$ if $(i, j) \notin E$ meaning that agent A_i does not wish to communicate with agent A_j.

Each agent is equipped with a directional antenna. The antenna radiation pattern is represented by a function g. As g can be different from one agent to another, we use g_i representing the antenna radiation pattern of agent A_i. It is expressed in decibels and in the spherical system of coordinates described in [10, Chapter 2.2]. Figure 6.2 gives an example of three antenna gain patterns in a plane (two directional antennas and one omnidirectional antenna). Depending on the antenna orientation between two neighbor agents (there exists a link between these 2 agents in G), these two agents may be able to communicate or not. When they are able to communicate, this orientation has also an impact on the power of the received signal. The higher the received power, the more likely the communication will be of good quality and will use a high transmission rate.

The objective of the agents is then to cooperatively solve the following optimization problem:

$$\max_{\phi \in [0;360[^N} f(\phi) := \sum_{i \in \{1,\dots,N\}} \sum_{\substack{j \in \{1,\dots,N\} \\ j \neq i}} a_{i,j} * S_{i,j}$$

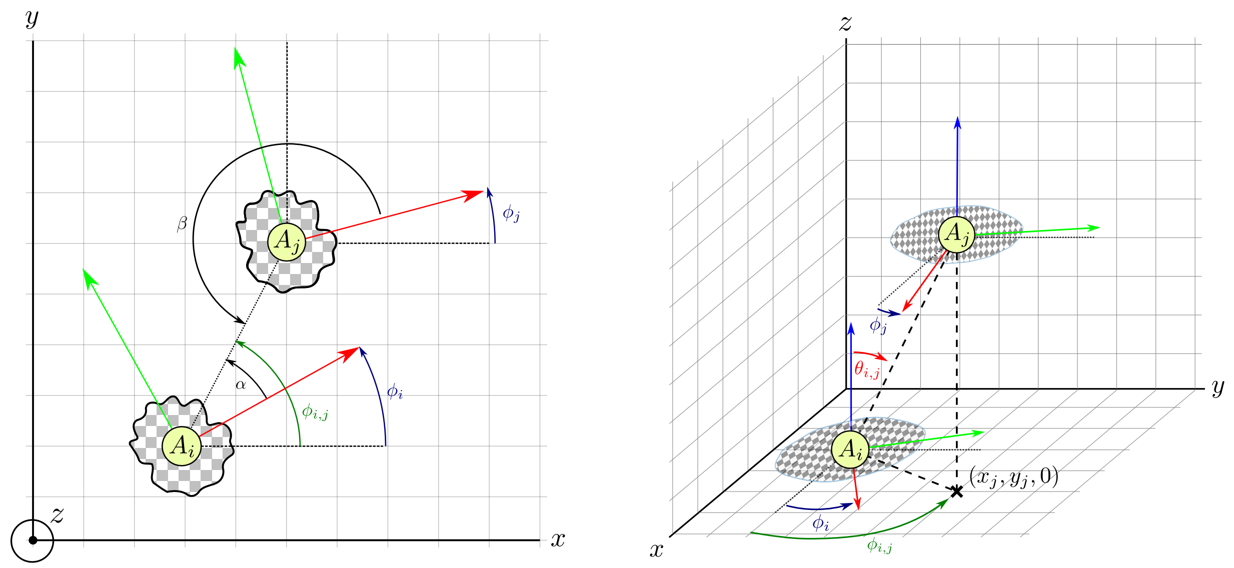


Figure 6.1: Left: Bird view of two agents A_i, and A_j with yaw ϕ_i and ϕ_j. The angles $\alpha = \phi_{i,j} - \phi_i$ and $\beta = \phi_{i,j} + \pi - \theta_j$ are the azimuthal angles at which the radiation patterns are considered. Radiation patterns are represented as checker boarded areas. Right: 3D View of the same agents.

with

$$S_{i,j} = e_j + g_j(\pi - \theta_{i,j}, \phi_{i,j} + \pi - \phi_j) + g_i(\theta_{i,j}, \phi_{i,j} - \phi_i) - C_{i,j}$$

if $S_{i,j} \geq Th$

$$= 0 \text{ otherwise}$$

$S_{i,j}$ represents the received power, at agent A_i, of the signal sent by agent A_j and $\boldsymbol{\phi}$ is the yaw orientation vector giving the yaw orientation of each agent (ϕ_i is the yaw orientation of agent A_i). The scalar e_j represents the transmission power of agent A_j in dBm and the scalar $C_{i,j}$ represents the loss induced by the channel between agents A_j and A_i, in dB. The antenna gains used during the communication between agent A_i and agent A_j depend on their position and their relative orientation. Assuming agent A_i is located at (x_i, y_i, z_i) and agent A_j is located at (x_j, y_j, z_j), we have

$$\theta_{i,j} = \text{atan2}(\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}, z_j - z_i)$$

and

$$\phi_{i,j} = \text{atan2}(y_j - y_i, x_j - x_i)$$

which represent respectively the relative polar and the relative azimuth angles between agents A_i and A_j. These quantities are represented on Figure 6.1. $S_{i,j}$ is a non-null value if $S_{i,j}$ is higher than a given threshold Th representing the minimal signal-to-noise ratio required to receive and decode data.

Finding a solution to this optimization problem involves determining the different agent antenna orientations to optimize the sum of the powers of the received signals in the network. In the next section, we propose a distributed solution in which each agent determines its antenna orientation based on local measurements, without knowing its antenna gain pattern nor the ones of the other agents, or their positions.

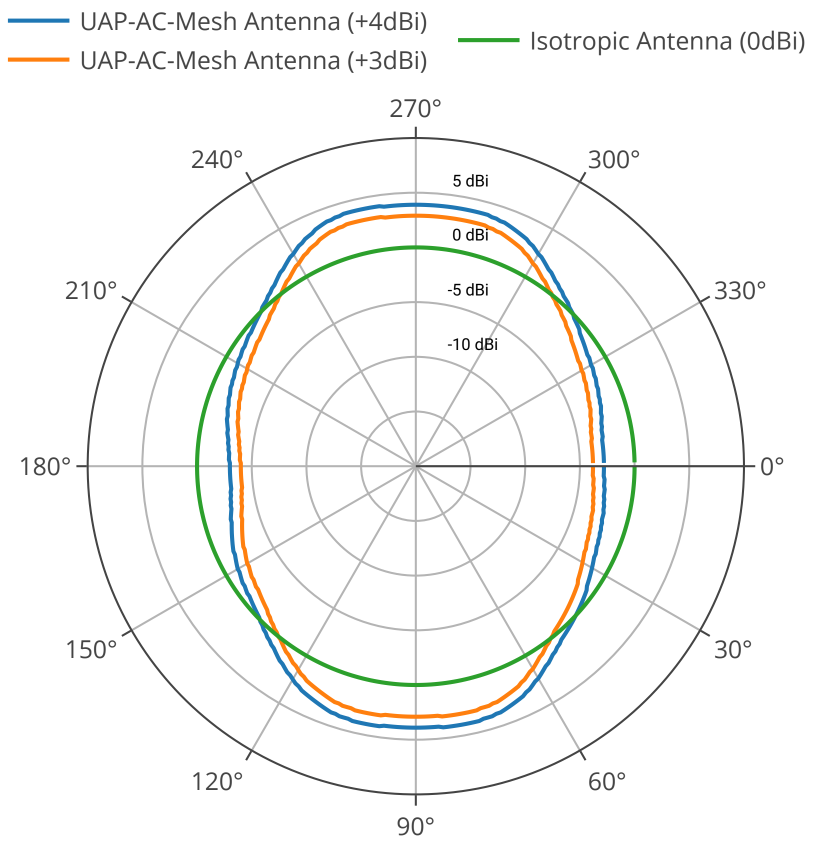


Figure 6.2: Radiation pattern of the antennas used during the simulations for $\theta = 90°$ (horizontal plane), in dBi (decibel relative to the isotropic antenna).

Optimization of Antenna Orientation

As the explicit expression of g is unknown from the agents, the proposed solution will be based on measurements that each agent can carry out. More precisely, agent A_i can measure $S_{i,j}(t)$ at time t if the following conditions are met: agent A_j is transmitting at time t, $a_{i,j} = 1$ and $S_{i,j}(t)$ is bigger than the given threshold Th (for the SNR). When agent A_i carries out such a measurement, it knows its yaw orientation $\phi_i(t)$. These measurements will be stored in a measurement vector M: each agent A_i maintains $M_{i,j} = [m_{i,j,0}, m_{i,j,1}, \ldots, m_{i,j,359}]$ for each agent A_j such that $(i, j) \in E$. The scalar components $m_{i,j,k}$ corresponds to the measurement of the mean received power, at agent A_i, of the signals sent by agent A_j when A_i has a yaw orientation equals to k. Because we are not requiring the knowledge of G and E from the agent A_i, $M_{i,j}$ is created "on the fly" when the connection between A_i and A_j is first established. It is then initialized to $M_{i,j} := [\textsc{None}, \ldots, \textsc{None}]$.

Each agent executes its own algorithm without being synchronized with its neighbors. The proposed algorithm consists of an infinite loop. In each passage in the loop, each agent realizes different steps. First, the agent fetches the frames it has received since the last loop execution, in its network interface queue, and updates its measurement vectors. Then, if the agent lacks some data in its measurement vector with at least one neighbor, it seeks which orientation to move to, to get this measurement. Finally, if its measurement vectors are complete, it tries to optimize its orientation based on their values. Each agent runs the algorithm while it changes its orientation according to online results and while it communicates with its neighbors if required by the data traffic.

Algorithm 1: Antenna Orientation Optimization (agent A_i)

```
$Window \leftarrow 360$
$Count \leftarrow 0$
$Goal \leftarrow None$
while $true$ do
    % Measurement updates
    if $m_{i,j,\lfloor\phi_i(t)\rfloor} == \text{NONE}$ then
        $m_{i,j,\lfloor\phi_i(t)\rfloor} \leftarrow S_{i,j}(t)$
    else
        $m_{i,j,\lfloor\phi_i(t)\rfloor} \leftarrow \text{mean}\left(m_{i,j,\lfloor\phi_i(t)\rfloor}; S_{i,j}(t)\right)$
    % New orientations to explore
    if there exists $k$ such that there exists $j$ such that $m_{i,j,k} = \text{NONE}$ then
        find the $k$ minimizing $|\lfloor\phi_i(t)\rfloor - k|$
        $Goal \leftarrow k$
        if $k == \phi_i(t)$ then
            $Count \leftarrow Count + 1$
        else
            $Count \leftarrow 0$
        if $Count \geq 10$ then
            $Count \leftarrow 0$
            $m_{i,j,k} \leftarrow -100$ for any $j$ such that $m_{i,j,k} == \text{NONE}$
    % Orientation optimization
    if there exists no $j$ or $k$ such that $m_{i,j,k} = \text{NONE}$ and ($Goal$ is NONE or $Goal == \lfloor\phi_i(t)\rfloor$) then
        Find $l$ in $[\lfloor\phi_i(t)\rfloor - Window/2; \lfloor\phi_i(t)\rfloor + Window/2]$ maximizing:
            $$\sum_{j\in[1;N]} a_{i,j} * m_{i,j,l}$$
        $Goal \leftarrow l$
        $Window \leftarrow Window/2$
    % Change of orientation
    Set $d_i$ to reach $Goal$
```

The proposed algorithm is based on the hill climbing approach [84]. We have chosen hill climbing for two reasons: 1) it is an anytime algorithm (it can find better and better solutions as long as it keeps running) and 2) even if it does not guarantee convergence towards a global optimum, it provides an efficient way to find a good solution in a decentralized multi-agent problem. Algorithm 1 describes the algorithm executed by each agent.

