Analysis and Implementation of Reinforcement Learning on a GNU Radio Cognitive Radio Platform

by

Yu Ren

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Master of Science
in Electronic and Computer Systems.

Victoria University of Wellington
2010

Abstract

Spectrum today is regulated based on fixed licensees. In the past radio operators have been allocated a frequency band for exclusive use. This has become problem for new users and the modern explosion in wireless services that, having arrived late find there is a scarcity in the remaining available spectrum.

Cognitive radio (CR) presents a solution. CRs combine intelligence, spectrum sensing and software reconfigurable radio capabilities. This allows them to opportunistically transmit among several licensed bands for seamless communications, switching to another channel when a licensee is sensed in the original band without causing interference. Enabling this is an intelligent dynamic channel selection strategy capable of finding the best quality channel to transmit on that suffers from the least licensee interruption.

This thesis evaluates a Q-learning channel selection scheme using an experimental approach. A cognitive radio deploying the scheme is implemented on GNU Radio and its performance is measured among channels with different utilizations in terms of its packet transmission success rate, goodput and interference caused. We derive similar analytical expressions in the general case of large-scale networks.

Our results show that using the Q-learning scheme for channel selection significantly improves the goodput and packet transmission success rate of the system.

Acknowledgments

I would like to thank my supervisors Dr. Peter Komisarczuk and Dr. Pawel A. Dmochowski for their feedback, patience and support in helping me produce this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Cognitive Radio

The area of radio technology has seen a recent rapid growth in wireless service users. From being non-existent just two decades ago, services such as GPS location based tracking, mobile telephony and broadband are today part of everyday consumers' lives. As advances occur and the cost of existing technology decreases, the introduction of improved services and now feasible services, such as wireless sensors, is also occurring.

For New Zealand 2009 was a watershed year, seeing the introduction of the HDTV (High Definition Television) capable Freeview digital television service alongside analogue broadcasting and Telecom's XT Mobile Network 3G mobile telephone network.

The result has been a proliferation in wireless services. The spectrum these applications operate in is a finite resource and there are issues of scarcity. The traditional static spectrum allocation policy has been for a regulation authority (the Ministry of Economic Development (MED), Radio Spectrum Group (RSM) in New Zealand) to grant a service sole usage rights to a spectrum band, which contributes to inefficient spectrum use. Licensed service users or *primary users* (PUs) may be active only some of the time or not even present in certain areas. A report from the NRNT

Project reveals that the average spectrum usage rate of bands under 3GHz is only 5.2% [53]. When the band is unoccupied, or usage of it will not interfere unduly with licensed user communications, we say there is channel *whitespace*.

Cognitive Radio (CR) is an emerging class of reconfigurable, spectrum aware, intelligent radios, typically associated with being implemented on *software defined radio* (SDR) which gives the capability to flexibly modify physical transmission parameters in software. The technology has been proposed to utilize whitespace for communications as part of a *Dynamic Spectrum Access Network* (DSAN). Unlicensed CR devices, or *secondary users* (SUs), are permitted to opportunistically use spectrum allocated to licensed PUs when it is temporarily unoccupied. The use of whitespace is conditional on the PU not being interfered with. CRs can detect where there is whitespace and sense and adapt to the presence of a PU, by switching frequencies or modifying its transmission waveform, to ensure non-interference, for a net gain in overall spectrum utilization. A CR in this role is also known as a *spectrum agile radio*.

1.2 Thesis Scope and Contributions

The CR concept was first introduced by Joseph Mitola [57] in 1999 and has many open research issues. One is *spectrum management*, which deals with the problem of learning to exploit the high-quality whitespace by the radio. Existing work in the literature takes a diverse approach, including predicting when a channel is free [41] [93] of PU activity to having SUs play a game to decide on the channel to use [60]. A survey of the literature however shows that most research is being done in simulators and emphasize improvements in throughput, neglecting to mention PU interference.

Previous work at the Victoria University of Wellington (VUW) has focused on intelligent opportunistic spectrum access techniques and design

of upper layer DSAN protocols [94] [96] [97], in particular the development of a novel ad-hoc Dynamic Channel Selection (DCS) scheme using Q-learning Reinforcement Learning (RL) to maximize the packet transmission success of the SU. The scheme has been found to give promising results in simulation [96] [97] but has yet to be trialled on a real system.

The thesis seeks to answer the research question, *"what performance does the Q-learning channel selection scheme achieve in a real deployment scenario?"* This is achieved by implementing the scheme on GNU Radio SDRs that are then organized into a wireless network experiment to find out the practical performance. GNU Radio is a free open source software framework for developing SDR. It is designed to operate with the Universal Software Radio Peripheral (USRP), which lets a host PC running GNU Radio create and process radio signals, with the actual RF transmission and reception handled by the USRP. As is the nature with using real systems, equipment and time limit the scale of the experimental scenario. An analytical model is derived to generalize performance to the realistic large-scale network case.

