Francisco Manuel de Oliveira Esteves

Licenciado em Ciências da Engenharia Electrotécnica e de Computadores

Implementation of a Cognitive Radio Access System

Dissertação apresentada para obtenção do Grau de Mestre em Engenharia Electrotécnica e de Computadores, pela Universidade Nova de Lisboa, Faculdade de Ciências e Tecnologia.

Orientadores : Rodolfo Oliveira, Professor Auxiliar, FCT/UNL
Luís Bernardo, Professor Auxiliar, FCT/UNL

Júri:
Presidente: Doutor Paulo da Costa Luís da Fonseca Pinto,
Professor Catedrático da FCT/UNL
Vogais: Doutor Rodolfo Alexandre Duarte Oliveira,
Professor Auxiliar da FCT/UNL
Doutor Luís Filipe Lourenço Bernardo,
Professor Auxiliar com agregação da FCT/UNL
Doutor Fernando José da Silva Velez,
Professor Auxiliar da Universidade da Beira Interior (UBI)

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Março, 2013

Implementation of a Cognitive Radio Access System

"To invent, you need a good imagination and a pile of junk."

Thomas Edison

To the love of my life, Ana.

Agradecimentos

Quero começar por agradecer ao meu orientador, o Professor Rodolfo Oliveira pela paciência, dedicação, disponibilidade e amizade demonstradas ao longo desta dissertação. Quero também agradecer ao meu coorientador, o Professor Luís Bernardo, pela igual disponibilidade e dedicação demonstradas.

Agradeço aos meus colegas de laboratório e a todos os meus amigos pelos bons momentos passados, especialmente ao longo do período de desenvolvimento desta dissertação e pelo seu contributo para a mesma.

Ao meu colega António Furtado, cujo contributo foi essencial para esta dissertação, deixo o meu especial agradecimento.

Ao meu colega José Vieira, um agradecimento especial pela ajuda prestada na revisão desta dissertação.

Quero agradecer aos meus avós, Albino e Donzília, por tudo o que fizeram e continuam a fazer por mim, pois é a eles que devo grande parte da minha educação. Obrigado por todo o carinho, amor e dedicação que sempre me deram, independentemente do momento ou situação. Para mim são pessoas com um valor incalculável às quais eu espero conseguir retribuir o que têm feito por mim.

Quero agradecer aos meus pais, que sempre me deram tudo, irmã e a toda a minha família o apoio e carinho que sempre me deram ao longo dos anos e que fizeram com que eu sempre me sentisse feliz.

Quero deixar um agradecimento especial aos meus tios Jorge e Irene por me terem tratado como um filho e me terem ajudado sempre que precisei.

Aos meus primos, Tomás, Salomé e Sílvia quero deixar o meu agradecimento pois são os melhores primos do Mundo!

Aos meus futuros sogros, nunca é demais agradecer por tudo o que têm feito por mim e pelo carinho que sempre me deram. Estiveram sempre disponíveis para me ajudar no que fosse necessário e a fazer por mim o que fosse preciso independentemente da situação. Nunca conseguirei agradecer o suficiente, por tudo o que fizeram e por me terem tratado como um filho.

Por último, queria agradecer à pessoa mais importante da minha vida. Por me ter apoiado incondicionalmente em tudo e por todo o carinho que me deu em todos os momentos, especialmente nos mais difíceis. Acima de tudo, foi uma verdadeira amiga na qual sempre pude confiar e pedir ajuda quando mais precisei. Ana, obrigado por tudo o que sempre fizeste por mim. Se consegui terminar o curso, a ti o devo.

Resumo

Actualmente, com o aumento do número de dispositivos sem fios, o espectro de rádio disponível é cada vez menor. A grande maioria dos dispositivos sem fios, habitualmente denominados de dispositivos licenciados ou utilizadores primários, devido ao facto de operarem em bandas previamente licenciadas para o efeito, não são suficientemente dinâmicos para se adaptarem a esta nova realidade. Dessa forma, são necessárias novas políticas de alocação de espectro que apresentem maior rendimento na utilização dos recursos disponíveis.

Entre as soluções para o aumento de desempenho, encontram-se os sistemas de rádio cognitivo. Estes dispositivos, também denominados de dispositivos não licenciados ou utilizadores secundários, são capazes de utilizar partes do espectro que durante algum tempo possam não estar a ser utilizadas pelos dispositivos licenciados. A introdução destes dispositivos nas redes actuais é o ponto de partida para a criação de redes cognitivas onde os dispositivos licenciados continuam a usar normalmente os recursos da rede e os não licenciados são capazes de utilizar os espaços livres no espectro.

Neste trabalho o software de controlo de uma placa de rede sem fios comum é modificado para o teste de alguns destes conceitos. Inicialmente, são efectuadas validações no software a fim de se obter o controlo total do mecanismo de transmissão e recepção de dados. De seguida, o mecanismo de controlo da janela de contenção é testado e usado para controlar o débito de dados e o accesso ao meio do dispositivo.

Na segunda parte deste trabalho é desenvolvido e implementado um protocolo para estimar a probabilidade de acesso de um utilizador primário. Depois de vários testes práticos e teóricos é feita uma comparação entre os resultados de forma a caracterizar o seu desempenho. Esta análise mostra como e com que nível de sucesso é possível estimar a presença na rede de um utilizador primário através de um ou mais utilizadores secundários.

Palavras Chave: Sistemas de Rádio Cognitivo, Controlo de Acesso ao Meio, Janela de Contenção, Probabilidade de Acesso, Utilizadores Primários e Secundários.

Abstract

Over the last two decades we have observed a rapid increase of the number of wireless devices. This fact poses big challenges to solve, namely because the available radio spectrum is limited and is becoming scarce. The most part of the actual wireless devices, usually called licensed devices or primary users, due to the fact of being licensed to operate in a specific band, are unable to dynamically adapt to other bands. In consequence, new spectrum allocations policies are being implemented.

The cognitive radio systems are among the actual solutions to increase this performance. These devices, often called unlicensed devices or secondary users, are capable of using spectrum holes that are not being used by the licensed users for a brief period of time. The introduction of these devices to the current wireless networks will be the starting point for the creation of cognitive radio networks, where licensed devices still use the spectrum normally and the unlicensed devices form the cognitive radio network that will take advantage of the spectrum holes freed, left by the licensed devices.

In this work, an open-source wireless driver for a commercial wireless network card is modified in order to test some of these concepts. First, some key validations are made in order to have a full control of the driver's transmission/reception mechanism. After this validation, the driver's contention window control mechanism is tested and used for controlling the data throughput and the device's access to the medium.

In a second part of this work, a primary user activity estimation scheme is present. With this scheme, after several practical tests it is possible to establish a comparison between the protocol simulation results and the practical tests in order to evaluate its performance. This analysis assesses the level of accuracy of the estimation of the primary user presence, when done by one or more secondary users on the network.

Keywords: Cognitive Radio Systems, Medium Access Control, Contention Window, Access Probability, Primary and Secondary Users.

Acronyms

ACK *Acknowledgment*

AMC *Adaptive Modulation and Coding*

AMM *Alternative Mode Monitoring*

AP Access Point

API *Application Programming Interface*

BSD *Berkeley Software Distribution*

CA *Collision Avoidance*

CCK *Complementary Code Keying*

CCTS *Control-Channel-Clear-To-Send*

CR *Cognitive Radio*

CR-ALOHA *Cognitive Radio ALOHA*

CR-CSMA *Cognitive Radio Carrier Sense Multiple Access*

CRN *Cognitive Radio Network(s)*

CRTS *Control-Channel-Request-To-Send*

CSMA *Carrier Sense Multiple Access*

CSMA/CA *Carrier Sense Multiple Access with Collision Avoidance*

CTS *Clear-to-Send*

CW *Contention Window*

DCF *Distributed Coordination Function*

DIFS *Distributed Interframe Space*

GUI *Graphical User Interface*

HAL *Hardware Abstraction Layer*

IMI *Initial Mode Identification*

LAN *Local Area Network*

MAC *Medium Access Control*

NS-2 *Network Simulator v2*

OFDM *Orthogonal Frequency Division Multiplexing*

PU *Primary User*

QAM *Quadrature amplitude modulation*

QOS *Quality of Service*

QPSK *Quadrature phase-shift keying*

RTS *Request-to-Send*

SCA *Statistical Channel Allocation*

SCA-MAC *Statistical Channel Allocation Medium Access Control*

SIFS *Short Interframe Space*

SNR *Signal-to-Noise Ratio*

SU *Secondary User*

UDP *User Datagram Protocol*

VAP *Virtual Access Point*

WLAN *Wireless Local Area Network*

Contents

List of Figures

List of Tables

Chapter 1

Framework

1.1 Introduction

In the last few years, the World has been watching the exponential growth of the number of wireless devices and networks. Nowadays, almost every house, public space or government facility have wireless networks, and they are used simultaneously by many devices. These networks are used to provide internet access, data transfer and many other services, therefore, they have a huge role in the actual society.

This brings a new concern, that has been neglected since the early days of the wireless networks, due to the lack of wireless devices, the radio spectrum occupancy. Every device requires permanent network access with good quality and with a high transmission bit rate. The increasing demand of radio spectrum access will eventually lead to a congestion, resulting in an unfair access opportunity for all devices and no possibility of assuring the required quality of service (QOS).

The available spectrum frequencies are becoming scarce and the traditional radio devices are unable to use the available spectrum in an efficient manner due to their rigid architecture and incapacity to adapt to new radio spectrum characteristics. These devices, denominated licensed devices or primary users (PUs), leave several "gaps" in radio spectrum during their operation, making the radio spectrum usage efficiency not as good as it could be. To fully increase the efficiency, these gaps become the target of cognitive radio devices, denominated non licensed devices or secondary users. This allows the accommodation of a large number of devices in the same radio spectrum, providing an overall

increase of the usage efficiency.

Cognitive radio devices differ from the traditional radio devices due to the ability of "sensing" the spectrum or being able to know which part of the spectrum is free to be used without causing interference to the licensed users.

Sensing is the main characteristic of these devices, because they are able to have a much higher knowledge of the radio spectrum than the traditional radio devices. This feature allows the device to know how the radio spectrum is and when it is occupied.

The cognitive radio devices must be able to transmit information using several free frequencies bands without causing any interference to the licensed devices.

Unlike the traditional radio, which is built almost entirely by hardware, the cognitive radio is constituted mainly by software, which confers the device an intelligence capability that will be determinant, to allow learning and evolving during the device functioning.

The cognitive radio is probably the future of radio devices, because the natural trend is to require much more efficiency in spectrum usage to allow an higher number of devices to establish communication with good quality of service, making possible the coexistence of non licensed and licensed devices on the same band.

1.2 Motivation

The cognitive radio devices must operate using the available radio spectrum frequency bands, which constitutes a difficult task since the licensed users (Primary Users) to use the spectrum, must be kept free of interference. This, along with the ability of sensing the spectrum are the main challenges for the cognitive radio.

The sensing is performed using several techniques, but the energy detector is the most common one, which consists in the comparison of the energy of the detected signal with a threshold value in order to determine if there is any device transmitting at that instant.

The threshold value can be defined using several methods but its efficiency is always questionable. Higher levels of interference will decrease its accuracy while increasing the probability of false alarm of the PUs.

This work proposes to use a set of nodes (secondary users) operating a contention-based medium access mechanism, such as the one adopted in IEEE 802.11, to estimate the

activity of the primary users.

The IEEE 802.11 standard is the most common standard for data wireless communications. Since this technology is present in almost all mobile devices, it becomes one of the main motivations for this work, whereas the massive adoption of this standard can be used as a way to develop cognitive radio systems.

The contention mechanism adopted in IEEE 802.11 standard is used in this work to indirectly give us a measure of PU's activity. This fact motivates some changes that could be made in order to achieve better network performance. The possibility of estimating the activity of PUs through the used of a contention-based protocol in the SUs constitutes the main motivation for this work.

1.3 Objectives and Contributions

The creation of a contention based network to act as a secondary network is one of this work's objectives. This network will be formed by several secondary nodes that will estimate the primary nodes' activity. To achieve this goal it was decided to use the IEEE 802.11 protocol and it was chosen the Madwifi software driver. Several steps must be taken starting with the understanding of the IEEE 802.11 wireless driver structure (Chapter 3), the Madwifi, which is used to control the wireless devices. Having a clear notion how it works and how it is structured, will allow the needed modification to implement new protocols and therefore the creation of a cognitive access system.

The next step will be validating the results obtained through the driver statistic mechanism, in order to confirm the frame timings that are needed to setup each transmission.

After this, the primary user (PU) estimation protocol must be implemented through a theoretical approximation (Chapter 4) and its performance tested for validation in Chapter 5 by establishing a comparison between the theoretical and practical results.

This work contributed to obtaining a complete framework for testing the IEEE 802.11 wireless networks. This framework has several applications and tools that allow real time statistical data gathering and specific IEEE 802.11 protocol parameters customization.

The development of this test framework is itself a contribution of this work because it can support the development of new protocols for wireless networks.

1.4 Structure of the Dissertation

This dissertation is split in six Chapters and three appendixes. The first Chapter ("Framework"), introduces the problems addressed in this work and identifies the main objectives of the work.

The second Chapter ("Related Work") refers the actual state of the art for the cognitive radio networks. An approach is made to the cognitive radio sensing methods and the current protocols that were proposed for this technology, when applied to the IEEE 802.11 networks.

The third Chapter ("Madwifi Description and Validation") describes the structure of the Madwifi IEEE 802.11 wireless driver used for implementing the proposed cognitive access system. The driver's performance validation represents the last part of this chapter.

The fourth Chapter ("Primary Users Activity Estimation") is dedicated to the PU activity estimation. The estimation is meant to be done by the SUs on the network. This chapter describes the theoretical approach behind the proposed estimation scheme.

The fifth Chapter ("Performance Analysis") explains how the PU activity estimation was implemented and characterizes the obtained results from both simulation and practical results. A performance analysis is done for all test results and a comparison of all considered scenarios is also presented.

The sixth Chapter ("Conclusions") describes the results obtained with the proposed cognitive access system. A resume of all the development and a plan for future work ends the description.

The Appendix A explains how the IEEE 802.11 frame timings were obtained for the Madwifi driver and how the throughput is estimated for each transmission.

The second Appendix, the Appendix B, contains the source code of all applications used in the results validation (Chapter 3), such as the Matlab application, scripts and all java applications.

The third Appendix, the Appendix C, has all the source code of all applications used in the Performance Analysis (Chapter 5), such as the Matlab application, scripts and all java applications.

Chapter 2

Related Work

2.1 Introduction

In the last few years, the growth of wireless networks has been exponential. Today is fairly common to have several wireless networks operating in the same location. These networks can be infra-structured or infrastructure-less (*ad-hoc*). The infra-structured network is the most common type of wireless network, although, *ad-hoc* networks are becoming more usual and their number is increasing very rapidly due to the growth of mobile devices with wireless networking capabilities.

With such a high number of wireless networks and devices it is very important to grant good performance and a very low error rate in each transmission between devices and a fair access policy for every device. This brings a huge concern, because the wireless spectrum is reaching the limit in terms of occupancy, so it is very important to implement mechanisms capable of managing the spectrum occupancy, granting a fair access policy to every device and minimizing the number of collisions or transmission errors.

The most common wireless devices are equipped with radio transceivers implementing the 802.11 DCF standard [IEE07]. These devices are unable to dynamically manage the spectrum occupancy because they are statically allocated to a given frequency band.

The main goal of this work is to use the "gaps" in the spectrum freed by licensed users to transmit information and therefore, reach a better spectrum occupancy and efficiency.

To take advantage of these "gaps" in the spectrum, a deeper knowledge of the spectrum is needed from the non-licensed devices that will use them. This knowledge is acquired

through the sensing mechanism implemented in the device. By other words, sensing in a radio environment, is the ability of having a notion of the spectrum occupancy. The most common sensing method is the energy-based, and is present in the majority of the IEEE 802.11 radio devices. A deeper knowledge of the spectrum occupancy will be helpful since it can be translated in a more efficient network resources usage.

The time spent in the spectrum sensing is also one of the main concerns. If a node spends too much time to decide about the occupancy state of the channel, the decision can be easily outdated and the practical usage of the decision can be highly inefficient

A Cognitive Radio (CR) is a non-licensed device that is capable of sensing the radio environment and decide which are the best transmission parameters for the current radio environment, as well the frequency in the available spectrum to be used. CR devices are defined by the capability of sensing the surrounding environment and having a notion of the spectrum occupancy. Using that information, CR devices become able to make adjustments for their transmission parameters in order to be more efficient.

There are two kinds of users in a cognitive radio network (CRN). The primary users (PUs) or licensed users, and the secondary users (SUs) or non-licensed users. The PUs are the owners of the spectrum and they should use the spectrum whenever they want. The SUs, can use the spectrum in an opportunistic way and must reduce the level of interference caused to the PUs. The CR device can only take advantage of the unused part of the spectrum or a specific frequency channel used by a PU when it is free, but only for a brief period of time.

This chapter introduces the most known *sensing* techniques at a physical level and Medium Access Control (MAC) protocols for cognitive radio networks (CRNs) in order to introduce the work already done in this field.

2.2 Cognitive Radio Sensing Dimensions

There are many dimensions of sensing in CR that can be taken into consideration. These dimensions are related with the domain that is selected to detect the presence of licensed users.

The dimensions usually adopted in the literature include:

- **Frequency** - The frequencies available in the radio spectrum can be used by CR devices to achieve a better spectrum usage. The main idea is to take advantage of available bands in the radio spectrum [Yuc09], [Mar10], [Hay10].

- **Time** - In the same band there are also time opportunities, because the band is not used continuously in time-domain and there will be also unoccupied "gaps" that can be explored by SUs [Yuc09], [Hay10], [Cha10].

- **Geographical space** - Based on location (latitude, longitude, and elevation) and distance of PUs, CR devices can take advantage of the the path loss to reuse the bands occupied by PUs. The channel occupancy is determined by the interference level at a given position and is only valid for a given instant of time. When interference is detected, the CR device assumes that a primary is transmitting [Hao09], [Yuc09], although, there is always a great concern about the hidden cognitive terminal problem .

- **Code** - It is possible to perform simultaneous transmissions if the SUs uses a different code than the one in use by the PU [Yuc09], [Lo10]. Although, it is extremely important to grant the usage of an orthogonal code to avoid any type of interference with the PU as well the channel occupancy in terms of codes at the moment of the transmission. This will also require a good time synchronization between the SUs and the PUs.

- **Angle** - The CR device must be aware of the PU location as well its beam direction (azimuth and elevation angle). If the SU is aware of the PU beam direction and location it can transmit information simultaneously without causing interference, because it will transmit to a different location with a different angle [Yuc09], [Lu11].

2.3 Sensing Architectures

There are two types of sensing architectures. The single-radio and the dual-radio architecture.

- **Single-radio** - This scheme uses a single radio to sense and transmit information. The downside of this architecture is that a time period is allocated for spectrum sensing, and as consequence the information gathered during the sensing time can be easily outdated during the transmission period.
  The efficiency also suffers from this factor because the time used for sensing can not be used for transmitting. On the other hand, this will be a simpler architecture in terms of complexity and will have a much lower cost when compared with the dual-radio architecture [Lui13], [Var01], [Cha05].

- **Dual-radio** - In this architecture [DaS13], [Var01], [Cha05] one of the radios is dedicated to the spectrum sensing and the other for all the transmission/reception procedures like a standard radio. This architecture allows a much better spectrum sensing efficiency, but the disadvantages are much more significant because it will increase dramatically the system complexity as well the power consumption, hardware requirements and cost.

2.4 Sensing Techniques

There is a variety of sensing techniques and they can be applied according the needs of the CR systems. These are the most common spectrum sensing techniques:

- **Energy based** - This is the most commonly used technique of spectrum sensing. It has a low computational and implementation complexity, therefore, it is widely used in the CR systems.
  This detector establishes a comparison between the detected signal and a threshold value previously defined according to the noise floor and specific detection and false alarm probabilities [Yuc09], [Lui13]. This method may exhibit low accuracy concerning the primary detection, because the PU signal can be classified as noise and

when the signal-to-noise ratio (SNR) is very low, this kind of detector achieves poor performance.

- **Waveform based** - This technique is more reliable than the previous one [Yuc09], [Gha12]. It uses a comparison between radio signals to perform the sensing, although, this is only applicable to systems with known signal patterns because the received signal will be correlated with a copy of itself, that is previously known. As the signal length increases, the precision of this sensing method also increases.