Description of the Algorithm As the orientation $\phi_i(t)$ and the power measurement $S_{i,j}(t)$ of the received signal depend on the instant at which these 2 parameters are considered, they are expressed in function of the time t. The $Window$ variable represents the search space. Initially, the search space includes all the possible orientations ($[0;360[$). In order to speed up the algorithm convergence, the size of the space search is divided by 2 as soon as a maximal solution is found in the current space search (line 18 of Algorithm 1). The $Count$ variable represents the maximum number of loop passages during which the agent stays in the same orientation. If the agent stays in a given orientation for too long while trying to fill its measurement vector, the agent considers that it is not a good orientation and sets a very low value to the corresponding measurement element (line 13 of Algorithm 1). The $Goal$ variable represents the orientation the agent is currently trying to reach. In the first loop passages, $Goal$ corresponds to unexplored orientations for which no measurement value has been collected.

Once measurements have been collected for all the orientations and neighbors (line 22), then an optimal orientation (in respect to the defined objective function) can be computed (line 23). Then, the parameter d_i, representing the direction to follow (*i.e.* right, left, or do not move), is updated in order to reach the orientation $Goal$ (line 27). Note that finding an optimal orientation does not mean the end of the algorithm. The search continues with new possible measurements and on a reduced search space (line 25). A new optimal solution can thus be found.

Evaluating this algorithm is a difficult task because the algorithm is distributed and executed in parallel by all the agents in an asynchronous way, but also because it depends on the data traffic, the medium access, the used transmission rates, the channel quality and the agent controller. Moreover, we are interested in the network performance. A dedicated simulation framework has therefore been developed to evaluate the proposed antenna orientation solution in a realistic context.

6.2 Simulation Framework

To simulate antennas and UAVs and to evaluate our proposition, we chose to develop a framework based on the ns–3 network simulator. This development is necessary as existing simulation frameworks do not offer the possibility to both easily simulate UAV controllers and the UAV communications using directional antennas. While ns–3 has some support for antenna modelling, this support is only compatible with Long-Term Evolution networks and not with Wi-Fi networks. Nodes, which represent the physical objects in ns–3, are point-like objects with

Cartesian coordinates, but no orientation coordinates are provided. These two facts together make it difficult to simulate spinning nodes embedding non-isotropic antennas for Wi-Fi networks, at least without rewriting much of the ns–3 models. UAV simulators based on ns–3 such as CUSCUS [96] or FlyNetSim [9] are focused on hardware-in-the-loop, software-in-the-loop or real-time simulations, and do not model the antennas. The developed simulation framework is available as an open source project at GitHub*.

Architecture

The architecture of our framework, as depicted in Figure 6.3, is divided into two main components. The first component is the network simulator ns–3, including the user simulation script or program (bottom left), backed by a custom ns–3 module implementing a *propagation loss model* and a *mobility model* (bottom right). The second component is the discrete-event antenna and UAV simulator, called Phi (top). A third optional component, the visualization front-end, can be plugged into the Phi simulator in order to follow the state of the simulations in "real" (simulated) time.

The goal of Phi is to simulate the behavior and dynamics of multiple UAVs equipped with non-isotropic antennas. Phi provides, according to the antenna orientation, the power gains to use in the ns–3 simulator, simulator that will in turn simulate the UAV network and the networking stack. The controllers and the sensors of the UAVs are therefore modelled by Phi. The simulator has been implemented in C++, which is also the language used by ns–3, but as the interface between Phi and ns–3 relies on message passing, the language could easily be changed.

Communications The two components (ns–3 and Phi) exchange Protocol Buffers messages, used to serialize and deserialize structures, and communicate using the ZeroMQ asynchronous messaging library. Interactions between ns–3 and Phi take the form of two types of messages: Meta messages that are used to set up and control the simulations' life cycle, and Meso messages that concern the simulation itself. The socket connecting the two components uses the ZeroMQ request-reply pattern, ensuring their synchronization, and currently uses a local inter-process communication transport. A third type of message, called Viz, serves as a way to serialize the state of the simulation in order to send it to the visualization front-end.

Simulation Life Cycle When a simulation is set up in ns–3, a Meta message (1) containing the number of agents and their types are sent to Phi's simulation manager component, which will instantiate the simulation, and reply with a simulation ID (2). This simulation ID is included in every subsequent Meta or Meso messages exchanged between the two components and allows a single instance of Phi to be used by concurrent ns–3 simulations. When the simulation ends in ns–3, a Meta message is sent to the Phi's simulation manager to end the simulation and release the resources.

* `Phi` codebase: https://github.com/rgrunbla/Phi.

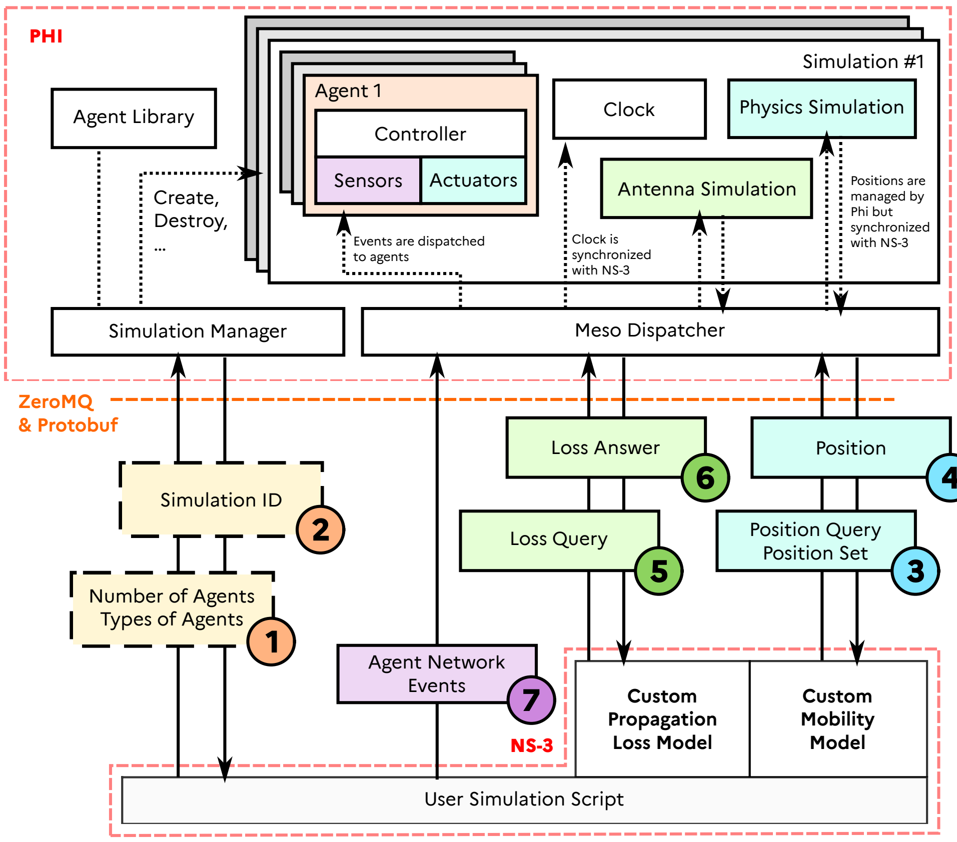


Figure 6.3: High level overview of the simulation framework architecture: main components and control and data paths.

Mobility Model and Physics Engine In ns–3, *mobility models* are in charge of tracking and changing nodes' positions, speeds and accelerations. These quantities are used by the propagation loss and delay models to compute a loss depending on the distance between nodes, and to compute the delay between the transmission and reception of the frames. A custom ns–3 mobility model has been developed, allowing to set the position in Phi from the ns–3 simulation (3), for example at the beginning of a scenario, as well as querying node positions (4), *e.g.* to calculate propagation delays. As the simulator was developed for scenarios where the UAVs have a static position and dynamic orientation, we chose to only consider constant rotation speeds for the UAVs. This allows to model their movements and rotations with simple multiplication operations without having to go through the use of a differential equation solver. This approach is also used by ns–3, which only supports constant speed or constant acceleration mobility models.

Propagation Loss and Delay Model In ns–3, *propagation loss models* and *propagation delay models* are used to model the propagation of the signal between any two nodes, by respectively calculating the signal power and the signal delay. These models can be chained, for example adding a model of Nakagami fading to a free space path loss model leading to a link budget calculation performed by the channel model. The custom module implements a propagation loss model which queries Phi (5) about the gains brought by the antennas of the agents, gains which are sent back to ns–3 (6). Phi does not model any other effect such as the free space loss as one can use the ns–3 models directly. No custom propagation delay model is needed, as this calculation can be done by ns-3 directly by using the positions set by the custom *mobility model*.

Clock Synchronization The clock state of a ns–3 simulation is included into every Meso message sent to Phi, for example in a propagation loss query, or a position query. The only way for a simulation in Phi to advance through time is to receive a Meso message and synchronize its clock with the value it contains. Before the clock update, all the events in the event queue of Phi that are scheduled to occur before the new clock value are executed, with each event being preceded by an update of the environment and agents states.

Agent Simulation and Environment Each agent simulated by Phi is specified by a type and the associated blueprint located in the *Agent Library* component. This blueprint contains the implementation of the controller, of the sensors and the actuators. These components are functions executed at their own frequency using events, *e.g.* 100 Hz for the controller or 10 Hz for a magnetometer. The controller is only capable of interacting with its environment through the use of a shared memory with sensors and actuators, in an asynchronous way. Messages originating from the ns–3 simulation intended for a specific agent are called *Network Events* $\textcircled{7}$ and are placed in a queue in the shared memory. Such messages are for example sent by ns–3 when a frame is received, and contain the frame characteristics, such as the reception power, or the MAC address of the transmitter if applicable.

6.3 Evaluation of the proposed antenna orientation solution

In this section, we present different scenarios used to evaluate the performance of our approach (Algorithm 1). The different scenarios share some parameters, described in Table 6.1, but differ in the number of nodes and their positions. We use the ns–3 Friis propagation loss model, also known as the free-space path loss model, which accurately models the path loss of air-to-air communications between UAVs [95]. All the simulations rely either on an isotropic antenna or a directional antenna whose orientation is regulated by Algorithm 1. The directional antenna represents the Ubiquiti UAP-AC-Mesh Antenna, also named as mesh antenna hereafter, whose radiation pattern is shown on Figure 6.2. The radiation pattern is provided by the constructor on its website [65] as an *ant* file type, covering the horizontal plane with a granularity of 1°. This antenna has been chosen for its small size and weight, making it compatible with airborne applications, as well as its balanced radiation pattern suitable for mesh applications. Two such antennas will be tested: one with a maximal gain of 4 dBi and one with a maximal gain of 3 dBi. Considering a link with two agents equipped with the directional antenna with a 4 dBi gain, 63% of all the possible orientations between the two agents yield a higher gain than a link with two isotropic antennas. On the other hand, if the directional antenna has a 3 dBi maximum gain,

Table 6.1: ns–3 simulation parameters

Simulation Parameter	Value
ns-3 Version	Dev Version (June 2020)
Simulation Duration	100s
Wi-Fi Standard	802.11 ac
Wi-Fi MAC type	Ad-Hoc
Rate adaptation algorithm	Minstrel-HT, Ideal or Intel
Spatial Streams	2x2:2
Channel Width	20 MHz
Antennas	Isotropic or Ubiquiti UAP-AC-Mesh
Propagation Loss Model	Friis
Routing	Static
Application	OnOff (constant bitrate, UDP)
UAV Rotation Speed	0 rad/s, ±0.50 rad/s
Controller Frequency	100 Hz
Magnetometer Frequency	10 Hz

only 38% of the possible orientations between the two agents result in a higher gain, compared with two isotropic antennas. While the high gain will obviously result in higher network performance than with the isotropic antenna, as long as the $\phi = 0°$ or $\phi = 180°$ positions are not used, we believe the results are still of interest in order to study the dynamics of the algorithm 1 and to illustrate the gains such a system can provide. We also want to check that our proposed solution does not converge to orientations leading to worse performance than with omnidirectional antennas.

