Chapter 3

Experimental Wireless Networking Research using Software-Defined Radios

Adriele Dutra Souza[3], Ariel F. F. Marques[2], Daniel F. Macedo[1], Diarmuid Collins[4], Gilson Miranda Júnior[2], Jefferson R. S. Cordeiro[1], Johann M. Marquez-Barja[4], José Augusto M. Nacif[3], Kristtopher Kayo Coelho[3], Luccas R. M. Pinto[2], Luiz A. da Silva[4], Luiz F. M. Vieira[1], Luiz H. A. Correia[2], Marcos A. M. Vieira[1], Pedro Alvarez[4], Wendley S. Silva[1]

[1]Universidade Federal de Minas Gerais (UFMG), Brazil

[2]Universidade Federal de Lavras (UFLA), Brazil

[3]Universidade Federal de Viçosa (UFV) - Campus Florestal, Brazil

[4]Trinity College Dublin (TCD), Ireland
e-mails: {damacedo,jeff,lfvieira,mmvieira,wendley}@dcc.ufmg.br,
{arielffmarques,junior.kdm,luccasrm,luiz.ha.correia}@gmail.com,
{adriele.souza, jnacif, kristtopher.coelho}@ufv.br
{collindi,marquejm,pinheirp}@tcd.ie

Abstract

Thanks to the popularization of software-defined radios (SDR), it is possible today to perform high-quality research in wireless protocols in real deployments. Although this technology is still a bit expensive, there are a number of initiatives that provide free access to SDR for research. Further, the number of free software libraries available for SDR has reduced the amount of effort required to conduct research using SDR. This short course will show by examples how to perform experimental research in wireless networking using software-defined radios that are available for free on open testbeds being developed on FUTEBOL, a joint Brazil-European Union project. We will adopt a hands-on approach, in which the students will perform many small assignments on real hardware. Those

assignments will demonstrate the maturity of SDR for research in wireless networking, and introduce the user to the many software tools and open source implementations of a variety of wireless standards.

3.1. Introduction

Software-Defined Radios (SDR) are a collection of hardware and software technologies where some or all of the radio's operating functions are implemented through modifiable software or firmware operating on programmable processing technologies (e.g. an FPGA, a generic CPU) [Wireless Innovation Forum 2017].

The US military was the first to employ SDRs, in order to provide flexible radios for large-scale operations [Dillinger et al. 2003]. The objective was to ensure the interoperability of military radios among government agencies (firefighters, police, intelligence agencies). Such a need comes from a practical reason since each agency performs its purchases independently, and thus the communication technologies employed may be incompatible from the point of view of frequency used, signal modulation technology, among others. Thus, the term "digital radio" was coined to define radios that adapt to different operating standards.

In 2011, the Wireless Innovation Forum commissioned a study to evaluate the rate of adoption of SDR in the telecommunications industry [Forum 2011]. The results indicate that more than 90% of the mobile infrastructure on that year employed SDR technologies of some kind. For markets where interoperability is a mandatory requirement, as in military and public safety applications, they have found that almost all transceivers and base stations employ SDR.

Software-defined radios are now a reality, as seen by the number of commercial and free platforms available in the market[1]. Given the viable commercial and academic platforms that are capable of implementing 3G and IEEE 802.11a/b/g/n radios and other technologies, several research groups have purchased SDR equipment for their research. SDR has lowered the cost of conducting experimental research on the physical layer and link layer. Thus, experiments that were previously possible only in laboratories of large companies can be done in laboratories of universities at a reasonable cost. To showcase the importance to the SDR in the area of wireless networks, only the WARP platform at Rice University counted 43 scientific papers that used their hardware in 2016[2].

However, as we will show in this chapter, SDR is a fairly accessible technology for experimental research in wireless networks. There are many software libraries available, which awesome allow SDR boards to run popular wireless standards. An SDR can be used to emulate from big to small devices (from RFID tags to a 4G eNodeB), from simple to complex communication standards (from a remote garage controller to a digital TV transmission). Further, initiatives such as GENI[3] in the US, Fed4Fire[4] in Europe, and

[1] See a list of some of the existing platforms: `https://en.wikipedia.org/wiki/List_of_software-defined_radios`

[2] `http://warpproject.org/trac/wiki/PapersandPresentations#LatestPapers`

[3] `http://geni.net/`

[4] `http://fed4fire.eu/`

FUTEBOL[5] in Brazil, provide free access to SDR hardware for research purposes.

This chapter will focus on the kinds of experiments that can be performed in USRP boards [Ettus 2017], as well as how to setup and run an experiment on the SDR testbeds made available by the FUTEBOL project. At the end of this chapter, the reader should be able to understand the limitations of SDR, how to choose the type of SDR for his/her experiment, as well as how to perform an experiment remotely on the FUTEBOL testbed. Readers that want to learn about the theoretical aspects of SDR, as well as the basics of how to program SDR using GNU Radio, should refer to [Silva et al. 2015].

3.1.1. Existing research using SDRs

SDR technologies enable different types of research that previously were mostly performed using simulations and/or analytical methods. Since SDR is a versatile hardware, the same board can be used over and over again for different wireless projects, becoming a must-have for groups that perform systems research in wireless networks. Below we present a list of recent advances in wireless communications that employed SDR as an evaluation platform:

- **Using network coding to increase wireless capacity.** With network coding, it is possible to send a packet that is the combination of several packets into wireless links, increasing the overall flow of the network. Earnings depend on the traffic pattern, ranging from a small percentage up to several times [Katti et al. 2008, Vieira et al. 2013].

- **Recover lost frames using signal processing.** Radios store incoming signals from a collision, and attempt to subtract received frames correctly (after a retransmission), often allowing the frame to be involved in a collision without it having to be retransmitted [Lin et al. 2008].

- **Rateless codes** In traditional wireless communication, data is transmitted using a certain modulation. Each modulation requires a certain Signal to Interference plus Noise Ratio (SINR) threshold for proper decoding of its data, and in order to deal with variations in SINR, existing protocols (e.g. Wi-Fi and WiMax protocols) use automatic modulation change mechanisms. Rateless codes [Gudipati and Katti 2011, Perry et al. 2012, Shokrollahi 2006] use a variable amount of symbols to encode the data. The transmitter sends the data using different codes, and the receiver uses those codes to identify which word is most likely to have been used to generate the received signals. Rateless codes require special protocols of the [Iannucci et al. 2012] binding layer, which can also be tested using SDR. The main benefit of rateless codes is that it provides bandwidth at the link much closer to Shannon's theoretical capacity.

- **Full-duplex radios.** Existing commercial radios are half-duplex since one receiving antenna would be saturated with signals transmitted by another adjacent antenna. However, research employing dedicated hardware and SDRs can achieve

[5] `http://www.ict-futebol.org.br`

such levels of noise cancellation as to allow full-duplex radios to be compliant with IEEE 802.11b [Hong et al. 2012] and IEEE 802.11ac [Bharadia et al. 2013].

- **Cooperative MIMO for enterprise WLANs.** OpenRF [Kumar et al. 2013] uses the concept of coding vectors in order to perform beamforming, which cancels out the interference of neighboring APs. With the help of a centralized controller and a local scheduling algorithm running on each access point, it is possible to exploit the MIMO capabilities of the IEEE 802.11n standard to improve the SINR in the stations. This, in turn, reduces latency variations and increases network throughput.

Although the most common use of SDR is in telecommunications research, it can also be used by networking researchers. Nowadays it is possible to find full protocol stacks that are either open source or available freely for research. This is allowing SDR to be used on networking-related papers, dealing with issues such as MAC layer design, management of wireless networks, performance evaluation of 4G networks, security, etc. Here are some examples.

- **Cloud RAN**. USRPs are being used to emulate Cloud RAN deployments [Beyene et al. 2014]. Aspects such as where to run the physical and MAC layers (near the eNodeBs or in data centers) and how to control the RAN system can be studied using real deployments.

- **Cognitive Radios**. Research on cognitive radios requires equipment that is able to change its operating frequency and survey the usage of the spectrum by employing different algorithms to detect the existence and types of transmissions on the medium. One such study is presented by Souryal et al., which measured the usage of the spectrum using USRPs as their platform [Souryal et al. 2015].

- **New MAC protocols**. SDR can be used to change parameters of the MAC layer or even implement a completely new protocol. Usually, the MAC protocol is implemented in the firmware of the wireless transceivers, and as such, it is very hard to change it. Thus, without SDR such types of research would probably be performed using simulators. However, this task is feasible in SDRs. One example is LA-MAC, a load-aware MAC protocol that was tested using USRPs [Hu et al. 2009].

- **Investigating the security of wireless protocols**. SDR can be used to capture traffic and then analyze it to identify vulnerabilities in wireless protocols. It can also be used to build proof-of-concept attacks against those protocols, as well as to propose defenses. Gollakota et al. employed USRPs to propose a device called *shield*, which prevents outsiders from eavesdropping the messages of implantable medical devices (IMDs) [Gollakota et al. 2011].

3.2. Getting your hands dirty

This section presents the Universal Software Radio Peripheral boards, which are the most popular SDR platform among researchers. Alternative platforms will also be presented, as well as the practical issues that arise when using SDR to perform experimental research. Next, we will describe the lifecycle of an experiment in remote testbeds.

3.2.1. An introduction to USRP boards

Universal Software Radio Peripheral (USRP) is a framework for the development of digital radios, providing a complete infrastructure for signal processing. The system is characterized by its high flexibility and cost-benefit. USRPs are developed by Ettus Inc [Ettus 2017], which is a subsidiary of National Instruments.

USRPs are an attractive platform for SDR research for many reasons. First, Ettus open sourced schematics for some of the USRP models, and the driver that allows the communication of the boards with a computer is also open source. The USRP hardware driver (UHD) is compatible with many operating systems, such as Windows, Linux, and Mac. Second, USRPs are compatible with GNU Radio [Gilmore and Blossom 2017], which is a GNU library of software that implements several algorithms related to signal processing and communications. Further, there is a large community of people using USRPs for research or as a hobby, and as such it is relatively easy to find support online. Finally, popular scientific software such as MatLab and others support USRPs.

USRPs are composed of an FPGA, components for baseband processing, and daughterboards. In general, the hardware processes the RF signals, converting them into digital signals to be processed either at the FPGA or at a host computer. The USRP communicate with a PC using a high-speed bus, which may be a USB interface or a network interface. The daughterboards are interchangeable cards that provide the filters and amplifiers that are necessary to support a certain application, that is, a certain range of frequencies. Some of the models do not support daughterboards, and as such, they have a fixed range of frequencies.

There are a number of USRP models, with varying interfaces and capabilities. The Networked series connects to the PC using Ethernet. The bus-based series connects to the PC using USB. This series has a "mini" line, which are USRPs with a small form factor (at the moment of the writing, the size of a business card). The X series products are the higher end products, with more capable FPGAs as well as higher sampling rates. Finally, the Embedded series provides a more rugged product that is coupled with an ARM processor. It is ideal for operations on the field.

USRPs are supported by a number open source software libraries or initiatives. There are open source implementations of many communication standards provided as out-of-tree GNU Radio components (that is, they are not officially supported by GNU Radio), such as IEEE 802.11, IEEE 802.15.4, RFID. It is worth noticing that USRPs can be used today even to implement a full mobile network using open source telecommunication stacks such as OpenBTS [ope 2017] and OpenAirInterface [OAI 2017].

3.2.1.1. Alternatives to USRP

Although USRPs are very popular SDR platforms, other platforms can also be used in research experiments. This section lists those that are most commonly found in universities and research centers around the world.

- **SORA** stands for *Microsoft Research Software Radio* [Tan et al. 2009], and is an SDR platform developed by Microsoft Research (MSR) Asia in Beijing. The SORA hardware is very simple, having only a baseband decoder, and all processing is done by an x86 CPU. Due to that architecture, the platform has very stringent bandwidth requirements and developers must optimize their implementations using assembly for better performance.

- **WARP** is a research-oriented SDR that is also used in many testbeds [Murphy et al. 2006, Amiri et al. 2007]. Its hardware is more capable than most USRPs since it is able to decode up to 40MHz channels. Further, it has libraries implementing high-speed communication standards such as IEEE 802.11. As in the USRPs, it has an FPGA that can be used for time-critical parts of the code. However, the hardware is costlier than USRPs.

- **Soft-MAC platforms**. Many platforms allow only the MAC layer to be modified [Tinnirello et al. 2012, Neufeld et al. 2005, Nychis et al. 2009, Rao and Stoica 2005]. The benefits are a lower cost of the devices, as well as the use of simpler programming languages. Some of those platforms even run on commodity wireless interfaces, since they are essentially a firmware update of a commercial interface. However, the kind of programmability that is allowed is limited. For example, in FLAVIA [Tinnirello et al. 2012] the only actions allowed are those defined by the developers of the platform. As such, it would not be possible for example to test a transmission power control protocol on such platforms.

- **RTL-SDR**. The chip RTL2832U is employed in many USB-based digital TV decoders, and it is in effect an SDR [RTL-SDR]. This is a very cheap SDR platform, which can be found for less than 40 dollars. However, its capabilities depend largely on the USB stick's manufacturers. Most are able only to receive signals, and the range of allowed frequencies is usually limited to those of digital TV. Although RTL-SDR is aimed at hobbyist, it could be used for very simple research projects or for teaching purposes. The advantage is that the software specific for RTL-SDR is usually very simple to use.

3.2.1.2. Limitations of SDR

Although SDR is very flexible, there are limitations associated with its generic hardware and its high CPU demands. This section will discuss those limitations so that experimenters understand when not to use SDR to conduct research, and what types of issues must be taken into account in order to select the proper SDR platform and configuration.

High demand for computing and I/O resources. In order to implement high-speed networks, SDR requires fast I/O from the SDR board to the processing unit, as well as a fast processing unit. The amount of processing increases with the complexity of the modulation as well as the bandwidth of the channels. Due to that, many SDR platforms support FPGAs so that the time critical and high bandwidth operations are processed in hardware, near the transceiver.

Limitations due to the choice of filters and antennas. Every daughterboard has limitations as to the frequency range that it is able to operate. Likewise, omnidirectional antennas are tailored to a certain operating frequency. This is not that important when the experimenter has direct access to the equipment since he/she can change the antenna and the daughterboard. However, when using a testbed, the antenna, and the daughterboard are usually fixed. For example, when using an antenna optimized for 2.4Ghz signals in an experiment that employs 900MHz will reduce the quality of the communication. Further, some experiments will not be feasible, since the filter does not allow the SDR to process the required frequency. As an example, an USRP using the Ettus WBX daughterboard[6] cannot be used to decode AM radios, because the WBX cannot pick up signals on frequencies lower than 50MHz.

Limited oscillator resolution. The resolution of the oscillator dictates the capability of the board to implement protocols that require more precise timings. For high-speed networks, for example, 3G and 4G networks, the devices must employ more precise oscillators (e.g a GPS-based oscillator). When the oscillator is not precise enough, the devices' clocks drift from one another, and as such the symbols will be decoded incorrectly.

Delays of the operating system and I/O buses. The delays incurred by the I/O subsystem as well as the operating system overhead must be taken into account when running high-speed networks. For example, in order to support IEEE 802.11g in SORA, Microsoft Research Asia changed the Windows scheduler so that a core is always dedicated to SDR processing [Tan et al. 2009]. OpenAirInterface recommends the use of low latency Linux kernels[7].

3.2.2. Using testbeds

The creation of experimental facilities for more realistic research is a concern of the networking community. Researchers have identified the need for experiments within realistic conditions in order to reduce the gap between experimentation and real network deployments. Due to that, the community has deployed a number of testbeds. The funding agencies have recognized that need as well, so that initiatives such as GENI in the USA and FIRE in Europe, and more recently the RNP calls of joint Brazil-Europe projects, allocate budget for projects that will create and maintain open testbeds. As a consequence, there are a number of testbeds around the world that are available for experimentation. Those testbeds provide resources such as virtual machines, sensor nodes, physical machines, software-defined networking switches, and routers, as well as USRPs.