- **Cyclostationarity based** - This technique takes advantage of the fact that a high number of signals used in the wireless networks are cyclostationary [Yuc09], [Gha10]. There is always periodic transmissions being made and while the white noise is stationary, these transmissions will occur periodically.
  The gathered statistics after a certain period of time will be enough to identify a PU and also to distinguish the users that are transmitting on the given network.

- **Radio identification based** - This technique uses radio technology identification [Yuc09], [Ars06]. The main objective is to identify the radio technology used by the PUs on the sensed radio spectrum. This allows a very high precision in the PU detection which is directly proportional with the amount of known technologies by the CR device.
  There are two main tasks for this technique. The IMI (Initial Mode Identification) and the AMM (Alternative Mode Monitoring)[Yuc09]. In the IMI, the CR device will try to find a transmission mode in the network. In the AMM, a CR will be monitoring the network for alternative transmission modes, while it could be simultaneously transmitting information in other transmission mode.

- **Matched-filtering** - This is considered the best sensing method when the transmitted signal is known [Yuc09], [Elt11]. This sensing method is based on a linear filter which maximizes the ratio between the output signal and a given input signal. The great advantage of this sensing method is the short time needed to achieve a certain probability of false alarm or probability of miss detection, when compared to the already referred sensing methods.

- **Other Sensing Methods** - There are also other alternative sensing techniques [Yuc09]. Multi-taper spectral estimation, wavelet transform based estimation, Hough transform, and time-frequency analysis. These methods are also valid but less common, although, they can be used in specific scenarios.

2.5 Cooperative Sensing

The spectrum sensing accuracy from the SUs is limited by the channel noise and fading and can be improved with cooperative sensing. Cooperative sensing provides a faster sensing time, a more precise knowledge about the existent PUs and it can solve the hidden PU problem [Mis12]. This kind of sensing can be implemented in two different forms, the centralized sensing [Qiu10] and the distributed sensing [Qiu10]:

- **Centralized sensing** - In this type of sensing, there is a central unit which has the task of collecting all sensing information from the CR devices and broadcast that information to other CR devices.

  When the network has a significant number of devices, a huge bandwidth is needed to report all the sensing information. To alleviate this problem, only CR devices with reliable sensing information are allowed to transmit it to the central unit [Qiu10].

- **Distributed sensing** - In distributed sensing [Qiu10], the CR devices will not depend on a central unit to decide which part of the radio spectrum will be used. The devices share the sensing information with each other, but they will make their own decisions independently. This will require much less bandwidth to transmit the sensing information between SUs.

  The distributed sensing does not need any kind of backbone structure in that view it is a cheaper solution.

2.6 CSMA and MAC in Cognitive Radio

As mentioned in Section 2.1, CR has as primary goal: achieving radio spectrum efficient usage without causing undesired interference to PUs.

The SUs detect if the PUs are using the radio spectrum through carrier sense. This type

of carrier sense is massively employed in CSMA (Carrier Sense Multiple Access) schemes. CSMA is indicated for distributed systems because there is no need for a centralized station and therefore it is adopted in Wireless Local Area Networks (WLAN). IEEE 802.11 DCF is the most successful protocol for WLANs, which adopts CSMA.

CSMA usually uses an energy-based detection technique, sometimes enriched with specific protocol's information. In CSMA's energy detection, the occupation of the channel is determined by establishing a comparison between the cumulative detected power (energy) and a pre-set threshold. Energy detection needs a longer sampling interval to increase the reliability of the spectrum's occupancy decision. When the PU's signal is strong enough, accurate decision is taken most of the times. But as the sampled SNR decreases, so does the technique's accuracy.

In CSMA, the access opportunity is determined by the user who does not detect any access from any other user, therefore, the access opportunity is higher when the number of detectable users is smaller and vice-versa. This will affect the MAC level channel capacity, and the main goal will be obtaining an optimal detection duration for the CSMA, in order to achieve the optimal MAC level channel capacity. There are also proposals for protocols that allow simultaneous transmissions for the PUs and SUs. Most of MAC protocols for CRs are based in CSMA and these protocols try to achieve the best CRN performance when compared to the standard MAC protocol, which are also CSMA-based.

2.6.1 CSMA based MAC Protocols for Cognitive Radio Networks

The standard CSMA only allows transmissions when the radio channel is idle. This restrains any kind of further improvement in the sensing mechanism for a standard radio device. Although, there are some proposals focusing specific constraints and enabling simultaneous transmissions for the CR devices, even when the PU is transmitting. This will improve the throughput of the CRN and the spectrum usage efficiency as well.

One of the most important characteristic of wireless communications, is that the received signal strength decreases as the link distance increases, and vice-versa. This constitutes an opportunity for the CR devices, because the PU will not be able to use all the spectrum resources when in a long distance scenario, therefore, the radio spectrum can be used in a

more effective manner.

The key is to allow CRs to transmit simultaneously during the PUs transmission when the interference to and from the PUs is acceptable. This is a challenge by itself because there is a need to adapt the CSMA mechanism, in order to allow simultaneous transmissions. The CSMA, by default, inhibits any transmission if the channel is sensed as busy, therefore, it is imperative to change this mechanism and due to this need, a new class of CSMA based MAC protocols was proposed, as it is mentioned in [Che08].

2.6.1.1 Rate-Distance Nature of Wireless Communications

The rate-distance relationship in wireless communication systems is very important since its early days. When the distance between two communicating terminals or between a terminal and a base station increases, the power of the signal received decreases and therefore, the data rate also converges to a smaller value.

Nowadays, the wireless systems use Adaptive Modulation and Coding (AMC) and automatically adjust the transmission power, modulation, coding scheme and the data rate based on the received signal strength from the other devices on the network. This factor is determinant for the selection of a lower data rate and a higher interference robustness transmission scheme (e.g. QPSK) in a scenario when high receiving reliability is required and low power signal is received.

A scenario of a short distance between the communicating devices will lead to a higher data rate and also to a higher level of reliability (lower rate of errors during the transmission).

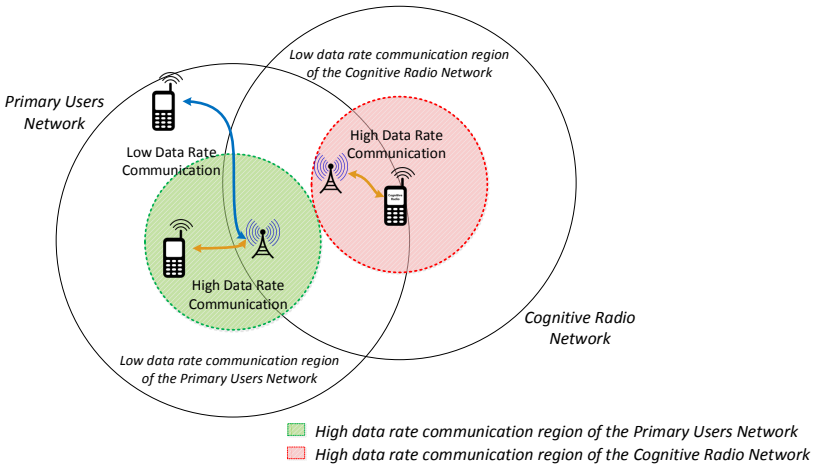


Figure 2.1: Cognitive Radio Network and Primary Users Network coexisting.

The Figure 2.1 shows how this rate-distance nature enables the coexistence of the CRN and the PUs Network and simultaneous transmissions between the terminals.
It is important to notice that a CR communication can be established within the coverage area of the PUs network. To make this possible, the interference level from and to the PU will determine the transmission rate.

2.6.1.2 New CSMA based MAC Protocols

A new class of CSMA-based MAC protocols was proposed in [Che08] for CRNs. These new protocols improve the CSMA in order to operate in CRNs. Both PU's network and the CRN, are carrier sensed based, with a four-way handshake (Figure 2.2).

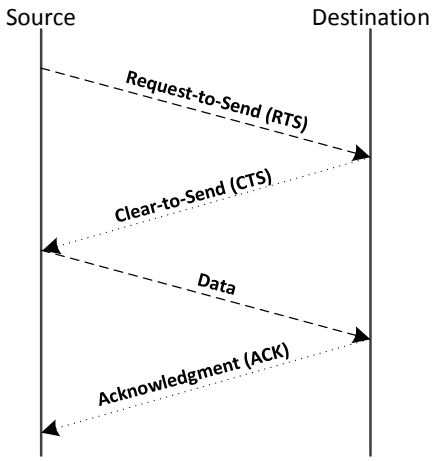


Figure 2.2: Four-way handshake process.

The basic operational mode for the system is described in [Che08] and has the following order:

- After sensing the carrier for a specific τ_p time before the transmission, the PU will send a request-to-send (RTS) (if the channel was sensed as idle) to the base station of the PU network, in order to begin the packet transmission.
- If a corresponding clear-to-send (CTS) is received successfully, then the data packets will be transmitted, followed by an acknowledgment (ACK).
  In case of failure in the CTS reception, the data packets will be sent when the channel becomes idle again after a new sense.
- If a CR or any other PU device starts a transmission, the channel will be sensed as busy.

The operating mechanism of these protocols can be defined in four steps:

1. When a new packet is waiting for transmission and the channel is observed busy, a CR starts to sense the channel for a time duration τ_s after the busy period. At the end of this sensing time, it will send a RTS to the CRN base station to gain priority over the channel, even if the channel is sensed as busy (a PU is transmitting a packet).
2. If the RTS is received by the CR device with no collision detected, then the CR receiver will calculate the best transmitting power and rate. If the rate and transmission power could be calculated, then the CR receiver will send them back to the CR transmitter through the CTS, otherwise no CTS will be sent.
3. If the CTS is received by the CR transmitter, then it starts transmitting the data packet with the calculated data rate and transmission power that were received in the CTS and previously calculated by the CR receiver. In case of a missed CTS reception, the CR transmitter waits until the end of the next carrier sensing period to send another RTS.
4. At last, if the data packet is correctly received, an ACK packet is sent by the CR receiver as confirmation.

These protocols use adaptive power and rate communication scheme between the CR devices. This is essential in order to achieve the best rate and transmission power when a PU is present and to allow simultaneous transmissions from the CR devices. The main goal of the protocol is to achieve the best rate and transmission power for the CR devices without increasing the interference level in a way that could cause failures in the PUs current transmissions. To make this possible, it is essential to guarantee an acceptable interference level of the PUs. This level is used to calculate the best rate and transmission power for every CR device as it is described in [Che08].

2.6.1.3 Achievements

As mentioned in [Che08], this protocol, allows the coexistence between the PUs network and the CRN and also provides simultaneous packet transmission for the CR devices during any PU transmission, increasing significantly the average throughput of the CRN.

2.6.2 Two-level MAC Protocols for Cognitive Ratio Networks

Other MAC protocols, as the one presented in [Won11], were designed to solve the problem that arises when CRNs try to access the medium that is occupied by PUs, are the slotted CR-ALOHA and the CR-CSMA.
These protocols use a two-level opportunistic spectrum access (OSA) strategy. The primary objective for this strategy is scheduling the SUs' packets transmissions in an efficiently manner and simultaneously protect the PUs transmissions.

2.6.2.1 Slotted CR-ALOHA and CR-CSMA Spectrum Access

The random access model for these protocols is based on CSMA for CRNs where no control channel is used and SUs share the same channel with PUs.
As referred in Subsection 2.6.2, these protocols were implemented using a two-level OSA strategy and both protocols have the first level in common.

- **First level** - This level is dedicated to the spectrum sensing and occurs in the beginning of each MAC frame (Figure 2.3) before accessing the medium to transmit. Due to the interference caused by the PUs, SUs are forced to fix their detection

probability based on a specific threshold value. The amount of bits sent can be easily adjusted by changing the sensing time of the SUs. The SUs can also use the detection and false alarm probabilities to choose when to transmit packets.

- **Second level for CR-CSMA and slotted CR-ALOHA** - On this level, similar operations to the standard MAC protocols take place. This level handles the SUs packet transmissions. The operation scheme implemented in this level is different for both protocols, and this feature distinguishes them from each other, despite using the same MAC frame structure.

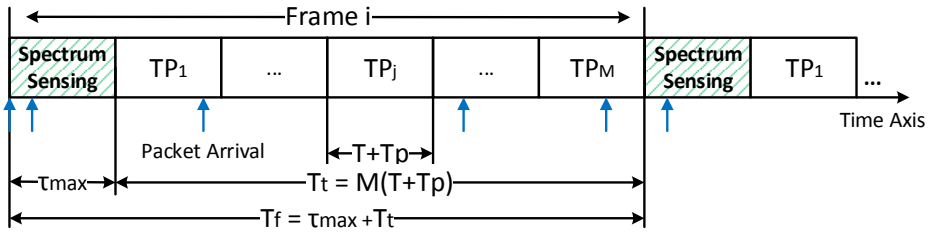


Figure 2.3: MAC frame structure for the two-level OSA strategy [Won11].

These two levels combined form the CR-CSMA and slotted CR-ALOHA protocols. The main characteristic of these protocols is the new MAC frame represented in the Figure 2.3 and the framework created to take advantage of this frame structure.

When a PU starts transmitting, the SUs on the network must vacate the channel within $\boldsymbol{T_d}$ seconds and as result, the spectrum sensing should be executed also within $\boldsymbol{T_d}$ seconds. This will require discontinuous frame-based channel access for the SUs, which diverges from the traditional continuous access of the conventional networks.

Each frame must have a duration of $\boldsymbol{T_f}(T_f \leqslant T_d)$ seconds. The $\boldsymbol{T_f}$ duration includes a $\boldsymbol{\tau_{max}}$ duration for spectrum sensing and a $\boldsymbol{T_t}$ duration for data transmission. The $\boldsymbol{T_t}$ duration includes $\boldsymbol{M}$ transmission periods ($\boldsymbol{T_{ps}}$) and each one consists in a packet transmission time of $\boldsymbol{T}$ seconds and a propagation delay of $\boldsymbol{T_p}$ seconds. Each MAC frame has duration $\boldsymbol{T_f} = \tau_{max} + M(T + T_p)$.

These protocols, mentioned in [Won11], have common features, such as the sensing method and assumptions for the operation scheme.

The energy detector (previously described in Section 2.4), is the method used by these

protocols due to its simplicity and easiness of implementation.

SUs implement a positive ACK scheme, which informs the SU if the packet has been transmitted successfully. If the transmission fails, no ACK is received and the packet is retransmitted after the backoff period.

2.6.2.2 Slotted CR-ALOHA Operation Scheme

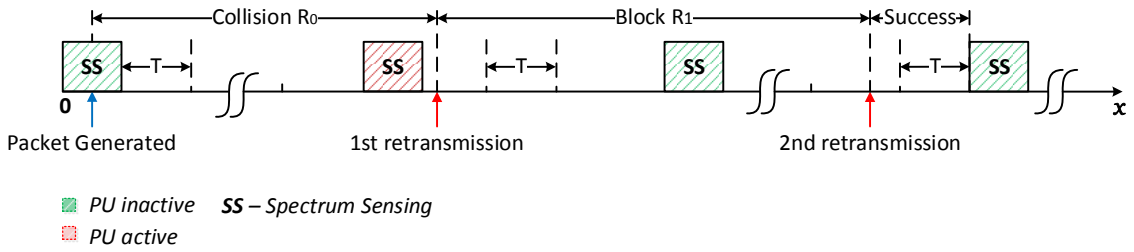


Figure 2.4: Operation scheme for slotted CR-ALOHA.

The protocol was developed from the standard slotted ALOHA, but with a different discrete channel access time and a constrain to protect the primary network was added.

For each frame (Figure 2.3), the data transmission will have a duration of $\boldsymbol{l}$, which is slotted with a slot size equal to the transmission period (TP) or $1 + \alpha$.

Figure 2.4 illustrates the operation scheme for the slotted CR-ALOHA, which can be described by the following rules:

1. If the SU senses the channel as available in the current frame, any packet that arrives in the Mth slot of the previous frame will be transmitted in the first slot. This will be applied to the spectrum sensing duration as well. Otherwise, if a packet arrives in the jth slot ($j \neq M$), it will start to transmit at the beginning of the $(j+1)$th slot.

2. If the channel is sensed as unavailable, any packet arrival within the current frame and up to the $(M\text{-}1)$th slot will wait until the end of the current frame and then will be uniformly retransmitted after the backoff period.

3. The transmission is considered successful, only when just one packet is transmitted, otherwise is assumed that a collision occurred, and as consequence, the packet will be

retransmitted after random delays to avoid another collision or to minimize possible repeated collisions.

4. If a packet arrives in the Mth slot of one frame, than it will only be processed in the following frame.

2.6.2.3 CR-CSMA Operation Scheme

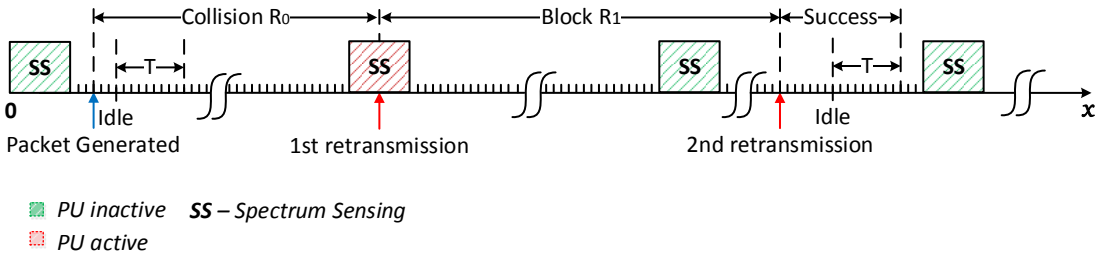


Figure 2.5: Operation scheme for CR-CSMA.

Unlike the slotted ALOHA, in the CR-CSMA it is assumed that the data transmission duration $\boldsymbol{l}$ is divided into mini-slots with length $\boldsymbol{\sigma}$, which are denominated as slot-time (**ST**). For this protocol, a timing structure similar to the Distributed Coordination Function (DCF) in the 802.11 protocol [IEE07] was adopted and the slot time concept is a part of that legacy.

The CR-CSMA requires that each packet marked for transmission should initiate the transmission procedure in the beginning of the next **ST**. In order to improve the channel access time, it was assumed that one slot or transmission period (**TP**) contains m mini-slots ($m\sigma$). Due to the short carrier-sensing time being much smaller than the spectrum sensing time, the carrier-sensing time was not taken into account for the operation scheme. The CR-CSMA operation scheme is represented in Figure 2.5 and operates according the following rules:

1. If the SU senses that the PU is inactive during the current frame, any packet arrival during the Mth slot of the previous frame or during the spectrum sensing duration of the current frame, it will be transmitted in the first slot of the actual frame. Otherwise, if the packet arrives in the jth slot ($j \neq M$), in case of the channel being

idle, the packet will be transmitted at the beginning of the next **ST**. In case of being busy, the SU will keep sensing the spectrum until the channel becomes idle again and then will attempt to transmit the packet.

2. In case of detecting the PU as active, if any SUs' packet arrives within the current frame up to the (M-1)th slot, it will wait until the end of the current frame and then the SU will choose an uniformly distributed backoff time to retransmit the frame.

3. A transmission is considered successful if no collisions occur during the transmission period (**BF**). If not, the transmission will fail and a retransmission will occur after a random delay to avoid collisions.

4. If any packet arrives during the Mth transmission packet of any frame, it will be marked for retransmission, and will be transmitted in the next frame.

2.6.2.4 Performance Analysis for CR-CSMA and slotted CR-ALOHA

The slotted CR-ALOHA, based in the standard slotted ALOHA suffers a considerable loss in terms of performance when applied to a CR environment, because any PU activity will affect its performance considerably.

The CR-CSMA will have a better performance when compared to the slotted CR-ALOHA. This is due to the fact that SUs are only allowed to transmit from the beginning of a slot in the slotted CR-ALOHA and they do not need to sense the channel. In CR-CSMA, the SUs are much more aggressive in terms of medium access, because they are always sensing the spectrum waiting that the channel becomes idle.

Despite the discrepancy in terms of performance, they have some characteristics in common like the performance/interference factor, the performance/agility factor and the optimal frame length.

Reaching the optimal agility factor, which means the best capability of a SU to leave a channel as soon as a PU starts transmitting is critical for the SU performance because there is always a "tradeoff" between the performance and agility. This is also valid for the frame length. Longer frames allow more SUs to compete for the channel, but shorter frames allow the periodic spectrum sensing and as consequence, the channel utilization is

reduced. An optimal frame length can be also achieved according the agility/performance ratio.