As in the previous chapters, three rate adaptations algorithms, named Minstrel-HT, Intel and Ideal, have been considered. Our evaluation will thus also study the impact of these three algorithms on the performance of our solution.

We present the obtained results with the 4dBi antenna in the next three subsections, and we then present the results obtained with the 3dBi antenna. The initial orientations of the nodes are distributed uniformly over $[0; 2\pi]$, and each simulation is repeated 20 times with different initial orientations. Several results are reported with the box plot representation. In this case, the lower side of the rectangle represents the first quartile, $Q1$, the upper side represents the third quartile, $Q3$, the plain line represents the median, Q2, while the dotted line represents the mean value. The extreme lines (outside the rectangle) represents the lower and upper fences, computed as 1.5 times the inter-quartile range below Q1 and above Q3. Any value outside the range $[4 * Q1 - 3 * Q3; 4 * Q3 - 3 * Q1]$ is considered as an outlier, and represented as a point.

Scenario #1: Simple

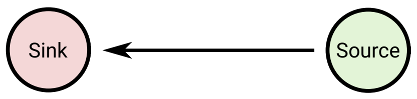


Figure 6.4: Simple Scenario: The source and the sink are separated by a fixed distance, and their position does not change during a simulation (but their orientation might).

In this scenario, two nodes are separated by a fixed distance, with one node acting as a source and one node acting as a sink (receiver), as shown on Figure 6.4. The two nodes are either both equipped with

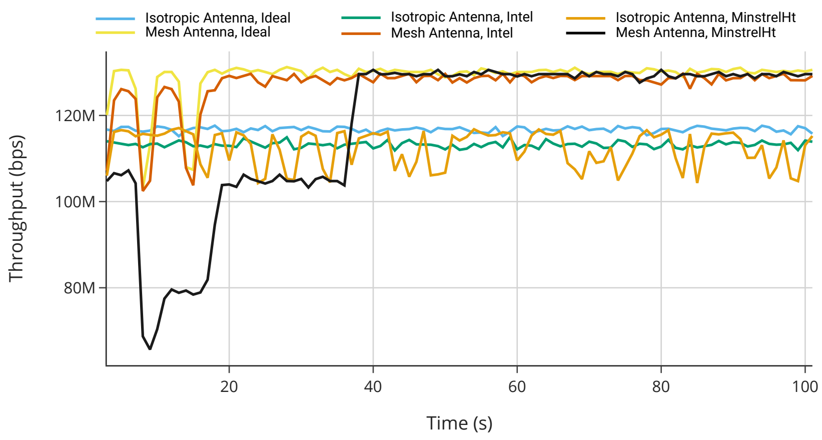


Figure 6.5: Evolution of the application throughput in function of time for Scenario #1, with 2 nodes 100 m apart and a saturating UDP application rate of 180 Mb/s.

omnidirectional antennas, in which case Algorithm 1 is not used, or both equipped with directional antennas using our antenna orientation algorithm. The throughput of the source is set to 180 Mb/s, which exceeds the maximum physical throughput for the Wi-Fi physical layer parameters used in the simulation, which is 173.3 Mb/s. The received throughput at the sink is plotted on Figure 6.5 as a function of time, rate adaptation algorithm, and the used antenna, for a single simulation and when the distance between the two nodes is $100m$. We can observe that the three rate adaptation algorithms are almost the same when an omnidirectional antenna is used, and the received throughput remains stable throughout the simulation (with some variations with Minstrel-HT). When the directional antenna is used, we observe two main phases. The first phase, where the throughput varies a lot corresponds to the execution of the antenna orientation algorithm: as the channel between the two nodes changes, the rate adaptation algorithms react and change the transmission rates, affecting the received throughput. The second phase starts after the antenna orientation algorithm has converged to its best solution in terms of received power. The received throughput remains fairly stable during this phase as the only source of change is the RAA decisions. We can however observe that when Minstrel-HT is used, it takes more time to reach the stabilized received throughput, which is consistent with our previous observations about the algorithm.

The convergence time for the antenna orientation algorithm and the convergence time on the received throughput for the simulations using the directional antenna are plotted on Figure 6.6 (with the box plot representation). The convergence time for the antenna orientation algorithm is the elapsed time between the start of the algorithm and the time when the last agent stops to change its orientation. The convergence time on the received throughput is the elapsed time between the start of the algorithm and the time when the received throughput on the sink is different to at most 5% of the final achieved received throughput. We can note that the convergence time of our algorithm is always smaller than 20s in Scenario #1. The convergence time on the received throughput is also smaller than 20s for Ideal and Intel, and it is never larger than 40s with Minstrel-HT. We can observe a strong correlation between the two quantities for the Ideal and the Intel RAAs, which underlines those algorithms are fast to react to changes in the channel, while the throughput convergence time of Minstrel-HT illustrates the inertia of the algorithm, which can be linked to its sampling approach. Using physical layers metrics, such as signal strength, and not application layer metrics, such as the received throughput, appears therefore justified for such an algorithm, as higher layer metrics may introduce important delay with certain RAA.

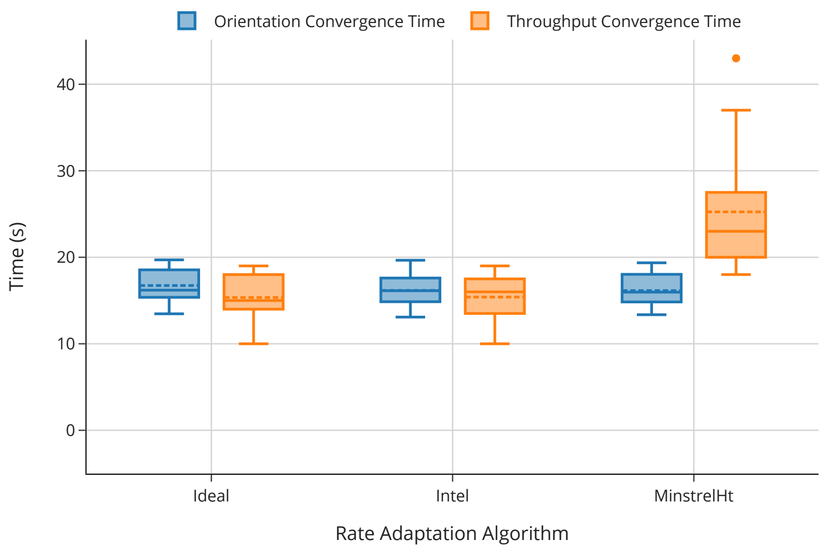


Figure 6.6: Comparison of the convergence time for the antenna orientation algorithm and application throughput for Scenario #1 at $d = 100m$.

Figure 6.7 shows the distribution of the achieved throughput for Scenario #1 when the two nodes are $100m$ away. The achieved throughput is measured when the antenna orientation has converged. The obtained results show that our antenna orientation solution improves the achieved throughput whatever the used RAA. For instance, with the Ideal RAA, the mean achieved throughput is 144.2 Mb/s with directional antennas compared to 116.8 Mb/s with omnidirectional antennas, whereas it is 136.9 Mb/s with directional antennas compared to 113.2 Mb/s with isotropic antennas for the Intel RAA. For Minstrel-HT the use of directional antenna with our orientation algorithm leads to 137.4 Mb/s compared to 111.5 Mb/s with omnidirectional antennas. We analyzed the antenna orientations obtained when our algorithm has converged for the different simulation repetitions and for the different RAAs. The values obtained on the antenna orientations vary but are mainly scattered on good positions as 95% of the achieved orientations lead to a better link budget than with isotropic antennas. These orientations lead to better link qualities which also lead to a use of higher transmission rates, which, at the end, results in higher achieved throughputs. Finally, one can note that the obtained values on the throughput are more dispersed with directional antennas than with isotropic antennas. This is explained by the fact that the obtained orientations vary, which results in different link budgets implying more various throughputs, though these latter are, most of the time, better than the ones obtained with isotropic antennas.

Scenario #2: Sink

In this scenario, one node serves as a sink while other nodes serve as sources. The sources are located on a circle with a fixed radius r, while the sink is located at the center of the circle, as shown on Figure 6.8. The sink can be seen as a UAV receiving video feeds from the sources, and sending them to the ground using another network component not studied here. The sink is equipped with an isotropic antenna. We

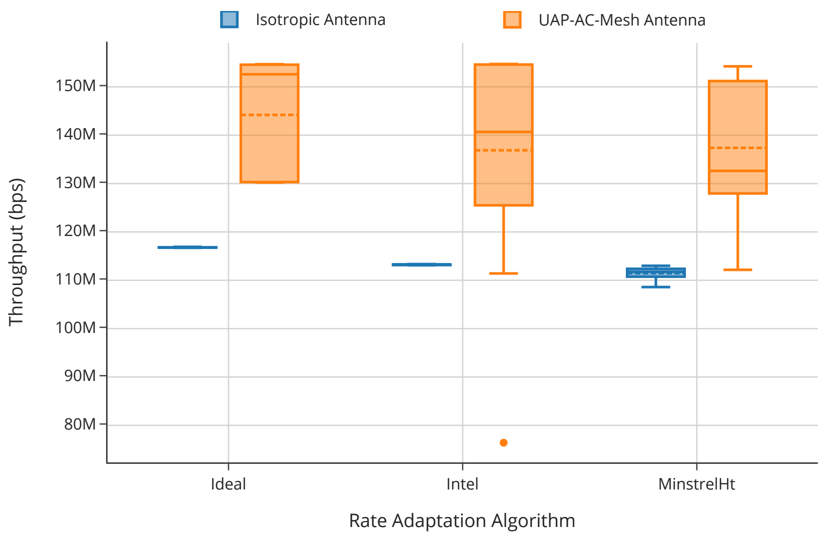


Figure 6.7: Comparison of the achieved throughput for Scenario #1 at $d = 100m$.

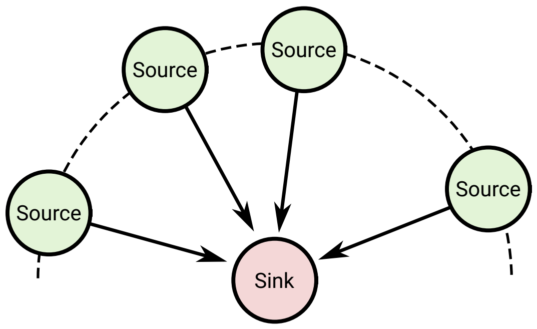


Figure 6.8: Sink Scenario

have observed, on the different simulations, that the antenna orientation algorithm converges in less than 30 s. The distribution of the average received throughput per link, at the sink, for a radius of 100 m, for 4 sources, and for different application rates at the source, is shown on Figure 6.9. One can observe an increase in the obtained throughput when using the directional antenna, no matter which RAA is used. The increase is more limited with Minstrel-HT. As 100% of the simulations obtain a better link budget than with an omnidirectional antenna, this more limited improvement can be explained by the larger time needed for Minstrel-HT to converge towards the final throughput when the antenna orientation has ended, leading to a smaller throughput than with Ideal and Intel. When the application rate is low enough, *e.g.* 10 Mb/s, it can be fulfilled by both the isotropic and the directional antennas in any direction, leading to very similar obtained throughput.