The access to those devices is free of charge and is performed remotely. The conditions to request access as well as the steps required to setup and run an experiment vary from testbed to testbed, however they follow a set of steps which are described below:

1. The research team identifies the requirements of the experiment.
2. With the requirements, the team identifies which testbed, or which set of testbeds, could be used to perform this experiment.

[6] `https://www.ettus.com/product/details/WBX`

[7] `https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/OpenAirKernelMainSetup`

3. The Principal Investigator (PI), which is the lead researcher on the team, creates a project within the testbeds. Usually, this requests involves a simple description of the experiments as well as a rough estimate on the type and number of devices employed, as well as the duration of the experiment. Other steps may be necessary, for example, if the experiment involves privacy and ethics issues, or if there is a need for a license to use a chunk of the spectrum. In this case, the researcher may be required to request approval from the appropriate university of government bodies.

4. Once the project is approved, the PI creates accounts for the other researchers in the experiment.

5. The researchers create *slices* on the testbed, which are composed of devices and links among those devices. The slices have a maximum lifetime, however, the slices can be destroyed and recreated several times during the lifetime of a project. The slices are described by a configuration file, which defines the type of devices to be used, their location as well as the software to be employed.

6. The slice is activated, and the researchers access the slice to install their tools and code so that the experiment can be run.

7. The experiment is run, and the researchers download the relevant data to be analyzed.

8. The slice is released, either because the experiment ended, or because the maximum time of the slice has expired.

For a more detailed description of the lifecycle of an experiment on a testbed as well as the software required to setup and run a testbed, please refer to [Gomez 2013]. From now on, in this chapter we will describe how to setup and run experiments in testbeds federated in Fed4Fire, more specifically using the resources provided by the FUTEBOL project. However, the process will be similar in other testbeds and for other types of resources.

3.2.3. Using FUTEBOL to conduct research

The EU-BR FUTEBOL project envisages the creation of a federated control framework to integrate testbeds from Europe and Brazil for network researchers from academia and industry. FUTEBOL's major goal is to allow the access to advanced experimental facilities in Europe and Brazil for research and education across the wireless and optical domains. To that end, the project will deploy testbed facilities in a number of European countries as well as in three sites in Brazil, as shown in Figure 3.1. FUTEBOL resources are available in the UFES, UFMG, and UFRGS in Brazil, and in VTT (Finland), IT Aveiro (Portugal), TCD Dublin (Ireland) and University of Bristol (England). Those resources are interconnected using FIBRENET in RNP, and Géant in Europe. Authorization and management of users credentials are performed using Fed4Fire, meaning that any researcher in academia or in the industry with a valid Fed4Fire will be able to access the FUTEBOL resources.

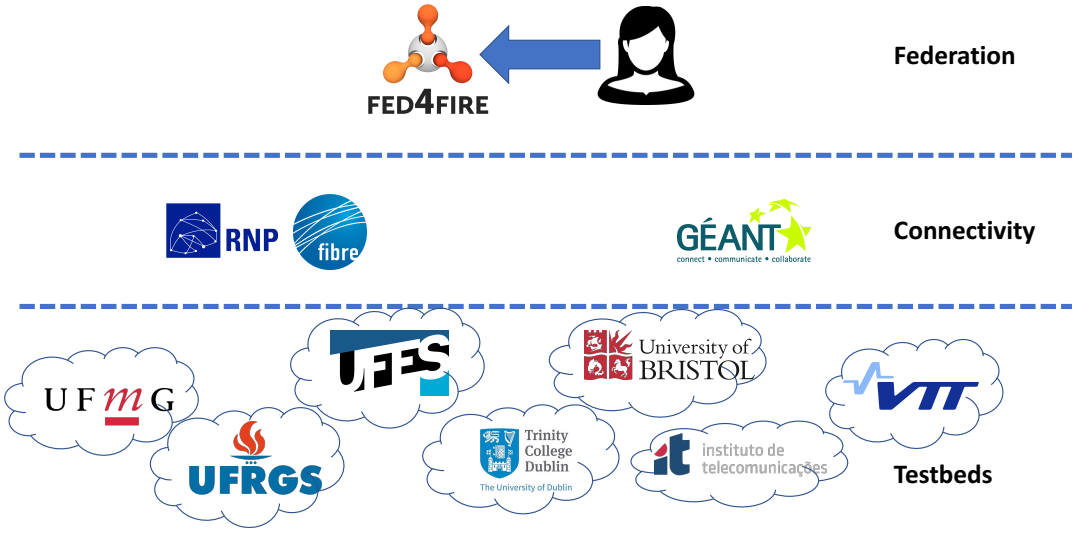

Figure 3.1: The overall architecture of the FUTEBOL federated testbed.

For the moment, USRPs are available in FUTEBOL in the Trinity College Dublin (TCD) as well as in the UFMG islands. However, in the near future, USRPs will also be available in UFRGS and in UFES. For more information on the types of USRPs available as well as their topology, please refer to the website of each island. Links are available at the main FUTEBOL website.

Creating an account. The first step to use FUTEBOL is to create an account and a project in Fed4Fire. The Principal Investigator (PI) should request an account, which can be created at `https://authority.ilabt.iminds.be/`. With an account in hand, the PI can create a project that will use Fed4Fire. This is a simple form that requests a description of the project and a project name. Once this is done, the PI requests the researchers associated with the project to create accounts on Fed4Fire and associates them as members of the project.

Downloading your certificate. After creating a login and receiving an authorization message, the reader can download the certificate and store it locally on your local computer. Save the file with `.pem` extension. This certificate will be requested to access the remote environment through JFed.

Using JFed. Before using JFed, the reader needs to install Java 8 manually. This link provides more details about Java installation: `http://jfed.iminds.be/java8_on_linux/`. To install JFed, the user can use the `.deb` format, available at `http://jfed.iminds.be/downloads/`.

After the installation, we can run JFed and use the .pem certificate, choosing the option *Login with PEM-certificate*, as seen at Fig. 3.2.

The following sections describe experiments using USRPs and assume that the user knows how to create an experiment in the FUTEBOL testbed. This involves creating an account in Fed4Fire, using JFed or other tools to define the number of USRPs that the user needs, and then request the reservation of those resources. For a step-by-step description of those steps, please refer to `http://futebol.dcc.ufmg.br/tutorials`.

Figure 3.2: jFed login screen

`html`. This website provides a sample file that books one virtual machine with one USRP in the UFMG testbed and can be modified to book a large number of USRPs.

3.3. Cognitive radios

In recent years, users have used communication services based on social networks, web chats, email and an infinity of applications that require the devices to have greater processing power, memory, as well as fast and efficient connections. Current mobile devices are equipped with multi-core processors, higher capacity memory systems and a diversity of communication technologies such as Bluetooth, Wi-Fi, and LTE. The demand for communication devices with these characteristics has increased the use of the licensed or primary frequencies and also of the secondary or unlicensed frequencies (ISM - Industrial Scientific and Medical). The intensive use of these devices has caused interferences among primary and secondary frequencies and consequently the spectrum pollution.

The coexistence of different networks and devices that operate at the same frequency, or in adjacent frequencies, may lead to harmful interference, which limits the capabilities of the applications and, in some cases, results in the complete shutdown of those networks. In 2006, the authors [Zhou et al. 2006] predicted that if nothing were done to avoid interference and coexistence problems, the growth of wireless networks could cause the complete overlapping of communication channels. In addition, studies of [McHenry et al. 2006] showed that at some places the 2,400 MHz frequency spectrum, which is used by several Wi-Fi devices, has an occupation of 90%.

To avoid these problems the telecommunication regulatory agencies have auctioned frequency bands each time highest (tens of GHz), but this requires high investment by telecom operators and exhaustive development of standards and devices by industry. On the other hand, the growth of mobile and ubiquitous computing has increased the demand for wireless communications. Several solutions have been proposed to reduce

interference in wireless channels such as smart antennas, multiple radios (MIMO), filters, transmission power control, but these solutions do not efficiently explore the frequency spectrum. Other proposals found in literature use solutions based on Dynamic Spectrum Allocation (DSA) to avoid interferences in wireless communication systems and optimize the frequency spectrum. DSA uses mechanisms that include spectrum sensing, choosing the best channel/frequency available and dynamically reconfigure the radio device. These mechanisms have been the groundwork for the development of cognitive or intelligent radios [Correia et al. 2015].

The term Cognitive Radio was defined by Mitola as "*radio technologies that can make possible more intensive and efficient spectrum use by users licensed within their own networks, and by spectrum users sharing spectrum access on a negotiated or an opportunistic basis*" [Mitola 1999]. Moreover, these technologies include, among other things, the ability of devices to determine their location, sense spectrum use by neighboring devices, change frequency, adjust output power, and even alter transmission parameters and characteristics. Cognitive radio technologies enable spectrum exploring in space, time, and frequency dimensions that until now have been unavailable.

In [Commission 2003] the Federal Communications Commission states that cognitive technologies can reconfigure radios according to environment characteristics in real-time, offering to regulatory agencies the potential for more flexible, efficient, and comprehensive use of available spectrum while reducing the risk of harmful interference.

Cognitive radios (CR) emerge as a viable solution to avoid interference, improve network throughput and optimize frequency spectrum usage. Spectrum usage can be optimized by opportunistic frequency exploration in spatial and temporal dimensions, for example: if in a region there is a license for primary frequency use, but the frequency is not exploited, then secondary users could use it; or else, random or seasonal usage of licensed frequencies allow secondary users to opportunistically exploit them. In both cases, it is mandatory for secondary users to sense the spectrum so that, at any indication of transmission of primary users, the channel is released and there is an immediate migration to another frequency. To develop CR is important to create an abstract model with all its components, tools, services and applications. In this way, several CR can be connected to form a cognitive radio network (CRN) or a CRAHN (Cognitive Radio Ad Hoc Network). This model should consider that all elements of the CRN can perform spectrum sensing, have the capacity to exchange information, whether centralized or not and choose the best communication channel.

There are still many technical issues to overcome in cognitive radio. Among many challenges that still need to be solved, two of the most important are presented by [Yucek and Arslam 2009]: the problem of the hidden primary user, and the problem of the spread spectrum primary user. These two issues can lead a cognitive radio to incorrectly choose a frequency that seems to be empty, interfering in primary user's signal. To define the hidden primary user problem consider that there is a primary user A in range to transmit to B, and a secondary user C that is in range of B but not in range of A. C senses the spectrum and because it is out of range of A, it may conclude that there is no primary user. As B is in range to communicate with both A and C, but C cannot detect A, if C begins its transmission it will interfere with transmissions from A to B. Another problem is the

spread spectrum primary user. In this case, primary user's low power transmission may seem like background noise, as the cognitive radio may sense the spectrum and interpret primary user's signal as noise. In order to avoid this problem, the cognitive radio should sense a large of the frequency spectrum to identify the primary user.

Several papers propose the development of cognitive radio networks. Akyildiz et al. proposed a framework for spectrum management in cognitive radios [Akyildiz et al. 2008]. This framework is based on a cross-layer model in which the MAC layer reconfigures the radio based on application requirements as well as network state. The framework model proposed by Akyildiz et al. for spectrum management in cognitive radios is divided into four stages:

1. Spectrum sensing: A CR should be able to monitor the frequency spectrum to avoid interference with primary frequencies, and to search for unused frequencies (spectrum holes). The CR should consider the primary users in the region registered on regulatory agencies, as well as the time of sensing. Information collected by CR can be treated individually by nodes or concentrated and treated centrally by a node.

2. Spectrum decision: With the information about spectrum holes, the CR can choose an unused frequency. The decision method for selecting the best channel can use algorithms based on RSSI (Received Signal Strength Indicator), Artificial Neural Networks (ANN), Correlation, Analytic Hierarchy Process (AHP), Random Forests (RndF) and others. In addition, the decision method can also be influenced by local policies and regulations.

3. Spectrum sharing: The CRN is formed by several CRs that are trying to access the spectrum, often in the presence of other devices that are operating at near frequencies. The network must be coordinated to avoid collisions and to prevent overlapping of spectrum portions. This is achieved by the exchange of messages between nodes containing the best selected channel (or a list of best channels) to be used without causing interference or collisions.

4. Spectrum mobility: When the CR receives information about the best available channel, it reconfigures its radio to new frequency. Spectrum mobility is also employed to avoid that if a portion of the spectrum in use is required by a primary user, the communication must be immediately interrupted and can be continued in another available channel.

This framework proposed by Akyildiz et al. is only a theoretical model that did not present any real implementation.

A framework with a focus on spectrum decision and using Bayesian networks was developed to model spectrum correlation in CRN [Li and Qiu 2010]. The work was based on a graphical modeling and used a numerical simulator to implement analytical models and demonstrate the problems of interference between primary and secondary users. In spite of this, this framework did not consider the complexity and the dynamism of frequency spectrum.

A framework developed for SDR (Software Defined Radio) using GNU Radio was proposed by [Jagannath et al. 2015]. The objective was to implement modules for development and testing of new techniques for automatic classification of multiple signals. Despite the accuracy signal classification, and their importance in spectrum sensing, the authors did not present their application in CRN.

The authors [Marques et al. 2016] proposed an architecture for development of spectrum decision methods for CRN. The architecture was implemented in GNU Radio using broad spectrum bands on real hardware. From this research, it was possible to construct a generic framework component that can be altered to test new spectrum sensing and spectrum decision methods. In addition, it allows the inclusion of application requirements and definition of other quality of service policies. Its modular organization facilitates the testing of different medium access control protocols and spectrum sharing message exchange.

The following experiments will use this framework to demonstrate the viability of CR implementation in SDR. The framework is based on the abstract model proposed by Akyildiz et al. and therefore follows the four stage model for CR development.

3.3.1. Framework architecture

The framework architecture is based on a cross-layer model that integrates all modules and resources for spectrum exploration. The main feature of this model is to enable dynamic spectrum access, and it is similar to the four stage model proposed by Akyildiz, composed by Sensing, Decision, Sharing and Mobility. The abstraction of the cross-layer communication model for CRN is shown in Figure 3.3, with all layers, modules, blocks and interfaces used to define the communication in a CRN.

The application layer can define the quality of service requirements and message traffic type. The quality of service policies can be established by the application to define requirements in terms of bandwidth, latency, and others. These policies directly influence MAC layer's behavior, for example, radio parameters, duty cycle operation, and sensing frequency. The traffic generation module is responsible for generating traffic patterns used for message exchanges. This module was developed to facilitate experimentation, generating traffic for evaluation of framework modules (or blocks in GNU Radio), and simulate the behavior of a real application. Three different distributions were implemented for packet sending interval: uniform, constant, and exponential. In addition, this module includes continuous and discrete distributions [Saucier 2000]. For the experiments presented in this work, only the uniformly distributed traffic model was considered.

Transport and Network layers were not implemented at this stage of the framework. As the experiments focused on spectrum sensing, decision, sharing, and mobility, the services provided by these layers were not essential. It is important to note that the CRN implemented in this work is based on unicast communications. Nevertheless, the modular design of the framework allows the insertion of additional modules to include transport and routing capabilities to form a CRN with multihop communication (CRAHN).