The performance/interference ratio will depend almost exclusively on the interference requirements of protecting the PUs network. Less interference restrictions will be translated in a higher throughput for the SUs network.

2.6.3 SCA-MAC for Wireless Ad-hoc Networks

The SCA-MAC, identified in [Kuo07], is another protocol created to solve the interoperability issue and achieves a much higher efficiency of spectrum utilization for the CR environment (Section 2.6).

This protocol is a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) based cognitive MAC protocol that uses Statistical Channel Allocation (SCA) for wireless ad-hoc networks. It allows CR devices to do real time opportunistic access to any part of the spectrum, independently of being used by PUs or SUs.

This protocol is capable of evolving in terms of knowledge about the radio environment. This knowledge increment by the CR is achieved from several sensing operations and collecting the spectrum usage statistics in each one of its operations.

The probability of successful transmission can be increased by this cumulative statistic, as well as the probability of interference with the PUs, which can be also reduced.

The SCA-MAC is designed to allow channel aggregation, so it can use several channels in a single transmission. To reach a good control over the influence of the PUs, the protocol is able to evaluate their impact in real time, and therefore, predict the successful rate based on the packet length and with the collected statistics make decisions among alternative choices.

2.6.3.1 SCA-MAC Operation Scheme

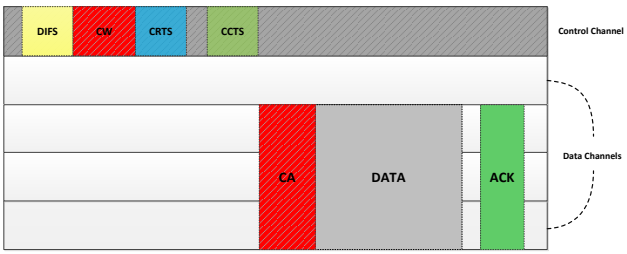


Figure 2.6: Operation scheme for SCA-MAC.

As is represented in the Figure (2.6), the SCA-MAC protocol relies on three major operations:

1. **Environment sensing and learning** - A CR device must be able to sense a part of the spectrum in one attempt and gather detailed information about the spectrum usage. One solution to this problem is using the sensing technique referred in Section 2.4, the Cyclostationary technique. Another less advanced method with a lower complexity, is sensing the spectrum several times periodically. This method is adopted by the SCA-MAC protocol.

   The sensing period is adjustable and is predefined. When the protocol is executed, a run length of the idle/busy period is kept for each channel. When the idle duration ends due to a transmission from the PU, the run length is recorded in a circular buffer (e.g. a circular buffer of size 300 can record the run length of the last 300 idle periods).

   This gathered data will provide the statistics for each channel and the knowledge needed to enable an intelligent channel allocation.

2. **CRTS/CCTS exchange over the control channel** - Due to the opportunistic channel access, it is necessary to determine which channel should be used by the sender to communicate with the receiver and a mechanism is needed to initiate the communication between them.

   This protocol implements a control channel, which is common to all users and has the role of initiate the hand-shaking process between both parts.

The control channel access is managed by a CSMA/CA mechanism which makes the SCA-MAC a decentralized protocol, despite the common control channel.

If a device wants to initiate a transmission, it listens to the control channel and waits until it becomes idle again. Then, it waits a DIFS period for the channel to remain idle before starting the countdown of the contention window (CW). After this countdown, if the channel remains idle, it transmits a Control-Channel-Request-To-Send (CRTS) packet.

When the receiver receives the CRTS, it tries to find the best transmission opportunity based on its own statistics and current channel status. Then, it will reply with a Control-channel-Clear-To-Send (CCTS) packet, which contains the information related to the best opportunity for transmission. If a collision is detected in the CRTS or CCTS packet, the negotiation process must be repeated with a double CW size.

The CCTS carries the Collision Avoidance (CA) window for the next transmission on the data channel. This backoff window, based in the traditional CW from the CSMA/CA, was implemented in order to reduce the probability of collision of two transmissions occurring simultaneously in the same data channel.

$$N = |CA| = \begin{cases} 2 & , n = 0, \\ 2^n & , 1 \leq n \leq 5, \\ 32 & , n \geq 5. \end{cases} \tag{2.1}$$

The Equation (2.1) from [Kuo07], describes the conditions to be met for obtaining the CA window, implemented in the SCA-MAC protocol. The number of neighbors of the receiver, represented by n, can be obtained by the information contained in the CCTS. The probability of collision at the receiver, depends almost only on the number of neighbors (n). The sender will then choose the CA window based on the rules expressed in equation equation (2.1), which are also valid for the retransmission scenario.

3. **DATA transmission and ACK over data channels** - In case of a successful exchange of the CRTS/CCTS packets, both sender and receiver will use the agreed data channels for the data transmission. The sender starts the countdown of a counter randomly selected from the CA window received in the CCTS and, if the channel is idle in the end of the countdown, it will begin the DATA transmission.
   The receiver will reply with an ACK after waiting a SIFS period. In case of transmission failure the sender has to do all the negotiation process again in the control channel.

2.6.3.2 Statistical Channel Allocation (SCA)

To allocate channels and grant acceptable interference levels simultaneously to the PUs it is imperative to evaluate the successful rate of any transmission in the future. Channel aggregation and the packet length are key variables for the successful rate. The available number of potential combinations with the available channels in the spectrum increases the system complexity, therefore is necessary to lower the complexity in order to allow an efficient channel allocation

The parameters that impact in the channel allocation are enumerated as follows:

1. **Optimum operation range** - It is very simple to calculate the successful rate for a single channel. Although, this is not true when a successful rate needs to be calculated for several channels.
   To reduce the complexity of this operation, a new parameter, r, was introduced in the calculations. It represents the operating range, or the spectrum range which a node must search for transmission opportunities. It translates the level of availability of the existing spectrum "holes".
   If the spectrum is very crowded, this parameter can be increased, or in other words, the operating range can be increased, allowing the CR device to search for other transmission opportunities in other channels.

2. **Maximum channel aggregation** - This feature is essential to achieve higher performance in DATA transmission because it enables the CR device to transmit a Data packet over multiple channels. It will decrease the transmission time and increase

the successful rate at the same time.

3. **Closest possible opening** - Normally, the CR device will choose the first idle channel to transmit, but if an higher successful rate is needed, the device may have to wait for more channels to become idle. In case of occurring several transmission opportunities, the device will choose the one with the shortest waiting time.
   The main concern will be the interference levels to the PUs, because they need to be below the successful rate threshold (α_T), therefore, the interference level should be under $1 - \alpha_T$.

2.6.3.3 Achievements

The SCA-MAC is able to maintain a success rate at a desired level and also achieving an higher level than the standard MAC in terms of throughput. This is a direct consequence of the spectrum opportunities prediction. Due to its superior performance over the standard MAC protocol, the SCA-MAC is a good choice for CR environments when applied to wireless ad-hoc networks.

2.7 Conclusion

Currently, the radio spectrum is becoming more loaded by the increasing number of mobile devices. The common radio spectrum users, the PUs or licensed users, are unable to use the available spectrum in a totally efficient manner, therefore, a more intelligent spectrum usage is required. The PUs are unable to adapt due to their rigid design, therefore there is a need for network users that are capable of taking advantage of this gaps. These kind of users, SUs, or non-licensed users are capable of overcoming that challenge, largely due to its capacity of sensing the radio spectrum and deciding what spectrum part to use according to the current spectrum occupation and without causing interference with the existing PUs.

There are several MAC protocols proposed for the CR environment adapted from the IEEE 802.11 standard. They grant a more efficient resource usage, performance and also the capability of coexistence between the PUs and the SUs.

In a nutshell, every MAC protocol designed for the CR environment has “improvements”

over the MAC protocol adopted in IEEE 802.11, and they are capable of dealing with the increasing needs of spectrum access.

Probably, in a near future, the radio devices will be constituted by software, instead of almost all hardware, and the software will be in charge of the device's capabilities in terms of adaptation and intelligence.

Chapter 3

Madwifi Description and Validation

3.1 Introduction

This chapter details the main aspects of the Madwifi (Multiband Atheros Driver for Wireless Fidelity) project and particularly the Linux® Kernel driver for wireless LAN devices equipped with Atheros® chipsets. It is split in five sections. The second section describes the driver features, main structure, driver version and the computer environment used in the laboratory. The third section explains the driver software organization as well the data structures from the driver data structure and the Kernel modules which constitute the Linux® Kernel driver. The fourth section describes the driver's transmission/reception scheme and all the important features related. The fifth section deals with the proposed module to achieve input and output data validation from the driver. The last section focuses on data analysis and validation from the proposed module.

3.2 Madwifi Description

3.2.1 Early History

Madwifi [Pro12] (Multiband Atheros Driver for Wireless Fidelity) was an open source project started in 2005 which had as primary objective, the development of Linux® kernel

drivers for wireless LAN devices with Atheros® chipsets.

Madwifi original author was Samuel Leffler. He started the project with the purpose of building a meshed community of wireless networks because the existing supported hardware was too limited at that time. He contacted Atheros® to get support for the open source community and he started to work on BSD (Berkeley Software Distribution) systems. His plan was to do the same work on the Linux® systems, but every people that he had invited refused and he kept working on Madwifi almost alone. At that time, Greg Chesson, an Atheros® employee, was the only one who supported Sam Leffler with Madwifi, but only on his spare time.

One of his primary goals while developing Madwifi for Linux® systems was to implement a 802.11 network layer stack that could be device-independent. As the driver development took place on Linux® systems, the FreeBSD 802.11 stack (net80211) was ported to Linux®.

This stack (net80211) was improved for one year on Linux®, but the Linux® Kernel developers came to believe that net80211 was not the 802.11 stack that they wanted due to design issues, so Sam Leffler felt that he should stop the development due to the lack of support from Linux® Kernel developers, and as a direct consequence, Sam Leffler left the project by the beginning of 2005 as well with Greg Chesson who also left Atheros® and changed his career. At that time, some volunteers decided to keep the development of Madwifi going on and they opened up the development process and also invited many developers to combine efforts in the improvement of one of the most advanced 802.11 drivers available for Linux®. Then, the official Madwifi project was born. The project was closed in March of 2012 due to the lack of significant bugs as well of new improvements in the latest version.

During the active time, the project has become an asset for Atheros® devices and for a large number of Linux® users. The most recent Linux® driver for wireless LAN devices with Atheros® chipsets, compatible with 802.11g standard, is designated by *ath5k* and is based on Madwifi driver, but, the biggest difference between them is that *ath5k* is not dependent from HAL (Hardware Abstraction Layer) (Subsection 3.3.3) unlike Madwifi, so the *ath5k* driver is now able to call hardware functions directly.

3.2.2 Driver Features

The Madwifi driver is considered to be one of the most advanced drivers for WLAN devices for Linux® systems. It has several features that allow users to have more control over their wireless devices and also allowing new experiments for academic and professional purposes. It supports several operational modes, like *sta (Station)*, *ap (Access Point)*, *adhoc (Ad-Hoc)*, *ahdemo (Ad-Hoc Demo)*, *monitor (Monitor)* and *wds (Wireless Distribution System)*.

The monitor mode is one of the features that makes this driver a very good tool for research at this level. It allows "sniffing" raw 802.11 frames from 802.11 frequency spectrum, making possible the analyzing of real-time traffic between several stations or access points. This feature was used in the current work to make possible having a more accurate perception of the frames that were transmitted between the stations or access points.

This driver allows running several Virtual Access Points (VAPs) on a single wireless network card. There is a base device that is usually named "wifi0" and VAPs sit on top of it. It is possible to have one VAP in monitor mode and other in station mode or any other configuration. By default, a single VAP is created in station mode and is called "ath0". Other VAPs that could be created next, will be named with the same rule *(athX)*, but only the number is incremented by default. This feature allows, for example, "sniffing" 802.11 frames with one VAP while being connected to an access point through another VAP. These VAPs can be managed using a tool called *wlanconfig* which allows their creation and destruction with several different nodes. This tool comes along with the driver package.

Madwifi also supports the Wireless Extension (WE), a generic API that is distributed with the Wireless Tools package for Linux®. This package has a set of tools to manipulate the Wireless Extensions (WE) and the Madwifi driver takes advantage of them enabling on the fly changes and customizations to the driver parameters. Tools like *iwconfig*, *iwlist*, *iwspy*, *iwpriv* and *ifrename* allow a more detailed control over the driver using the operating system terminal to pass specific configuration parameters to the driver.

3.2.3 Driver Software License

The Madwifi driver is open source but it depends on HAL (Hardware Abstraction Layer) which is closed source. The driver is provided under three-clause BSD and GPL v2 dual license. The binary HAL is provided under a proprietary license.

3.2.4 Driver Version and Linux® Environment

In this work, the chosen Linux® operating system is the *Ubuntu 12.04 LTS 32 Bits* with the Kernel version *3.2.0-35-generic-pae*. The Madwifi version used is the latest stable release, the *v0.9.4 r4180*, for 2.6.25 or higher Linux® Kernel versions.

3.2.5 Hardware Version and Specifications

The hardware used in this project was the PCI 2.2 network card *D-Link DWL-G520* with the Atheros® *AR5212/AR5213* Chipset *(rev 01)*. This chipset uses CSMA/CA (Carrier sense multiple access with collision avoidance) with ACK (Acknowledgement) for the Medium Access Control (MAC) and can switch between two modulation technologies, Orthogonal Frequency Division Multiplexing (OFDM) or Complementary Code Keying (CCK). These modulations are used according the data bit rate which can reach up to 108 Mbps with Super-G technology.

3.3 Madwifi Software Organization

The Madwifi driver (Appendix A.2) is formed by four main parts. The net80211 stack, the *ath* part, HAL and rate algorithms. These parts are the main Linux® Kernel modules. The driver has eighteen kernel modules, which are loaded at operating system startup according the user setup and some of them may not be loaded.

3.3.1 Net80211 Stack

As mentioned in Subsection 3.2.1, *net80211* was not accepted by Linux® Kernel developers, as a default 802.11 network layer stack for future Linux® releases. It was ported from BSD systems, as the FreeBSD system, and becomes the 802.11 stack for Madwifi driver.

Net80211 only supports Atheros® devices, even though, it supported several WLAN devices from other hardware manufacturers in BSD systems.

This stack contains generic 802.11 functions and callback functions that can be overridden by devices. It is implemented by several Kernel modules, but the main one, the *wlan.o*, is always running with another module depending on the operating mode that has been chosen. If the network adapter is operating in station mode, it will load the *wlan_scan_sta.o*. If the network adapter is operating in access point mode, it will load the *wlan_scan_ap.o*. Both kernel modules have specific functions for the selected operating mode.

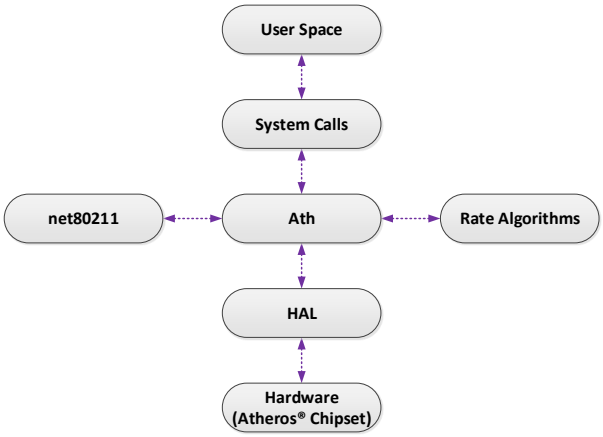


Figure 3.1: Madwifi driver structure.

3.3.2 Ath

In the Madwifi driver, the *ath* part is responsible for defining Atheros® specific callback for the net80211 stack and ensuring the correct hardware accesses using HAL. It is the core of the Madwifi driver because every packet processing, queuing, dequeuing, transmission and reception is defined in *ath*. This is illustrated in Figure 3.1, where it is possible to see the role of *ath* in Madwifi driver structure. It is more like a central piece in the puzzle that links all other three key pieces that were referred in the beginning of Section 3.3.

This part is implemented as a Kernel module, the *ath_pci.o*, which does the hardware probing to find the correct PCI device, so it can allocate all the structures and data, related to the correct device that is being used.

3.3.3 HAL (Hardware Abstraction Layer)

As stated at the end of Subsection 3.2.1 and in the beginning of the Subsection (3.2.3), the Madwifi driver depends on HAL (Hardware Abstraction Layer) to operate properly. This software is essential because it establishes the connection between Atheros® hardware and the driver itself.

The Atheros® chipsets are able to use a wide range of frequencies and the host software is able to control many radio aspects. To reach an worldwide agreement about transmission power and frequency, many manufacturers like Atheros®, implemented several solutions to enforce these standards. HAL is one of their solutions for these known security issues, and in this case was the solution adopted by Atheros®.

HAL is provided in binary form only and any attempt to modify it is very likely to be unsuccessfully. Although, HAL enables a good interaction between the chipset and the driver without compromising the radio transmission rules.

HAL is split in two parts, the precompiled binary part and the open source software part that is constituted by three files, *ah_os.c*, *ah_os.h*, and *ah_osdep.h*. These files compiled with the binary file will produce the loadable Kernel module *ath_hal.o*.

Any modification required due to changes in the Linux® operating system can be achieved by editing the *ah_os.c* file and mapping HAL and kernel functions calls to one another.

3.3.4 Rate Algorithms

Madwifi comes with three different bit rate control algorithms. *Onoe*, *amrr* and *sample*. A fourth one, *Minstrel* was imported to Madwifi from Linux® *mac80211* stack by Felix Fietkau on January 2005 which was originally designed and implemented by Derek Smithies. This algorithm bases its rate choices on the packets transmission success rates.

Onoe algorithm chooses bit rates by measuring the number of retries per packet. If packets need to be retransmitted at least once, it decreases the bit rate and only will increase it when at most 10% of packets do not need to be retransmitted.

Ammr uses another method to define the bit rate. It uses exponential backoff to avoid increasing the bit rate.

The last rate control algorithm, the *sample* algorithm, periodically sends frames at dif-

ferent bit rates so it can predict the best bit rate to be set. It also sends frames from different sizes so it can gather enough data to choose the best bit rate for the current radio environment.

All four algorithms are implemented as a single Kernel module that can be selected by the user from the operating system terminal or when Kernel modules are loaded, allowing different approaches to the bit rate control mechanism.

3.3.5 Madwifi Source Code Organization

Madwifi is built mainly in C language and the source code (Appendix A.2) is split equally in four main parts as described in the previous Subsections of this Section (3.3).

Despite the existence of tools in the driver package, the driver's core is constituted by the *ath* part, the *net80211*, *hal* and the bit rate control algorithms.

All code blocks will generate the Kernel modules that will be running in the Kernel space.

3.3.5.1 Main Data Structures Organization

The driver has some key structures [Str13] that store all important data and are used by all kernel modules. These structures are passed from one function or procedure to another as a parameter, although, the main driver structure is the *ath_pci_softc*, which is allocated when the driver probes for the right PCI wireless network card. There are also two important structures that belong to the Linux® Kernel and they will be addressed in the next two Subsubsections (3.3.5.2 and 3.3.5.3).

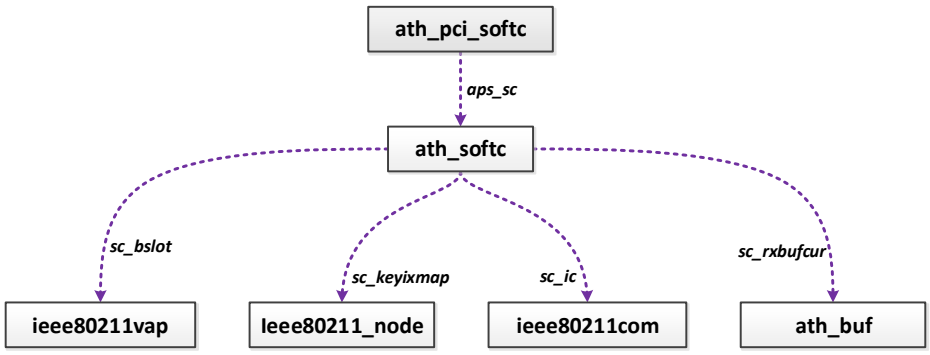


Figure 3.2: Madwifi driver main data structures with the respective pointers.