We have measured whether the different sources are receiving a fair "share" of the received throughput at the sink or not with the Jain's fairness index [47]. The Jain's fairness index can be used to measure whether the different sources are receiving a fair "share" of the received throughput at the sink or not. If we denote by $t_1, \ldots, t_n$ the throughput received at the sink from the agents $A_1, \ldots, A_n$, then the index can be defined as:

$$J(t_1, \ldots, t_n) = \frac{(\sum\limits_{i \in \{1,\ldots,n\}} t_i)^2}{n \cdot \sum\limits_{i \in \{1,\cdot,n\}} t_i^2}$$

This index takes values between $1/n$, the worst case representing an

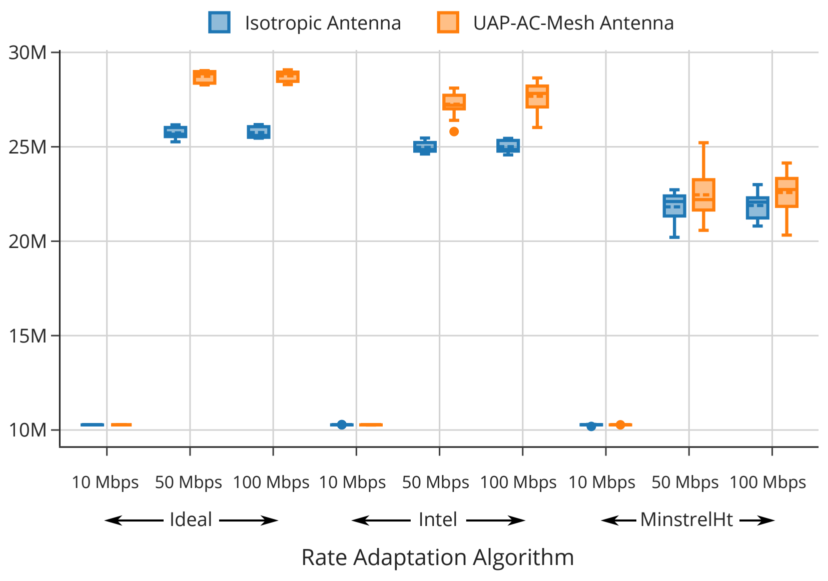


Figure 6.9: Comparison of the average received throughput per link for Scenario #2 with $r = 100$ m and 5 nodes.

unfair share of the resources, and 1, the best case, representing a fair share of the resources. The results for $r = 100m$ and $n = 3, 5, 10$, and an application throughput of $50Mb/s$ are shown on figure Table 6.2. The results obtained for $r = 100m$ and $n = 3, 5, 10$, and an application throughput of 50 Mb/s show that the use of the mesh antenna does not decrease the fairness between the nodes with rather a slight increase of the Jain's index values (between 0.94 to 1 for directional antennas compared to 0.92 to 1 with isotropic antennas). Overall, the use of the mesh antenna does not decrease the fairness between the nodes.

Scenario #3: Chain

In this scenario, one node serves as a source, one node serves as a sink, and the other nodes serve as relays between the source and the sink as depicted on Figure 6.11. The source and the sink are separated by a fixed distance d, and the relays are equidistantly placed between them.

We plot the distribution of the received throughput at the sink on Figure 6.12, for a distance between the source and the sink of $d = 1000$ m, and for 5 and 10 nodes in total, that is to say for respectively 3 and 8 relays, for an application throughput of 50 Mb/s. While for a chain of 5 nodes, the use of the directional antenna with the antenna orientation algorithm improves the overall throughput, for any RAA, no improvement is observed for a chain of 10 nodes with the Intel and Minstrel-HT RAAs.

We observe that with 5 nodes, while the percentage of failed MAC transmissions at the source is higher with the directional antenna, the frame transmission rate also increases, leading to lower air-time per frame, allowing more frames to be exchanged, as shown in Table 6.3. This property is also verified on the different links of the chain. This results in higher throughput with directional antennas than with omnidirectional ones. On the other hand, with 10 nodes, a too high number of retransmissions has been observed whatever the used antenna, leading to low throughput in both cases. One can also note that Intel exhibits poor performance, in this scenario, compared to Ideal and Minstrel-HT.

Number of nodes	Wi-Fi Manager	Antenna Type	Mean	Standard Deviation
3	Ideal	Isotropic	1.00	0.00
		Mesh	1.00	0.00
	Intel	Isotropic	1.00	0.00
		Mesh	1.00	0.00
	Minstrel-HT	Isotropic	1.00	0.00
		Mesh	1.00	0.00
5	Ideal	Isotropic	0.92	0.06
		Mesh	0.94	0.07
	Intel	Isotropic	0.93	0.06
		Mesh	0.94	0.07
	Minstrel-HT	Isotropic	0.94	0.04
		Mesh	0.94	0.05
10	Ideal	Isotropic	0.93	0.07
		Mesh	0.95	0.04
	Intel	Isotropic	0.95	0.03
		Mesh	0.96	0.03
	Minstrel-HT	Isotropic	0.92	0.05
		Mesh	0.95	0.04

Table 6.2: Jain Index mean and standard deviation for the scenario #2, with an application throughput of 50Mb/s and at distance d=100 m, across 20 experiments.

Antenna Type	Rate Adaptation Algorithm	Transmission Failure	Transmission Number
Isotropic	Ideal	16.3%	203.1k
	Intel	21.2%	180.3k
	Minstrel-HT	21.0%	187.7k
Mesh	Ideal	24.5%	311.6k
	Intel	19.0%	230.0k
	Minstrel-HT	25.7%	278.8k

Table 6.3: Comparison of the number of MAC transmission and percentage of transmission failure at the source in Scenario #2, with 5 nodes and 50Mb/s of application rate.

It can be explained by the conservative behavior of the Intel RAA when too many retransmissions are triggered.

Results with a 3 dBi antenna

All the previous results were obtained with an antenna whose maximum gain was 4 dBi. We also evaluated our solution with the same pattern radiation but with a maximal gain reduced to 3dBi (see Figure 6.2). With such an antenna, only 38% of the possible antenna orientations between two agents will result in higher network performance than with the isotropic antenna.

Figure 6.13 reports the distribution of the achieved throughput between two nodes (Scenario #1) separated of 100 m and when the source throughput is 180 Mb/s for 20 repetitions of the simulation with different initial orientations. The results show that even if the achieved throughputs are smaller than the ones obtained with a 4dBi antenna, the antenna orientation algorithm is able to find good orientations resulting in better performance than with omnidirectional antennas. The reduction of the performance gain, when switching from a 4dBi antenna to a 3 dBi antenna is more pronounced for Intel and Ideal than for Minstrel-HT.

Figure 6.14 reports the distribution of the average achieved throughput per link for Scenario #2 with 5 nodes and each source transmitting with an application rate of 50 Mb/s. The results show that the achieved

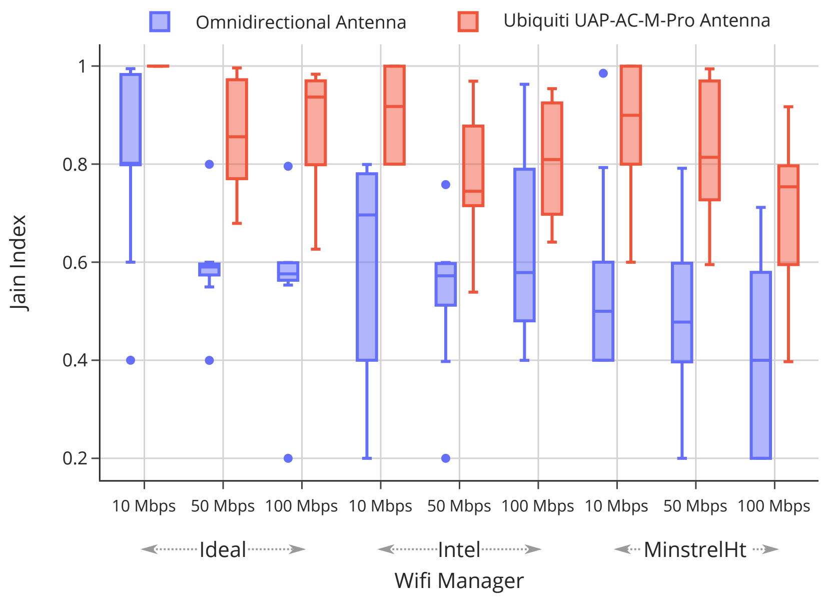


Figure 6.10: Distribution of the Jain's fairness index for r = 300 m, 5 nodes, and 20 random initial position and orientations.

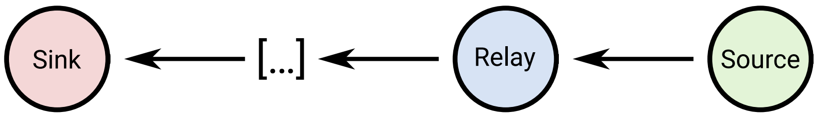


Figure 6.11: Chain Scenario

throughputs are not so different from the ones obtained with a 4dBi antenna, even if the results with Minstrel-HT are a bit smaller than with the 4 dBi antenna.

Figure 6.15 reports the distribution of the average achieved throughput per link for Scenario #3 with 5 nodes on a chain whose source and sink are separated by 1000 m. The source transmits with an application rate of 50 Mb/s. The results show that the achieved throughputs are reduced compared to the ones obtained with a 4dBi antenna. Intel is the less impacted by this reduction in the performance gain, even if the mean achieved throughput is smaller with Intel than with Ideal and Minstrel-HT. We can also note that the obtained results are more scattered with Ideal and Minstrel-HT than with Intel. This dispersion is very important for Minstrel-HT, leading to smaller results than with an omnidirectional antenna for a set of simulation repetitions.

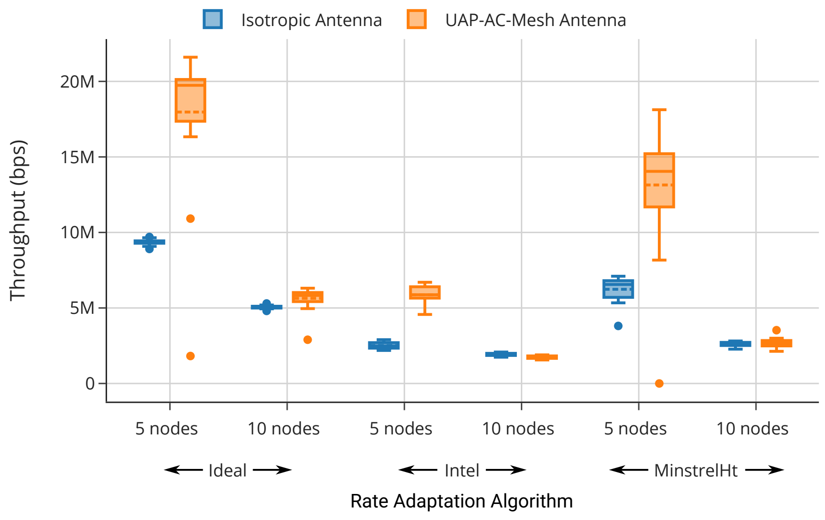


Figure 6.12: Comparison of the average received throughput at the sink for Scenario #3 with $d = 1000$ m, 5 and 10 nodes.