On MAC layer two major modules were defined: cognitive engine and medium

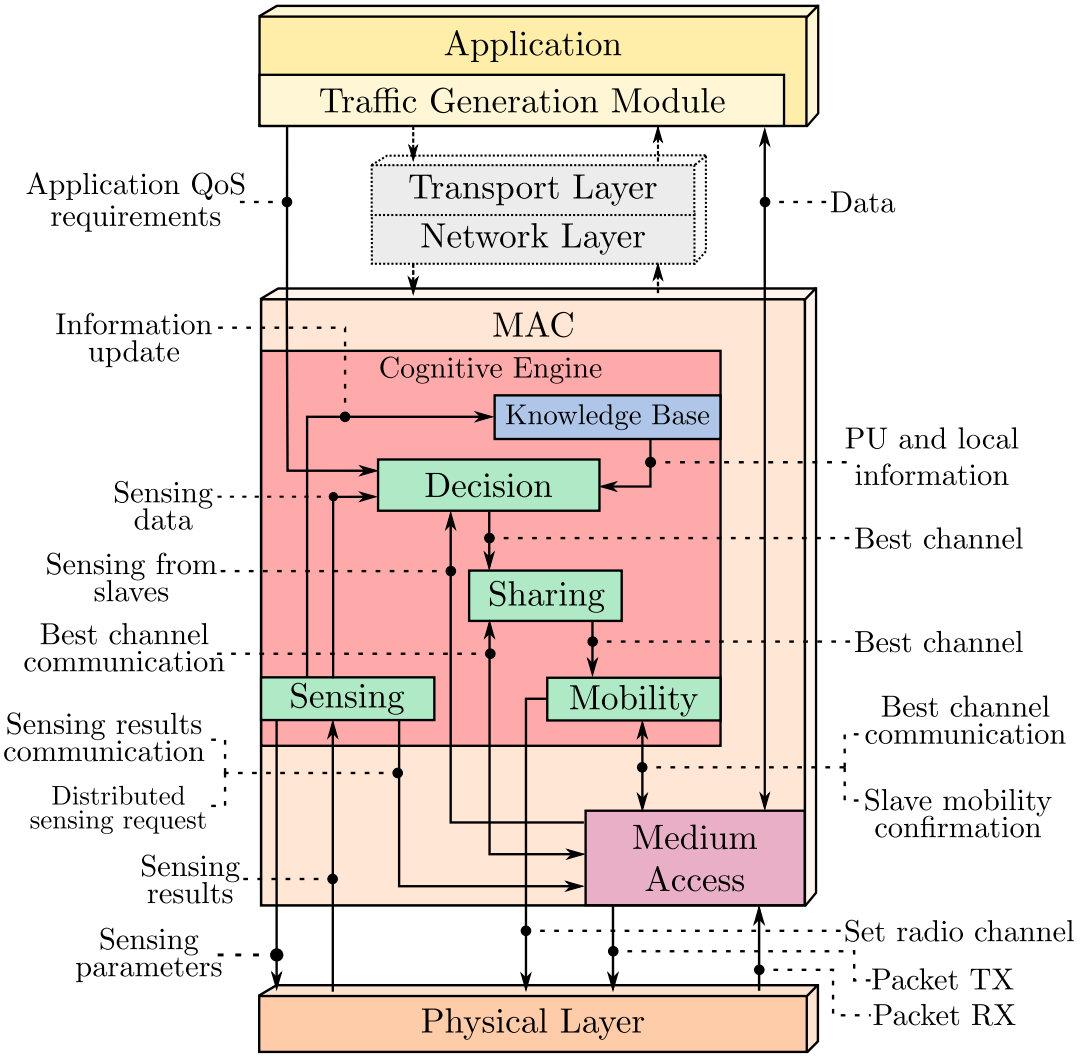

Figure 3.3: Framework architecture.

access. The cognitive engine is composed of the four stages of a cognitive radio and a knowledge base. Licensed users or primary users (PU) register the contracted frequencies in regulatory agencies to operate telecommunication services. The knowledge base may consist of: a database provided by regulatory agencies, information collected through local spectrum sensing, and information entered manually by the administrator, based on policies or regulatory laws of countries or regions.

Knowledge base information can be dynamically updated by data collected from local spectrum sensing. This information can be preprocessed or not and will be used as input to spectrum decision methods (SD). In this framework, a database provided by ANATEL [8] was inserted as a block, which can be used to get information of PUs in a specific region at a given time. In addition, the local spectrum sensing data also can be entered into this database.

The Spectrum Sensing (SSe) communicates with the physical layer by sending

[8] Agência Nacional de Telecomunicações - (Brazilian Regulatory agency)

sensing parameters and commands, and collecting the results. Spectrum sensing can be done in two ways: distributed, executed by all nodes; or centralized, executed only by the master node. In distributed mode, the master node requests sensing from all slave nodes by messages transmitted through medium access block. The slaves execute spectrum sensing and send information back to the master node, which combines all the received information in the spectrum decision module.

The main block of the cognitive engine is the Spectrum Decision (SD). All intelligence methods can be implemented within this block. This framework provides four methods for spectrum decision: Artificial Neural Network (ANN), Analytical Hierarchical Process (AHP), Random Forest (RndF) and a simple method based in received signal strength (CogMAC) [Marques et al. 2016]. These decision methods are affected by the input parameters: QoS, spectrum sensing and knowledge base. Based on these inputs, the methods decide which is the best channel that avoids interference with primary users and use the spectrum opportunistically. After the SD choose the best channel, it is necessary to share this information with the other nodes.

The Spectrum Sharing (SSh) module communicates the best channel to the slave nodes. In this framework a 6 GHz common control channel (CCC) is used to initialize communication between nodes, to exchange control messages, and as a fallback channel. The choice of this channel is based on empirical data and on the list of PUs provided by ANATEL. The message exchange is also controlled by the Medium Access module.

Information about the best channel is also passed to Spectrum Mobility (SM) block. The master sends messages informing the best channel to all slave nodes, and then migrates to the best channel after receiving mobility confirmation of at least one slave.

The medium access module is responsible for collision avoidance, and controls transmission of packets originated from upper layers and from the cognitive engine. Before transmitting a packet, it performs a Clear Channel Assessment (CCA) to verify medium state. If the channel is free the message is sent by radio. Otherwise, if the medium is busy, the message is delayed by a backoff algorithm. The transmission is retried until the message is delivered or application's timeout occurs. Furthermore, packet reception is also controlled by medium access module.

The physical layer has functions of adjusting radio parameters and transmitting and receiving packets over a wireless channel. In addition, all spectrum sensing is performed on this layer within a range of 800 MHz to 5.8 GHz.

The communication model is shown in the next section.

3.3.2. Communication model and message types

The framework uses a communication model based on the Master-Slave paradigm. Communication is established between two adjacent nodes (one hop distance), so there is no multihop communication. Figure 3.4 shows the message flow between the master node and two slaves nodes.

The master node sends messages in broadcast using the CCC (6 GHz) to start the neighbor discovery phase (ND). Nodes in range reply with $ACK(ND)$. After the discovery

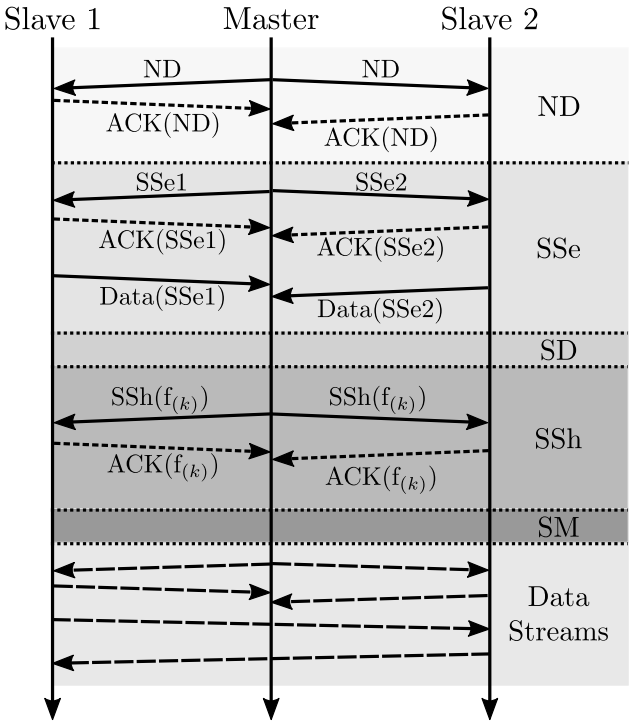

Figure 3.4: Message flow between three nodes.

phase, the master node sends sensing requests messages (SSe_n) to all its 1-hop neighbors, waits for confirmation messages and enters idle mode. If a neighbor node does not reply after several retransmissions it is considered disconnected.

Slave nodes (S_n) perform spectrum sensing and send the collected data to the master node. All spectrum sensing is performed in 1 MHz steps within the frequency range of 800 MHz to 5.8 GHz. Spectrum decision (SD) phase initiate when the master node receives the spectrum sensing data ($DATA(SSe_n)$) from slave nodes. The best channel decision can be performed by different algorithms. In this framework, there are four algorithms and a knowledge base that contains the historic of primary users in that region (provided by a regulatory agency). The decision method receives as input parameters of QoS, collected sensing data from slaves, and PU historic. After processing, the method returns the best channel.

The best channel selected is individually informed to each slave node by the master by sending $SSh(f_{(k)})$ messages. Upon receiving at least one $ACK(f_{(k)})$ the master enters spectrum mobility phase (MD) and configures its radio to the best channel.

After the spectrum mobility phase (MD), all nodes can communicate with their neighbors. This communication can be established between master with any slave or between slaves.

3.3.3. Cognitive radio experiments

In real environments interference and noise influence radio communication. These characteristics have high complexity to be mathematically modeled, resulting in inaccurate transmission analysis through simulations.

This section presents experiments using real hardware, the USRP B200 and B210 daughterboards. These are Software Defined Radio (SDR) devices that have the capability of accessing frequency spectrum in ranges between 70 MHz and 6 GHz.

The network communication model is based on the master and slave paradigm, and the experiments use one master and one slave node. The master node is responsible for centralizing all information of spectrum sensing and apply spectrum decision methods to select the best channel according to application requirements. Slave node performs sensing and sends the collected information to the master node. Although, the network allows the use of multiple slave nodes.

The main objective of these experiments is to present a CRN composed by two nodes, which employ intelligence methods to dynamically access the spectrum, selecting channels with low noise ratio, and avoid interference on primary users.

This CRN uses the four stage model presented on Figure 3.3. In these experiments, only the sensing and decision methods are used, nevertheless, the other methods are fully implemented in the framework. The experiment will be performed in two steps, with each one focusing on a specific method: the first experiment demonstrates spectrum sensing phase, and the second presents the spectrum decision.

Preparing the Environment

The requirements to perform the experiments are described below:

- GNU Radio version 3.7.x.
- Ubuntu 14.04.
- Framework code, available on `http://github.com/GrubiCom/FrameworkCRN`
- 2 USRPs models B200 or B210, with proper cabling and antennas.
- 2 computers with VOLK library support[9].

The framework can be downloaded from GitHub with the command:

```
git clone http://github.com/GrubiCom/FrameworkCRN
```

Two scripts in the framework are used to configure the environment. The first, `setup_dependencies.sh`, installs the software necessary for running the framework. The second script, `build_blocks.sh`, configures additional GNU Radio blocks of the framework. The following commands must be run on the machines used for master and slave USRPs:

```
user@ubuntu:~$ sudo chmod +x setup_dependencies.sh
user@ubuntu:~$ sudo setup_dependencies.sh
user@ubuntu:~$ chmod +x build_blocks.sh
user@ubuntu:~$ ./build_blocks.sh
```

[9] https://gnuradio.org/doc/doxygen/volk_guide.html

Cognitive Radio Experiment 1: Spectrum Sensing

The objective of this experiment is to perform spectrum sensing using USRP with the master-slave architecture. At the end of this experiment, it will be possible to verify the state of the sensed spectrum. The GNU Radio blocks that will be used in the master-slave architecture are represented in the figures 3.5 and 3.6.

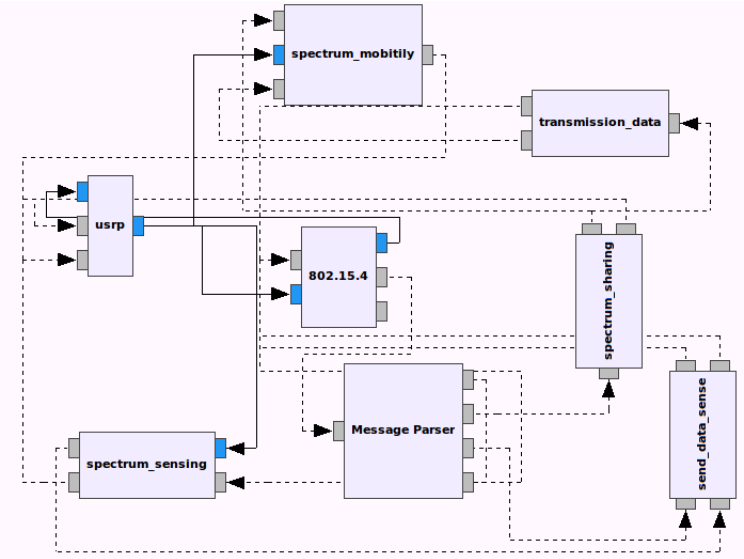

Figure 3.5: Slave node block diagram.

In Figure 3.5, the block diagram of the slave nodes is presented. This type of node does not implement spectrum decision, since it does not execute this phase.

Block diagram of the master node is presented in Figure 3.6. This type of node, on the other hand, implements all four stages of cognitive radio, and concentrates the sensing data to execute spectrum decision. For this experiment, the necessary files are located on folder `gr-pmt_cpp/grc`. Spectrum sensing can be run in two different modes:

- In graphic mode, go to File menu and open `slave.grc` file on slave computer, then open `master.grc` on master computer. To execute the code of each node, click on the green arrow on GNU Radio interface (Execute the flow graph).
- Using the terminal, running the command `grcc -e slave.grc` on slave computer, and `grcc -e master.grc` on master computer.

During the execution in graphic mode, the terminal on GNU Radio's interface of the master node shows a time counter while it waits for sensing replies. On the slave node, the terminal shows the current sensed frequencies.

After finalizing the spectrum sensing, the slave stores the sensed results on `/tmp/` folder. The file `sense.txt` contains raw data, with up to 16 samples of each frequency, during 8 ms. For each frequency, the slave selects the sample with highest noise, and stores on `send.txt` file, which will be transmitted to the master node for spectrum decision.

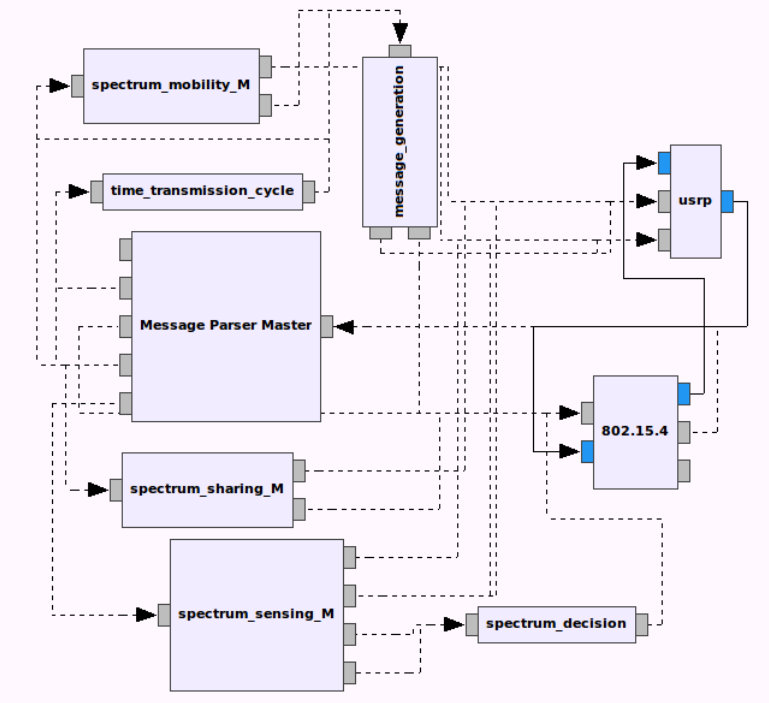

Figure 3.6: Master node block diagram.

On the master node, data received is stored on `/tmp/res_sense.txt`. The collected data can be visualized on slave using the script `slave_freq_plot.sh`, while on master node the results can be viewed with `master_freq_plot.sh`.

At this point, the first experiment is finalized, and the CRN is ready to start the second stage, spectrum decision.

Cognitive Radio Experiment 2: Spectrum Decision

Following the experiment 1, it is necessary to select the best channel. This is performed by the master node, during the spectrum decision phase. In this phase, the master combines sensing information received with information on its knowledge base, and application QoS requirements.

The decision method used in this experiment is based on ANN. In Figure 3.6, the decision block is named `spectrum_decision`, being able to decide the best channel with accuracy of 99.9996%. This stage is carried on immediately after spectrum sensing, and no additional executions are needed. After spectrum decision, the best channel frequency is stored on `/tmp/master_channels.txt`, and can be visualized with `best_freq_plot.sh`, both on master and slave computers.