3.3.5.2 Net_device

This structure is allocated when the Kernel boots and each network interface will be described by this kind of structure defined in *<linux/netdevice.h>*.

The *alloc_netdev* Kernel function defined in *<linux/netdevice.h>* allows other specific driver structures to be allocated along the primary *net_device* structure. This feature is used when the Madwifi driver module, the *ath_pci*, is launched into the Kernel space and probes for the correct PCI device. The *net_device* structure is then allocated with the *ath_pci_softc* structure, which contains all specific driver structures.

This process takes place at the *ath_pci_probe* function in the *ath_pci* module, when the pointer *dev* for the *net_device* structure receives a new allocation using the Kernel function already referred, the *alloc_netdev* function, so each device will have an instance of the structure *ath_pci_softc*.

The default name for the pointer which points to this structure along the source code is denominated "*dev*", so by default "*dev*" should be pointing to an instance of this structure.

3.3.5.3 Sk_buff

This is a very important structure that belongs to the Linux® Kernel, and is used by Madwifi driver. It stores all information related with a frame and consequently gives access to all data from that frame (pointers, variables, etc.). This structure keeps frames for every device, so it is used for all devices present in the system

The Figure 3.3 shows the relationship between the *sk_buff* and the packet itself.

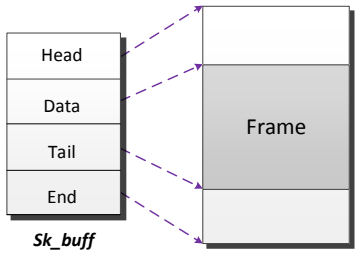


Figure 3.3: Sk_buff structure and the relation with the packet.

Each *sk_buff* has access to next and previous *sk_buff* as well the access to the list of *sk_buffs* where is inserted. Along the Madwifi source code it is possible to find several references

to this structure. Usually, the pointer that points to an instance of this structure is called "*skb*".

3.3.5.4 Ath_softc

After the *ath_pci_softc* structure allocation, as it was referred in the Subsubsection 3.3.5.2, the driver will use all structures related to the main one *(ath_pci_softc)*. This structure has another structure inside it, the *ath_softc*, which has all the other specific structures for the driver. This structure also have the pointer to the *net_device* structure.
The *(ath_softc)* allows the access of all other specific driver structures. This structure is the top of the tree of structures designed in Madwifi driver. From this structure it is possible to reach all driver's structures, including *ath_stats*, *net_device_stats* and *ieee80211vap* structures which will be very useful for this work, because they contain important statistics from the device.
The default name for the pointer which points to this structure along the source code is usually "*sc*", so by default the "*sc*" should be pointing to an instance of this structure.

3.3.5.5 Ieee80211vap

The structure *ieee80211vap* is related to the VAP technique explained in Subsection 3.2.2, which enables the Madwifi driver to manage several VAPs from only one physical access point. These VAPs can operate in different modes (*station*, *monitor*, etc.).
This structure has beneath it all the other structures from *net80211* stack that are related to the operation mode for the VAP in question.
As other main structures, this structure usually has also a default pointer name wherever is needed to point to an instance of this structure and is called "*vap*".

3.3.5.6 Ieee80211_node

This structure represents each station that is in range of the current network.
All node's related information is stored in this structure as well the specific information from each node device. Usually, the pointer that points to an instance of this structure is called "*ni*".

3.3.5.7 Ieee80211com

The *ieee80211com* is a a structure that stores common data between one or more VAPs. Any state shared by the main physical device and the net80211 stack is accessible through this structure. The pointer "*ic*" is used by default to point to an instance of this structure.

3.3.5.8 Ath_buf

This structure is very similar to the *sk_buff* (Subsubsection 3.3.5.3) but is specific for the 802.11 Atheros® wireless devices.
The *sk_buff* structure keeps packets from all devices that are being used by the operating system, but this structure just keeps specific packets from Atheros® wireless devices.

3.4 Madwifi Driver Transmission/Reception Scheme

The transmission/reception scheme is the most important mechanism in the Madwifi driver. It is responsible for all transmissions and receptions of data packets, beacons and management packets. This scheme also includes the monitoring feature provided by the Madwifi driver which allows raw data capture.
In this work, the transmission scheme is very important due to the need of changing hardware queue properties in real-time, such as the Contention Window (CW) values.

3.4.1 Contention Window (CW)

As mentioned in the beginning of Section 3.4, the CW is one of the most important properties in the hardware transmission queues of the Madwifi driver that will be used in this project. Each hardware transmission queue has associated a minimum and a maximum CW value that will be used by the Atheros® Wireless Lan Card chipset to control the transmission mechanism according to the 802.11 standard.
The 802.11 wireless transmission scheme uses by default a random "backoff" to control the transmission mechanism when the medium is busy. This is denominated random "backoff" because the "backoff" time or the number of slots to wait will be determined by a randomly generated number between the maximum (CWmax) and minimum (CWmin)

admissible values of the CW.

Usually, the minimum value is 31 and the maximum is 1023, although, in this project is essential to control the chosen "backoff" value through the CW values (CWmin and CWmax) to force a specific "backoff" value. This means changing the CWmax and CWmin values anytime during a transmission and also having the same values for the CWmax and CWmin if required.

Although, the HAL (Subsection 3.3.3) establishes some rules related to the CW values. It would be very useful to setup any value for the "backoff" through the CW values (CWmin and CWmin), but the HAL only allows exponential form values for this parameters. This can be described by the equation (3.1).

$$CWmin/max = 2^x - 1 \qquad x \in [1:10] \tag{3.1}$$

The previous equation illustrates the calculation done by the HAL to obtain the desired CW value. It is also important to notice that HAL will accept values between 0 and 1023 in decimal format, but it will use the previous equation (3.1) to obtain the desired value by approaching the given value in decimal format with the closest window value obtained through the equation.

Summarizing, only 10 values are allowed for the CWmax and CWmin parameters, which are obtained with the equation 3.1 and represented in Figure 3.4.

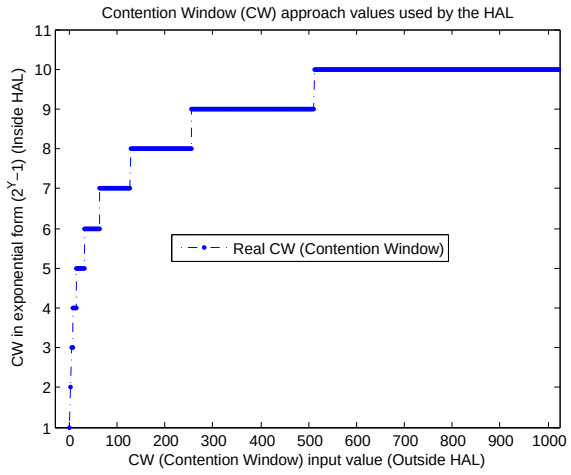


Figure 3.4: HAL Contention Window (CW) restriction.

It is noticeable that only 10 steps, corresponding to 10 different CW values, are allowed. Despite the fact of having any decimal number as input for the CW value, the HAL will only use the closest step value as CW.

This behavior (Appendix B.1) will decrease the freedom degrees related to the transmission "backoff" parametrization, but it is extremely important to predict it because any wrong approach that could be committed later would be problematic.

3.4.2 Transmission

The transmission scheme has several functions that are called during the process. The most important functions are represented in Figure 3.5. There are three types of transmission, the data frame transmission, the management frame transmission and the beacon transmission. These types are represented in Figure 3.5 in three distinct colors.

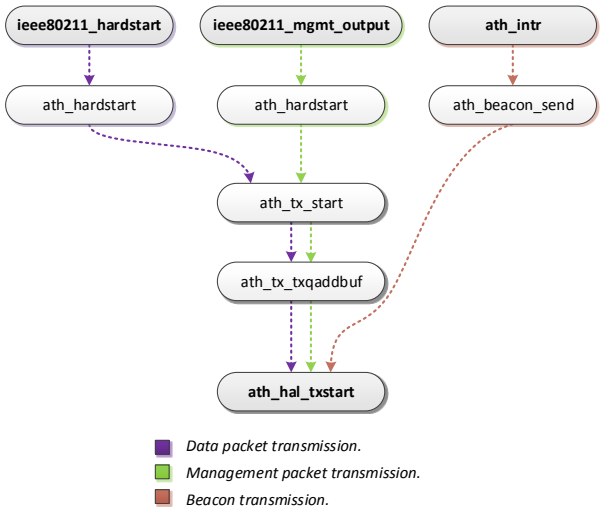


Figure 3.5: Madwifi driver transmission scheme.

After each successfully transmission processed by the HAL, an interrupt is triggered (*ath_intr()*) and the function *ath_tx_tasklet()* is called to update the transmission information.

The post transmission process finishes after executing *ath_tx_processq()* function, which is called to update the transmission queue (Figure 3.6).

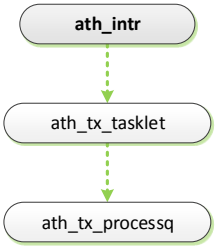


Figure 3.6: Madwifi driver post transmission scheme.

3.4.2.1 Data Frames

The data frame transmission, represented by the purple color in the Figure 3.5, starts with the Kernel call *ieee80211_hardstart()* (*dev->hard_start_xmit* of the current VAP) to transmit a frame, then, the driver function *ath_hardstart()* (*dev->hard_start_xmit* of the physical interface) is called and the data frame (*ethernet frame*) is encapsulated into the 802.11 frame. The function *ath_tx_start()* is called after and will encrypt the frame if the encryption is enabled, it will map the skb (*sk_buff*) to DMA (Direct Memory Access) and it will choose the correct transmission queue according to the frame priority that is set if the QoS (Quality of Service) control is set to be active.
The function *ath_tx_txqaddbuf()* will then insert the buffer into the selected transmission queue and call the HAL function *ath_hal_txstart()* to begin the transmission through the radio transceiver.

3.4.2.2 Management Frames

The management frames' transmission process is similar to the data frame transmission described in the Subsubsection 3.4.2.1, although, these frames are generated by the 802.11. The function *ieee80211_mgmt_output()* is responsible for the management frame transmission process. It calls the *ath_mgtstart()* (similar to the function *ath_hardstart()* described in Subsubsection 3.4.2.1) that will format the management frame according to the 802.11 standard. The rest of the process is identical to the one described in the Subsubsection 3.4.2.1.

3.4.2.3 Beacons

The beacon frame is controlled by HAL. The default time for the beacon transmission cycle is 100 ms, and when the trigger is fired by the HAL, an interrupt (*ath_intr*) is issued and the function *ath_beacon_send* is invoked to create and send the beacon frame.
In the final stage of the transmission, the beacon is passed directly to the HAL by calling the function *ath_hal_txstart()*, so it can be transmitted through the radio.

3.4.3 Reception

As it is clear in Figure 3.7, the frame reception mechanism is based in one interrupt. This interrupt is triggered by the HAL and the driver treats the interrupt with the function *ath_intr()*. The received frame is processed by the Linux® tasklet *ath_rx_tasklet()*, the *sk_buff* is retrieved and the receiving node is identified. After this process the function *ieee80211_input()* is called and it will treat the frames according their type (figure 3.7).

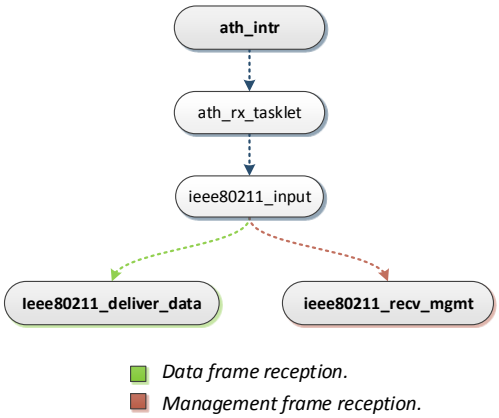


Figure 3.7: Madwifi driver reception scheme.

3.4.3.1 Data Frames

The frame's reception mechanism implemented in Madwifi driver is explained in the beginning of the Subsection 3.4.3. The mechanism is commonly adopted for data frames and management frames, but it differs after the last common function call, the *ieee80211_input()*. When the frame is identified as a data frame, the function *ieee80211_input()* will call the

ieee80211_deliver_data() to decapsulate the frame to the Ethernet format and deliver it to the Linux® Kernel. By this way a frame can reach the network layer implemented in the operating system.

3.4.3.2 Management Frames

Management frames are processed equally as the data frames and based on the same scheme as the one referred in Subsection 3.4.3. The only difference is that the function *ieee80211_input()* will identify the frame received as a management frame and will call the *ieee80211_recv_mgmt()* function to handle it instead of the *ieee80211_deliver_data()* function (Subsubsection 3.4.3.1).

3.4.4 Frame Sniffing (monitor mode)

One of the advantages of supporting multiple VAPs is the possibility of enabling simultaneous transmission and reception of data frames through a VAP in station mode and be able to "*sniff*" frames from all incoming and outgoing transmissions through another VAP in monitor mode.

It is possible to have a sense of what is being transmitting and received by any AP on this channel in real-time. This enables data confirmation for any transmission. The sniffing scheme has specific functions to treat all raw data that is captured on the virtual interface. Figure 3.8 illustrates how incoming and outgoing frames are captured when the VAP is in monitor mode.

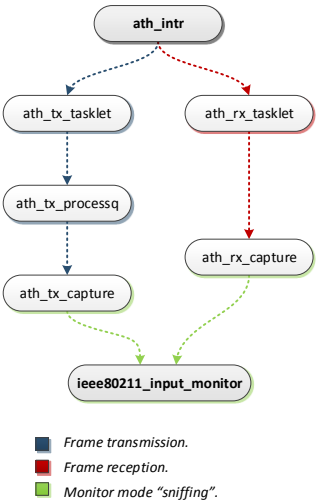


Figure 3.8: Madwifi driver sniffing mechanism (monitor mode).

3.4.4.1 Outgoing Frames "Sniffing"

When the *ath_tx_processq()* function is executed after a successful transmission (Subsection 3.4.2), if the VAP is in monitor mode, the *ath_tx_capture()* function is called and the *ieee80211_input_monitor()* function will handle the "sniffed" frame.

3.4.4.2 Incoming Frames "Sniffing"

The same process happens to the frames that are received and treated by the driver. If the monitor is enabled in the VAP then the function *ath_rx_capture()* is called by the *ath_rx_tasklet()* function to capture the received frame. Then *ath_rx_tasklet()* will call the function *ieee80211_input_monitor()* (like in the transmission process (Subsection 3.4.4.1)) to handle the "sniffed" frame.

3.5 Proposed Module for Validation of i/o (input/output)

The Madwifi driver runs in the Linux® Kernel space of the operating system by loading several modules described in Section 3.3. In this work, three modes were used for the VAPs: Station, Access Point and Monitor mode. Each one of these modes used in the VAPs, loads the needed modules into the Kernel space.

Usually, the driver uses the modules *ath_pci*, *wlan_scan_sta* (Station mode) or *wlan_scan_ap* (Access Point mode), *ath_rate_sample* or other modules for bit rate control, *ath_hal* and *wlan*.

One of the objectives planned for this work is to create a module dedicated to gather statistics from the wireless network card so it can make decisions based on that information, such as changing the CW for the different transmission queues.

3.5.1 *ath_sensing* module

This module will be loaded into the Kernel alongside with other Madwifi modules that were referred in the beginning of this Section (3.5). The Figure 3.9 illustrates the loading process of the proposed module.

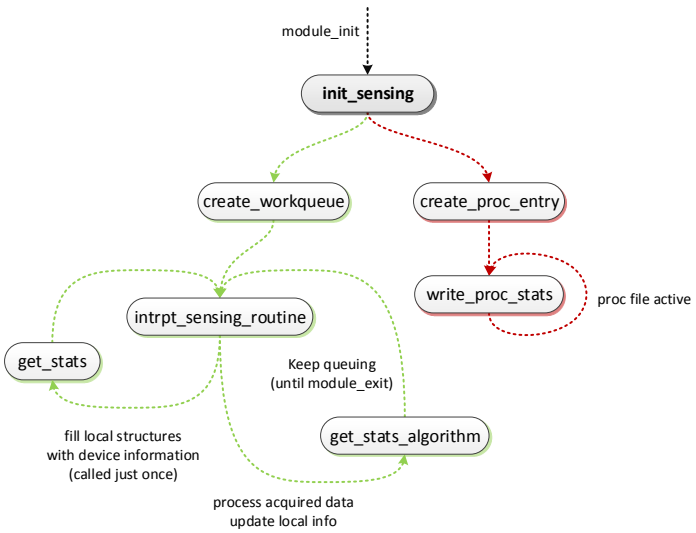


Figure 3.9: Initialization process for the proposed module.

The Kernel module is initialized with the normal initialization call from the Kernel, the *module_init*, which calls the module initialization function that was created to start the module and the respective services, the *init_sensing* function.

The module exit procedure is represented at Figure 3.10. It removes the proc entry, kills the running task and destroys the workqueue.

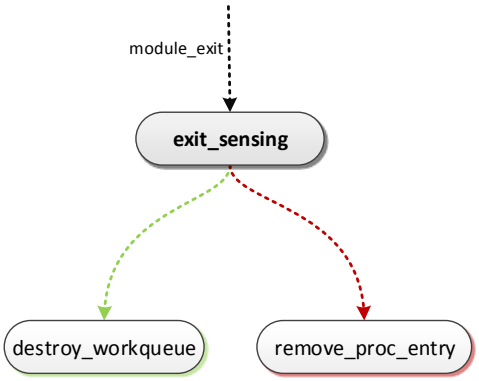


Figure 3.10: Unload process for the proposed module.

3.5.1.1 Workqueue and Sensing Routine

The module has a workqueue defined in the *<linux/workqueue.h>* file, which make possible to have some kind of threading system running in the Kernel module. The *workqueue* is a queue that allows queuing several tasks that will be executed every cycle time that has been chosen when the task "sensing" is initially queued.

The task will generate an interrupt routine each cycle time and it will be used to gather statistics from the device and update the module local statistics. The interrupt routine will start gathering statistics and applying the implemented algorithm to control the driver transmissions if the device and the VAP had already been allocated in the *ath_pci* module. The *ath_pci* allocations take place at the *ath_pci_probe* (in *if_ath.c* file) function and in the *ath_vap_create* function (in *if_ath_pci.c* file). They will fill the structures *ath_stats* and *net_device_stats* in the local copy of the sensing module. After the first run, the algorithm that was developed to control the CW is applied by comparing the old statistics with the newest ones and the local statistics are updated.

3.5.1.2 Proc Entry for Module Data Output

After having the sensing module working as described in Subsection 3.5.1 and Subsubsection 3.5.1.1 the next goal is to achieve real-time data output from the module to allow external data analysis.

This was possible due to the implementation of the Linux® proc filesystem in this module.

The Linux® proc filesystem allows communication between the Kernel space and the user space. This enables data reading from the Kernel module and write to it as well. In this case, the proc file reading was prioritized because on each read from the module and by having the time elapsed from the last read it is possible to have accurate data from the module.

The proc file entry is created in the module initialization and it is removed only at the module exit function (*exit_sensing*).

3.6 Validation Results

To validate the module output data and the consequent transmission results, several tools were used and new applications and scripts were created to analyze all information retrieved from the module. A Matlab® GUI (Guided User Interface) application and scripts were created, as well as two java applications.

The goal of the validation was to transmit a specific number of packages with a pre-established packet size and with a transmission cycle defined in microseconds (Equation 3.2). The chosen packet generation rate for all test scenarios was 11 Mbps. According the frame timings (Appendix A.1), the chosen packet generation rate guarantees a saturated operation, meaning that each transmitting node has always a frame to transmit.

The tests were made in a normal noisy environment (several transmissions occurring at the same time in the same channel) and in a controlled environment (inside a Faraday cage) to achieve the optimum results possible. The following equation (3.2) shows that 11 Mbps traffic generation rate can be achieved when 990 bytes are transmitted every 720 microseconds.

$$\begin{aligned} Bit\ Rate\ (Mpbs) &= \frac{TotalPacketSize\ (bits)}{TxCycle\ (\mu sec)} \\ &= \frac{990 \times 8\ (bits)}{720\ (\mu sec)} \\ &= 11\ Mbps \end{aligned} \tag{3.2}$$

Several transmissions were done by using applications developed in Java (Java Programming Language) specifically for this case. All transmissions used the UDP protocol and both applications communicated through an UDP socket. The transmissions statistics

were obtained through these applications and with the Madwifi tool *athstats*.
To validate these results, an comparison was established between the *ath_sensing* module output data from the proc filesystem and the *athstats* tool.