The contributions of this thesis are firstly the performance analysis of the Q-learning channel selection scheme in a real wireless network experiment. Second the development of a SU cognitive system in GNU Radio. A flexible physical (PHY) layer, including spectrum sensing for detecting PU activity, and an ad-hoc cognitive medium access control (MAC) layer capable of managing opportunistic access between SUs need to be implemented to support the DCS. These layers' services have so far been taken for granted in previous work. For instance, perfect sensing and PU avoidance is commonly assumed. This is not physically possible if a transceiver cannot detect other users when it itself is transmitting. The experience of using GNU Radio adds a third contribution. This thesis is hoped to assist future students as a familiarization guide to GNU Radio and USRP.

1.3 Thesis Structure

This thesis is structured as follows. Chapter 2 contains background material on CR. The description of our CR implementation on GNU Radio is divided among Chapters 3 and 4. In Chapter 3, the design of the physical and MAC layers to support spectrum agility are discussed. The MAC protocol mechanisms to co-ordinate access to different channels are described and we explain the design decisions leading to the choice of spectrum sensing technique. The Q-learning channel scheme and its implementation in GNU Radio is described in Chapter 4, along with the motivation for using this form of learning. Learning background material is also discussed in this section. The test scenario is introduced in Chapter 5. The performance metrics to be measured by the experiment and logging and co-ordination tasks are detailed. A model of the scenario is derived in Chapter 6 using Markov chain analysis that predicts the goodput and packet transmission success rate of the SU using the scheme and the level of PU interference in large-scale networks. The results of the wireless experiment are the subject of Chapter 7. The performance of the scheme is characterized and compared against what can be ideally expected and analytical predictions. Chapter 8 sums up the key results of the research and future work.

Chapter 2

Background

This chapter examines cognitive radio background material. CR is first introduced in Sections 2.1-2.2 as the technology behind Dynamic Spectrum Access Networks, a proposed solution to the spectrum allocation problem and spectrum scarcity. CR is then defined in Section 2.3 in terms of possessing three key properties, adaptability (SDR), sensing and intelligence, which are then separately discussed.

2.1 The Spectrum Allocation Problem

The wireless spectrum defines the span of electromagnetic frequencies that can be used to transmit and receive signals for communications. Traditional spectrum assignment practise has been for regulation authorities, such as the US Federal Communications Commission (FCC) and here in New Zealand the Ministry of Economic Development (MED), to allocate fixed spectrum bands for exclusive use to communications services. However with more wireless services being introduced, the remaining spectrum left to be allocated has been found to be limited. Referring to the MED spectrum allocation chart [7] virtually all RF frequencies from 9kHz-40GHz are already reserved. The regulatory bodies have tried to solve this problem by converting some frequencies to dual use and opening up

industrial, scientific and medical (ISM) bands, originally reserved for purposes other than communication, for use, but these too are experiencing overcrowding. For example Wi-Fi and Bluetooth both share the 2.4GHz ISM band, with resulting interference.

At the same time this licensing model is inefficient. To use as an example, a mobile telephone network will not have active calls all the time. Yet because this service has exclusive rights to the band use is denied when it is inactive, effectively wasting spectrum. According to a report by the FCC the utilization of allocated spectrum ranges from 15% to 85% [14].

2.2 Dynamic Spectrum Access Networks

New radio technologies in the form of software defined radio and CR promise a more efficient network paradigm, in the form of Dynamic Spectrum Access Networks (DSANs) [16], for allocating the spectrum. DSANs contain CR users not holding a license to transmit on a portion of the spectrum, referred to as Secondary Users (SUs) in the literature. For communications SUs opportunistically exploit unused *whitespace* in the spectrum, defined as spectrum bands that are temporarily unoccupied by licensed users or where its use will not interfere with licensed user activities. SUs may transmit on a *channel* medium that is part of the whitespace. Interference to licensed users, referred to as Primary Users (PUs), must be avoided by having the SU stop transmitting on the channel when it is in use by a PU. If PUs can tolerate some interference, the SU can adjust its transmission waveform, for instance the power level, to fall within these bounds rather than abandoning the channel. What constitutes an acceptable level of interference is set by regulatory authorities as part of the SU's *policy rules*, defining what form of secondary access is legal. In DSANs, licensed user idle times do not represent wastage of the spectrum as it may be reused.