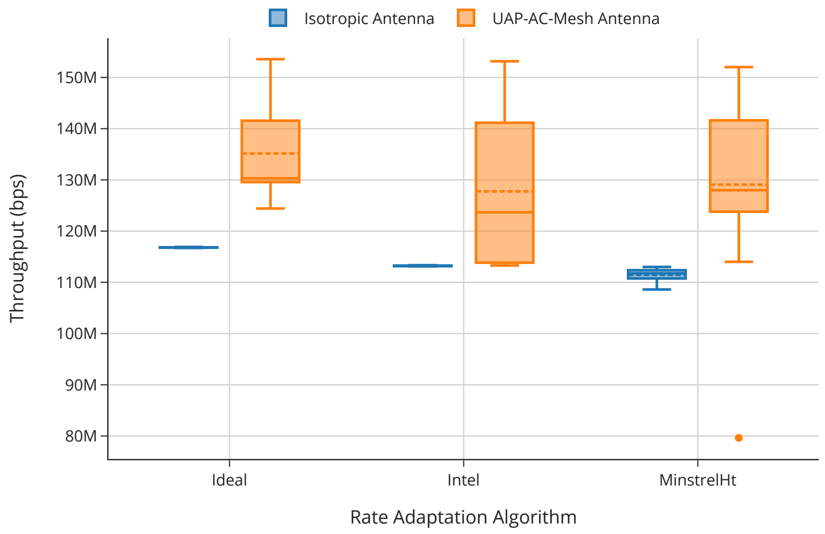


Figure 6.13: Comparison of the achieved throughput for Scenario #1 at $d = 100m$ over 20 random initial orientations with the 3 dBi directional antenna.

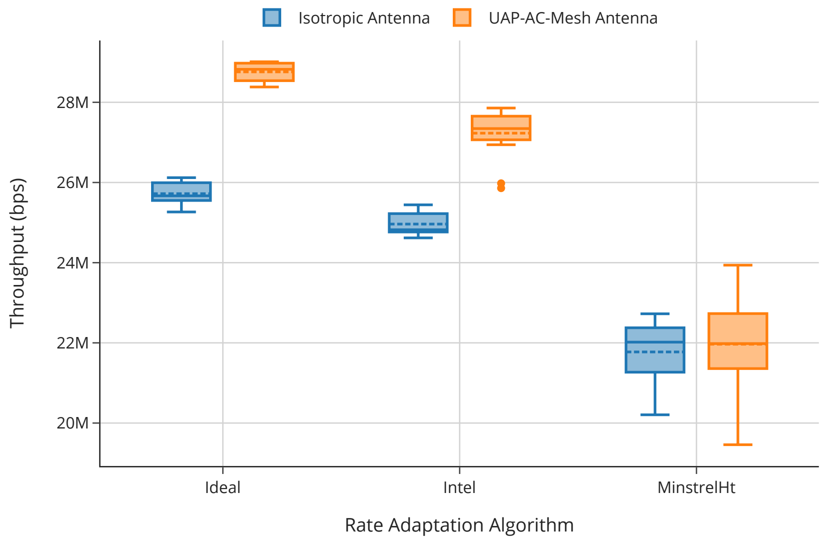


Figure 6.14: Comparison of the average received throughput per link for Scenario #2 with $r = 100$ m, 5 nodes, and 20 random initial position with the 3 dBi directional antenna.

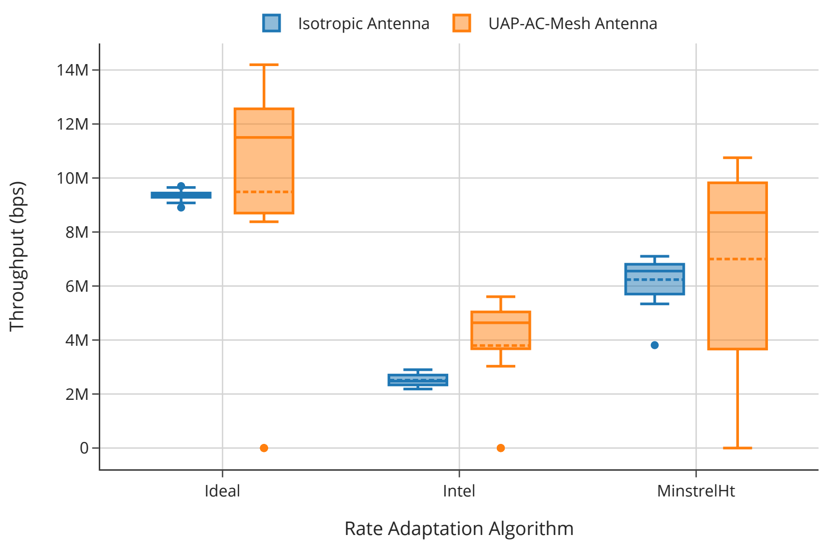


Figure 6.15: Comparison of the average received throughput at the sink for Scenario #3 with $d = 1000$ m, 5 nodes, and 20 random initial orientations with the 3 dBi directional antenna.

6.4 Conclusion

The antenna radiation pattern is clearly an important factor when looking at the performances of a communication network. While Wi-Fi simulations in ns–3 only support isotropic antennas, in reality, perfect isotropic or even omnidirectional antennas do not exist, which means some progress can be made in this regard in the simulator. Is it possible to take advantage of the non isotropism of the antenna radiation pattern, for drone networks ? By taking into account the antenna radiation patterns in ns–3, thanks to a customized simulation framework, we illustrated that it is possible for a Wi-Fi based drone network to exhibit some gain in performance, in a few simulation setups, with a relatively simple optimization algorithm. We believe this algorithm can be used for many drone applications, as it does not imply changing the position of the drones, does not need any synchronization or centralization. Indeed, the relative orientation of the drones (for multicopters) can be controlled without modifying their 3D position, and by only changing their own orientation. a priori knowledge of the radiation pattern of the antennas is not needed, nor knowledge of the peers positions. While we observed improvements in terms of performances for the drone networks, it depends on the underlying rate adaptation algorithms used by the WNIC, as well as the radiation patterns of the antennas. While this conclusion is limited to the studied simulation scenarios and the considered simulation environment, we believe that, given small modifications of the orientation algorithm, the conclusions would translate well in the real world.

Conclusion 7

The initial goal of this thesis was to explore controlled mobility for Wi-Fi based drone networks. Retrospectively, most of the three years have been focused on sub-problems of controlled mobility one could consider as side problems, but are, in our opinion, prerequisites. Indeed, how can one claim to try to take advantage of mobility when its effects are not well understood, and when the simulations do not take into account certain aspects related to this mobility? New results bring new problems, and we feel many doors were opened while only a few were closed.

The contributions of this thesis can be summarized into several main axes:

1. We improved the understanding of rate adaptation algorithms used in many real-world devices, by looking at the behavior of Intel devices and reverse engineering their rate adaptation algorithm, as well as implementing it in the popular network simulator ns–3.
2. We studied the relationship between performances in terms of throughput, rate adaptation algorithms, and the nodes' mobilities, speeds and positioning. The different overall behavior of Minstrel-Ht and Iwl-Mvm-Rs, two rate adaptation algorithms used in real devices, was illustrated, which underlines that studying those algorithms is important when dealing with mobile nodes, such as drones.
3. We illustrated how drone mobility can help to improve network performances by looking at the effect of drones antenna radiation patterns in ns–3 and devising a mobility algorithm to take advantage of the non isotropism of the antennas.

Initially, our approach of the thesis topic was an experimental one. During the first months of the thesis, after some initial state of the art, our main goal became to experiment using drones and Wi-Fi networks to be rooted in the reality on the ground. First tests were conducted using a Parrot Bebop recreational quadcopter that we had to equip with an embedded system supporting Wi-Fi, small and light enough it could be retrofitted on the recreational drone. This task was complicated by the fact we wanted the hardware to use moderately recent amendments, e.g. 802.11ac, and possess at least 2 antennas for transmission and reception. Finding hardware open enough one can install and run an operating system like Linux was the first main difficulty in this thesis, and remained so thereafter. The lack of open, hackable hardware is in this regard detrimental to academia and experimentation with Wi-Fi networks. We then moved very quickly to another drone model, Intel Aero, equipped with an onboard computing board itself equipped with a suitable Wi-Fi card.

Our initial results were disconcerting. Focusing on the application throughput obtained between a laptop, the source, and the drone-mounted access point, the sink, throughput values seemed to change randomly, with performance spikes and troughs, even without much movement, on the ground. This was quickly linked to spikes and troughs in the transmission rates used by the laptop, which used an Intel card, like the new Intel Aero drones we were going to be experimenting with in the following months. The second main difficulty was therefore to understand the operation of Intel's rate adaptation algorithm, which led to a big part of the work done during this thesis. Implementing the algorithm in the ns-3 simulator also started a shift towards simulation for the rest of this thesis. This implementation allowed us to highlight how important are rate adaptation algorithms in the context of drone mobility, and allowed us to better understand the ns–3 simulator, which, although state of the art in Wi-Fi network simulation at our academic level, can be improved in many ways.

Understanding how ns–3 works, and understanding its results, could be considered as the third main difficulty we tackled during this thesis. While widely used, ns–3 is plagued with many bugs which can be hard to discover. This is in part due to the design of the simulator which tends to break silently, which means the reason behind pretty much any simulation results need to be thoroughly understood. Doubting any ns–3 simulation result still seems to be the best attitude to adopt, and while bugs still exist in the simulator, we can hope their number is lower now than in the past.

Last, but not least, conducting real experiments, whether they involve a few Wi-Fi devices in a corridor or flying objects that weigh about a kilogram is not an easy task. We expected to start experimenting again using real drones, outside, during the first part of 2020. Difficulties in the control of the drone at our disposal had been reduced, and we expected to start airborne measurement campaigns of the state of the Wi-Fi spectrum, as well as other experiments. A few test flights had even been carried out. The course of this year will have decided otherwise. While it is already easy to shoot yourself in the foot in simulation, it is somehow easier with experimentation, which is also more time-consuming. This is still necessary, and while our controlled mobility solution seems promising in simulation, without an experiment to actually test it in real conditions, it is not possible to verify if this is indeed the case. Our developed framework to study the effect of the antennas' radiation patterns does not claim to be a long-term solution for the study of this class of problems, but more of a demonstrator.

Perspectives

Intel RAA Regarding Iwl-Mvm-Rs, we can identify a few extensions to our work that would be welcome:

- Currently, the Intel rate adaptation algorithm is only suitable to very specific hardware, namely the hardware sold by Intel. Its general applicability is therefore limited. Re-using the techniques employed in the original algorithm and designing a more generic RAA to be used primarily in simulation would allow to make it suitable as an alternative for Minstrel-Ht, which is currently the only other RAA used both in simulations (in ns–3) and in the real world supporting recent standards.

- As a *living* algorithm, which undergoes regular changes, the Iwl-Mvm-Rs ns–3 implementation needs to be maintained in sync with the kernel implementation. In a perfect world, its implementation in simulation would also be verified against the "hardware" implementation. Whereas this was possible until recently to do so by *just* reading the Linux kernel source code or instrumenting it, since the 802.11ax / Wi-Fi 6 amendment, the algorithm has moved from the Linux kernel to the Intel chipsets closed source firmware, which makes those tasks harder. The best way to achieve this goal is probably to compare simulator and hardware traces, received *over-the-air*, to verify they agree with each other. This assumes it is possible to study the behavior of the hardware in a sufficiently controlled environment to avoid outside interferences to create behaviors hard to recreate in simulation.

This latest point can also be made for other RAA, such as Minstrel-Ht, whose implementation in ns–3 has been found to be wrong during the writing of this thesis, and this since its first introduction in the simulator.