3.6.1 Data Transmission Application

This application (Appendix B.1) generates a UDP packet on each cycle (the cycle time is defined by the user in microseconds). This packet has a preselected size, that was chosen by the user before the beginning of the transmission. The packet size can be also adjusted by the user if needed. The packet has only the sequence number of the current transmission inside it as valuable data. This sequence number is used to control the missing packets on the reception side by the data reception application (3.6.2). The number of packets that are intended to be sent is selected before the transmission.
When the transmission starts, the application increases the sequence number of each packet during the transmission and it will count the packets sent through the network until reaching the maximum number previously established. The starting and finishing time is also stored by the application and displayed to the user, allowing the determination of the transmission duration.

3.6.2 Data Reception Application

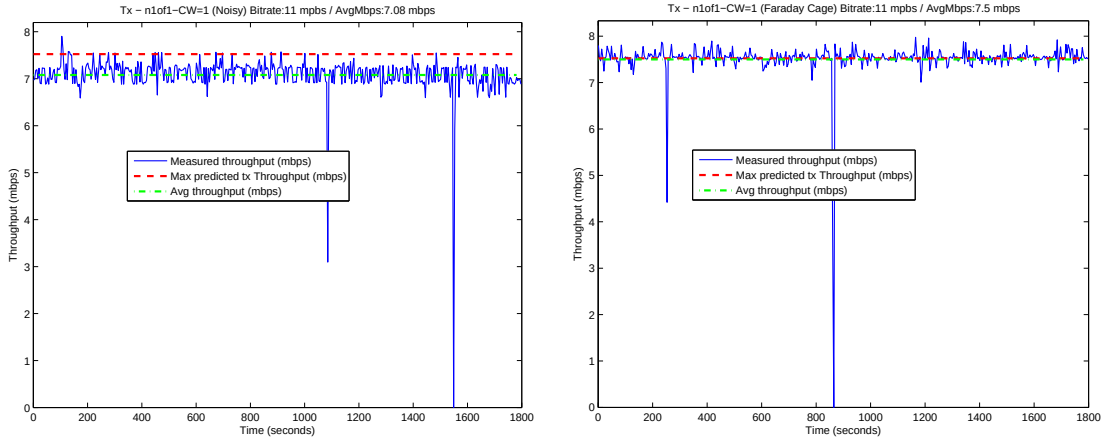
This application (Appendix B.2) was created with a simple purpose: receive UDP packets from the client application. This application receives the UDP packets destined to it and checks each packet for the sequence number. The sequence number provides key information for the missing packets, because the application stores the last sequence number received on the last packet and compares it to the newest sequence number received. The difference between the sequence numbers gives the missing packets since the last packet and this differential is displayed by the application.
This application uses a UDP server socket that will be listening in the specified port for incoming packets. The application only receives as input, the number of packets predicted for the transmission, which will enable the calculation of the ratio between the received packets and the missing ones.

3.6.3 Athstats

Like it was referred in the beginning of Section 3.6, the *athstats* tool is very useful because it displays current statistic for the Atheros® device. This tool was modified to report statistic data output to a text file in order to analyze the output data with a Matlab® script (Appendix B.2).

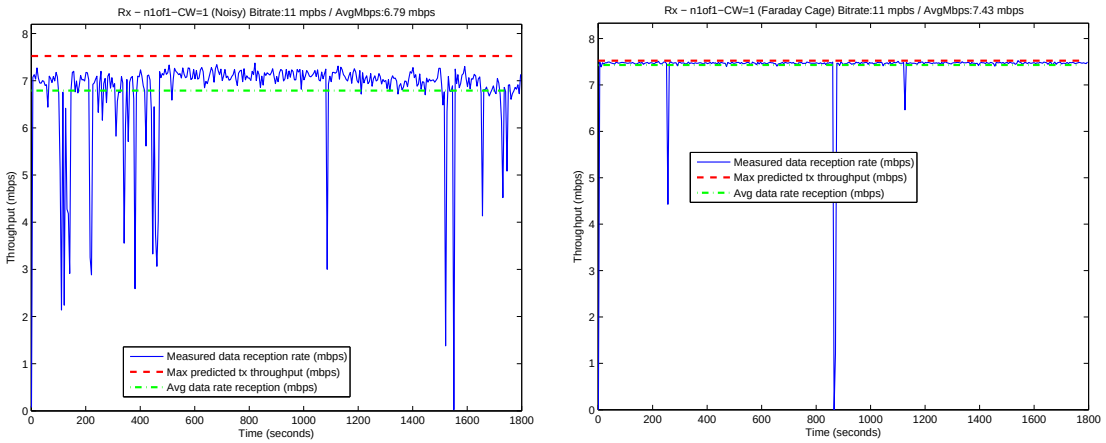
The data analysis produces results that will increase the knowledge about the current medium and the data bit rate achieved in the each transmission with a specific CW.

It was also essential to establish a comparison between two different radio environments. A normal daily environment with a good amount of interference and a controlled environment protected from external interference. This last test environment was achieved in a Faraday Cage in a laboratory environment.



(a) Noisy environment (b) Controlled environment (Farady Cage)

Figure 3.11: Transmission throughput comparison between the environments (a) and (b).



(a) Noisy environment

(b) Controlled environment (Farady Cage)

Figure 3.12: Reception throughput comparison between the environments (a) and (b).

Figures 3.11 and 3.12 depict the output data obtained with the *athstats* tool for a test transmission during 30 minutes, with data packets of 990 bytes, with a theoretical packet generation rate of 11 Mbps and using a CW equal to 1. Other CW values according to the ones allowed by the HAL (Subsection 3.3.3) and the results were analyzed in the Subsection 3.6.5.

These tests were essential to confirm the theoretical results for the throughput bit rate formula (1) described in Appendix A.1 and the CW effect in the transmission throughput. The controlled radio environment (Faraday Cage), allowed much better results than the normal radio environment with considerable noise levels, due to the lack of external interference. This results also permitted to confirm the discrepancy in terms of throughput when the transmission is affected by any kind of interference.

3.6.4 Real-time Proc File Analyser from the Proposed Module

As referred before (beginning of Section 3.6), the proc file output that was implemented in the proposed module allows real-time data analysis. This data is used to gather information about the device packet transmission.

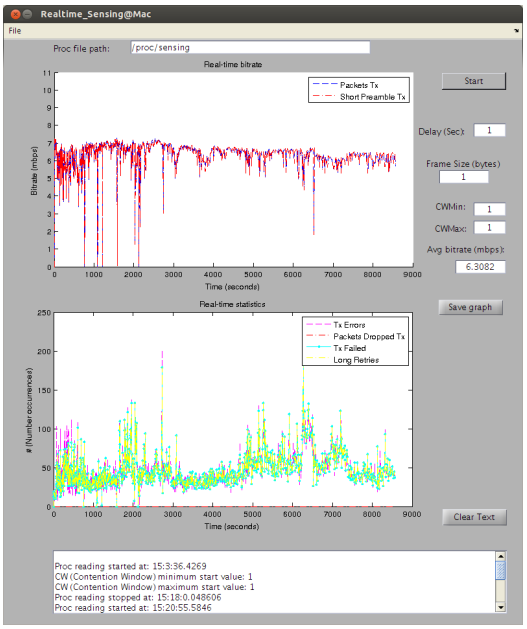


Figure 3.13: Proc file analysis application during one transmission (noisy environment).

The Matlab® application (Appendix B.3), represented in Figure 3.13, reads the proc file data and plots it. The gathered data can also be used for other calculations. It is possible to establish a comparison between this Figure (3.13) and the Figure 3.11, because both represent the data output from the Station (*client*) in the same transmission.

This data is similar to the one obtained by the *athstats* tool (Subsection 3.6.3) and only the acquisition times could differ in a few microseconds due to the operating system delays. The proc file output has all statistics from the driver related to the current transmission and also has the time gap between the data acquisitions. This time gap is essential to get a more precise value for the transmission throughput. It is possible to collect the average throughput in real time and the transmission errors/failures for the current transmission.

3.6.5 Gathered Data Analysis

As referred in Subsection 3.6.3, the *athstats* output was the key to validate the parametrization of the CW (showed in Figure 3.4) effect over the throughput bit rate. The CW, mentioned in Subsection 3.4.1, is an important parameter for this work, and therefore is essential to confirm the frame timings (Appendix A.1) and the maximum theoretical throughput bit rate for each CW value with a data packet of 990 bytes, given by the equation 1 described in the Appendix A.1.

These tests were made in a noisy radio environment without any kind of guarantees in terms of interference from other devices and in a controlled environment, inside a Faraday Cage, to experience a low level of interference. In each test, one node transmitted data at the maximum theoretical throughput of 11 Mbps (990 bytes packets transmitted each 720 μseconds). The packets were transmitted continuously for 30 minutes in order to gather a good amount of statistical information to allow consistent results in data analysis.
These tests have confirmed the impact of the CW in the transmission throughput, as well as the error associated to this prediction.

	Throughput				
CW	**Theoretical**	**(Noisy Environment)**	**Error**	**Faraday Cage**	**Error**
1	7.52 Mbps	7.08 Mbps	5.85 %	7.50 Mbps	0.27 %
3	7.46 Mbps	6.75 Mbps	9.52 %	7.42 Mbps	0.54 %
7	7.34 Mbps	6.52 Mbps	11.17 %	7.32 Mbps	0.27 %
15	7.1 Mbps	6.38 Mbps	10.14 %	7.08 Mbps	0.28 %
31	6.67 Mbps	5.94 Mbps	10.94 %	6.64 Mbps	0.45 %
63	5.95 Mbps	5.03 Mbps	15.46 %	5.91 Mbps	0.67 %
127	4.89 Mbps	4.09 Mbps	16.36 %	4.84 Mbps	1.02 %
255	3.61 Mbps	2.18 Mbps	16.9 %	3.56 Mbps	1.39 %
511	2.37 Mbps	1.93 Mbps	18.57 %	2.33 Mbps	1.69 %
1023	1.4 Mbps	1.14 Mbps	18.57 %	1.38 Mbps	1.43 %

Table 3.1: Maximum theoretical vs. real throughput according several CW values in a noisy and in a controlled radio environment (Faraday Cage).

The results represented in Table 3.1 show that in a noisy environment the deviation between the practical measurements and the theoretical values is higher than the deviation observed when the measurements are taken in the Faraday cage. Since the frame's retransmission mechanism was disabled in the practical experiences, the results indicate that there is a lower rate of success when receiving a packet in the noisy environment. This means that in a scenario with lower interference, as is the case for the Faraday cage, the transmitted packets experience less errors due to the low level of interference. In this

scenario the transmission errors have a lower weight in the throughput, and we can observe from Table 3.1 that the results obtained in the Faraday cage are very close to the theoretical ones. This means that the theoretical values are successfully validated, namely as the level of interference becomes more negligible.

3.7 Conclusion

Madwifi has a good potential for implementing new functionalities for wireless networks, despite the lack of information regarding the internal driver organization.

In order to implement a cognitive medium access scheme, which is the one of the main goals of this work, it was essential to have a clear understanding about the driver characteristics and working mechanism.

This chapter has focused the driver organization as well the driver flexibility for changes in the transmission parameters such as the CW. The CW control was the main concern in the first part of this work because it is a key factor for the adaptive contention mechanism proposed.

Having a better understanding of the driver statistics gathering mechanism is also crucial to validate the statistical results when the driver parameters, such as the CW, are modified. The driver statistics tool (*ath_stats*) was determinant for reaching a solid validation of the CW. The proposed module (*ath_sensing*) had also a key role in this validation by allowing real time readings of the CW values and transmission statistics.

With this information it was possible to validate the frame timings (Appendix A.1) for the chosen transmission parameters (bit rate, frame length, etc.), as well as having a more detailed description of the CW behavior. The CW calculation algorithm, described in Subsection 3.4.1, does not allow the best flexibility possible in terms of allowed input values due to HAL's constraints.

Despite the lack of flexibility in the CW setup, it was possible to achieve a good validation for its behavior, as well as a stable foundation for the implementation of the PU activity estimation (Chapter 4).

Chapter 4

Primary User Activity Estimation

4.1 Introduction

This chapter presents the theoretical foundations to estimate the activity of the primary users.

The SUs must be able to minimize, and if possible to avoid any kind the interference to the PUs when transmitting data between them. Therefore, being able to estimate the PUs' activity is fundamental to the implementation of the cognitive access system proposed.

The statistical data gathered by any SU on the network will be extremely valuable to improve the precision of the PU estimation mechanism.

The PU estimation will take into account the probabilities of transmission, collision and physical error for each node in the network from the SUs' perspective. Each SU will use its transmission statistics to calculate the probability of successfully transmitting and with that information estimate the PU activity.

The theoretical approach followed to design the estimator is described in this Chapter.

4.2 Individual Medium Access Probability

4.2.1 Single Node's Access Probability

Each SU runs a contention-based protocol, with a given medium access probability. This probability is directly related to the CW value used in the IEEE 802.11 DCF backoff process. The backoff period duration is given by the equation (4.1).

$$\textbf{\textit{Backoff duration}} = rand(0, CW) \times Slot\ duration \tag{4.1}$$

The random value obtained between 0 and the given CW value, multiplied by the slot duration will represent the backoff time. Basically, this time is the number of time slots used by the backoff counter for the backoff period. After waiting a DIFS period, and during the time that the medium is sensed as idle, the backoff counter will be decremented until it reaches 0, and then, the transmission begins.

$$\mathbf{CW} = \begin{cases} Min(2 \times CW, CW_{max}) & Transmission\ failure \\ CW_{min} & Transmission\ successfully \end{cases} \tag{4.2}$$

The equation (4.2) represents the behavior of the CW after a failed or successfully transmission. The CW will be doubled after each backoff period if the transmission fails. Eventually, with continuous failures, the CW value will reach the maximum allowed value (CW_{max}). When the transmission is successful, the CW value will be restored to the minimum CW value allowed (CW_{min}). Both CW_{min} and CW_{max} are previously established by the network card driver.

The backoff process can be illustrated by a Markov chain (Figure 4.1) where the ω represents the CW value and it is assumed that no transmission retries will occur.

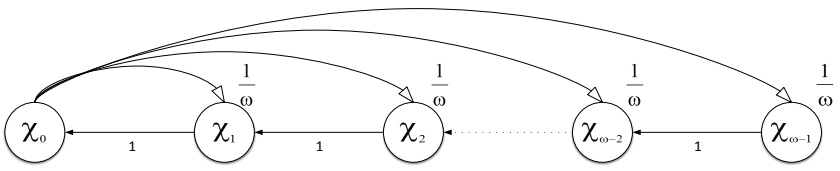


Figure 4.1: Markov chain illustrating the backoff process.

χ_0 represents the state where the transmission occurs for the given node. The probability of a node reaching the contention state χ_{w-1} is given by:

$$P_r(\chi_{w-1}) = \frac{1}{w} \times P_r(\chi_0) \tag{4.3}$$

Following the same rationale, the probability of a node reaching the state χ_{w-2} is given by:

$$\begin{aligned} P_r(\chi_{w-2}) &= P_r(\chi_{w-1}) + \frac{1}{w} \times P_r(\chi_0) \\ &= \frac{1}{w} \times P_r(\chi_0) + \frac{1}{w} \times P_r(\chi_0) \\ &= \frac{2}{w} \times P_r(\chi_0) \end{aligned} \tag{4.4}$$

Generalizing, it is possible to have a global equation that expresses the probability of a node reaching a generic contention state χ_{w-j}, given by:

$$P_r(\chi_{w-j}) = \frac{j}{w} P_r(\chi_0), \qquad 1 \leq k \leq w \tag{4.5}$$

Knowing that the probability of a node being in any state from χ_0 up to χ_{w-1} is equal to 1,

$$\sum_{k=0}^{w-1} P_r(\chi_k) = 1 \tag{4.6}$$

and using the arithmetical progression:

$$\sum_{k=1}^{n} a_k = \frac{n}{2}(a_1 + a_n), \tag{4.7}$$

the equation (4.6) can be re-written as:

$$P_r(\chi_0) + \sum_{k=1}^{w-1} P_r(\chi_k) = 1. \tag{4.8}$$

In the next step we isolated χ_0 because this state represents the effective access to the channel, and the probability of staying in this state represents the individual node's medium

access.

$$\begin{aligned} & P_r(\chi_0) + \frac{w-1}{2}[P_r(\chi_1) + P_r(\chi_{w-1})] = 1 \\ & \Leftrightarrow P_r(\chi_0) + \frac{w-1}{2}\left[\frac{w-1}{w}P_r(\chi_0) + \frac{1}{w}P_r(\chi_0)\right] = 1 \\ & \Leftrightarrow P_r(\chi_0)\left[1 + \frac{w-1}{2}\right] = 1 \\ & \Leftrightarrow P_r(\chi_0) = \frac{2}{w+1} \end{aligned} \tag{4.9}$$

Therefore, the node's medium access probability will be designated by τ:

$$\boldsymbol{\tau} = P_r(\chi_0) = \frac{2}{w+1} \tag{4.10}$$

Taking into account that every node is in saturation and as consequence, they will always have frames to transmit, the individual medium access probability for each node will be given by the equation (4.10).

4.2.2 Error and Collision Probabilities

The main concern when estimating the node's access probability is physical errors that can influence the correct reception of the frame. Physical errors and frame collisions always occur in contention-based wireless transmissions. These two factors can be determinant when estimating the node's access probability, therefore, it is important to be able to deal with these factors in order to achieve the best access probability possible for any node.
A transmission is considered successful when:

- The transmitted frame is not involved in a collision with SUs nor PUs
- No physical errors occur during the frame's reception.

After having defined these constraints, it is possible to translate them into an equation that expresses the SU's probability of a successful transmission:

$$P_s = (1 - P_c)(1 - P_e) \tag{4.11}$$

In Equation (4.11), P_c represents the probability of occurring a collision, and P_e represents the probability of occurring an error at the physical layer. Equation (4.11) establishes a

simple relationship between the two sources that influence the transmission success.
The collision probability can be translated by the next equation (4.12), where τ_{PU} represents the Primary User's (PU) access probability and τ_{SU} represents the Secondary User's (SU) access probability accordingly the number of SUs' nodes (n):

$$P_c = 1 - (1 - \tau_{PU})(1 - \tau_{SU})^{n-1} \tag{4.12}$$

Hereafter we assume that each node knows:

- The number of its successful transmissions (n_{TX_S}).
- The total number of transmissions attempted (n_{TX_T}).

We also assume that each node has access to the following information during the backoff period:

- The number of slots observed idle (S_I).
- The number of busy slots (S_B).

With these parameters it is possible to approximate the success and conditional collision probabilities:

$$\hat{P}_s = \frac{n_{TX_S}}{n_{TX_T}} \qquad \hat{P}_c = \frac{S_B}{S_B + S_I} \tag{4.13}$$

Taking the equation (4.11) and approximating it in the same way we obtain

$$\begin{aligned} \hat{P}_s &= (1 - \hat{P}_c)(1 - P_e) \\ \Leftrightarrow P_e &= 1 - \frac{\hat{P}_s}{1 - \hat{P}_c} \end{aligned} \tag{4.14}$$

Then, an approximation for the physical error probability can be given by:

$$\hat{P}_e = 1 - \frac{n_{TX_S}}{n_{TX_T}} \times \frac{1}{1 - \frac{S_B}{S_B + S_I}} \tag{4.15}$$

Knowing how to estimate the success and collision probability values it will be possible to calculate the access probability for the Primary Users (PU) from any Secondary User (SU) node in the network.