After choosing the best frequency for data transmission, the master node informs the frequency to slave nodes. Slave nodes acknowledge the reception of this information, and upon receiving at least one confirmation, the master node modifies its channel to the best frequency. With the nodes operating in the new channel, data transmission starts and lasts for 60 seconds, when the network repeat the cognitive process.

3.4. Dynamic change of the MAC protocol in WPANs

This section will present how the MAC protocol influences the performance of a WPAN network. It is known that each family of protocols (contention-based and contention-free) perform best on certain scenarios: contention-free MAC protocols (e.g TDMA) are best for crowded networks, however they may have a large overhead and underutilize the medium on uncongested networks. On the other hand, contention-based MAC protocols (e.g. CSMA/CA) present a very low overhead, however, their performance degrades when many stations compete for the medium.

In this experiment, we'll use a rule-based system to change from one MAC protocol to the other in an IEEE 802.15.4 network. We will code our own rules into a set of existing GNU Radio modules, and will evaluate how our system behaves for a varying number of stations transmitting data.

3.4.1. MAC protocols

Multiple access methods in wireless networks are used when the medium is shared and some form of organization is needed so that different nodes can transmit their data satisfactorily. There are several protocols that fulfill this role and they can be divided into two broad categories: multiple access protocols contention-based and contention-free, as shown in Figure 3.7.

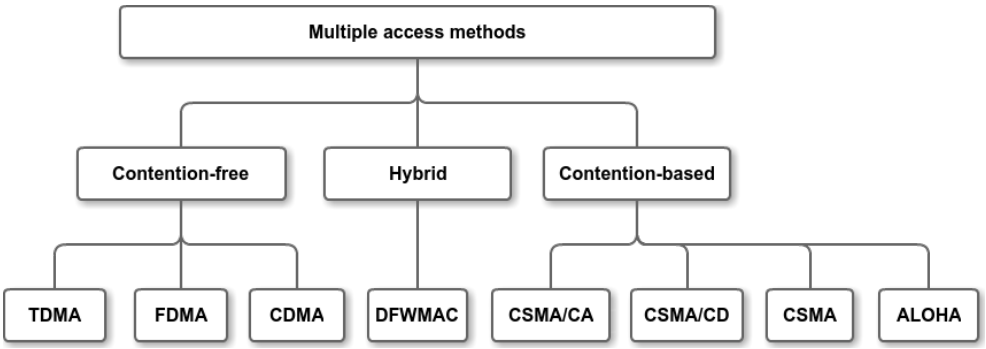

Figure 3.7: Multiple access control methods

In contention-based methods, the transceivers do not have strong restrictions to access the medium. In this case, access does not require coordination to use the channel and the decision to transmit is taken locally. There is also no specific time delimitation within which the transmission should be performed. For these reasons, contention mechanisms are adopted in the protocols to reduce the number of collisions and to avoid channel saturation and thus the network to function satisfactorily. Generally, when a collision occurs, the node waits a random time and repeats the transmission attempt.

The first protocols that used this approach were Pure ALOHA and Slotted ALOHA. This approach was improved by including the verification of the medium before the transmission attempt, resulting in the CSMA (Carrier sense multiple access) and later, its variations with collision detection (CSMA/CD) and collision avoidance (CSMA/CA). In the approach of these protocols, the attempt of transmission is direct, depending only on the carrier sense and the waiting of some previously defined timings. This is positive because

in case of no collisions, the use of time is optimized. However, with this approach, the number of collisions tends to increase with increasing devices trying to communicate. At certain point, the number of collisions may be large enough to hamper the operation of the network [Takagi and Kleinrock 1985, Ziouva and Antonakopoulos 2002].

In contention-free methods, the medium is accessed in a coordinated way, and there is no need for mechanisms to resolve access conflicts and collisions. In this case, the allocation control of the channel can be done by a centralized entity, which coordinates the transmission order, or by some distributed approach as token passing or distributed queue. Some examples of protocols that do not use contention mechanisms are Frequency Division Multiple Access (FDMA), Time Division Multiple Access (TDMA), and Code Division Multiple Access (CDMA) [Busch et al. 2004].

TDMA-based protocols, for example, divide time into slots where each device transmits individually. Since a slot is reserved for only one device, there are a small number of collisions, no matter how many devices are trying to communicate. Therefore, it is a good approach when the network has many nodes trying to transmit. Despite this advantage, when there are few transmission attempts, the control messages used in the slots allocation procedure consume a relative large time. So, these protocols are best when the network is congested, but they lose performance when there are few nodes trying to communicate.

There is also a hybrid approach, which uses combined contention-based and contention-free methods. In this approach, each mode operates for a certain time and there is a switch between the two modes at the end of each period. In the Contention Period (CP), the medium access organization is distributed, and in the Contention Free Period (CFP), the medium use is coordinated by a access point. DFWMAC is an example of protocol that uses this approach [Diepstraten and WCND-Utrecht 1993].

3.4.2. Experiments

In the experiments performed in this part of the course, we use the FS-MAC platform [Cordeiro et al. 2017] to automatically switch between a contention-based (CSMA/CA) and a contention-free protocol (TDMA). This platform works with these two protocols putting each one in operation according to some predefined rules. These rules are part of a fuzzy system that uses the information about *number of senders* and *packet delivery latency* to infer the contention level of the network and define which protocol should operate at each moment. Further details on the architecture and operation of the FS-MAC platform can be found at the address:
`https://github.com/jeffRayneres/FS-MAC`.

In the following sections we describe the settings required to use the FS-MAC platform, then we describe the experiments.

3.4.2.1. Configuration

To perform the experiments, we must install the FS-MAC platform and its dependencies. The dependencies, with their respective addresses, are:

- *gr-ieee802-15-4*: `https://github.com/bastibl/gr-ieee802-15-4`
- *gr-foo*: `https://github.com/bastibl/gr-foo`
- *gr-eventstream*: `https://github.com/osh/gr-eventstream`
- *gr-uhdgps*: `https://github.com/osh/gr-uhdgps`

The *gr-ieee802-15-4* project provides the ZigBee stack used as the basis for building the FS-MAC platform, and the *gr-foo* project is a dependency on that stack. The *gr-eventstream* and *gr-uhdgps* projects are used in the CSMA/CA protocol of the FS-MAC platform in the Carrier sense process. A detailed description of the *gr-ieee802-15-4* installation can be found in [Silva et al. 2015]. In addition to these projects, we must also install the libraries *liblog4cpp5-dev* and *python-matplotlib*.

The FS-MAC platform can be downloaded from the address `https://github.com/jeffRayneres/FS-MAC`, or the user can use the command:

```
git clone https://github.com/jeffRayneres/FS-MAC
```

After downloading the project in the FUTEBOL testbed, go to the /gr-fsmac directory and run the following commands:

```
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig
```

After executing these commands, the FS-MAC platform will be installed and the environment will be prepared for running the experiments.

In the experiments we use resources provided by the FUTEBOL project, so in directory */gr-fsmac/examples* of the FS-MAC platform there is an RSpec file to be used with jFed for resource allocation. To improve execution, we do not use the graphical mode of the GNU Radio Companion, so all information generated during the experiments, including which protocol is in operation, will be displayed on the terminal window.

3.4.2.2. Dynamic Change Experiment 1

The objective of this experiment is to exchange the MAC protocol in operation on the network automatically according to some rules based on the amount of contention on the wireless medium. The file *decision.py* in the */gr-fsmac/python* directory of the FS-MAC platform, implements *fuzzy* system that receives as input the number of senders in the network and the average latency of package delivery. Its output is the effectiveness of

a certain MAC protocol. The system employs fuzzy rules that determine which protocol should operate at a given moment, according with the network contention level. The fuzzy system was modeled as follows:

Linguistic variables: The model considers three linguistic variables, which are (*i*) Average latency of packets delivery (*AL*), *ii*) Number of senders (em NS) and *iii*) Adaptability of the protocol (*ADP*). They all accept the fuzzy terms *LOW* and *HIGH*.

Fuzzy rules:

```
CSMA
If NS is LOW and AL is HIGH then ADP is HIGH
If NS is LOW and AL is LOW then ADP is HIGH
If NS is HIGH and AL HIGH then ADP is LOW
If NS is HIGH and AL is LOW then ADP is HIGH

TDMA
If NS is LOW and AL is HIGH then ADP is LOW
If NS is LOW and AL is LOW then ADP is LOW
If NS is HIGH and AL is HIGH then ADP is HIGH
If NS is HIGH and AL is LOW then ADP is LOW
```

Membership functions: The membership functions are shown in the graphics of Figures 3.8, 3.9, 3.10 and 3.11 (source [Cordeiro 2017]):

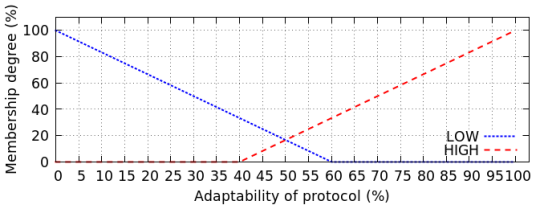

Figure 3.8: Membership functions for linguistic variable “Adaptability”.

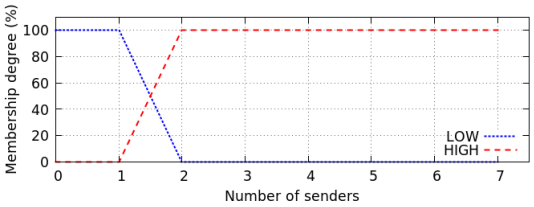

Figure 3.9: Membership functions for linguistic variable “Number of senders”.

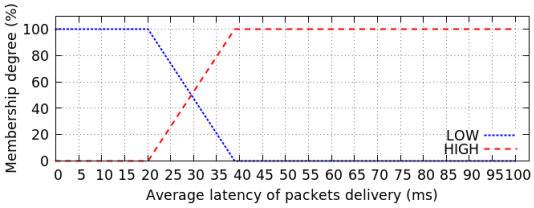

Figure 3.10: Membership functions for linguistic variable “Average latency - CSMA”.

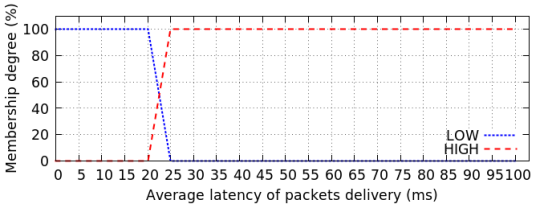

Figure 3.11: Membership functions for linguistic variable “Average latency - TDMA”.

In this experiment, we use three USRPs to simulate sensor nodes. Each USRP is connected to a computer installed with GNU Radio version 3.7. We call *StationN* the set

formed by a USRP connected to a computer, so in the experiments we use a group formed by Station1, Station2 and Station3.

In the directory */gr-fsmac/examples* of the FS-MAC platform, there are three files configured to be used in the stations. The names of these files are *transceiverStation1.py*, *transceiverStation2.py* and *transceiverStation3.py*. These files are scripts generated by GNU Radio Companion, they contain all the necessary settings for the transmissions in this experiment. These settings include:

- Preparation of a message with 110 bytes to be sent.
- Setting the send interval to 20 ms.
- Setting the transmit power to 60 dBm.
- Setting the frequency used to 2.48 GHz.
- Setting the transmitter and receiver MAC address.

In addition to transmitting data packet and acknowledgement packet, Station1 operates as FS-MAC Coordinator. In the FS-MAC platform, the Coordinator node is the one that coordinates the exchange of the protocol in operation when necessary.

The experiment starts with the transmission of packets from Station1 to Station2. After some time, without interruption of Station1's transmission, the new transmission starts from Station2 to Station3. After some time, keeping the transmissions in progress, a new transmission starts from Station3 to Station1. When each sender is included, we check in the terminal which protocol is in operation. With the rules configured in the *fuzzy* system, in the *testbed* that we use, the contention level that indicates the moment of exchange of the protocol must occur when the network changes from one to two senders. Thus, when there is only one sender, the system operates with CSMA/CA, with two senders or more, the system automatically starts to operate with TDMA.

3.4.2.3. Dynamic Change Experiment 2

The purpose of this experiment is to change the rules of the *fuzzy* system so that the MAC protocol exchange in the network occurs at a different contention level from the previous experiment. To do this, we need to change the *decision.py* file in the */gr-fsmac/python* directory of the FS-MAC platform, we need to modify the functions *senders function()* (line 270) and *data function()* (line 250). These functions are responsible to calculate the membership of the values of *Packet delivery latency* and *Number of senders* to the HIGH and LOW sets. They reflect the membership functions of Figures 3.9, 3.10 and 3.11. We'll modify them to reflect the membership functions shown in Figures 3.12, 3.13 and 3.14.

In this experiment we changed the membership functions so that the FS-MAC platform identifies the moment of exchange of the protocol in operation when the contention level is slightly higher than in the case of the previous experiment. After the functions have changed, we must reinstall the FS-MAC platform.

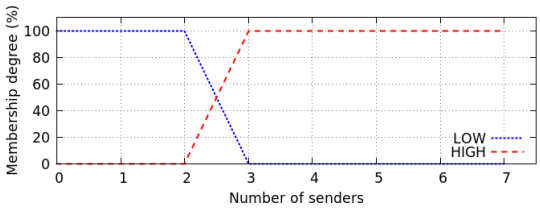

Figure 3.12: Membership functions for linguistic variable “Number of senders”.

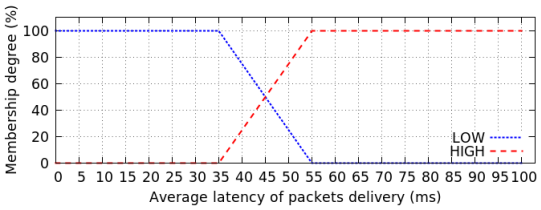

Figure 3.13: Membership functions for linguistic variable “Average latency - CSMA”.

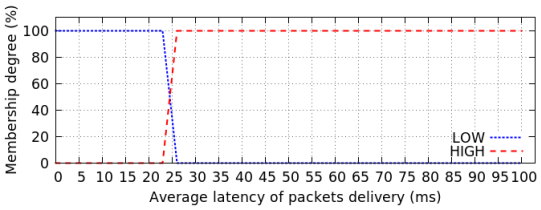

Figure 3.14: Membership functions for linguistic variable “Average latency - TDMA”.

In terms of execution, this experiment follows the same dynamics of Experiment 1, that is, it starts with one sender and adds other sender up to a total of three. When each sender is added, we check in the terminal which protocol is in operation. With changes in the *fuzzy* system, the contention level that indicates the moment of exchange of the protocol must occur when the network changes from two to three senders. Thus, when there are one or two senders, the system operates with CSMA/CA, with three senders, the system automatically starts to operate with TDMA.

3.5. Reliability in WBANs

Wireless Body Area Networks (WBANs) consist of a wireless network composed of several wearable or implantable computing devices. Although WBANs can be applicable in various fields [Movassaghi et al. 2014], this section focuses on healthcare, in which the application is the monitoring and transmission of physiological signals to medical servers. In this part of our course, we will perform a reliability experiment in WBANs. First, we evaluate the classic IEEE 802.15.4 protocol stack, measuring the amount of transmitted data from both the sensor and the base station points of view. With this data in hand, we will evaluate the simplified protocol stack, quantizing the packet delivery efficiency. Next, we move forward using a more robust IEEE 802.15.4 protocol stack. This stack implements acknowledgment functionality providing single channel communication between devices with support to a three-way handshake. Finally, we will compare both protocols packet delivery efficiency.

3.5.1. General dependencies

In order to perform the experiment, we need to install some modules and complementary libraries.

First, we should download the module **gr-foo** developed by [Bloessl et al. 2013].