Simulations Setups The simulation *setups* in which we studied our controlled mobility solution and the performance of the different RAAs were all *custom* setups we chose because they seemed interesting. Whereas in many fields, benchmarks exist, allowing to quickly evaluate the performance of some proposed solution, Wi-Fi networks research mostly rely on specific and custom simulation scenarios. While one could say such scenarios are simple enough they can be recreated at will in this or that simulator, it is, in our opinion, a waste of time and the progress made by each new work is difficult to quantify. A library of well-defined benchmarks, with common implementations in simulators like ns–3 or OMNeT++ / INET, would allow researchers to compare their works more easily, and better evaluate their results. In particular when dealing with mobility, where dozens of mobility models parametrized by continuous parameters exist, common scenarios deemed important for Wi-Fi research (and wireless research in general) should be made easier to use.

Such scenarios could also be used to gauge the quality of network simulators, compare them with each other, identify the impact of the global evolutions they undergo, and find bugs more quickly. In particular, we found many bugs in ns–3 during this thesis, without even actively trying to find them, just by simulating some moderately complex scenarios, and observing *weird* behaviors. We believe more extensive testings would catch many behaviors we observed, and more generally, more verification and validation are needed for such projects on which much depend.

Controlled Mobility and Robot Networks Today, the simulation of robot networks, including drone networks, is in a gray area. Network simulators seem pretty good at simulating network protocols and part of the networking stacks of modern operating systems. Robotic simulators seem pretty good at simulating the mobility, dynamics, control of the robots. Some hybrid simulators exist, trying to rely on both robotic and network simulators, but they are the subject of even less "technical attention" from the community than the components they bridge. Several attempts have been made to create these kinds of bridges, which underline their importance, but without a continuous technical support and development they tend to become slowly unusable. Looking specifically at ns–3, we believe a few key changes in the simulator could make it easier to interface with robotic simulators. Such changes include transitioning to better models for the nodes, which are currently only modeled as dimension-less, point-like objects. Such changes are also of interest for the simulation of antennas, another weak point on ns–3.

Our proposition of controlled mobility to take advantage of the antenna's radiation patterns needs more evaluation, in more diverse scenarios, in particular including global mobility for the nodes. Other approaches at the distributed optimization of the antennas' orientation should also be explored, e.g. machine learning techniques, as some works already proposed in the context of drone networks. This, of course, depends on correctly simulating mobility in the chosen network simulator.

Concluding Remarks

The complexity of Wi-Fi networks leads us to believe that studying Wi-Fi networks has to be rooted in some level of experimentation. Relying only on the IEEE 802.11 standards as they are written and on simulation based on them is not enough as the standard does not cover many important aspects, such as rate adaptation. Between the state of what is implemented in simulation, what is implemented in the Linux kernel, what is implemented in Wi-Fi network interface controllers, and what is described in the academic literature, gaps exist. We are not certain they are narrowing. This is not made better by the inability of hardware vendors to respect a standard a few thousands page long changing every 5 years or so, whether they are responsible for it.

Of course, experimentation is not enough. The hardware on which we experimented during this thesis was closed hardware, and the conclusions of pretty much any experimentation involving closed hardware whose behavior is, in some part, not understandable, are limited. Simulators like ns–3 aim to abstract this hardware layer, but they are of course imperfect. In particular, we feel that ns–3, which to us is the best shot at the simulation of Wi-Fi networks, lacks the workforce to stay relevant and up-to-date with the current "hardware state-of-the-art".

There is, however, room for hope. More and more experimentation platforms lower the entry cost in terms of complexity for hardware platforms, and allow for greater reproducibility of Wi-Fi experiments. Open Wi-Fi transceiver based on software defined radios, like OpenWiFi, are also in development. This may open up the possibility for academics to change low levels parameters, currently reserved to hardware vendors.

Bibliography

[1] Evan Ackerman and Eliza Strickland. 'Medical delivery drones take flight in east africa'. In: *IEEE Spectrum* 55.1 (2018), pp. 34–35 (cited on page 18).

[2] *ACM: Artifact Review and Badging Version 1.1*. 2020. URL: https://www.acm.org/publications/policies/artifact-review-and-badging-current (cited on page 36).

[3] ADSBexchange.com LLC. *World's largest co-op of unfiltered flight data*. 2020. URL: https:///www.adsbexchange.com/ (visited on 08/21/2020) (cited on page 21).

[4] *AERPAW: Aerial Experimentation and Research Platform for Advanced Wireless*. 2020. URL: https://aerpaw.org/ (cited on page 36).

[5] Hugues Anguelkov. *Reverse-engineering Broadcom wireless chipsets*. 2019. URL: https://blog.quarkslab.com/reverse-engineering-broadcom-wireless-chipsets.html (cited on page 49).

[6] Ardupilot Dev Team. *SITL Simulator (Software in the Loop)*. 2016. URL: https://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html (cited on page 34).

[7] Mahdi Asadpour, Domenico Giustiniano, and Karin Anna Hummel. 'From ground to aerial communication: Dissecting WLAN 802.11 n for the drones'. In: *Proceedings of the 8th ACM international workshop on Wireless network testbeds, experimental evaluation & characterization*. 2013, pp. 25–32 (cited on pages 5, 26).

[8] Mahdi Asadpour, Domenico Giustiniano, Karin Anna Hummel, and Simon Heimlicher. 'Characterizing 802.11n Aerial Communication'. In: *Proceedings of the Second ACM MobiHoc Workshop on Airborne Networks and Communications*. ANC '13. Bangalore, India: Association for Computing Machinery, 2013, pp. 7–12. DOI: 10.1145/2491260.2491262 (cited on pages 5, 26).

[9] Sabur Baidya, Zoheb Shaikh, and Marco Levorato. 'FlyNetSim: An Open Source Synchronized UAV Network Simulator based on ns-3 and Ardupilot'. In: *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. ACM. 2018, pp. 37–45 (cited on pages 34, 89).

[10] Constantine A Balanis. *Antenna theory: analysis and design*. John wiley & sons, 2016 (cited on page 84).

[11] *BARKHANE : Engagement de la composante aérienne dans une vaste opération*. French. June 2020. URL: https://www.defense.gouv.fr/fre/actualites/operations/barkhane-engagement-de-la-composante-aerienne-dans-une-vaste-operation (cited on page 14).

[12] Bay Area Compliance Labs Corp. *Japan MIC Test Report For SZ DJI Technology Co.,LTD*. Sept. 2018. URL: https://www.tele.soumu.go.jp/giteki/file?AFK=211_N_1_181228N211_%94F%8F%D8_152_*****_*****&AFT=2&AFN=2 (cited on page 23).

[13] Émile Bayard, Yan Dargent, A. Férat, Alexandre Ferdinandus, Jacques Guiguet, Frédéric Lix, Parent., Paul Dominique Philippoteaux, H. Rousseau, Roux., Félix Thorigny, and Valnay. *Album de la science : savants illustres, grandes découvertes*. French. 1897 (cited on page 13).

[14] B. Bloessl, M. Segata, C. Sommer, and F. Dressler. 'Performance Assessment of IEEE 802.11p with an Open Source SDR-Based Prototype'. In: *IEEE Transactions on Mobile Computing* 17.5 (2018), pp. 1162–1175 (cited on pages 44, 49).

[15] E. Borgia. 'Experimental evaluation of ad hoc routing protocols'. In: *Third IEEE International Conference on Pervasive Computing and Communications Workshops*. 2005, pp. 232–236 (cited on page 21).

[16] Pierre Brunisholz, Franck Rousseau, and Andrzej Duda. 'Anatomie des retransmissions dans les implémentations de 802.11'. French. In: *Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication (CORES 2018)*. Roscoff, France, May 2018 (cited on page 11).

[17] Hal Chamberlin. *Musical Applications of Microprocessor*. Sams, 1985 (cited on page 69).

[18] Serge Chaumette. 'Collaboration Between Autonomous Drones and Swarming'. In: *UAV Networks and Communications*. Ed. by Kamesh Namuduri, Serge Chaumette, Jae H. Kim, and James P. G.Editors Sterbenz. Cambridge University Press, 2017, pp. 177–193. DOI: 10.1017/9781316335765.009 (cited on page 4).

[19] J. Chen, J. Xie, Y. Gu, S. Li, S. Fu, Y. Wan, and K. Lu. 'Long-Range and Broadband Aerial Communication Using Directional Antennas (ACDA): Design and Implementation'. In: *IEEE Transactions on Vehicular Technology* 66.12 (2017), pp. 10793–10805 (cited on pages 5, 29, 30).

[20] C. Cheng, P. Hsiao, H. T. Kung, and D. Vlah. 'Performance Measurement of 802.11a Wireless Links from UAV to Ground Nodes with Various Antenna Orientations'. In: *Proceedings of 15th International Conference on Computer Communications and Networks*. 2006, pp. 303–308 (cited on page 25).

[21] *CityLab: The City of Things Smart Cities FIRE Testbed*. 2018. URL: https://doc.lab.cityofthings.eu/wiki/Main_Page (cited on page 35).

[22] Edison Pignaton De Freitas, Tales Heimfarth, Ivayr Farah Netto, Carlos Eduardo Lino, Carlos Eduardo Pereira, Armando Morado Ferreira, Flávio Rech Wagner, and Tony Larsson. 'UAV relay network to support WSN connectivity'. In: *International Congress on Ultra Modern Telecommunications and Control Systems*. IEEE. 2010, pp. 309–314 (cited on page 18).

[23] S. Derek. *Minstrel*. http://madwifi-project.org/browser/madwifi/trunk/ath_rate/minstrel/minstrel.txt. 2005 (cited on page 43).

[24] Milan Erdelj, Enrico Natalizio, Kaushik R Chowdhury, and Ian F Akyildiz. 'Help from the sky: Leveraging UAVs for disaster management'. In: *IEEE Pervasive Computing* 16.1 (2017), pp. 24–32 (cited on page 23).

[25] Aymen Fakhreddine, Christian Bettstetter, Samira Hayat, Raheeb Muzaffar, and Driton Emini. *Handover Challenges for Cellular-Connected Drones*. Seoul, Republic of Korea, 2019. DOI: 10.1145/3325421.3329770 (cited on page 23).

[26] *Fed4Fire+: Federation for Future Internet Research and Experimentation+*. 2017. URL: https://www.fed4fire.eu/the-project/ (cited on page 35).

[27] Fédération Française des Télécoms. *Tempête Alex | Comment les opérateurs se sont mobilisés pour réparer les réseaux fixes et mobiles ?* Oct. 2020. URL: https://www.fftelecoms.org/nos-travaux-et-champs-dactions/reseaux/tempete-alex-comment-les-operateurs-se-sont-mobilises-pour-reparer-les-reseaux-fixes-et-mobiles/ (cited on page 4).

[28] Wi-Fi Alliance. *Wi-Fi Alliance® introduces Wi-Fi 6*. Oct. 2018. URL: https://www.wi-fi.org/news-events/newsroom/wi-fi-alliance-introduces-wi-fi-6 (cited on page 10).

[29] F. Fietkau. *Minstrel HT: New rate control module for 802.11n*. https://lwn.net/Articles/376765/. 2010 (cited on pages 43, 63).

[30] Flightradar24 AB. *Live Flight Tracker*. 2020. URL: https://www.flightradar24.com/ (visited on 08/21/2020) (cited on page 21).

[31] Dario Floreano and Robert J Wood. 'Science, technology and the future of small autonomous drones'. In: *Nature* 521.7553 (2015), pp. 460–466 (cited on page 15).

[32] Global Mobile Data Traffic Forecast. 'Cisco visual networking index: global mobile data traffic forecast update, 2017–2022'. In: *Update* 2017 (2019), p. 2022 (cited on page 10).

[33] Eric W. Frew and Timothy X. Brown. 'Airborne Communication Networks for Small Unmanned Aircraft Systems'. In: *Proceedings of the IEEE* 96.12 (2008), pp. 2008–2027 (cited on page 22).