4.2.3 PU Access Probability Estimation

Each SU in the network must be able to calculate the PU access probability.
To make this possible, the PU access probability estimation can be achieved through the equations (4.11) and (4.12):

$$\begin{aligned}
&P_s = (1-P_c)(1-P_e) \Leftrightarrow \\
&\Leftrightarrow P_s = [1-(1-(1-\tau_{PU})(1-\tau_{SU})^{n-1})](1-P_e) \Leftrightarrow \\
&\Leftrightarrow P_s = (1-\tau_{PU})(1-\tau_{SU})^{n-1}(1-P_e) \Leftrightarrow \\
&\Leftrightarrow 1-\tau_{PU} = \frac{P_s}{(1-P_e)(1-\tau_{SU})^{n-1}} \Leftrightarrow \\
&\Leftrightarrow \tau_{PU} = f(P_s, \tau_{SU}, P_e, n) = 1 - \frac{P_s}{(1-P_e)(1-\tau_{SU})^{n-1}}
\end{aligned} \tag{4.16}$$

The error probability can be also approximated with:

$$P_e = f(P_s, \tau_{SU}, \tau_{PU}, n) = 1 - \frac{P_s}{(1-\tau_{PU})(1-\tau_{SU})^{n-1}} \tag{4.17}$$

Splitting the problem into a two equation system formed by the equations (4.12) and (4.14), will allow obtaining an individual equation for the PU access estimate through the SU node's probability of success.

$$\begin{cases}
P_s = (1-P_c)(1-P_e) & \textbf{(1)} \\
P_c = 1-(1-\tau_{SU})^{n-1}(1-\tau_{PU}) & \textbf{(2)}
\end{cases} \tag{4.18}$$

Developing the first equation, using the second equation:

$$\begin{aligned}
&\Leftrightarrow P_s = (1-P_c)(1-P_e) \Leftrightarrow \\
&\Leftrightarrow P_s = (1-\tau_{SU})^{n-1}(1-\tau_{PU})(1-P_e) \Leftrightarrow \\
&\Leftrightarrow \tau_{PU}(\hat{P}_s, \tau_{SU}, n) = 1 - \frac{P_s}{(1-\tau_{SU})^{n-1}(1-P_e)}, \text{ where } P_s \approx \hat{P}_s = \frac{n_{TX_S}}{n_{TX_T}}
\end{aligned} \tag{4.19}$$

Developing the second equation:

$$\begin{aligned} &\Leftrightarrow P_c = 1 - (1 - \tau_{SU})^{n-1}(1 - \tau_{PU}) \Leftrightarrow \\ &\Leftrightarrow \tau_{PU}(\hat{P}_c, \tau_{SU}, n) = 1 - \frac{1 - P_c}{(1 - \tau_{SU})^{n-1}}, \text{ where } P_c \approx \hat{P}_c = \frac{S_B}{S_B + S_I} \end{aligned} \tag{4.20}$$

With these equations it is possible to calculate the PU access probability from a SU node using the error probability or the collision probability and the success probability from the node itself.

It is important to notice that all tests were conducted inside a Faraday cage to minimize the interference levels, and therefore, the error probability can be neglected when estimating the PU's access probability from any SU node on the network.

The number of SU nodes on the network can be easily known through the network's access point. Each node can receive the number of SU nodes present in the network through a broadcast message from the access point and then combine all the information to estimate the PU access probability.

When the probability of error approaches zero, the main equation for the PU access probability estimation (4.19) can be easily simplified to:

$$\begin{aligned} P_s &= (1 - \tau_{SU})^{n-1}(1 - \tau_{PU}) \Leftrightarrow \\ &\Leftrightarrow \tau_{PU} = 1 - \frac{P_s}{(1 - \tau_{SU})^{n-1}} \end{aligned} \tag{4.21}$$

$\hat{P}_s$ is the probability of success obtained from each node internal statistics for each transmission and will be used to calculate the PU's access probability. Doing the following approximations, $P_s \approx \hat{P}_s$ and $\tau_{SU} = \frac{2}{w+1}$, we finally obtain:

$$\boldsymbol{\tau_{PU}} = 1 - \frac{\hat{P}_s}{(1 - \frac{2}{w+1})^{n-1}} \tag{4.22}$$

Finally, knowing the CW value (w), estimated the SU probability of success ($\hat{P}_s$) and the number of SU nodes (n) accessing the medium, the access probability of the PU can be determined using the Equation (4.22).

4.3 Conclusion

The developed PU activity estimation scheme allows any SU node to be aware about the PU access probability, or in other words, be aware about the possibility of having a PU transmitting on the network by predicting its activity.

The ability to predict the PU activity is one of the main objectives of this work. There are some pre-established conditions in order to adopt the proposed scheme. Each SU node must be able to know its success probability, the number of SU nodes on its network and the adopted CW value. After fulfilling these constraints, the scheme can be employed. The performance of the proposed scheme is analyzed in the next Chapter (5).

Chapter 5

Performance Analysis

5.1 Introduction

In the previous Chapter (4), we have presented an estimation scheme to enable the PU's activity estimation from any SU node on the network. To achieve this, several modifications to the Madwifi driver were performed, new applications were created specifically for this implementation and several results were analyzed.

A set of simulations tests were performed in the network simulator version 2 (NS-2) and a practical implementation of the subjects studied in the simulations was tested in a real network that was set up inside a Faraday Cage to ensure a controlled radio environment with lower interference levels from external sources.

A comparison is performed between the results of the practical tests and the computer simulations. The performance of the proposed scheme is analyzed through the results from both tests.

5.2 Madwifi's Implementation

The Madwifi driver, like it was described in Chapter 3, was the central piece for this work. After controlling it and having a full understanding about its functioning and the parameters it operates, the next logical step would be implementing new mechanisms on top of it or modifying the actual driver to achieve the desired behavior.

In this part of the work, there was a need to be able to control the driver's internal

parameters in real time and obtain the maximum amount of statistical data. This was imperative in order to implement the developed protocol for the PU's access probability estimation, and therefore, the CW values and the transmission details were considered the most relevant information for this implementation.

In the end of this implementation it is expected to have a reliable software test environment for the estimating scheme and also a good launch pad for implementing new protocols or mechanisms in the future.

5.2.1 Driver's Sensing Module input/output

Like it was described in the Section 3.5 of Chapter 3, a Kernel module (Appendix A.2.2) was created to ensure real time communication between the driver and the user's space. This module permits real time modifications and readings of the CW values, although, it can also be used to retrieve valuable statistical information from the driver. The module uses the proc file system, to act as an intermediary between the driver and the user's space.

The first module version, referred in the Section 3.5 of Chapter 3, was modified to allow input data from the user's space.

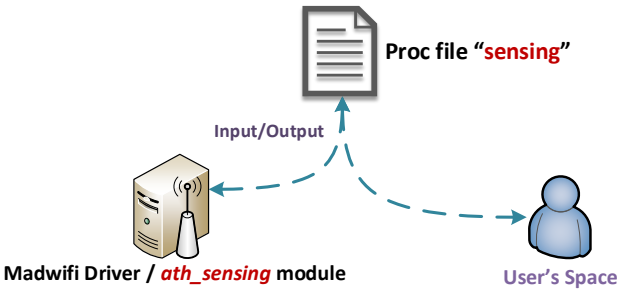


Figure 5.1: Proc file system used by the *ath_sensing* module.

Figure 5.1 illustrates how the communication is made between both communicating ends. The proc file is updated periodically and can be read or written by an application running in the user's space. Mainly, this feature provides direct access to the CW values by allowing their reading and modification.

The sensing routine that is running inside the *sensing* module, produces statistical data that is printed to the proc file periodically and provide an important source of information

about the driver's transmission/reception behavior. This data can be obtained in real time from the user's space.

The input values for the proc file, will be the maximum and minimum values of the CW (CW_{min}, CW_{max}). These values can be passed to the driver through the Operative System's command line using a simple command (ex: "echo cw_{min}:cw_{max} > /proc/Sensing_MAC/sensing"). This command can be executed directly into the Operative System's command line or through an application.

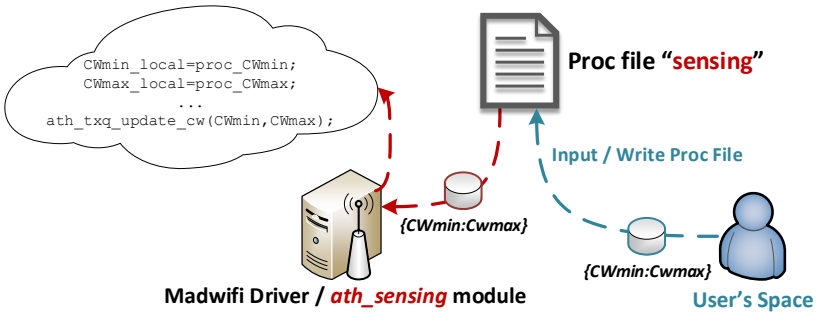


Figure 5.2: Proc file system writing from user's space to *ath_sensing* module.

The figure 5.2 shows how the information is passed from the user's space to the driver. The contention window values are passed through a command that will write them in the proc file after its execution. The proc file will be read by the driver's sensing module and the CW local values are then updated. The sensing routine (Subsubsection 3.5.1.1, Chapter 3) running periodically, will update the driver's CW values into the hardware registers.

5.3 Developed Applications

The Chapter 3 refers two applications that were developed to ensure specific communication between nodes. These applications were the first step for developing the final test framework to validate the proposed PU activity estimation scheme. They were crucial to validate the driver's performance and its specific parameters (Section 3.6 of Chapter 3). These applications were based in a client/server scheme for the transmission of UDP pack-

ets with configurable generation rates and sizes. This client/server scheme was the start line to develop a test framework with several SUs and a PU transmitting UDP packets simultaneously.

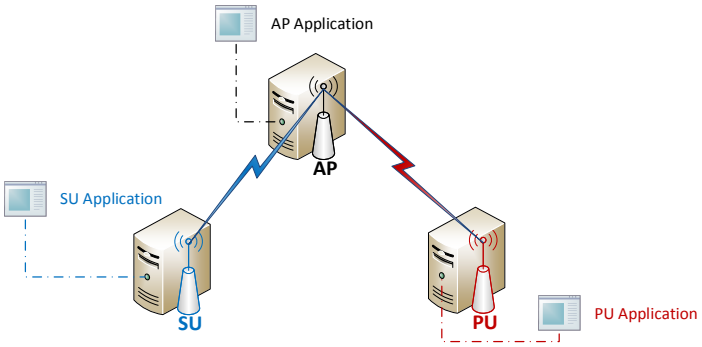


Figure 5.3: starting point of the Test framework with the SU, PU and AP nodes.

The figure 5.3 illustrates how the first test environment was set up. It represents the first network created to ensure a good operation of all node's specific application. The number of SUs can change depending on the test scenario that is requested.

After the initial tests, a second network functionality was implemented. The driver's sensing module (Subsection 5.2.1) interaction through an application that could be running on every node was a priority. This was needed to allow direct communication between the driver and the user's space. This application had to coexist with the running applications referred, such as the AP, PU and SU applications, without causing any interference to them. This new scenario is represented in Figure 5.4.

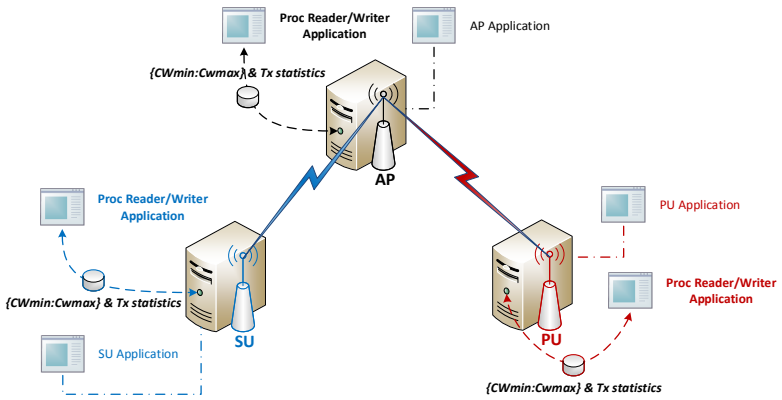


Figure 5.4: Proc file reader/writer running in each node.

5.3.1 PU Application

The PU application (Appendix C.1.2) was designed to translate a PU behavior. The PU usually is not constrained in terms of medium access by any SU in the network, but in this case, the SUs can interfere with PU's medium access. This work tries to minimize the impact of SUs in the PU medium access, by setting the PU CW values to the minimum value (CWmin=1,CWmax=1), and the SUs' CW to higher values (ex: CWmin=31,CWmax=31).

5.3.1.1 PU Behavior

The PU behavior adopted in all tests was implemented within the PU application. This behavior controls how the PU's transmissions are scheduled by using a previously established access probability (τ_{PU}).

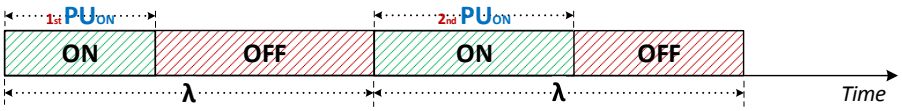


Figure 5.5: PU's access scheme based on the exponential calculation.

The PU's behavior is represented in Figure 5.5. Being λ the complete period of activity and inactivity of the PU, formed by an active period where the PU is transmitting (PU_{ON}) and an inactive period where the PU is not transmitting (PU_{OFF}).
The PU's activity is modeled by an exponential distribution for the ON (active) durations. λ can be defined as:

$$\lambda = PU_{ON} + PU_{OFF} \tag{5.1}$$

$$\frac{1}{\lambda} = \frac{PU_{ON}}{\tau_{PU}} \tag{5.2}$$

Equation (5.2) provides the mean value ($\frac{1}{\lambda}$) for the exponential distribution used to obtain the active durations of the PU. The initial λ value for a given PU access probability is calculated using a PU_{ON} duration equal to the PU's static frame duration ($T_{PU} = 720\mu s$). λ provides the mean value ($\frac{1}{\lambda}$) to use in the exponential distribution calculation to obtain newer PU_{ON}/τ_{PU} values. Although, these values must obey to a restriction. If the

generated value for the PU_{ON} duration is less than the PU's frame duration, the PU_{ON} value is recalculated through a new realization from the exponential distribution.
The access probability (τ_{PU}) for the PU is specified in steps of 5% and it is possible to have a PU transmitting with access probabilities from 5% up to 100%.

5.3.2 SU Application

The SU application described in Appendix C.1.1 is in charge of transmitting packets.
This application has a running task during the test period and will generate packets with a configurable size and according to a given packet generation rate.
Since the goal is transmitting while in saturation, the packet generation rate must be higher enough to guarantee a constant packet transmission, which means, that there will be always packets waiting for transmission in the transmission queue.
This application can also be controlled by a remote control panel unit (Subsection 5.3.5). It can also report periodically the transmission evolution by reporting to the main control panel the number of sent packets.

5.3.3 Access Point Application

This application (Appendix C.1.3) was developed specifically for handling the data packets reception from any node transmitting packets. It acts as a simple UDP server and filters the received packets from every known IP address. It can also calculate the loss packet ratio for every node by computing the difference from each packet number at its arrival. With this, it is possible to know how many packets were loss for every node or device.
This application evolved from the first version that was referred in Subsection 3.6.2 of Chapter 3. It was modified in order to enable the construction of an internal database to keep record of any connected node's packet transmission statistics. It was also modified to be controlled by a remote control panel unit (Subsection 5.3.5) for reporting the packet's statistics.

5.3.4 Proc Reader/Writer Application

The proc reader/writer application (Appendix C.1.4) establishes a connection with the driver through the proc file from the sensing module. It also starts or stops the driver's statistical readings from *athstats's* tool (Subsection 3.6.3, Chapter 3), so it can generate an output trace file to be latter analyzed.

This application is controlled by a remote control panel unit (Subsection 5.3.5) and all gathered data is reported to the main control panel unit. The application is also responsible for changing the CW values and control the data gathering process through the *athstats* trace file generation.

5.3.5 Control Panel Application

This application (Appendix C.1.5) was developed to be the brain of the test framework. It is responsible for controlling the data reading and writing through the Proc Reader/Writer application running on every node of the network and also for controlling every test transmission for all network nodes.

The application is split in two interfaces. One for controlling the proc reader/writer applications that are running on every node and other for controlling the packet transmission/reception applications that are also running on every node.

This application is meant to be running in a dedicated terminal so it can operate as the control center of the network.

5.3.5.1 Node's Transmission Control

This part of the control panel application was developed with the objective of controlling all node's transmissions. A registration procedure is implemented, so every node that connects to the control panel is registered and has a connection slot assigned to it.

With this interface is possible to have some information about the current transmission in each node and configure the test duration and also start the transmissions in all nodes simultaneously. The number of connected nodes can vary depending on the current test needs.

5.3.5.2 Node's Proc Reader Control

This part of the control panel application was created to control all the proc reader/writer applications from all network nodes.
The interface shows all node's CW values and also allows real time modification of them for any node.
One of the main features is the possibility to control the data gathering of all nodes through the *athstats* tool. Each proc reader/writer application executes a command in the Operative System's command line to interact with the tool.
An add-on was also implemented in order to enabled real time plot reading of the transmissions, so it will be possible to have live graphs of the current transmissions. This feature was implemented with an API from an open source tool, the Live Graph tool.

5.3.5.3 Live Graph Tool

Live Graph[Gra08] is an open source tool with a very useful API. This API was used for implementing a file writing system in the control panel application for gathering the reported data of the proc reader/writer application from the network nodes.
With this API, a trace file can be produced with the gathered data and it can be plotted in real time using the Live Graph tool.
One of the main advantages of using this API was the fact of being able to have data series in the trace file that can be plotted in real time, so it is possible to use it to see the transmission evolution in any node and choose what data series to plot.

5.4 Final Test Environment and Network Setup

The previous applications work together to enable the creation of a test framework with all the needed functionalities such as real time data reading, CW values modification, transmission control of any node and data trace file generation for every network node.

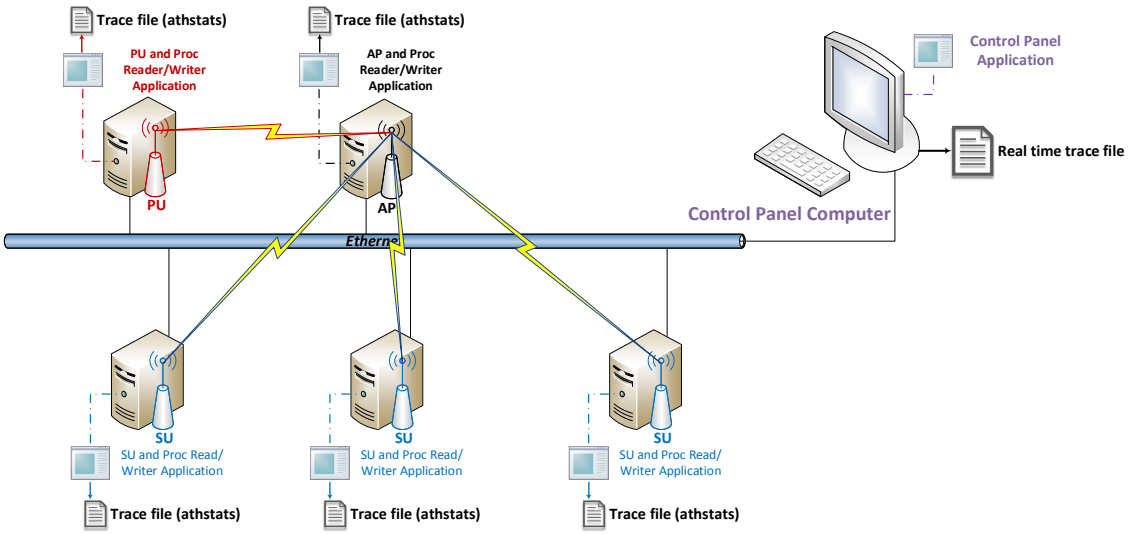


Figure 5.6: Complete diagram of the test framework and network.

The Figure 5.6, illustrates the complete network setup that was used in the final tests to access the PU activity estimation scheme.

All nodes are connected to the control panel computer through the ethernet network which is only used for control and data report purposes, allowing the wireless network to be exclusively dedicated for the needed tests. The three SUs, the PU and the AP will be transmitting information through the wireless network while periodically reporting the gathered statistical data to the control panel application. All nodes are controlled by the control panel application and they all store the gathered statistical data locally in data trace files to be later analyzed.

5.4.1 Setup Parameters

For each transmission, some parameters were set up. These parameters were defined for each application and also for the Madwifi driver.

For each one of the nodes running the Madwifi driver, the following parameters were adopted:

- **Bit rate and IEEE 802.11 mode** - IEEE 802.11g mode at a fixed bit rate of 11Mbps was adopted.

For every transmission/reception application (PU, SU and AP) running on each node several setup parameters were defined:

- **Test duration** - The time duration of each test performed. Usually not less than ten minutes.
- **Packet size** - It was chosen a packet size of 990 bytes for every test performed. The packet size can also be adjusted if needed.
- **SUs' Packet generation rate** - The time interval between the packet generation. By default, each packet will be generated every 720 milliseconds with a size of 990 bytes, so the maximum theoretical bit rate is achieved to guarantee traffic saturation for each node.