This module contains blocks responsible for the configuration of sending and displaying the data packets. For this experiment, we will use only the part in charge of visualizing the received data packets. The reader can get this module by running the following command on the terminal:

```
git clone https://github.com/bastibl/gr-foo.git
```

Other non-native GNU Radio plug-ins to download are **gr-eventstream** and **gr-uhdgps**. Both composing the set of blocks entrusted by performing the Carrier Sense function. However, prior to the installation of these, it is necessary to install the following additional libraries with their respective commands:

```
sudo apt-get install liblog4cpp5-dev
sudo apt-get install python-matplotlib
```

To obtain the **gr-eventstream** module, execute the following command:

```
git clone https://github.com/osh/gr-eventstream.git
```

Similarly the command for the **gr-uhdgps** module is:

```
git clone https://github.com/osh/gr-uhdgps.git
```

Finally, we should download the module **gr-traffic_generator**, which is in charge of generating dynamic messages in a variable interval of time. The traffic generator block receives as parameters one value for the size of the message and another value for a time interval. These parameters can be generated from the distribution type (Pareto, Poisson, Weibull, Zipf or Uniform) with the help of the Distribution block, or defined by the user. The reader can download this module by running the following command:

```
git clone https://github.com/AdrieleD/gr-traffic_generator.git
```

The installation of each additional module downloaded is simple and follows the GNU Radio standard. Simply access their respective folders by creating a folder named build in the root of the project, access that new folder and execute the following commands:

```
cmake ../
make
sudo make install
sudo ldconfig
```

To uninstall, from within the **build** folder, just run the command:

```
sudo make uninstall
```

After the installation of each of the modules and complementary packages, the environment is able to receive the packages related to the respective experiments. Each experiment will depend on the main module containing its corresponding protocol stack, one simplified and one more robust, respectively.

3.5.2. WBAN Experiment 1

The objective of this experiment is to transmit data from a sensor node to a receiving station. This experiment uses real communication devices, where it is simulated sensor nodes and a base station. Furthermore, we will use a simplified protocol stack, which is only responsible for packaging and sending the data. At the end of the experiment, it will be possible to visualize the data transmitted from one device to another and compare them to each other. The purpose of this verification is to evaluate possible data loss and interference by using the simplified stack. We should download and install the **gr-mac1** by executing the following command:

```
git clone https://github.com/AdrieleD/gr-mac1.git
```

After performing all the steps for installing the protocol stack module, we can view the **gr-mac1** library in GRC. To complete the installation of this module, we need to install the hierarchical block, opening the *gr-mac1/examples/ieee802_15_4_OQPSK_PHY.grc* file in GRC and compile it (Generate the flow graph / F5 key) or using the following command line (we recommend the user to restart the GRC environment after this procedure):

```
grcc ieee802_15_4_OQPSK_PHY.grc
```

Please open the *gr-mac1/examples/transceiver_OQPSK_TX.grc* file to check if installation has been successful. All blocks should be properly connected and with no error messages. It is important to emphasize that to carry out the experiment, we need at least two computers, one to act as transmitting node and another as base station. Although the protocol stack used in both is the same, we need to make some changes according to the function that the application will perform. If you have access to the GCR graphical interface, you should edit the *gr-mac1/examples/transceiver_OQPSK_TX.grc*. When it comes to the receiving application, traffic generating blocks are not necessary, so they can be disabled or even deleted. If you are using the testbed terminal you do not need to perform any modification.

For the experiment to work properly, we should also update the address configuration, found in the *gr-mac1/lib/mac.cc* file. Both the source address and the destination address must be different. If such a change is necessary, it can be done by changing lines 60 and 355. In Table 3.1, we are show an example of how to configure two computers. In addition, we need to change the location where the file containing the experiment data will be saved. This configuration is performed in the File Sink block of the application which represents the base station. For the transmitting node, the use of the Wireshark Connector and File Sink blocks is not required. After any and all file editing of the blocks, it is

essential to reinstall the blocks so that they can be effective. This is possible through the following commands:

```
cd gr-mac1/build
sudo make uninstall
sudo make install
sudo ldconfig
```

Table 3.1: The configuration of source and destination addresses for two computers.

	Computer 1	Computer 2
Source Address	60 char mac_addr_1 = 0x41;	60 char mac_addr_1 = 0x40;
Destination Address	355 addr0[0] = 0x40;	355 addr0[0] = 0x41;

After completing these steps, the environment is configured and able to start the experiment. Just open the *gr-mac1/examples/transceiver_OQPSK_TX.grc* file (Figure 3.15) and execute (F6 key) on the GNU Radio graphic interface. It is also possible to use the testbed terminal. In this case, you can compile and execute the respective files according to the desired computer function. The following commands initialize the computer as a receiver or transmitter, respectively:

```
grcc -e transceiver_OQPSK_RX.grc
grcc -e transceiver_OQPSK_TX.grc
```

In this experiment, the computer acting as base station should be initialized first. In this way we will start the communication between the nodes, always remembering that each computer with its respective USRP unit is equivalent to a node. Thus, for example, we can have 4 transmitter nodes and 1 base station, we need 5 computers and 5 USRPs.

The transmitting application is configured to send the "Hello world!" message endlessly with a one-second interval between messages. These settings can be changed by configuring the properties of the Periodic Message Strobe block. Once the experiment is started, the sensor node configured as the base station is able to receive and confirm the messages transmitted by any transmitting node, as long as both are configured on the same channel. The selection of the channel occurs manually, manipulating the settings of the Variable block with ID "freq".

Since the content of the message sent by the transmitting entities is static and predetermined, the visualization of these via an interface or the persistence in file becomes irrelevant. However, the confirmation of the sending and receiving of the packet by the base station is essential so that the full transfer of data between the devices can be identified. The persistence of data by the receiving station is a must. Once stored, they can be analyzed, compared and used to produce information. This data retention is the responsibility of the Wireshark Connector block, which saves the messages in a .pcap file. In this way, the reader can check the newly generated .pcap file if the message sent by the transmitter was correctly received by the USRP. This file can be better analyzed with the help of Wireshark (Figure 3.16), where we can view the transmitted messages and their respective text content.

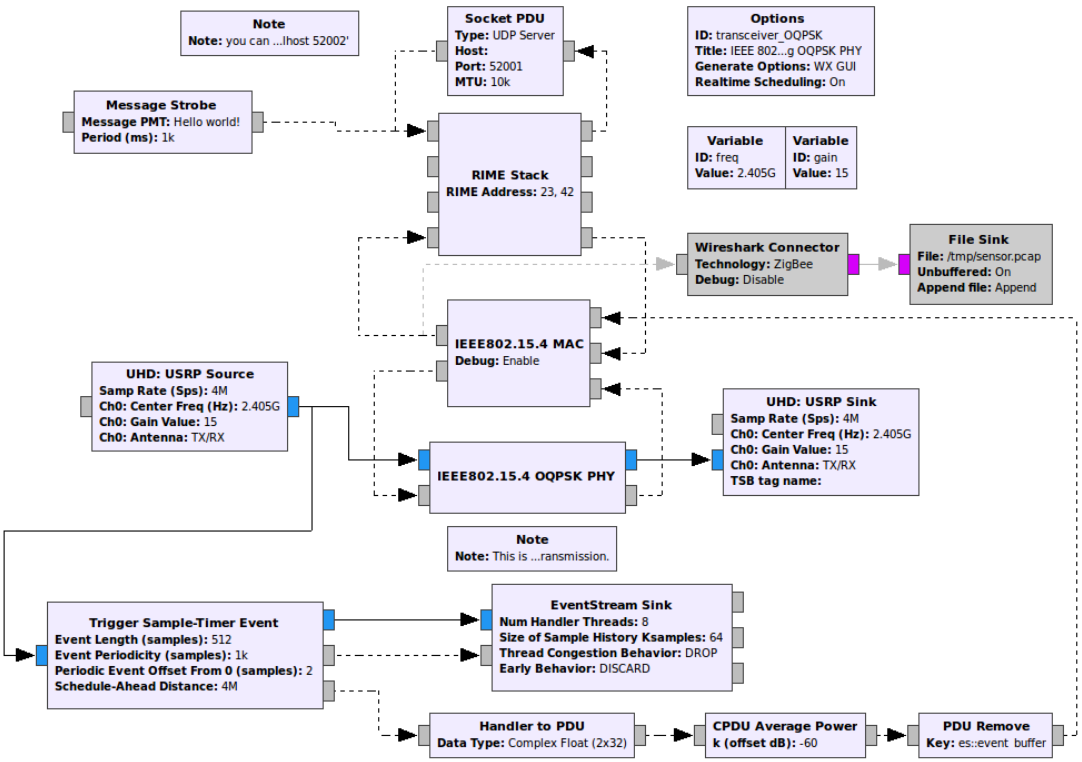

Figure 3.15: Graphical display of protocol stack for experiment 1.

3.5.3. WBAN Experiment 2

This experiment is an improvement of the previous one. We will also perform a data transmission between the devices, sensors and base station, but using of a more **robust protocol stack**. The stack used in this protocol was implemented based on the stack of the first experiment, but it has features that provide a higher quality and reliability in data transmission. Included in these functionalities is the Frequency Hopping technique, which provides interference inhibition and better use of the frequency spectrum. Three-way Handshake is another utility, responsible for establishing a single channel of communication between two devices. Another objective of this experiment is to evaluate, as in the previous one, the quality of data delivery. We also provide a quality comparison of the data transmitted by both protocols. First, we should download the **gr-mac2** module through the following command:

```
git clone https://github.com/AdrieleD/gr-mac2.git
```

We should also carry out the complementary modules installation process. As well as in the previous experiment, after completing all the steps of installing the module referring to the protocol stack, it becomes possible to view the gr-mac2 library in GRC. To complete the installation, it is essential to compile the hierarchical block *gr-mac2/examples/ieee802_15_4_OQPSK_PHY.grc* in GRC (Generate the flow graph / F5 key). Similarly to experiment 1 it is advisable to restart the GRC environment and check the installation success by opening the *gr-mac2/examples/transceiver_OQPSK_TX.grc*

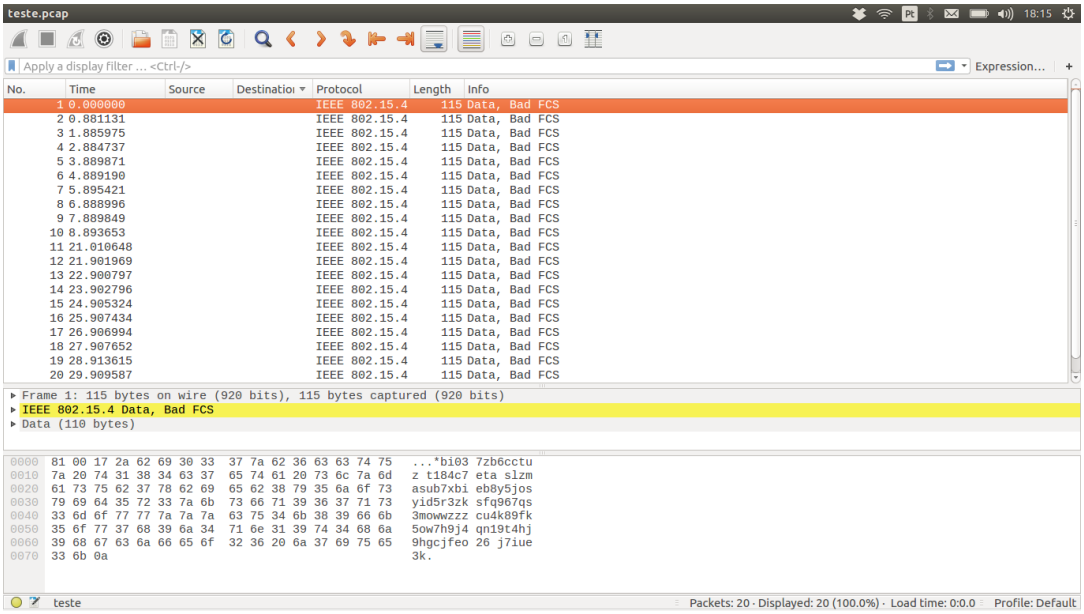

Figure 3.16: Viewing data packets through Wireshark.

file where all blocks must be properly connected and without error messages. You can also perform this step executing the following command in the testbed terminal:

```
grcc ieee802_15_4_OQPSK_PHY.grc
```

This experiment requires at least two computers, one acting as a transmitter and another as a receiver. The stack used for both projects is the same, but we should make some configurations to act according to their specific functions. Please refer to **WBAN Experiment 1** instructions on how to configure the nodes as transmitter or receiver.

After we have completed all the above procedures, the environment is configured and the user can start the experiment. To do this, simply click the Execute the flow graph icon or use the F6 hotkey. In this experiment, the initialization order of the devices is not important, but it is suggested that the node with base station characteristics be started first. Similarly to the first experiment, each node is conditioned to the existence of a computer with a USRP unit. You can also execute the experiments using the following commands to initialize the computer as a receiver or transmitter, respectively:

```
grcc -e transceiver_OQPSK_RX.grc
grcc -e transceiver_OQPSK_TX.grc
```

Unlike experiment 1, the content and time between messages are dynamic. They can be changed according to the configuration of the Distribution, **Traffic_Generator** and **Traffic_Generator_Random** blocks. In order to be viewed by the transmitting application, it is necessary to include a Message Debug block connected to the output of the traffic generator module. By enabling the "Debug" option of the IEEE 802.15.4 MAC block, it is also possible to observe the attempts to connect and transmit the packets with their respective confirmations, in addition to the end of the connection. A log is made available at the end of all submissions with information regarding data transmission.

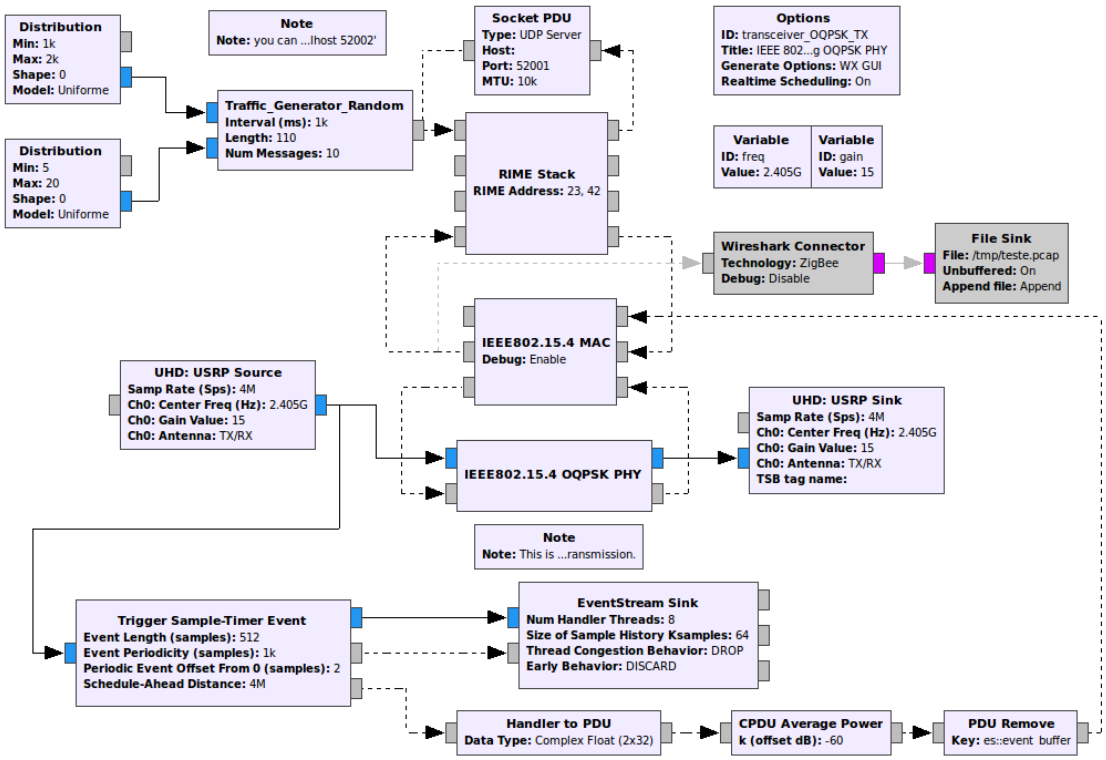

Figure 3.17: Graphical display of protocol stack for experiment 2.