[34] Paul Gardner-Stephen, Romana Challans, Jeremy Lakeman, Andrew Bettison, Dione Gardner-Stephen, and Matthew Lloyd. *The serval mesh: A platform for resilient communications in disaster & crisis*. Oct. 2013. DOI: 10.1109/GHTC.2013.6713674 (cited on page 5).

[35] Matthew S Gast. *802.11 Wireless Networks: The Definitive Guide, Second Edition*. O'Reilly Media, Inc., 2005 (cited on page 9).

[36] Nikolaus Gebhardt, Michael Zeilfelder, et al. 'Irrlicht Engine - A free open source 3d engine'. In: *irrlicht.sourceforge.net* (2002) (cited on page 32).

[37] N. Goddemeier, S. Rohde, and C. Wietfeld. 'Experimental validation of RSS driven UAV mobility behaviors in IEEE 802.11s networks'. In: *2012 IEEE Globecom Workshops*. 2012, pp. 1550–1555 (cited on page 26).

[38] Mahanth Gowda, Ashutosh Dhekne, and Romit Roy Choudhury. 'The case for robotic wireless networks'. In: *Proceedings of the 25th International Conference on World Wide Web*. 2016, pp. 1317–1327 (cited on pages 5, 29).

[39] Rémy Grünblatt, Isabelle Guérin-Lassous, and Olivier Simonin. 'Simulation and Performance Evaluation of the Intel Rate Adaptation Algorithm'. In: *MSWiM 2019 - 22nd ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. Miami Beach, United States: ACM, Nov. 2019, pp. 27–34. DOI: 10.1145/3345768.3355921 (cited on page 7).

[40] Rémy Grünblatt, Isabelle Guérin-Lassous, and Olivier Simonin. 'Study of the Intel WiFi Rate Adaptation Algorithm'. In: *CoRes 2019 - Rencontres Francophones sur la Conception de Protocoles, l'Évaluation de Performance et l'Expérimentation des Réseaux de Communication*. Saint-Laurent-de-la-Cabrerisse, France, June 2019, pp. 1–4 (cited on page 7).

[41] Rémy Grünblatt, Isabelle Guérin-Lassous, and Olivier Simonin. 'Leveraging Antenna Orientation to Optimize Network Performance of Fleets of UAVs'. In: *MSWiM'20 - The 23rd International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. Alicante, Spain, Nov. 2020. DOI: 10.1145/3416010.3423225 (cited on page 7).

[42] Y. Gu, M. Zhou, S. Fu, and Y. Wan. 'Airborne WiFi networks through directional antennae: An experimental study'. In: *2015 IEEE Wireless Communications and Networking Conference (WCNC)*. 2015, pp. 1314–1319 (cited on pages 5, 28).

[43] Isabelle Guérin-Lassous and Laurent Reynaud. 'Improving the Performance of Challenged Networks with Controlled Mobility'. In: *Journal on Mobile Networks and Applications (MONET)* (Jan. 2017) (cited on page 29).

[44] S. Hayat, E. Yanmaz, and C. Bettstetter. 'Experimental analysis of multipoint-to-point UAV communications with IEEE 802.11n and 802.11ac'. In: *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. 2015, pp. 1991–1996 (cited on pages 5, 27).

[45] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. 'Performance anomaly of 802.11b'. In: *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*. Vol. 2. 2003, 836–843 vol.2 (cited on page 42).

[46] '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'. In: *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)* (2016), pp. 1–3534 (cited on pages 12, 42).

[47] Rajendra K Jain, Dah-Ming W Chiu, William R Hawe, et al. 'A quantitative measure of fairness and discrimination'. In: *Eastern Research Laboratory, Digital Equipment Corporation, Hudson, MA* (1984) (cited on page 95).

[48] Xianjun Jiao, Wei Liu, Michael Mehari, Muhammad Aslam, and Ingrid Moerman. 'openwifi: a free and open-source IEEE802.11 SDR implementation on SoC'. In: *IEEE VTC2020, the 91st Vehicular Technology Conference*. 2020, pp. 1–2 (cited on page 44).

[49] A Kamerman and L Monteban. 'WaveLAN II: a high-performance wireless LAN for the unlicensed band'. In: *Bell Labs Technical Journal* (1997), pp. 118–133 (cited on pages 42, 43).

[50] Julius Knapp. *Clearing the Air on Wi-Fi Software Updates*. 2015. URL: https://www.fcc.gov/news-events/blog/2015/11/12/clearing-air-wi-fi-software-updates (cited on page 43).

[51] Jason Koebler, Joseph Cox, and Jordan Pearson. *Customs and Border Protection Is Flying a Predator Drone Over Minneapolis*. May 2020. URL: https://www.vice.com/en_us/article/5dzbe3/customs-and-border-protection-predator-drone-minneapolis-george-floyd (cited on page 14).

[52] S. Li, C. He, M. Liu, Y. Wan, Y. Gu, J. Xie, S. Fu, and K. Lu. 'Design and implementation of aerial communication using directional antennas: learning control in unknown communication environments'. In: *IET Control Theory Applications* 13.17 (2019), pp. 2906–2916 (cited on page 30).

[53] Magnus Lindhé and Karl Henrik Johansson. 'Using robot mobility to exploit multipath fading'. In: *IEEE Wireless Communications* 16.1 (2009), pp. 30–37 (cited on pages 5, 29).

[54] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim. 'Placement Optimization of UAV-Mounted Mobile Base Stations'. In: *IEEE Communications Letters* 21.3 (2017), pp. 604–607 (cited on page 18).

[55] Emerson Alberto Marconato, Mariana Rodrigues, Rayner de Melo Pires, Daniel Fernando Pigatto, Alex Roschildt Pinto, Kalinka RLJC Branco, et al. 'Avens-a novel flying ad hoc network simulator with automatic code generation for unmanned aircraft system'. In: *Proceedings of the 50th Hawaii international conference on system sciences*. 2017 (cited on page 34).

[56] Stephen McCann. *Official IEEE 802.11 Working Group Project Timelines*. IEEE. July 21, 2020. URL: http://grouper.ieee.org/groups/802/11/Reports/802.11_Timelines.htm (visited on 07/29/2020) (cited on page 10).

[57] Ministère de l'économie et des finances. *Arrêté du 27 décembre 2019 définissant les caractéristiques techniques des dispositifs de signalement électronique et lumineux des aéronefs circulant sans personne à bord*. Dec. 2019. URL: https://www.legifrance.gouv.fr/eli/arrete/2019/12/27/ECOI1934044A/jo/texte (cited on page 21).

[58] Ministère de l'économie et des finances. *Décret n° 2019-1114 du 30 octobre 2019 pris pour l'application de l'article L. 34-9-2 du code des postes et des communications électroniques*. Oct. 2019. URL: https://www.legifrance.gouv.fr/eli/decret/2019/10/30/ECOI1901144D/jo/texte (cited on page 21).

[59] K. Miranda, E. Natalizio, T. Razafindralambo, and A. Molinaro. 'Adaptive router deployment for multimedia services in mobile pervasive environments'. In: *2012 IEEE International Conference on Pervasive Computing and Communications Workshops*. 2012, pp. 471–474 (cited on pages 5, 30).

[60] Karen Miranda, Nathalie Mitton, and Tahiry Razafindralambo. 'On the Impact of Routers' Controlled Mobility in Self-Deployable Networks'. In: 2015 (cited on page 30).

[61] Jean-Marc Moschetta and Kamesh Namuduri. 'Introduction to UAV Systems'. In: *UAV Networks and Communications*. Ed. by Kamesh Namuduri, Serge Chaumette, Jae H. Kim, and James P. G.Editors Sterbenz. Cambridge University Press, 2017, pp. 1–25. DOI: 10.1017/9781316335765.002 (cited on page 4).

[62] David Murray, Michael Dixon, and Terry Koziniec. 'An experimental comparison of routing protocols in multi hop ad hoc networks'. In: *2010 Australasian Telecommunication Networks and Applications Conference*. IEEE. 2010, pp. 159–164 (cited on page 21).

[63] Raheeb Muzaffar, Christian Raffelsberger, Aymen Fakhreddine, José López Luque, Driton Emini, and Christian Bettstetter. 'First Experiments with a 5G-Connected Drone'. In: *Proceedings of the 6th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*. DroNet '20. Toronto, Ontario, Canada: Association for Computing Machinery, 2020. DOI: 10.1145/3396864.3400304 (cited on page 23).

[64] Richard van Nee and Ramjee Prasad. *OFDM for Wireless Multimedia Communications*. 1st. USA: Artech House, Inc., 2000 (cited on page 13).

[65] Ubiquiti Networks. *UAP Antenna Radiation Patterns*. 2019. URL: http://web.archive.org/web/20200613210252/https://help.ui.com/hc/en-us/articles/115005212927-UniFi-UAP-Antenna-Radiation-Patterns (visited on 06/13/2020) (cited on page 91).

[66] Axel Neumann, Ester López, and Leandro Navarro. 'Evaluation of mesh routing protocols for wireless community networks'. In: *Computer Networks* 93 (2015), pp. 308–323 (cited on page 21).

[67] Francesco Nex and Fabio Remondino. 'UAV for 3D mapping applications: a review'. In: *Applied geomatics* 6.1 (2014), pp. 1–15 (cited on page 17).

[68] George Nychis, Thibaud Hottelier, Zhuocheng Yang, Srinivasan Seshan, and Peter Steenkiste. 'Enabling MAC Protocol Implementations on Software-Defined Radios.' In: *NSDI*. Vol. 9. 2009, pp. 91–105 (cited on page 49).

[69] Kirtan Gopal Panda, Shrayan Das, Debarati Sen, and Wasim Arif. 'Design and deployment of UAV-aided post-disaster emergency network'. In: *IEEE Access* 7 (2019), pp. 102985–102999 (cited on pages 5, 27).

[70] Guangyu Pei and Thomas R Henderson. *Validation of OFDM model in ns-3*. 2011 (cited on page 42).

[71] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. 'ROS: an open-source Robot Operating System'. In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5 (cited on page 19).

[72] *R2lab: An open testbed for reproducible networking research*. 2017. URL: https://r2lab.inria.fr/index.md (cited on page 35).

[73] Jean Razafimandimby, Karen Miranda, Dimitrios Zorbas, and Tahiry Razafindralambo. 'Fast and reliable robot deployment for substitution networks'. In: *Proceedings of the 10th ACM symposium on Performance evaluation of wireless ad hoc, sensor, & ubiquitous networks*. 2013, pp. 17–24 (cited on pages 5, 30).

[74] Laurent Reynaud and Isabelle Guérin-Lassous. 'Design of a force-based controlled mobility on aerial vehicles for pest management'. In: *Ad Hoc Networks* 53 (2016), pp. 41–52 (cited on page 29).

[75] Craig W Reynolds. 'Flocks, herds and schools: A distributed behavioral model'. In: *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. 1987, pp. 25–34 (cited on page 21).

[76] RFI. *French police deploy drones to enforce coronavirus restrictions*. Mar. 2020. URL: https://www.rfi.fr/en/france/20200322-french-police-deploy-drones-helicopters-to-enforce-coronavirus-restrictions-covivd-19-lockdown (cited on page 18).

[77] George F Riley and Thomas R Henderson. 'The ns-3 network simulator'. In: *Modeling and tools for network simulation*. Springer, 2010, pp. 15–34 (cited on page 33).

[78] Michael S. Schmidt and Eric Schmitt. *Pentagon Confronts a New Threat From ISIS: Exploding Drones*. Oct. 2016. URL: https://www.nytimes.com/2016/10/12/world/middleeast/iraq-drones-isis.html (cited on page 14).