For the PU application:

- **PU's channel occupancy** - The PU's channel occupancy can be also controlled through the control panel. It can be set up between 0% and 100% in steps of 5%.

For every proc reader/writer application running on each node controlled by the control panel application:

- ***athstats* tool data gathering** - The data gathering through this tool can be controlled from the control panel application.
- **CW values** - The node's CW minimum and maximum values can be also configured from the control panel application within a range of 1 to 1023 respecting the constraint $2^n - 1, \quad n \in [1 : 10]$.

5.4.2 Testing Procedures and Data Gathering

The test procedure can be summarized in the following steps:

1. The Control Panel application is started in the main command computer connected to the Ethernet network.
2. Every node from the test network establishes a connection with the control panel application through the Ethernet network.

Each SU node runs one instance of the SU's application and one instance of the proc reader/writer application to control the *athstats* tool for generating statistical data output.

Each SU node runs one instance of the PU application and also one instance of the proc reader/writer application.

The AP node runs one instance of the AP application and also one instance of the proc reader/writer application.

3. After connecting all node's applications to the control panel application through the Ethernet network the transmission parameters are configured, such as the test duration and the node's CW values.

4. The statistical data gathering is started on each node through the proc reader/writer applications.

5. The transmission is started in all nodes from the control panel application.

6. When the node's transmission time expires they automatically stop transmitting and the data gathering of the proc reader/writer application is stopped through the control panel.

These tests produce a considerable amount of statistical data to be further analyzed and establish a comparison between the theoretical/simulation results and the obtained practical results. The gathered data has all the needed information about the transmissions that occurred during the tests duration. This information is analyzed with several Matlab scripts (Appendix C.2) and the results are generated according to the gathered data for each node.

The obtained results constitute the main source to evaluate the performance of the proposed scheme through a comparative analysis.

5.5 Performance Results

The results obtained in the simulations (Appendix C.2.1) and in the practical (Appendix C.2.2) tests were analyzed in order to obtain the SU node's success probability. This

probability will be used to estimate the PU's access probability. Each node has a success probability according the contention window value and the PU access probability used in the simulations or in the practical tests. This data will provide a more thorough analysis of the test environment for these scenarios.

Since the PU's contention window value is set to the minimum allowed (CW=1), it is important to take into consideration that only CW values from 3 up to 1023 (CW values given by $2^n - 1$) were used in the tests for the SUs.

Four different situations were analyzed. With a PU always present, several tests were performed with the SUs varying the number of SUs from one up to four. The next two Subsections 5.5.1 and 5.5.2, analyze the results of one of the tests performed. This test considered three traffic-saturated SU nodes transmitting to the AP and one PU transmitting to the same AP with the behavior described in the Subsubsection 5.3.1.1.

5.5.1 Simulation Results

A large amount of statistical data was produced for every simulated scenario. This data is used as reference for further comparisons with the practical tests.

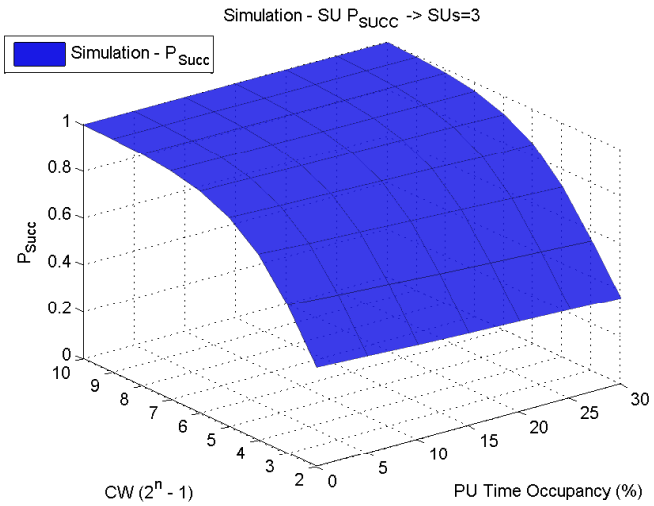


Figure 5.7: Simulated SUs' $\hat{P}_S$ accordingly the PU's time occupancy and the CW.

In the Figure 5.7 the transmitting success probability for a SU node is represented accordingly the CW values and the PU's access probability adopted in the simulations performed for this scenario. It is noticeable that when SUs adopt CW values less than 31 ($2^5 - 1$),

the success probability drops considerably. With smaller CW values, the SUs transmission aggressiveness increases dramatically.

For a better understanding of the PU's access probability in the SUs success probability, the relationship between these two values for different CW values is represented in Figure 5.8. shows

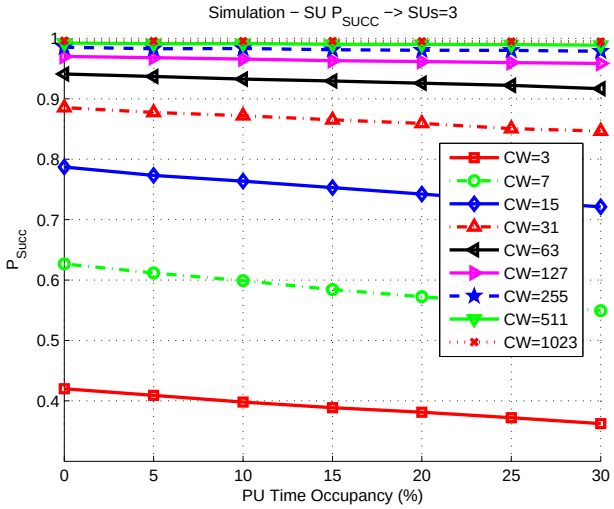


Figure 5.8: SUs' success probability depending on the PU's access probability.

Accordingly to the Figure 5.8 it is possible to confirm the trend observed in the Figure 5.7. Bigger CW values will guarantee a better transmission success probability.

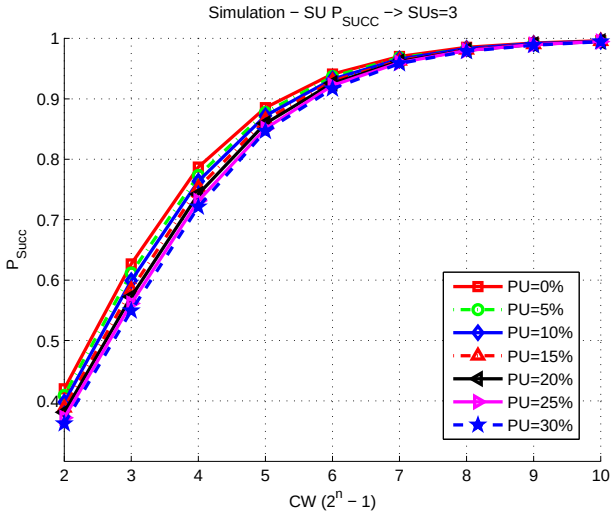


Figure 5.9: SUs' success probability depending on the CW value.

Figure 5.9 shows the similarity between the SU's probability of success for different values

of PU's access probability. The PU's access probability will influence the SU's success probability but not in such a higher level than the CW value. This is because increasing the CW leads to an increase of the contention period and the SU is less aggressive in terms of accessing the medium, decreasing the probability of being involved in a collision with other SUs, PU or both. More transmission aggressiveness will lead to more frame collisions and as result it will dramatically decrease the success probability. When using the same CW values for different PU's access probabilities, Figure 5.9 also shows that when PU's access probability increases the success probability of the SUs decreases, although, this decrease is very small when compared with the change of the CW.

5.5.2 Practical Results

These results, obtained through the practical tests in the Faraday Cage, will be used to establish a comparison with the simulation results previously described in Subsection (5.5.1). It will be applied the same analysis adopted in the simulation results. The results hereafter presented were obtained in the same scenario, where 3 SUs transmit to the AP and one PU transmits according to the model described in Sub-subsection 5.3.1.1.

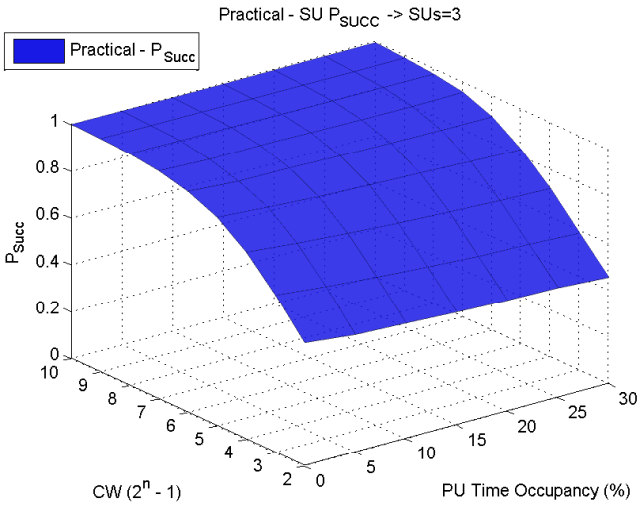


Figure 5.10: Practical SUs' $\hat{P}_S$ accordingly the PU's time occupancy and the CW.

Figure 5.10 illustrates the evolution of the SUs' success probability for the allowed CW values and the given PU's time occupancy.

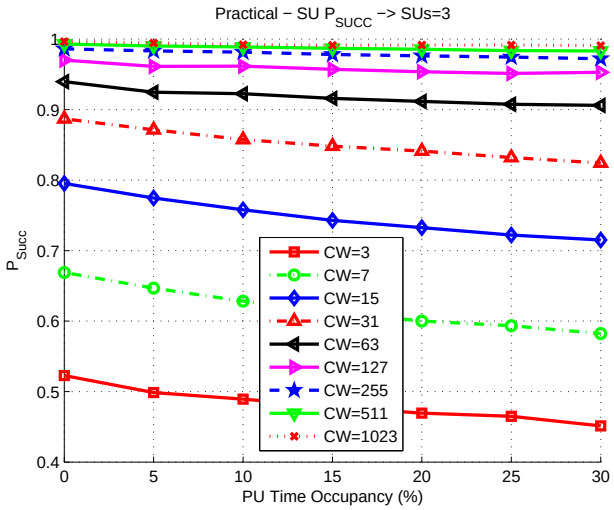


Figure 5.11: SUs' success probability depending on the PU's access probability.

As can be observed, smaller contention windows increase the number of frames collisions and consequently, decrease the SU's success probability. The PU's access probability has also an important role in the SU's success probability but not so significant as the CW value.

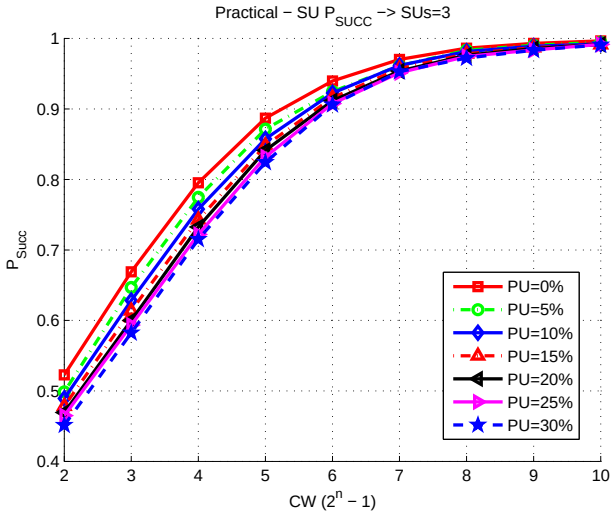


Figure 5.12: SUs' success probability depending on the PU's access probability.

Figure 5.12 exemplifies how the CW value affects the SU's success probability in a real transmission. It is clear that this parameter has the main role in the success probability variation, although, the PU's access probability also contributes for decreasing the success probability as it is increased.

5.5.3 Comparative Analysis

It is possible to establish a comparison between both simulation and practical results. It is clear that for both test scenarios, the SU's success probability behavior will be influenced mainly by the CW value used by the SUs' nodes. The increase of the PU's access probability will be also an important factor for decreasing the SU's success probability but with a smaller impact.

The results obtained in the practical tests were very similar to the ones obtained through simulations and therefore, the adopted model is valid for both situations. One thing to take into account, is the fact of having a bigger deviation in the practical results that arise from the fact that the interference is uncontrolled, which is not the case in the simulation. These errors can lead to a deviation in the expected results even when applying the same model used in the simulations.

5.5.3.1 PU's Activity Estimation Analysis

The scheme presented in Chapter (4) for estimating the PU's activity is now evaluated. The results presented in this section exhibit the confidence intervals for a confidence level of 95%.

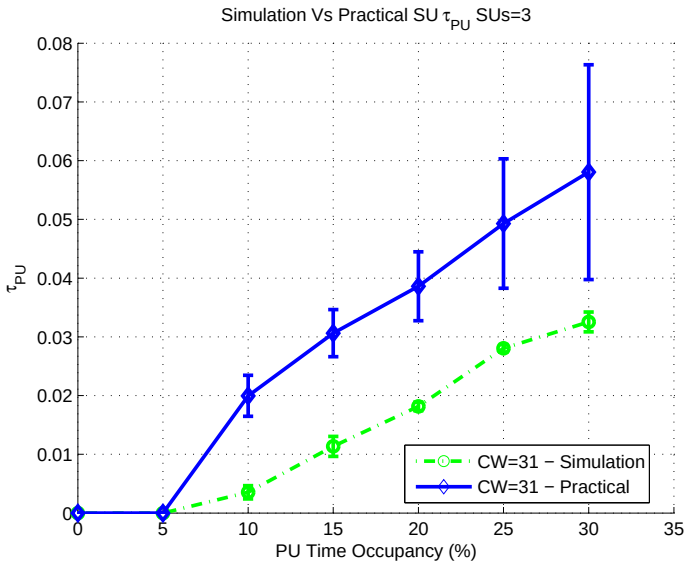


Figure 5.13: PU's access probability (τ_{PU}) with deviation, using $CW = 31$ for the SUs.

Figure 5.13 illustrates the comparison between the estimated access probability for the

PU through the SU nodes in both test scenarios. At first sight it is possible to conclude that the τ_{PU} values obtained in the simulation tests follow the same trend of the practical ones, although exhibiting a significant difference. A second observation is related with the relationship of figure's X and Y axis. Note that the percentage presented in X axis is the amount of time that the PU is occupying the channel. In fact the X-axis does not represents an access metric, but a time-occupancy metric. Y-axis represents the measured channel access of the PU by the SUs, which can not be compared with X-axis due to the fact the it is the probability of PU's access based on the sampling process performed by the SUs. This sampling process is completely constrained by the occupancy of the slots, which can not be compared with the amount of time a PUs uses the channel (X-axis).
As referred before, the practical results contain a bigger deviation than the results obtained through simulations and this fact can be due to physical layer errors that are only observed in the practical tests due to external interference.
The lack of knowledge from each node about the number of idle and busy slots during the backoff period will limit the estimation of the error probability, and as direct consequence, the estimation of the PU's activity will be affected (Chapter 4, Equation 4.22).

5.6 Conclusion

With these results and after a careful analysis it is possible to reach some valid facts.
After the results' analysis it is clear that the proposed model was correctly implemented. The comparison between the practical results and the results obtained through the computer simulations show that the results obtained in the practical test are very close to the results obtained in the simulations. Although, there are some discrepancies between the two test scenarios that can be easily justified. The practical experiments present a more random behavior due to the physical error probability. In the simulations, the results obey to a set of pre-established rules, and a null error probability was considered.
The error probability deviation can be found it the practical tests' results when the CW values are lower than 31 despite the PU's access probability used. This happens when the number of SU nodes increases and they become more aggressive with the decreasing of the CW for the backoff period. This will increase dramatically the number of frame collisions.

The estimation of τ_{PU} will be also affected by these factors. An increase in the number of physical errors will cause a deviation in the results for this estimation, although, since the simulations do not have such deviation, the estimation performed in the practical tests tends to be higher than the one obtained in the simulation tests.

Chapter 6

Conclusions

6.1 Final Considerations

This work proposed the creation of a contention based network to act as a secondary network formed by several SU nodes. This was the starting point for implementing a cognitive access system that constitutes the main achievement of this dissertation.

This dissertation starts presenting an overview about the actual cognitive radio systems and MAC protocols. Since this work focuses on the modification of the MAC layer of the IEEE 802.11 protocol in order to implement the cognitive system it was necessary to create a solid base for developing the proposed system. The cognitive system implementation was based on the Madwifi driver. This open source driver was the starting point to develop the cognitive network and the first step was being able to have a full control over the driver's functioning and internal parameters. After this step it was essential to validate the driver's performance and compare the obtained results with the theoretical results.

The lack of development and testing tools led to the need of developing new software packages for the Madwifi driver, such as the sensing module implemented for establishing a bridge between the user's space and the Madwifi driver by allowing real time changes of the driver's internal parameters and statistical readings. The proposed modifications jointly with the developed tools have supported the correct validation of the driver and most of its more important parameters, such as the CW values and protocol's behavior.

The next step consisted in developing the theoretical mechanism to estimate the PU's ac-

cess probability from the SU nodes. The proposed scheme for implementing the estimator proposed in Chapter 4 was developed and the last step of this work consisted in assessing the performance of the estimator through simulations and through a real implementation. This also led to the development of new testing tools for increase the versatility of the practical test framework.

The practical tests were conducted inside a controlled environment (Faraday Cage) and the results were analyzed and then compared with the results from the computer simulations. These results provided the needed information for estimating the PU's activity. Each SU node on the network produced an high amount of data for every test performed. The multiple tests performed for CW values and PUs' access probabilities (τ_{PU}) provided a large amount of data, such as the SU's transmission success probability (P_{Succ}). These results allied with the results obtained from the simulations provided the required statistical data for estimating the PU's access probability (τ_{PU}).

The results from both test scenarios were very similar when concerning the SU's transmission success probability (P_{Succ}). A minor deviation of the results was observed when the SUs' CW value was smaller than 31 due to the increasing number of physical errors. This was not so flagrant in the simulation test because the simulated nodes are experiencing a lower level of interference.

The estimation of the PU's medium access probability (τ_{PU}) presented an high deviation from the results obtained through simulation. This is mainly caused by the higher level of interference observed in the practical tests, which can not be simulated. As explained in Chapter 5, the validated scheme was admitting nonexistence of physical errors, namely due to the impossibility of estimating the probability of physical errors because the statistics related with the number of busy and idle slots described in Chapter 4 are not available in Madwifi driver.

This work provided a solid base for developing more mechanisms in the cognitive radio area and despite the amount of required work, the novelty factor introduced by this dissertation, constituted a strong motivation, since there is not any similar system or framework that could be used to developed cognitive systems using common wireless network devices.

6.2 Future Work

The implementation of the cognitive radio system described in this work, provided a strong test framework for developing and implementing new cognitive systems. This framework may be extended and suffer several improvements in the near future.

Testing new driver's parameterizations could increase the knowledge of the IEEE 802.11 network's behavior in order to implement new cognitive systems and also improve the test framework.

The PU's activity estimation could also be improved by gathering more statistical data in each SU node and increasing the number os SUs in the network. This surely would represent a leap forward in terms of efficiency for this mechanism. Improving the estimation mechanism is a good starting point for future works in this area, using the developed system and test framework.

One major step would be the automation of each SU in the network by making it able to adapt to the network conditions. An adaptive contention system could be the next step in the evolution of the current system by enabling each SU to adapt its CW according to the network's usage from the PUs.

These functionalities could be implemented in the near future and, after having a solid implementation, it would be interesting to test them in a real network radio environment with typical interference levels.

Bibliography

[Ars06] Tevfik Yiicek; Hiiseyin Arslan. Spectrum characterization for opportunistic cognitive radio systems. *Military Communications Conference*, pages 1–6, 2006.

[Cha05] Sai Shankar N; Carlos Cordeiro; Kiran Challapali. Spectrum agile radios: utilization and sensing architectures. *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium*, pages 160–169, 2005.

[Cha10] Anuj Saxena; Loveleen Chauhan. Novel approach for blind spectrum sensing in time and frequency domain. *Communication and Computational Intelligence (INCOCCI), International Conference*, pages 168–173, Dec 2010.

[Che08] Shao-Yu Lien; Chih-Cheng Tseng; Kwang-Cheng Chen. Carrier sensing based multiple access protocols for cognitive radio networks. *ICC 2008 proceedings*, 1(1):3208–3214, 2008.