In the base station application also with “Debug” option enabled, is visualized the establishment of the communication such as its closure, messages referring to received data and information about the “jumps” between the channels. For a detailed verification of the received data, the newly generated .pcap file is analyzed with the help of Wireshark, where we can see the transmitted messages and their respective text content as Figure 3.18 depicts. In this Figure, we can see: 1) Command to finish communication; 2) Finished MAC; 3) Total communication time; 4) Total number of sent packets; 5) Total number of confirmed packets; 6) Number of retransmissions; 7) Packet throughput; 8) Bytes throughput; 9) Packet delivery rate; 10) Packet latency.

3.5.4. Comparison between protocols

As is remarkable, the stack structure of the protocols used in the two experiments is similar (Figures 3.15 and 3.17). Taking into account a basic application, both would perform significantly. However, more complex applications, such as WBANs, require functionality that ensures reliability, accuracy, and agility in data transmission and confirmation. In addition to the basic characteristics necessary for communication between one or more transmitter nodes and a base station, the protocol of experiment 2 has other functionalities such as Frequency Hopping, Handshake. Table 3.2 details both protocols characteristics.

In the Handshake process, the transmitting node requests the establishment of communication by sending the request command (RTS). Considering the immediate availability of the base station, upon receiving this command a confirmation is sent along with

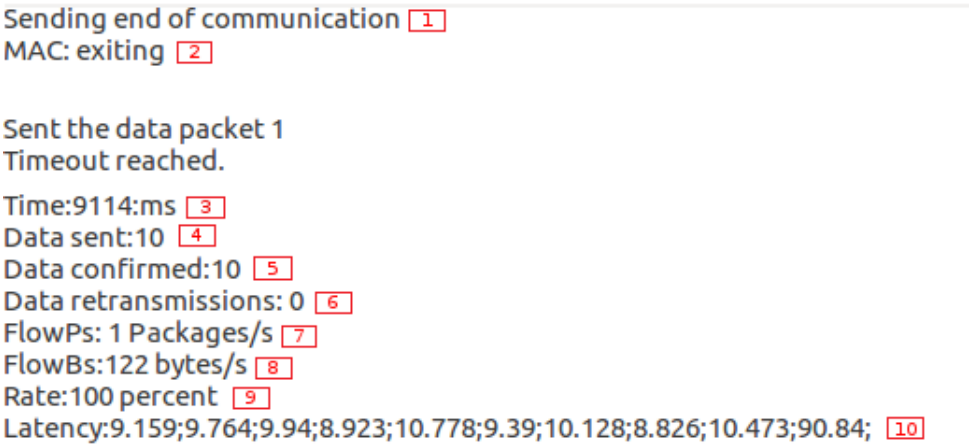

Figure 3.18: Log of the transmitted data.

Table 3.2: Comparison between the protocols used in experiments 1 and 2, respectively.

Features	Experiment 1	Experiment 2
Static message and fixed interval	x	*
Data frame	x	x
Data package	x	x
Broadcasting	*	*
Message addressed	x	x
Carrier Sense	x	x
ACK frame	x	x
ACK package	x	x
Dynamic message and variable interval		x
Control frame		x
Control package		x
RTS/CTS		x
Handshake		x
Frequency Hopping		x

* It has the functionality, however, it is not justified the use of it.
x It has the functionality

the CTS. After this confirmation, there is a dedicated communication between the two nodes and the data will be transmitted until the connection is terminated. The Frequency Hopping technique allows the node to make a previous evaluation of the quality of a channel, in this case, the transmitting node. If it has some interference or noise, 5 MHz "jumps" are performed until a channel free of these factors that are harmful to communication is found. For the base station, the "jumps" are performed every 50 ms within the 2.4 GHz band, in order to sweep the entire free frequency spectrum. In both nodes, the starting channel starts from a random choice.

Through the improvements mentioned above, it is possible to notice some im-

provements regarding the performance in data transmission. The transmission carried out in one fixed channel causes a saturation of this channel and the underutilization of the others. This causes congestion and packet loss caused by interference from other nodes or external devices. Packet loss for any reason leads to data retransmissions, which increase traffic on the channel and contribute to low data throughput. In addition to contributing to an increase in energy consumption. Problems of this nature are easily bypassed with the deployment of Frequency Hopping. On the Handshake side, it is possible to smooth application layer problems since the received packets are always from the same transmitter. Thus, there is no need for further checks and data ordering by devices. Other benefits, achieved through the establishment of a secure communication channel, were accuracy and reliability as the problems with interference were smoothed.

3.6. Experimenting with Orthogonal Frequency-Division Multiplexing (OFDM) Modulation

In this section, we introduce multi-carrier modulation and Orthogonal Frequency-Division Multiplexing (OFDM). OFDM is a modulation that is used in popular high-speed network standards, such as WiFi, DSL, and 4G. Trinity College Dublin's experiment consists of sending packets from the transmitter (Tx) N210 USRP to the Receiver (Rx) N210 USRP using an OFDM signal generated by GNU Radio [Blossom 2004]. The experiment will illustrate the capability of changing center frequency, bandwidth, gain, modulation depth, and cyclic prefix dynamically, and the impact of those parameters on the transmitted and received signal. Through this course, the reader will gain an appreciation for the factors that are most important in the use of OFDM for wireless communication by exploring its configuration and use on real radios. The goals of this section include:

- Understand the need for OFDM signaling in telecommunications networks.
- Comprehend and understand some of the theory behind multi-carrier modulation and the design of OFDM.
- Observe and examine OFDM performance and behavior under different conditions using real radio experimentation equipment.
- Exposure to advanced Future Internet Research and Experimentation (FIRE) testbed infrastructure.
- Experience using GNU Radio, an open-source software toolkit that provides signal processing blocks to implement software radios.

3.6.1. Multi-carrier Systems and OFDM Standards

The Collins Kineplex System was the first multicarrier system based on orthogonal subcarriers in HF military radio links. It was built in 1957. In 1966, a team at Bell labs filed a patent (granted in 1979) and published the first article on OFDM systems in IEEE Trans. Communications Technology in 1967 entitled: Performance of an efficient parallel data transmission system [Saltzberg 1967]. OFDM was quickly recognized as an

efficient data transmission method and research and standards continued to evolve in the 1980's, 1990's and 2000's with the addition of the following (see Figure 3.19):

- Power-Line-Communication.
- Broadcast: DVB-C2.
- ADSL/-2/-2+.
- Digital Subscriber Line (DSL) technologies.
- Broadcast: DAB, DVB-T/-T2, DVB-H, ISDB-T.
- Wireless Personal Area Network (WPAN): WiMEdia.
- Wireless Local Area Network (WLAN): IEEE802.11a/g/n/ac/ad, IEEE 802.15.4g, HiperLAN/2.
- Wireless Metropolitan Area Network (WMAN): IEEE 802.16a WiMAX.
- Mobile telephony: LTE (3.9G), LTE Advanced (4G)

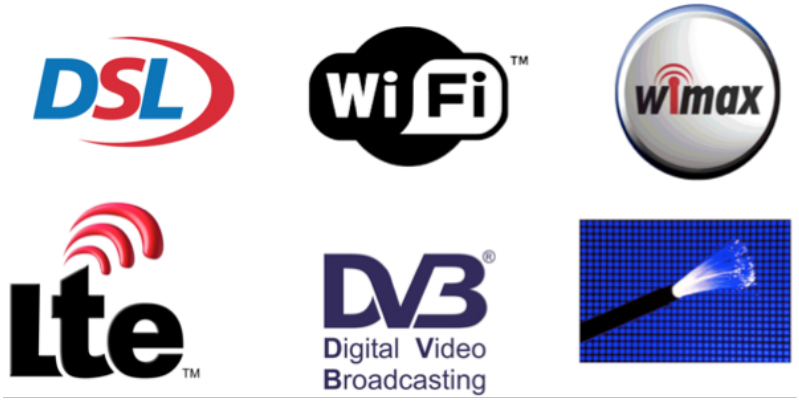

Figure 3.19: OFDM Standards.

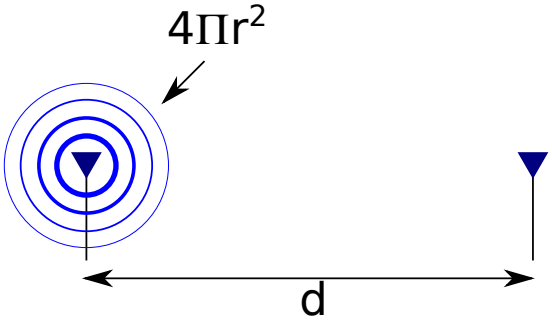

Figure 3.20: Signal Transmitter and Receiver.

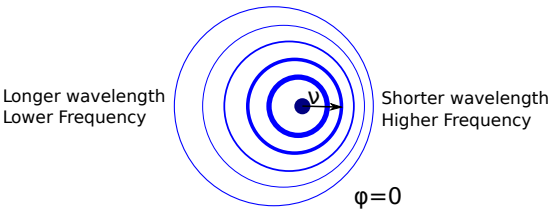

Figure 3.21: The Doppler Shift.

3.6.2. The Wireless Channel

A signal undergoes changes when it is transmitted on its way to the receiver, see Figure 3.20. A key metric of the joint impact of a wireless channel is the variation and attenuation in received signal envelope power over time and/or space, which is called fading. There are two types of fading used to describe the signal level at the receiver, large scale-fading and small-scale fading.

3.6.2.1. Large-Scale Fading

In large-scale fading, signal power falls quadratically with distance as a result of attenuation and diffraction, which occurs due to the signal traveling over large distances and using different frequencies i.e. signal path loss. Large objects such as trees, buildings, mountains, and so forth cause shadowing, and as a result received power can vary dramatically.

3.6.2.2. Small-Scale Fading

In small-scale fading, which is due to reflectors, scattering and receiver motion, multiple versions of the transmitted signal can be received from different path lengths spread over time. There are several types of small-scale fading. These include Multipath and Motion.

Multipath. If the channel is considered as a linear-time invariant system, the convolution of the channel impulse response h(t,τ) with the input stimulus x(t) (the transmitted signal) yields the system output y(t) (the channel output, i.e. the received signal). Delay spread $\sigma \downarrow \tau$ is the maximum difference between times of arrival of multipath components. The following YouTube video illustrates multipath small-scale fading [Ó Coileáin 2016].

Motion. If transmitter, receiver and/or interacting objects are in motion with the speed v under relative angle ϕ, the received signal gets shifted in frequency by Δf due to the Doppler effect, i.e., Doppler shift (see Figure 3.21). Different propagation directions result in different Doppler shifts per multipath component. Received envelope power depends on constructive or destructive addition of signals. The following short YouTube video gives a high level explanation of the Doppler Effect [Alt-Shift-X 2013].

3.6.3. Multi-carrier Systems

Multimedia applications require higher and higher data rates from wireless and wired communications systems. Mobile radio channels are fading channels that can be flat or frequency selective. For high bandwidth applications, channels are frequency selective. In conventional single-carrier modulation techniques this can only be achieved by, see Figure 3.22: transmitting shorter symbols => limited in the case of multi-path propagation (Inter-symbol interference (ISI)); and transmitting more bits per symbol => limited by noise and other distortions. In single carrier/mono-carrier system with symbol width 1/W, data is transmitted using only one carrier. Disadvantages include:

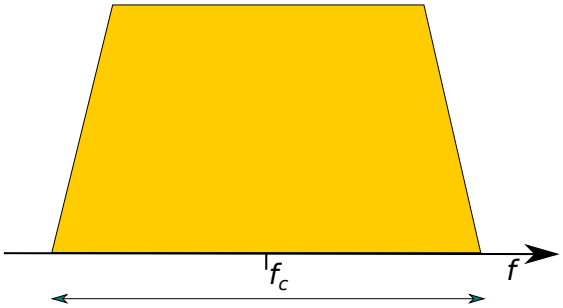

Figure 3.22: Single carrier or mono-carrier system.

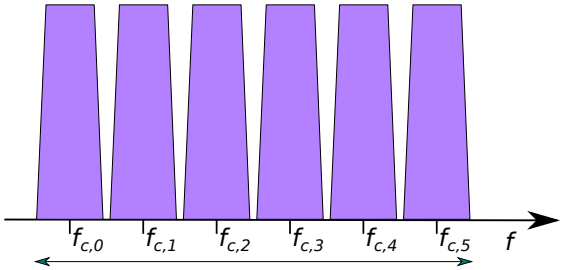

Figure 3.23: Multicarrier system.

- Event frequency selective fading.
- Equalization is complex.
- Very short pulses.
- Inter-symbol interference (ISI) is long.
- Poor spectral efficiency because of guard bands.

Multicarrier modulation is a technique where multiple low data rate carriers are combined by a transmitter to form a composite high data rate transmission, see Figure 3.23. To improve the spectral efficiency, guard bands between carriers need to be eliminated. In a classic multi-carrier system, the available spectrum is split into several non-overlapping frequency sub channels. The individual data elements are modulated into these sub-channels and are thus frequency multiplexed.

Symbol width=N_c/W and data stream is split up into multiple lower data rate sub-streams, see Figure 3.23. They are modulated and transmitted in parallel on different sub carrier frequencies i.e. Frequency Division Multiplexing (FDM). By parallel data transmission on NC sub-carriers, symbol duration TS can be increased by factor N_c to achieve the same data rate. Longer symbols are less susceptible to inter-symbol interference (ISI). Other advantages include:

- Flat fading per subcarrier.
- N_c short equalizers.
- N_c long pulses.
- ISI is relatively short.
- Poor spectral efficiency because of guard bands.
- It is easy to exploit Frequency diversity.
- 2D coding techniques are allowed.
- Dynamic signaling is possible.

3.6.4. Orthogonal Frequency-Division Multiplexing (OFDM)

Orthogonal frequency-division multiplexing (OFDM) is a multicarrier modulation technique for encoding digital data on multiple carrier frequencies. It is an FDM scheme that uses a large number of sub-carrier signals. These signals are orthogonal to each and carry parallel channels of data. In classic multicarrier systems, guard bands have to be inserted, resulting in poor spectral efficiency. A more efficient approach is to allow the spectra of individual subcarriers to overlap, see Figure 3.24. Zero crossings occur at every multiple of and hence no inter-carrier interference is present i.e., no overlap at sampling frequencies. The following YouTube video gives a high-level overview of OFDM technology [Huawei 2014].

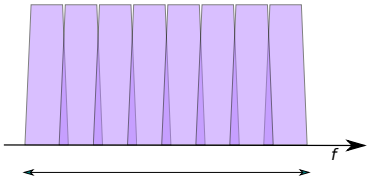

Figure 3.24: OFDM subcarrier tones are separated by the inverse of the signalling symbol duration.

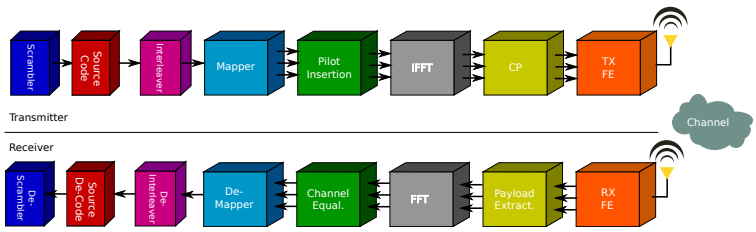

Figure 3.25: Framework architecture.

Problem: If individual subcarriers are overlapping isn't there interference between carriers?

Answer: No! If subcarrier tones are separated by the inverse of the signaling symbol duration, independent separation of frequency-multiplexed tones is possible. Additionally, sub-spectra may overlap in the frequency domain, which supports more efficient use of available spectrum and greater data rates are achievable.

In the remainder of this section, we give you a brief overview of some basic OFDM concepts that we will explore further in the experimentation section using TCDs IRIS FIRE testbed equipment [Collins 2016] [Collins et al. 2016]. These include symbol mapping/de-mapping, Inverse Discrete Fourier Transform (IDFT), Discrete Fourier Transform (DFT), equalization, cyclic prefix, and frequency sensitivity. Figure 3.25 illustrates the OFDM Systems Model, and how these concepts are interconnected.