[79] Guillaume Sanahuja et al. *Fl-AIR: Framework Libre AIR*. 2016. URL: https://devel.hds.utc.fr/software/flair (cited on page 32).

[80] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. 'AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles'. In: *Field and Service Robotics*. 2017 (cited on page 32).

[81] Skydio. *Introducing Skydio 2*. Oct. 2019. URL: https://www.skydio.com/pages/skydio-2 (cited on page 16).

[82] Jonathan M Smith, Marc P Olivieri, Alex Lackpour, and Nicholas Hinnerschitz. 'RF-mobility gain: concept, measurement campaign, and exploitation'. In: *IEEE Wireless Communications* 16.1 (2009), pp. 38–44 (cited on page 29).

[83] International Organization for Standardization. 'ISO 8373: 2012 (en) Robots and robotic devices–Vocabulary'. In: (2012) (cited on page 15).

[84] Russell Stuart, Norvig Peter, et al. *Artificial intelligence: a modern approach*. 2003 (cited on page 88).

[85] Hanif Ullah, Mamun Abu-Tair, Sally McClean, Paddy Nixon, Gerard Parr, and Chunbo Luo. 'Connecting Disjoint Nodes Through a UAV-Based Wireless Network for Bridging Communication Using IEEE 802.11 Protocols'. In: *EURASIP Journal on Wireless Communications and Networking* 2020.1 (2020), pp. 1–20 (cited on pages 5, 28).

[86] Hanif Ullah, Mamun Abu-Tair, Sally McClean, Patrick Nixon, Gerard Parr, and Chunbo Luo. 'An unmanned aerial vehicle based wireless network for bridging communication'. In: *2017 14th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC)*. IEEE. 2017, pp. 179–184 (cited on pages 5, 28).

[87] András Varga and Rudolf Hornig. 'An overview of the OMNeT++ simulation environment'. In: *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST (Institute for Computer Sciences, Social-Informatics and . . . 2008, p. 60 (cited on page 33).

[88] G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szörenyi, T. Nepusz, and T. Vicsek. 'Outdoor flocking and formation flight with autonomous aerial robots'. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 3866–3873 (cited on page 21).

[89] Nick Waters. *First ISIS, then Iraq, now Israel: IDF Use of Commercial Drones*. June 2018. URL: https://www.bellingcat.com/news/mena/2018/06/18/first-isis-iraq-now-israel-idf-use-commercial-drones/ (cited on pages 14, 18).

[90] Brian Glyn Williams. 'The CIA's covert Predator drone war in Pakistan, 2004–2010: the history of an assassination campaign'. In: *Studies in Conflict & Terrorism* 33.10 (2010), pp. 871–892 (cited on page 14).

[91] Starsky HY Wong, Hao Yang, Songwu Lu, and Vaduvur Bharghavan. 'Robust rate adaptation for 802.11 wireless networks'. In: *Proceedings of the 12th annual international conference on Mobile computing and networking*. 2006, pp. 146–157 (cited on page 42).

[92] Evşen Yanmaz and Samira Hayat. 'Aerial Wi-Fi Networks'. In: *UAV Networks and Communications*. Ed. by Kamesh Namuduri, Serge Chaumette, Jae H. Kim, and James P. G.Editors Sterbenz. Cambridge University Press, 2017, pp. 45–57. DOI: 10.1017/9781316335765.004 (cited on pages 19–21).

[93] Evşen Yanmaz, Samira Hayat, Jürgen Scherer, and Christian Bettstetter. 'Experimental performance analysis of two-hop aerial 802.11 networks'. In: *2014 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE. 2014, pp. 3118–3123 (cited on page 27).

[94] Evşen Yanmaz, Robert Kuschnig, and Christian Bettstetter. 'Channel measurements over 802.11 a-based UAV-to-ground links'. In: *2011 IEEE GLOBECOM Workshops (GC Wkshps)*. IEEE. 2011, pp. 1280–1284 (cited on pages 5, 26).

[95] Evşen Yanmaz, Robert Kuschnig, and Christian Bettstetter. 'Achieving air-ground communications in 802.11 networks with three-dimensional aerial mobility'. In: *2013 Proceedings IEEE INFOCOM*. IEEE. 2013, pp. 120–124 (cited on pages 5, 23, 26, 91).

[96] Nicola Roberto Zema, Angelo Trotta, Enrico Natalizio, Marco Di Felice, and Luciano Bononi. 'The CUSCUS simulator for distributed networked control systems: Architecture and use-cases'. In: *Ad Hoc Networks* 68 (2018), pp. 33–47 (cited on pages 34, 89).

[97] Nicola Roberto Zema, Angelo Trotta, Guillaume Sanahuja, Enrico Natalizio, Marco Di Felice, and Luciano Bononi. 'CUSCUS: An integrated simulation architecture for distributed networked control systems'. In: *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE. 2017, pp. 287–292 (cited on page 34).

[98] Yi Zhou, Nan Cheng, Ning Lu, and Xuemin Sherman Shen. 'Multi-UAV-aided networks: Aerial-ground cooperative vehicular networking architecture'. In: *ieee vehicular technology magazine* 10.4 (2015), pp. 36–44 (cited on pages 5, 27).

Special Terms

Numbers
3GPP 3rd Generation Partnership Project. 3

A
ACK Acknowledgement. 11, 43
ACM Association for Computing Machinery. 36
ADS-B Autonomous Dependent Surveillance-Broadcast. 21
AERPAW Aerial Experimentation and Research Platform for Advanced Wireless. 36, 37
A-MPDU Aggregation of MPDU. 53, 55, 58, 59, 64, 66, 68
A-MSDU Aggregation of Service Data Unit. 53, 58
AP Access Point. 11, 23, 27–29, 45–47, 54
API Application Programming Interface. 31, 45, 49
ARCEP Autorité de régulation des communications électroniques, des postes et de la distribution de la presse. iii, 4
ARF Automatic Rate Fallback. 42, 43
ARP Address Resolution Protocol. 33
AWGN Additive White Gaussian Noise. 42

B
BER Bit Error Rate. 41, 70, 71
BLE Bluetooth Low Energy. 35
BPSK Binary Phase-Shift Keying. 40
BSS Basic Service Set. 11

C
CCK Complementary Code Keying. 13, 52
CLI Command Line Interface. 33
CSMA Carrier Sense, Multiple Access. 11
CSMA/CA Carrier Sense, Multiple Access with Collision Avoidance. 9, 11, 42
CSMA/CD Carrier Sense, Multiple Access with Collision Detection. 11

D
DBPSK Differential Binary Phase-Shift Keying. 12, 40
DCF Distributed Coordination Function. 11, 49
DIFS Distributed Inter Frame Space. 11
DMG Directional Multi-Gigabit. 40, 41
DQPSK Differential Quadrature Phase-Shift Keying. 12, 40
DS Distribution System. 12
DSSS Direct-Sequence Spread Spectrum. 12, 40–42, 52

E
ERP Extended Rate PHY. 40, 41, 52, 54
EWMA Exponential Weight Moving Average. 68, 69

F
FCC Federal Communications Commission. 3, 9, 23, 43
FHSS Frequency-Hopping Spread Spectrum. 12, 23, 40, 41
FPGA Field Programmable Gate Arrays. 44

G
GFSK Gaussian Frequency-Shift Keying. 12, 40

GI Guard Interval. 13
GNSS Global Navigation Satellite System. 16, 19, 20, 29
GUI Graphical User Interface. iii, 33, 34

H
HALE High-Altitude Long Endurance. 16, 22
HAPS High-Altitude Platform Station. 16, 22
HE High-Efficiency. 52
HR/DSSS High-Rate Direct-Sequence Spread Spectrum. 12, 40, 41
HT High Throughput. iv, 40, 41, 43, 45, 47, 49, 51–53, 63, 64
HWMP Hybrid Wireless Mesh Protocol. 12

I
IBSS Independant Basic Service Set. 11, 12
IFS Inter Frame Space. 11
IMU Inertial Measurement Unit. 19, 20, 31
IOT Internet Of Things. 5, 35
IP Internet Protocol. 10
ISM Industrial, Scientific and Medical. 9, 13, 20
ITU International Telecommunication Union. 3, 20

L
LAN Local Area Network. 10
LDPC Low-Density Parity-Check Codes. 53
LGI Long Guard Interval. v, 53, 55, 56, 64
LOA Level Of Autonomy. 15
LoRa *Lo*ng *Ra*nge. 36
LOS Line Of Sight. 26
LTE Long Term Evolution. 24, 35

M
MAC Medium Access Control Layer. i, 9, 11, 33, 39, 49, 50, 114
MALE Medium-Altitude Long Endurance. iii, 16, 22
MBSS Mesh Basic Service Set. 11, 12
MCS Modulation and Coding Scheme. iv, v, 27, 40–42, 47, 49, 52, 53, 55, 58, 64, 66–68, 72, 73, 76, 80
MIMO Multiple Input, Multiple Output. 13
MLME MAC Sublayer Management Entity. 49
MPDU MAC Protocol Data Unit. 53, 55, 58, 59, 64, 66, 68, 113
MRC Maximum Ratio Combining. 13
MSDU MAC Service Data Unit. 58

N
NLOS Non Line Of Sight. 26

O
OFDM Orthogonal Frequency-Division Multiplexing. 13, 23, 40, 41, 52
OFDMA Orthogonal Frequency-Division Multiple Access. 39
OLSR Optimized Link State Routing. 21, 29, 33
OS Operating System. 43, 45, 49, 50

P
PHY Physical Layer. i, iii, v, 9, 11–13, 33, 40–43, 45, 49, 51–54, 63, 64, 83, 114
PLME PHY Sublayer Management Entity. 49
PN Pseudonoise. 12, 13
PPDU PHY Protocol Data Unit. 58

PPM Pulse-Position Modulation. 21

Q

QAM Quadrature Amplitude Modulation. 13, 40, 41
QOS Quality Of Service. 11
QPSK Quadrature Phase-Shift Keying. 40

R

RAA Rate Adaptation Algorithm. iv, v, 6, 27, 39, 42–45, 47, 50–54, 58–61, 63–65, 67, 69–71, 75, 76, 78–81, 83, 93–97, 104, 105
RAM Random Access Memory. 49
ROM Read Only Memory. 49
ROS Robot Operating System. 19, 31, 32
RSS Received Signal Strength. iii, iv, 25, 26, 29, 30, 46–48
RTS/CTS Request to Send, Clear to Send. 12
RTT Round-Trip Time. 30

S

SAR Search and Rescue. 18
SDR Software Defined Radio. 23, 29, 30, 35, 36, 44, 45, 47
SGI Short Guard Interval. 53, 55, 64
SIFS Short Inter Frame Space. 11, 49
SLAM Simultaneous Location And Mapping. 15, 16, 32
SME Station Management Entity. 49
SNR Signal-To-Noise. iii, 29, 30, 42, 70, 80
STA Station. 11, 12, 42, 45, 46, 49, 52, 53, 57, 72, 75, 78
STBC Space-Time Block Code. 13, 53

T

TDMA Time-division Multiple Access. 29
TVHT Television Very High Throughput. 40, 41

U

U-NII Unlicensed National Information Infrastructure. 13
USB Universal Serial Bus. 46
UWB Ultra Wide Band. 16

V

VHT Very High Throughput. v, 40, 41, 43, 45, 49, 51–54, 59, 63, 64, 68, 80

W

WLAN Wireless Local Area Network. 9, 10, 23
WNIC Wireless Network Interface Controller. 20, 27–29, 45, 46, 59, 60, 69, 101
WPAN Wireless Personal Area Network. 9
WSN Wireless Sensor Network. 18