[DaS13] Ryan E. Irwin; Allen B. MacKenzie; Luiz A. DaSilva. Resource-minimized channel assignment for multi-transceiver cognitive radio networks. *Selected Areas in Communications, IEEE Journal*, pages 442–450, 2013.

[Elt11] Amir Hossein Gholamipour; Ali Gorcin; Hasari Celebi; B. Ugur Toreyin; Mazen A. R. Saghir; Fadi Kurdahi; Ahmed Eltawil. Reconfigurable filter implementation of a matched-filter based spectrum sensor for cognitive radio systems. *Circuits and Systems (ISCAS), 2011 IEEE International Symposium*, pages 2457–2460, 2011.

[Gha10] Maria Iqbal; Abdul Ghafoor. Affordable cyclostationarity-based spectrum sensing for cognitive radio with smart antennas. *IEEE Transactions on Vehicular Technology*, 59:1877–1886, 2010.

[Gha12] Maria Iqbal; Abdul Ghafoor. Analysis of multiband joint detection framework for waveform-based sensing in cognitive radios. *Vehicular Technology Conference (VTC Fall), 2012 IEEE*, pages 1–5, 2012.

[Gra08] Live Graph. Live graph tool. Software Package retrieved from http://www.live-graph.org/, 2008.

[Hao09] You-Xi Tang; Jin-Xiang Xia; Zhang-Hong Hao. A distributed cooperative spectrum sensing based on network code in cognitive radios. *Apperceiving Computing and Intelligence Analysis. ICACIA. International Conference*, Oct 2009.

[Hay10] Wael Guibene; Aawatif Hayar. Joint time-frequency spectrum sensing for cognitive radio. *Applied Sciences in Biomedical and Communication Technologies (ISABEL), 3rd International Symposium*, Nov 2010.

[IEE07] IEEE. Ieee standard for information technology - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications, 2007.

[Kuo07] Alex Chia-Chun; David S. L. Wei; C.-C. Jay Kuo. A cognitive mac protocol using statistical channel allocation for wireless ad-hoc networks. pages 105–110. Wireless Communications and Networking Conference. WCNC. IEEE, March 2007.

[Lo10] Chi-Yuan Chen; Yao-Hsin Chou; Han-Chieh Chao; Chi-Hsiang Lo. Toward secure centralized spectrum sensing by utilizing geographical information. *Future Information Technology (FutureTech), 5th International Conference*, May 2010.

[Lu11] Jonathan S. Lu; I-Tai Lu. Propagation based spectrum sensing: A novel beamforming detection method. pages 11–15, 2011.

[Lui13] Antonio; Oliveira Rodolfo; Dinis-Rui; Bernardo Luis Luis, Miguel; Furtado. Towards a realistic primary users' behavior in single transceiver cognitive networks. *Communications Letters, IEEE*, pages 309–312, 2013.

[Mar10] Alexandru Martian; Ioana Marcu; Ion Marghescu. Spectrum occupancy in an urban environment: A cognitive radio approach. pages 25–29, 2010.

[Mis12] Jelena Misic. Cooperative sensing at the mac level in simple cognitive personal area networks. *Selected Areas in Communications, IEEE Journal*, 30:1711–1718, 2012.

[Pro12] The Madwifi Project. The madwifi project. Software Package retrieved from http://madwifi-project.org/, 2012.

[Qiu10] Fang Chen; Runhe Qiu. Centralized and distributed spectrum sensing system models performance analysis based on three users. *Selected Areas in Communications, IEEE Journal*, pages 1–4, 2010.

[Str13] Fossies Dox Madwifi Structure. Fossies dox - madwifi structure. http://fossies.org/dox/madwifi-0.9.4-current/, 2013.

[Var01] J.; Clemo-Gary R.; Haines-Russell J. Vardoulias, Georgios A.; Faroughi-Esfahani. Blind radio access technology discovery and monitoring for software defined radio communication systems: problems and techniques. *3G Mobile Communication Technologies, 2001. Second International Conference*, pages 306–310, 2001.

[Won11] Qian Chen; Ying-Chang Lian; Mehul Motani; Wai-Choong Wong. A two-level mac protocol strategy for opportunistic spectrum access in cognitive radio networks. pages 2164–2179. Vehicular Technology, IEEE Transactions on (Volume: 60 , Issue: 5), Jun 2011.

[Yuc09] H. Yucek, T.; Arslan. A survey of spectrum sensing algorithms for cognitive radio applications. *IEEE Communications Surveys & Tutorials*, 11(1):116–130, 2009.

Appendix

Appendix A

Specific Driver's Characteristics

A.1 Madwifi Frame Timings and Chosen Parameters

Frame Timings for Madwifi 0.9.4 Test Environment

The MAC (Medium Access Control) layer used was the CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) which has frame sequence and time delay described by the image below. Due to the usage of only 802.11g Stations and Aps a short slot time was used.

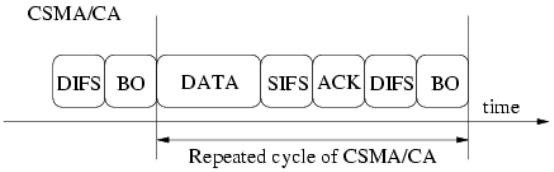


Image 1 - CSMA/CA Frame Sequence and Time Delay

ACK: 112 bits or 14bytes @ 2 Mbps
ACK (µsec): 152 µsec
SIFS (µsec): 10 µsec
DIFS (µsec): 28 µsec
BO (Backoff): Determined by the **CW** (Contention Window).

The High Rate direct sequence spread spectrum (HR/DSSS) system with CCK (Complementary code keying modulation scheme was used in this test environment. A short PPDU (PLCP (Physical Layer Convergence Procedure protocol) data unit) was set by default for this specifications, which translate the timings described after the 802.11 frame structure, illustrated by image 2.

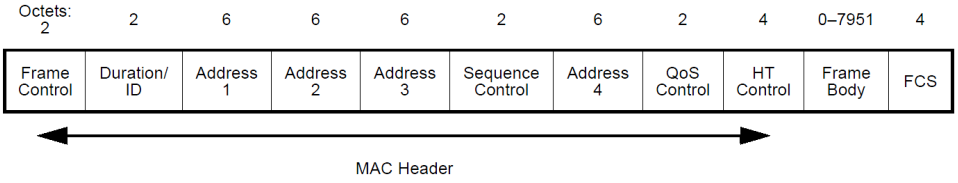


Image 2 – Standard Data frame structure PSDU (Physical Layer Service Data Unit).

FCS, QoS control and HT Control are disabled, so every data frame will have a MAC Header of 30 bytes + frame body.

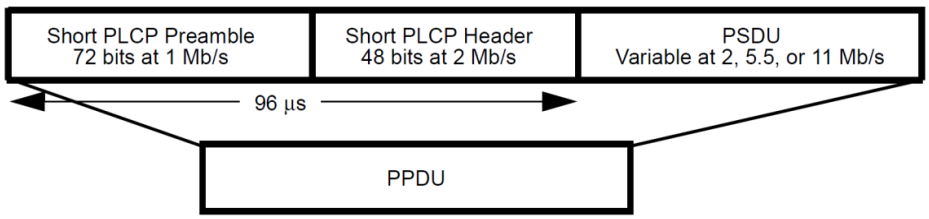


Image 3 - 802.11 Standard Frame Structure with Short PLCP.

The 802.11 frame will have the size off the PSDU (Physical Layer Service Data Unit) and the short PLCP combined (Short PLCP + PSDU (MAC Header + Frame Body)).

Slot (µsec): 9 µsec
PLCP Preamble: 72bits or 9 bytes @ 1Mbps
PLCP Header: 48bits or 6 bytes @ 2Mbps
Short PPDU/ (PLCP Preamble + PLCP Header): 15 bytes
Short PPDU (µsec): 96 µsec
Mac Header: 240 bits or 30 bytes @ 11 Mbps
FCS: disabled
IP: 160 bits or 20 bytes @ 11 Mbps
UDP: 64 bits or 8 bytes @ 11 Mbps
Payload (Data): 0 – 2312 bytes @ 11 Mbps

Expected Bitrate for data transmission (Mbps):

$$\frac{Payload * 8}{\frac{PLCP\ Preamble}{1e^6} + \frac{PLCP\ Header}{2e^6} + \left(\frac{CW}{2} * Slot\right) + \left(\frac{MAC\ Header + IP + UDP + Payload}{11e^6} * 8\right) + SIFS + DIFS + ACK} \quad (1)$$

A.2 Madwifi Source Code

Digital support files: "/Appendices/A/A2/Madwifi/"

This folder contains all the source code of the Madwifi driver with the implemented modifications developed in this dissertation.

A.2.1 Installation and Configuration Bash Scripts

Digital support file: "/Appendices/A/A2/A2_1/madwifi_install.sh"

Digital support file: "/Appendices/A/A2/A2_1/setup_card_parameters.sh"

A.2.2 Sensing Module

Digital support files: "/Appendices/A/A2/Madwifi/sensing/"

This folder contains the developed sensing module for Madwifi developed in this work.

Appendix B

Validation Tools and Scripts

B.1 Contention Window Behavior - Matlab® Code

Digital support file: "/Appendices/B/B1/CW_behavior.m"

The referred script file produces a graphical output of the Madwifi's contention windows behavior.

B.2 *athstats* output analyzer - Matlab® code

B.2.1 *athstats* analyzer: simulation duration - Matlab® code

Digital support file: "/Appendices/B/B2/B2_1/getTimeSim.m"

This function calculates the time duration between two given times.

B.2.2 *athstats* analyzer: expected bit rate - Matlab® code

Digital support file: "/Appendices/B/B2/B2_2/expected_bitrate.m"

This function calculates the expected throughput bit rate for specific CW and data payload values.

B.2.3 *athstats* analyzer: noisy environment test - Matlab® code

Digital support file: "/Appendices/B/B2/B2_3/validacao_parametros_noisy.m"

This script analyzes the gathered data from the Madwifi driver in a noisy environment.

B.2.3.1 Noisy Environment Trace Files

Digital support files: "/Appendices/B/B2/B2_3/TestesAth/Noisy/*"

Trace files generated during the practical tests in a noisy environment.

B.2.3.2 Noisy Environment Test Results

Digital support files: "/Appendices/B/B2/B2_3/Resultados/Noisy/*"

Results obtained after the trace files analysis.

B.2.4 *athstats* analyzer: faraday cage test - Matlab® code

Digital support file: "/Appendices/B/B2/B2_4/validacao_parametros_faraday.m"

This script analyzes the gathered data from the Madwifi driver in a controlled environment (Faraday Cage).

B.2.4.1 Controlled Environment Trace Files

Digital support files: "/Appendices/B/B2/B2_4/TestesAth/Faraday/*"

Trace files generated during the practical tests in a controlled environment.

B.2.4.2 Controlled Environment Test Results

Digital support files: "/Appendices/B/B2/B2_4/Resultados/Faraday/*"

Results obtained after the trace files analysis.

B.3 Proc File Real Time Analyzer - Matlab® Code

Digital support files: "/Appendices/B/B3/Realtime_stats/"

Source code of the Matlab real time proc file analyzer.

B.4 Test Applications - Java® Code

Applications developed for the first validation tests.

B.4.1 Data Transmission Application

Digital support files: "/Appendices/B/B4/B4_1/Cliente_UDP/"

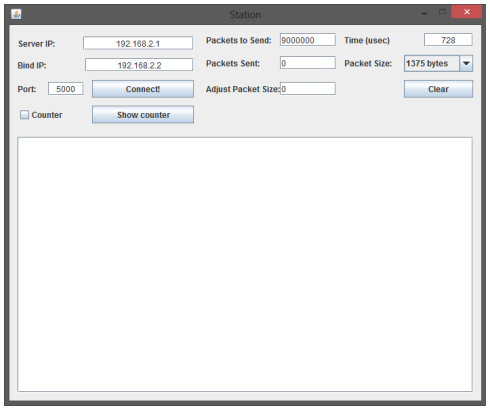


Figure B.1: Client application - User interface.

B.4.2 Data Reception Application

Digital support files: "/Appendices/B/B4/B4_2/Server_UDP/"

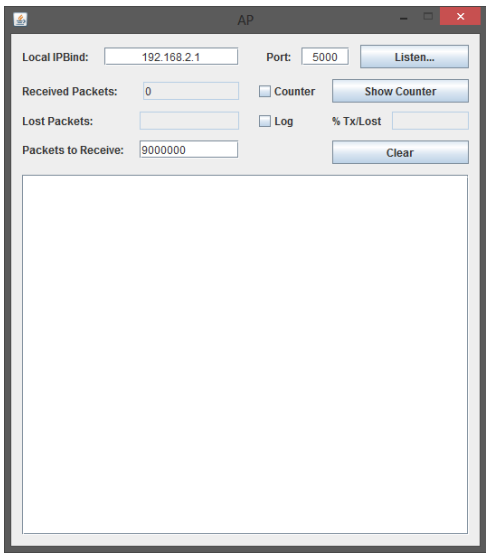


Figure B.2: Server application - User interface.

Appendix C

Test Applications and Scripts

C.1 Test Applications - Java® Code

Applications developed for the test framework.

C.1.1 SU Application

Digital support files: "/Appendices/C/C1_1/SU/"

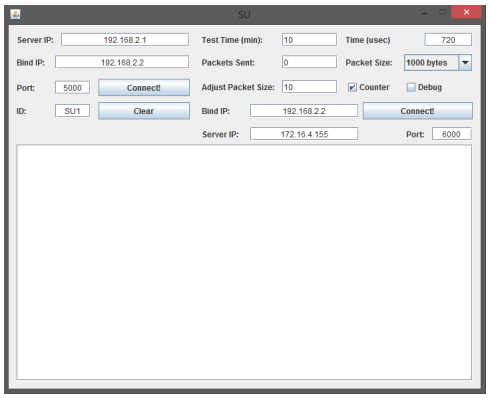


Figure C.1: SU application - User interface.

C.1.2 PU Application

Digital support files: "/Appendices/C/C1_2/PU/"

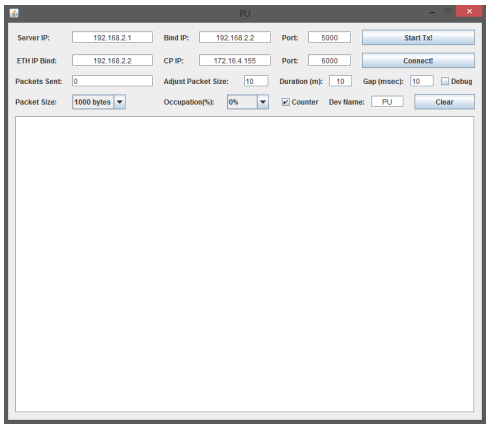


Figure C.2: PU application - User interface.

C.1.3 AP Application

Digital support files: "/Appendices/C/C1_3/AP/"

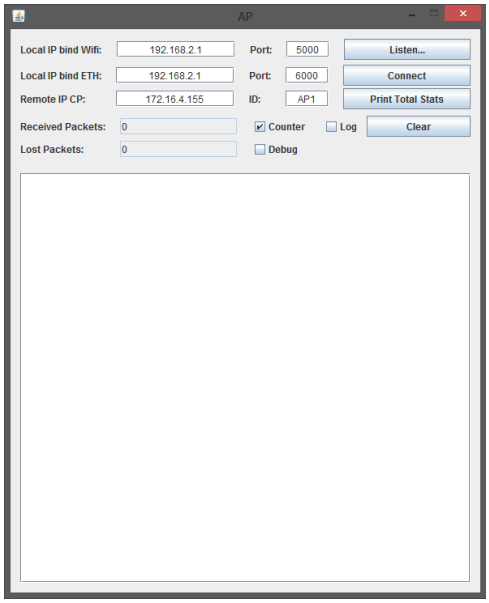


Figure C.3: AU application - User's interface.

C.1.4 Proc Reader/Writer Application

Digital support files: "/Appendices/C/C1_4/ProcReader/"

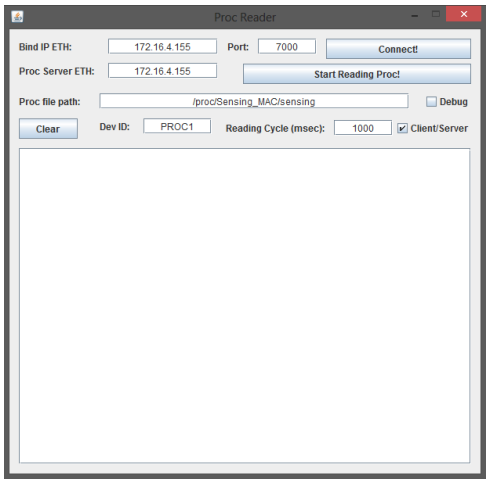


Figure C.4: Proc Reader/Writer application - User interface.

C.1.5 Control Panel Application

Digital support files: "/Appendices/C/C1_5/ControlPanel/"

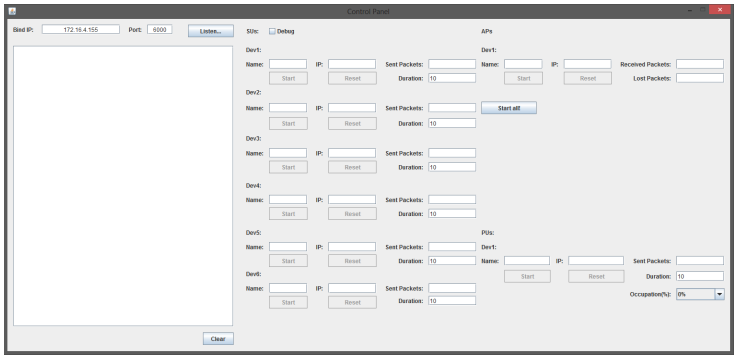


Figure C.5: Control panel application - Transmission control interface.

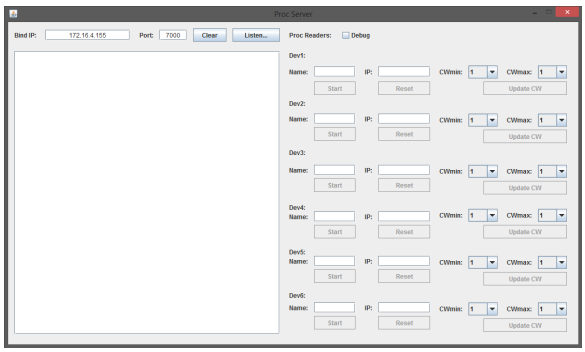


Figure C.6: Control panel application - Proc applications control interface.

C.2 *athstats* output analysis - scripts, trace files and results

C.2.1 Simulation Results Analysis

C.2.1.1 Matlab® Scripts Code

Digital support file: "/Appendices/C/C2_1/pu_simul.m"

This script will create a storage file with all the important data gathered from the simulation trace files.

Digital support file: "/Appendices/C/C2_1/Tstudent.m"

Digital support file: "/Appendices/C/C2_1/pu_results.m" This file will then use the gathered data to produce all the output graphics. The the student's t-distribution will be used for analyzing the test results deviation.

C.2.1.2 Trace Files

Digital support files: "/Appendices/C/C2_1/loggerSU/*.m"

Digital support files: "/Appendices/C/C2_1/loggerPU/*.m"

Due to the large amount of trace files generated (31.6GB), these folders contain a few trace files to serve as example.

C.2.1.3 Analysis Results

Digital support files: "/Appendices/C/C2_1/Resultados/"

Each script file (C.2.2.1) will produce a set of results that will be stored in the referred folder.

C.2.2 Practical Tests Results Analysis

C.2.2.1 Matlab® Scripts Code

Digital support file: "/Appendices/C/C2_2/pu_0_faraday.m"

Digital support file: "/Appendices/C/C2_2/pu_1_faraday.m"

Digital support file: "/Appendices/C/C2_2/pu_2_faraday.m"

Digital support file: "/Appendices/C/C2_2/pu_3_faraday.m"

Digital support file: "/Appendices/C/C2_2/pu_4_faraday.m"

These scripts analyze the trace files data created during the practical tests. Each one analyzes a different test scenario from 0 up to 4 SU nodes.

C.2.2.2 Trace Files

Digital support files: "/Appendices/C/C2_2/TestesAth/"

Due to the large amount of trace files, this folder contains almost all trace files generated during the practical tests. They are separated in folders depending on the number of SUs and CW values used.

C.2.2.3 Analysis Results

Digital support files: "/Appendices/C/C2_2/Resultados/"

Each script file (C.2.2.1) will produce a set of results that will be stored in the referred folder.