Symbol Mapping / De-Mapping. Symbol mapping (or de-mapping) involves loading (or unloading) data bits received from the source encoder and the interleaver (or channel equalizer) to (or from) complex subcarrier modulations such as QAM, PSK, and so forth, see Figure 3.27, as **Inverse Discrete Fourier Transform (IDFT).** The output from the mapper constitutes as input to the Inverse Discrete Fourier Transform (IFFT), which accepts complex input data. In IDFT, data is parallelized then treated as samples in the frequency domain. The IDFT process transforms these into time domain signals. Rectangular time-domain pulse shaping spectra of the subcarriers become a cardinal sine function or sync function in the frequency domain.

If number of sub-carriers NC is chosen as a power of 2 (2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048), the IDFT can be replaced by an IFFT, yielding a very efficient

implementation of a OFDM modulator (FFT for demodulator at receiver). For example, 8-PSK, which has 8 Phase Shift Keying, has three bits per sub-carrier per symbol, see Figure 3.26.

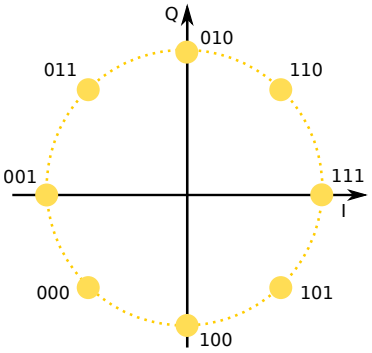

Figure 3.26: 8-PSK Constellation Diagram.

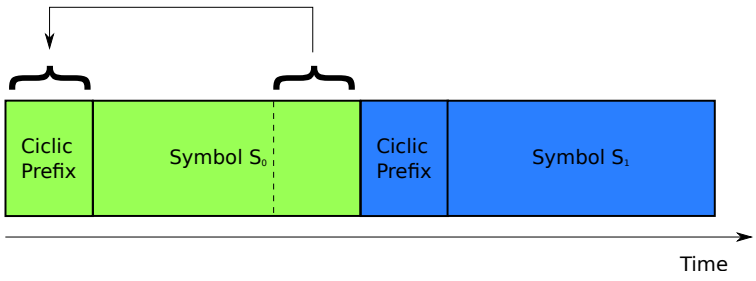

Figure 3.27: Cyclic prefix.

Equalization. The primary advantage of OFDM over mono-carrier schemes is its ability to cope with severe attenuation across channel frequencies such as small or large-scale fading (discussed above in Section 3.6.2) using a simplified equalization scheme. Equalization helps attenuate or adjust the balance between frequency components to flatten channel response, supporting the removal of frequency selective fading effects. This is achieved by:

- Insertion of known symbols (pilots) in the OFDM frame.
- Evaluating their distortions at the receiver.
- Assuming a relatively static channel, data symbols can be equalized.

In OFDM, each carrier becomes an infinite sinusoid (i.e. eigenfunction). As a result, the out of channel is a scaled version of the same function. The eigenvalues of the (circular) channel are the complex scalar terms that multiply each carrier. Thus symbols only experience magnitude and phase change, which makes equalization simple. Convolution in the time domain corresponds to multiplication in the frequency domain. However, this fact does not hold in discrete time. Circular convolution in the (discrete) time domain corresponds to multiplication in the (discrete) frequency domain. OFDM wants simple multiplication in the frequency domain. So, circular convolution is needed and not the regular convolution i.e., the real channel does regular convolution. The solution to this problem is to add a cyclic prefix, so regular convolution can be used to create circular convolution.

Cyclic Prefix. The cyclic prefix is added to the beginning of a symbol and is a repetition of the end of a symbol. Figure 3.27 shows the cyclic prefix added to a symbol over time. Its purpose is to help preserve sinusoids in multipath channels. Sinusoids are eigenfunctions of linear time-invariant channels. The cyclic prefix helps eliminate inter-symbol interference (ISI), which is the delayed replica of previous symbols interfering

with the current symbol. Additionally, they facilitate equalization by transforming linear convolution into circular convolution. Transmission time is limited to N symbols and this property is lost. The cyclic prefix restores this property by "simulating" an infinite-length sinusoid. Looking at the spectrum, Y(f)=X(f)·H(f), if H(f) is not approximately equal for all f, the original signal is destroyed. Cyclic prefix can make OFDM transmissions completely immune to ISI created by multipath propagation when cyclic prefix length T_cp is longer than the delay spread: T_cp $\geq \sigma \downarrow \tau$

Discrete Fourier Transform (DFT). At the receiver, OFDM de-modulation uses Discrete Fourier Transform (DFT) transformation to convert payload received to the frequency domain. Modulation symbols received from the DFT are de-mapped from complex subcarrier modulations such as QAM, PSK, and so forth, to bits, which are inputted into the deinterleaver and the source-decoder blocks.

3.6.5. OFDM Disadvantages: Timing and Frequency Sensitivity

OFDM transmissions are susceptible to timing and frequency offsets. Timing offsets are due to uncertainties of OFDM symbol boundaries, which can cause intersymbol interference, channel interference, and phase offset. Frequency offsets cause inter-carrier interferences (ICI), and a reduction of desired power in data received. Frequency offsets are caused by the Doppler shift or hardware imperfections e.g. imprecise up-down-conversion. This has the effect that operating on different frequency sub-carriers is no longer orthogonal. OFDM needs accurate frequency synchronization.

3.6.6. OFDM Data Rates

Doubling subcarriers in used bandwidth do not double the data rate. See Table 3.3 for comparisons of modulation depth and data rate.

Data Rate	Bandwidth	N	Code Rate	Modulation
6 Mbps	15	48	1/2	BPSK
9 Mbps	15	48	3/4	BPSK
12 Mbps	15	48	1/2	QPSK
18 Mbps	15	48	3/4	QPSK
24 Mbps	15	48	1/2	16-QAM
36 Mbps	15	48	3/4	16-QAM
48 Mbps	15	48	2/3	64-QAM
54 Mbps	15	48	3/4	64-QAM

Table 3.3: OFDM Data Rates and Modulations Depths

3.6.7. FIRE Testbed Environment

This OFDM course runs completely on Trinity College Dublin's (TCD's) IRIS testbed facility, which is located on TCD's campus in Dublin, Ireland. The testbed consists of 16 flexible Universal Software Radio Peripheral (USRP) N210 Ettus Research units aligned in a grid configuration, see Figure 3.28. Each USRP is connected to a virtual machine that runs a software-defined radio (SDR) system. In these experiments, we use the GNU Radio software development toolkit. GNU Radio offers signal-processing

Figure 3.28: Iris wireless laboratory ceiling mounted N210 USRP radio equipment.

blocks that implement software radios. A conceptual diagram of IRIS's virtualized cloud resources, radio hypervisor, user experiments and physical equipment is shown in Figure 3.29. The hardware that you will use will be configured automatically through a process called provisioning. This process will take care of the reservation of two virtual machines in TCDs FIRE testbed facility, the connection of the same to appropriate USRP hardware, installation of required operating system and tools, and initialization of experimentation services. These two virtual machines are under your sole control for use in experimentation. Data for monitoring wireless spectrum is sent to a database on the webserver which is displayed to you via a graph.

Each OFDM experiment requires a virtual machine and a USRP for transmitting a signal and a VM and a USRP for receiving a signal. Variable parameter changes from users are sent from the web interface to GNU Radio, which supports changing frequency, gain, modulation depth, and so forth. The Rx node sends data streams received for frequency, time, waterfall and constellation back to a gateway server for rendering in the web interface.

Testbed Configuration and Tools. Users can experiment with real radio hardware equipment on a Future Internet Research & Experimentation (FIRE) testbed facility [Collins 2016] or [Collins et al. 2016]. TCD/CONNECT has deployed the Smart Reconfigurable Radio Testbed based on the GNU Radio software-defined radio (SDR) system. The Iris testbed supports experimentation with a mature SDR running on a virtualized computational platform. The testbed is organized in experimentation units, each of which consists of three parts: a virtual computational platform, SDR software, and flexible radio front-end hardware. Through this organization, TCD encapsulates the elements required to use the GNU Radio SDR system to construct a broad range of radio

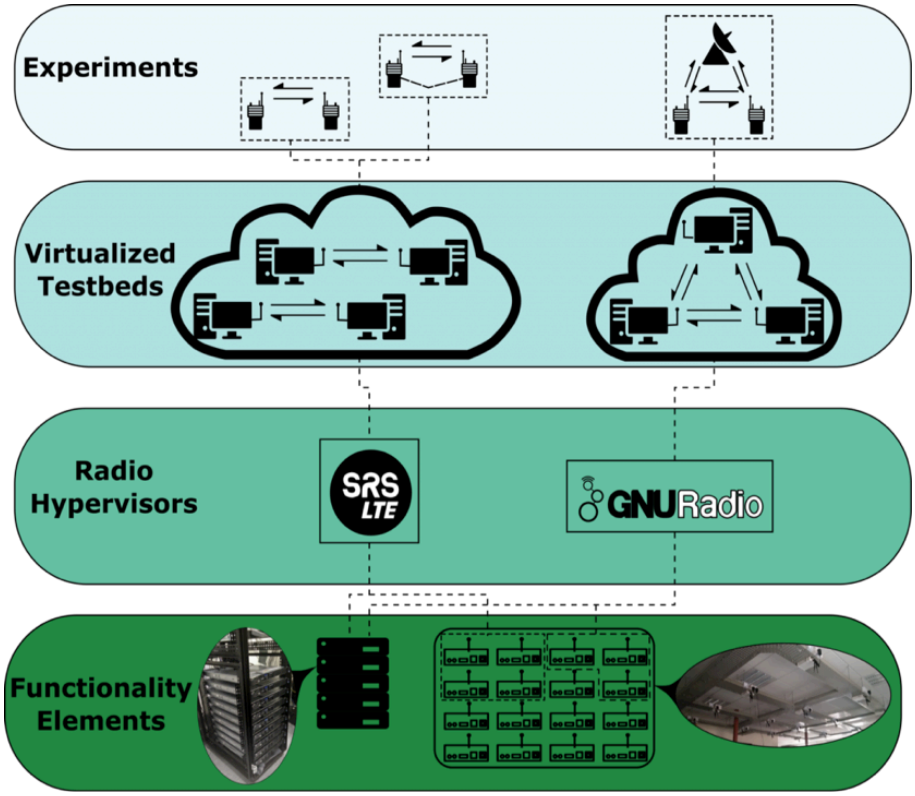

Figure 3.29: Conceptual diagram of IRIS's virtualized cloud resources, radio hypervisor, physical equipment, and user experiments.

systems. Each experimentation unit is designed to flexibly serve a range of needs: Linux (Ubuntu 16.04.01 LTS) provides a highly configurable computation platform, GNU Radio provides real-time radio reconfigurability, and a USRP offers a broad range of wireless interfaces. Radio hardware is housed on the ceiling of the dedicated indoor testing space to provide users with a clean operating environment. The management infrastructure allows users to deploy experimentation units to compose arbitrary radio systems and networks as desired. These facilities have enabled and facilitated several international research and education-related projects.

3.6.8. Exercises

The TCD OFDM course, which allows students to inspect the effect of configuring OFDM concepts and principles using the GNU Radio software radio equipment on the transmitter (Tx) and receiver (Rx) machines, is available at [Collins 2016] or [Collins et al. 2016]. There is no need to investigate every possible combination of configuration parameters. However, after each experiment, you should try to understand the effects and implications of your configuration changes on the OFDM radio and try to answer the accompanying questions.

3.7. Hints for using testbeds

This section will present some hints of how to use testbeds in the most efficient way (how to organize your code, how to run it...).

3.7.1. OFDM - Notes when running GNU Radio experiments

- When initializing experiments, please note that it can take up to ten minutes to provision the Tx and Rx nodes.
- Remember that each configuration change can take up to several seconds to take effect.
- It is important to remember that the transmitter and receiver need to be configured with the same frequency, modulation depth, bandwidth, and cyclic prefix when sending and receiving a signal. This is to give the message or packet the best chance of being received correctly.
- Due to the nature of radio communication - every packet may not be received. Consequently, don't be afraid to send lots of packets.
- After an experiment is launched, resources provisioned will only be available for two hours.

3.7.2. Make an installation script

Considering that during the stage of installation of the modules several commands are repeated, a script can be constructed to aid this process. Using Shell scripts we can automate the installation, assuming that the files will initially be uploaded to the VM or that they will be fetched using git commands.

3.7.3. Avoiding the graphical interface

For various reasons, such as screen freezes, the use of the graphical interface should be avoided when using remote testbeds. Here are some changes that need to be made to prevent a graphical interface from being displayed at runtime.

Before sending the `.grc` file to the remote environment, we need to delete or disable all blocks that instantiate the GUI. All GRC GUI blocks should be avoided or disabled, as shown in Fig. 3.30. In the `Options` block, at the top-left corner of GRC, we need to choose *No GUI* on *Generate Options*.

In the example of Fig. 3.30, `WX GUI Slider` and `WX GUI Chooser` are disabled to prevent the execution of the graphical interface. To disable a block just right click on the block and choose *Disable*. If the block that was disabled provides a variable for use in other GRC blocks, we need to add a `Variable` block with the same variable name that was previously contained in the block that used GUI. In addition, all `WX GUI` and `QT GUI` blocks should be disabled, as well as the all `WX GUI Sink` and `QT GUI Sink` instrumentation blocks. This procedure is required in an existing file or in a totally new project.

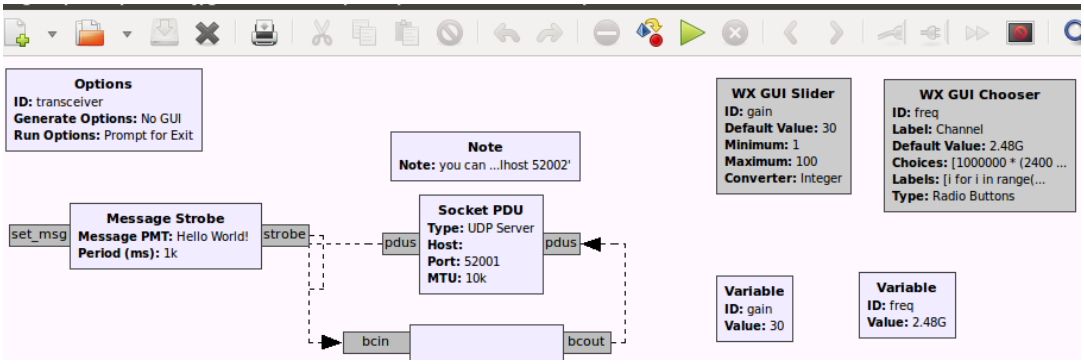

Figure 3.30: Changes on GRC to avoid graphical interfaces

After modifying all graphical GUI instances in GRC, it is important to note if there is any GUI execution in the source code. If so, it should be changed to terminal output.

After modifying the files to not display the graphical interface, the files can be sent to the remote environment and `.grc` can be executed remotely as follows:

```
vm$ grcc -e file.grc
```

If you only need to compile the file, then it is necessary to use the command:

```
vm$ grcc file.grc
```

Finally, all debug checks must be performed by the output terminal, for example, by using in the source code a `print "Message"` in case of Python, or in case of C language using `printf("Message")`.

3.7.4. Using the RSpec editor

In JFed, using the RSpec editor, we can create or modify an experiment and add new features. At the XML file, as shown in the following, we can change the fields *client_id* (line number 3), *component_manager* (line 3), *silver_type name* (line 4) and *disk_image name* (line 5).

```
1 <?xml version='1.0'?>
2 <rspec xmlns="http://www.geni.net/resources/rspec/3" type="request"
    generated_by="jFed RSpec Editor" generated="2017-02-03T16:50:46
    .711-02:00" xmlns:emulab="http://www.protogeni.net/resources/rspec/
    ext/emulab/1" xmlns:jfedBonfire="http://jfed.iminds.be/rspec/ext/
    jfed-bonfire/1" xmlns:delay="http://www.protogeni.net/resources/
    rspec/ext/delay/1" xmlns:jfed-command="http://jfed.iminds.be/rspec/
    ext/jfed-command/1" xmlns:client="http://www.protogeni.net/
    resources/rspec/ext/client/1" xmlns:jfed-ssh-keys="http://jfed.
    iminds.be/rspec/ext/jfed-ssh-keys/1" xmlns:jfed="http://jfed.iminds
    .be/rspec/ext/jfed/1" xmlns:sharedvlan="http://www.protogeni.net/
    resources/rspec/ext/shared-vlan/1" xmlns:xsi="http://www.w3.org
    /2001/XMLSchema-instance" xsi:schemaLocation="http://www.geni.net/
    resources/rspec/3 http://www.geni.net/resources/rspec/3/request.xsd
     ">
3   <node client_id="node01" exclusive="false" component_manager_id="
      urn:publicid:IDN+futebol.dcc.ufmg.br+authority+am">
4     <sliver_type name="usrp-vm">
```

```
5       <disk_image name="urn:publicid:IDN+futebol.dcc.ufmg.br+image+
            gnuradio"/>
6     </sliver_type>
7     <location xmlns="http://jfed.iminds.be/rspec/ext/jfed/1" x="410.0"
          y="85.0"/>
8   </node>
9   <node client_id="node02" exclusive="false" component_manager_id="
        urn:publicid:IDN+futebol.dcc.ufmg.br+authority+am">
10    <sliver_type name="usrp-vm">
11      <disk_image name="urn:publicid:IDN+futebol.dcc.ufmg.br+image+
            gnuradio"/>
12    </sliver_type>
13    <location xmlns="http://jfed.iminds.be/rspec/ext/jfed/1" x="10.0" y
          ="85.00000000000001"/>
14  </node>
15  <jfed-command:experimentBarrierSegment orderNumber="0" tag="Barrier
        segment 0"/>
16 </rspec>
```

The options to those fields are:

client_id: any nick;

component_manager: urn:publicid:IDN+futebol.dcc.ufmg.br+authority+am ;

sliver_type name: choose between available slivers: usrp-vm, iot-vm and raw-raspberry;

disk_image name: each sliver has a disk available, which depends on the testbed that you are using.

3.7.4.1. Adding one more node

In the `.rspec` file presented earlier, there are 2 nodes, called *node01* (see line 3) and *node02* (line 9). If we intend to add another node, we can copy the whole stretch of line 9 to line 14, modifying the node identifier (`node client_id`) and the `location xmlns` field, to avoid overlapping nodes in the jFed graphical environment.

3.8. Conclusions and further readings

This chapter presented how to use Software-Defined Radios (SDR) to conduct wireless research. SDRs are wireless transceivers that are able to run a number of wireless protocols, since they are implemented in software, not in hardware. Further, the GNU Radio project provides a number of algorithms and protocols that can be used to implement new protocols. SDRs are an important tool for wireless researchers because it allows the creation of new protocols, which can be tested in a realistic situation. By experimenting with a real device, the proposal is evaluated under more realistic settings than those found in most simulators, for example. As a consequence, the research becomes more relevant, and the gap from research to mass dissemination of the technology becomes shorter.

Although SDR devices are not cheap, it is possible to perform research using real hardware very easily by remotely using those resources, for free, over the Internet. There

are a number of testbeds spread all over the world, including testbeds in Brazil, which have USRPs available for researchers. In this short course, we focused on the FUTEBOL federation of testbeds, however other testbeds could be used.

In order to show that SDR is relevant for research in wireless communications and wireless networking, this chapter presented four simple experiments that highlight the capabilities of such a platform. Those experiments range from WBAN to cellular, from modulation techniques to MAC protocols, and provide a glimpse of the versatility of SDR. Further, all the code used in the experiments is available for use and modification by other researchers.

In the future, we expect more and more papers to be written using SDRs as their platform. Although today it may be a bit hard to find full stack implementations for some popular wireless standards, this limitation is quickly being addressed by the community. As a consequence, the complexity of building experiments with SDRs will go down with time. There is a push in the networking and telecommunications community towards experimental research, so wireless researchers should be aware of SDRs, how to use them and what are their limitations. Even if you do not plan to use it today for your experiments, you might need to use them in the near future.

Acknowledgements

The authors of this chapter have been funded by CAPES, CNPq, Fapemig and the FUTEBOL project. FUTEBOL has received funding from the European Union's Horizon 2020 for research, technological development, and demonstration under grant agreement no. 688941 (FUTEBOL), as well from the Brazilian Ministry of Science, Technology and Innovation (MCTI) through RNP and CTIC.

References

[OAI 2017] (2017). OpenAirInterface | 5G software alliance for democratising wireless innovation. `http://www.openairinterface.org`.

[ope 2017] (2017). OpenBTS | open source cellular infrastructure. `http://openbts.org`.

[Akyildiz et al. 2008] Akyildiz, I., Lee, W.-Y., Vuran, M. C., and Mohanty, S. (2008). A survey on spectrum management in cognitive radio networks. *Communications Magazine, IEEE*, 46(4):40–48.

[Alt-Shift-X 2013] Alt-Shift-X (2013). The doppler effect: what does motion do to waves? `https://youtu.be/h4OnBYrbCjY`.

[Amiri et al. 2007] Amiri, K., Sun, Y., Murphy, P., Hunter, C., Cavallaro, J., and Sabharwal, A. (2007). WARP, a unified wireless network testbed for education and research. In *Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on*, pages 53–54.

[Beyene et al. 2014] Beyene, Y. D., Jäntti, R., and Ruttik, K. (2014). Cloud-ran architecture for indoor das. *IEEE Access*, 2:1205–1212.

[Bharadia et al. 2013] Bharadia, D., McMilin, E., and Katti, S. (2013). Full duplex radios. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, SIGCOMM '13, pages 375–386, New York, NY, USA. ACM.

[Bloessl et al. 2013] Bloessl, B., Leitner, C., Dressler, F., and Sommer, C. (2013). A GNU Radio-based IEEE 802.15. 4 Testbed. *12. GI/ITG FACHGESPRÄCH SENSORNETZE*, page 37.

[Blossom 2004] Blossom, E. (2004). Gnu radio: tools for exploring the radio frequency spectrum. *Linux journal*, 2004(122):4.

[Busch et al. 2004] Busch, C., Magdon-Ismail, M., Sivrikaya, F., and Yener, B. (2004). Contention-free MAC protocols for wireless sensor networks. In *International Symposium on Distributed Computing*, pages 245–259.

[Collins 2016] Collins, D. (2016). Connect smart reconfigurable radio testbed. `https://iris-testbed.connectcentre.ie/ofdm_v2/login.php`.

[Collins et al. 2016] Collins, D., Barja, J. M., Kaminski, N., Blumm, C., Silva, L. D., Sutton, P., and Gomez, I. (2016). Introduction to orthogonal frequency-division multiplexing (OFDM) modulation method. `http://www.forgebox.eu/fb/preview_course.php?course_id=180`.

[Commission 2003] Commission, F. C. (2003). *FCC 03-322*. FCC.

[Cordeiro 2017] Cordeiro, J. R. S. (2017). FS-MAC: um sistema para flexibilização da subcamada MAC em redes sem fio.

[Cordeiro et al. 2017] Cordeiro, J. R. S., Lanza, E., Macedo, D. F., and Vieira, L. F. M. (2017). Fs-mac: Uma plataforma para a flexibilização da sub-camada mac em redes sem fio. In *XXXV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

[Correia et al. 2015] Correia, L. H., Tran, T.-D., Pereira, V. N., Giacomin, J. C., and Sá Silva, J. M. (2015). Dynmac: A resistant mac protocol to coexistence in wireless sensor networks. *Computer Networks*, 76(Complete):1–16.

[Diepstraten and WCND-Utrecht 1993] Diepstraten, W. and WCND-Utrecht, N. (1993). IEEE 802.11 wireless access method and physical specification. *Power*, 5:10.

[Dillinger et al. 2003] Dillinger, M., Madani, K., and Alonistioti, N. (2003). *Software Defined Radio: Architectures, Systems and Functions*. Wiley & Sons.

[Ettus 2017] Ettus (2017). Ettus Research. `http://www.ettus.com`.

[Forum 2011] Forum, W. I. (2011). Software defined radio - rate of adoption. `http://www.wirelessinnovation.org/sdr_rate_of_adoption`.

[Gilmore and Blossom 2017] Gilmore, J. and Blossom, E. (2017). GNU Radio - the free and open software radio system. `http://gnuradio.org/redmine/projects/gnuradio/wiki/`.

[Gollakota et al. 2011] Gollakota, S., Hassanieh, H., Ransford, B., Katabi, D., and Fu, K. (2011). They can hear your heartbeats: Non-invasive security for implantable medical devices. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 2–13, New York, NY, USA. ACM.

[Gomez 2013] Gomez, O. M. (2013). Implementation of the ofelia control framework (ocf) for open flow-based testbed facilities. Master's thesis, Universitat Politècnica de Catalunya (UPC).

[Gudipati and Katti 2011] Gudipati, A. and Katti, S. (2011). Strider: automatic rate adaptation and collision handling. In *Proceedings of the ACM SIGCOMM 2011 conference*, SIGCOMM '11, pages 158–169, New York, NY, USA. ACM.

[Hong et al. 2012] Hong, S. S., Mehlman, J., and Katti, S. (2012). Picasso: flexible RF and spectrum slicing. *SIGCOMM Comput. Commun. Rev.*, 42(4):37–48.

[Hu et al. 2009] Hu, W., Li, X., and Yousefi'zadeh, H. (2009). La-mac: A load adaptive mac protocol for manets. In *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*, pages 1–6.

[Huawei 2014] Huawei (2014). Huawei Learning Service Express OFDM. `https://youtu.be/tPQ_ahjCujY`.

[Iannucci et al. 2012] Iannucci, P. A., Perry, J., Balakrishnan, H., and Shah, D. (2012). No symbol left behind: a link-layer protocol for rateless codes. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, Mobicom '12, pages 17–28, New York, NY, USA. ACM.

[Jagannath et al. 2015] Jagannath, J., Saarinen, H. M., and Drozd, A. L. (2015). Framework for automatic signal classification techniques (fact) for software defined radios. In *CISDA*, pages 1–7.

[Katti et al. 2008] Katti, S., Rahul, H., Hu, W., Katabi, D., Médard, M., and Crowcroft, J. (2008). XORs in the air: practical wireless network coding. *IEEE/ACM Trans. Netw.*, 16(3):497–510.

[Kumar et al. 2013] Kumar, S., Cifuentes, D., Gollakota, S., and Katabi, D. (2013). Bringing cross-layer MIMO to today's wireless LANs. In *Proceedings of the ACM SIGCOMM 2013 conference*, SIGCOMM '13, pages 387–398, New York, NY, USA. ACM.

[Li and Qiu 2010] Li, H. and Qiu, R. C. (2010). A graphical framework for spectrum modeling and decision making in cognitive radio networks. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–6.

[Lin et al. 2008] Lin, K. C.-J., Kushman, N., and Katabi, D. (2008). ZipTx: Harnessing partial packets in 802.11 networks. In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, MobiCom '08, pages 351–362, New York, NY, USA. ACM.

[Marques et al. 2016] Marques, A. F. F., Miranda, G., Silva, L. M., Ávila, R. S., and Correia, L. H. A. (2016). Iscra - an intelligent sensing protocol for cognitive radio. In *IEEE – ISCC*, pages 385–390.

[McHenry et al. 2006] McHenry, M. A., Tenhula, P. A., McCloskey, D., Roberson, D. A., and Hood, C. S. (2006). Chicago spectrum occupancy measurements & analysis and a long-term studies proposal. In *Proceedings of the first international Workshop on Technology and policy for accessing spectrum (TAPAS), 2006*, pages 1–12.

[Mitola 1999] Mitola, J. (1999). Cognitive radio : model-based competence for software radios. NR 20140804.

[Movassaghi et al. 2014] Movassaghi, S., Abolhasan, M., Lipman, J., Smith, D., and Jamalipour, A. (2014). Wireless body area networks: A survey. *IEEE Communications Surveys & Tutorials*, 16(3):1658–1686.

[Murphy et al. 2006] Murphy, P., Sabharwal, A., and Aazhang, B. (2006). Design of warp: A wireless open-access research platform. In *European Signal Processing Conference*.

[Neufeld et al. 2005] Neufeld, M., Fifield, J., Doerr, C., Sheth, A., and Grunwald, D. (2005). Softmac-flexible wireless research platform. In *Proc. HotNets-IV*, pages 1–5.

[Nychis et al. 2009] Nychis, G., Hottelier, T., Yang, Z., Seshan, S., and Steenkiste, P. (2009). Enabling MAC Protocol Implementations on Software-defined Radios. In *6th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pages 91–105.

[Ó Coileáin 2016] Ó Coileáin, D. (2016). Multipath fading. `https://youtu.be/1rcCLfdR5qs`.

[Perry et al. 2012] Perry, J., Iannucci, P. A., Fleming, K. E., Balakrishnan, H., and Shah, D. (2012). Spinal codes. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, SIGCOMM '12, pages 49–60, New York, NY, USA. ACM.

[Rao and Stoica 2005] Rao, A. and Stoica, I. (2005). An overlay MAC layer for 802.11 networks. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 135–148.

[RTL-SDR] RTL-SDR. Rtl-sdr.com. `http://www.rtl-sdr.com/`.

[Saltzberg 1967] Saltzberg, B. (1967). Performance of an efficient parallel data transmission system. *IEEE Transactions on Communication Technology*, 15(6):805–811.

[Saucier 2000] Saucier, R. (2000). Computer generation of statistical distributions. Approved for public release; distribution is unlimited.

[Shokrollahi 2006] Shokrollahi, A. (2006). Raptor codes. *IEEE/ACM Trans. Netw.*, 14(SI):2551–2567.

[Silva et al. 2015] Silva, W. S., Cordeiro, J. R. S., Macedo, D. F., Vieira, M. A. M., Vieira, L. F. M., and Nogueira, J. M. S. (2015). Introdução a Rádios Definidos por Software com Aplicações em GNU Radio. In *Minicursos do XXXIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, chapter 5, pages 216–265. Sociedade Brasileira de Computação.

[Souryal et al. 2015] Souryal, M., Ranganathan, M., Mink, J., and Ouni, N. E. (2015). Real-time centralized spectrum monitoring: Feasibility, architecture, and latency. In *2015 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*, pages 106–112.

[Takagi and Kleinrock 1985] Takagi, H. and Kleinrock, L. (1985). Throughput analysis for persistent CSMA systems. *IEEE Transactions on Communications*, 33(7):627–638.

[Tan et al. 2009] Tan, K., Zhang, J., Fang, J., Liu, H., Ye, Y., Wang, S., Zhang, Y., Wu, H., Wang, W., and Voelker, G. M. (2009). Sora: High performance software radio using general purpose multi-core processors. In *USENIX International Symposium on Networked Systems: Design and Implementation (NSDI)*, pages 75–90.

[Tinnirello et al. 2012] Tinnirello, I., Bianchi, G., Gallo, P., Garlisi, D., Giuliano, F., and Gringoli, F. (2012). Wireless MAC processors: Programming MAC protocols on commodity hardware. In *IEEE INFOCOM*, pages 1269 –1277.

[Vieira et al. 2013] Vieira, L. F. M., Gerla, M., and Misra, A. (2013). Fundamental limits on end-to-end throughput of network coding in multi-rate and multicast wireless networks. *Computer Networks*, 57(17):3267–3275.

[Wireless Innovation Forum 2017] Wireless Innovation Forum (2017). Wireless innovation forum. `http://www.wirelessinnovation.org`.

[Yucek and Arslam 2009] Yucek, T. and Arslam, H. (2009). A Survey of Spectrum Sensing Algorithms for Congnitive Radio Applications. *Proceedings of the IEEE*, 97(5):805–823.

[Zhou et al. 2006] Zhou, G., Stankovic, J. A., and Son, S. H. (2006). Crowded spectrum in wireless sensor networks. Workshop on Embedded Networked Sensors.

[Ziouva and Antonakopoulos 2002] Ziouva, E. and Antonakopoulos, T. (2002). CSMA/CA performance under high traffic conditions: throughput and delay analysis. *Computer Communications*, 25(3):313 – 321.