The extra radio functionality required to implement a DSAN is sum-

marized by [16]. Firstly, *spectrum sensing* is needed by SUs to detect and classify whitespace and the presence of PUs. A decision must then be made as to what channel will be used based on the user's service requirements, the currently available whitespace, its quality, such as whether the whitespace is consistent and the channel quality, for example in terms of noise and waveform throughput, a task which is known as *spectrum management*. Should a PU be detected, the user is compelled to leave the channel to prevent interference, a function termed *spectrum mobility*. Like any other network a MAC-like protocol is needed to co-ordinate access among SUs, *spectrum sharing*, particularly to establish channel rendezvous in the selected medium. The design of these features are a focus in the research community. These primarilly physical and link (MAC) layers functions have a significant bearing on network application performance at higher levels. For instance, the spectrum management decision to switch to a better channel needs to be balanced against the brief communications interruption it will cause, termed *spectrum handoff*. For this reason researchers are considering a cross-layer approach to DSAN architecture design, particularly to supply a certain Quality of Service (QoS) [23] [94].

Work on the implementation of DSANs is being carried out by the NeXt Generation (xG) program and development of the IEEE 802.22 standard. The DARPA xG program aims to produce a radio system providing seamless unlicensed user communications across a diverse frequency range. Multiagent field trials of working prototypes were successfully conducted in August 2006 [54]. IEEE 802.22 is an in-progress Wireless Regional Area Network (WRAN) standard using opportunistic access of TV channels to communicate [26].

2.3 Cognitive Radio

Cognitive radio is an emerging class of reconfigurable, spectrum aware, intelligent radios. CRs as SUs are the enabling technology for DSANs,

having the required capability to autonomously adjust communications to where there is whitespace. The concept of CR originates from a 1999 paper by Joseph Mitola III and Gerald Q. Maguire, Jr. [57]. This paper envisaged software radios that applied advanced reasoning on its current situation and history to best satisfy the needs of the user. It would "search out ways to deliver the services the user wants even if the user does not know how to obtain them." [57] The "Mitola" CR is an ideal device that can perfectly meet any user need, even inventing new protocols to advance its goals.

Current CR definitions emphasize the SU role, although in this regard the term *spectrum agile radio* [16] is more appropriate. Yucek and Arslan [98] define CR as "an SDR that is aware of its environment, internal state, and location, and autonomously adjusts its operations to achieve designated objectives." The FCC classification [15] is a radio or system that senses its operational electromagnetic environment and can dynamically and autonomously adjust its radio operating parameters to modify system operation, such as to maximize throughput, mitigate interference, facilitate interoperability, access secondary markets."

These definitions identify three critical CR capabilities: awareness, reconfigurability and intelligence. Rondeau [71] uses the analogy of the knobs and meters on a regular stereo with the reconfigurable knobs standing in for radio parameters, such as transmission power and modulation, and for meters sensed metrics such as bit error rate or quality of service. A CR autonomously turns the reconfigurable knobs to achieve the meters, which it senses. The essential awareness task is *spectrum sensing* to identify spectrum whitespace. CRs are implemented on SDRs that provide reconfigurability, since transceiver functionality such as waveform modulation is shifted to be modifiable in software. The *cognitive engine* [71] describes the intelligent decision making process of autonomously developing a policy of actions to meet desired performance.

All CRs operate by implementing a form of the *cognition cycle*, shown in Figure 2.1, formulated in Mitola's original paper. This is described in

SU terms of intelligently deciding on the best whitespace channel to use. In the *observe* stage of the cycle sensing describes the spectrum. The environment description, past learning and a record of the radio's history is fed into *orient*. The *orient* stage determines the priority the radio needs to act, based on the information of what has occurred in the spectrum. Immediate priority information provokes an instant change in radio behavior (direct transition to *act*). This corresponds to when a policy rule is triggered, for instance if a PU is detected in the current channel communications must be halted as the action. Otherwise the cycle moves to *plan* or *decide*. As no immediate action is required, the information can be considered for ways to improve performance over the current course of action, such as switching to another channel if the quality has degraded. In *plan* alternative behaviors are generated, one of which is selected by *decide* to implement as seen to best meet the radio's goals. Evaluation of previous learned behavior is critical for the planning stage. The *learn* stage stores the performance outcome of past actions and the environmental context. If a similar state is occurring the engine can choose the same action that led to a good outcome previously.

The rest of this chapter discusses the literature on SU awareness and reconfigurability, as relevant to this thesis' goal of building a CR to physically test DSAN techniques. SDR architecture is considered first in Section 2.4, followed in Section 2.5 by a survey of spectrum sensing techniques. Background material on intelligence is deferred till Chapter 4 where it serves to introduce the Q-learning dynamic channel selection scheme.

2.4 Software Defined Radio

CR is implemented on software defined radio (SDR), known interchangeably as software radio. The term software radio was first coined by Joseph Mitola III in 1992 as a "class of reprogrammable or reconfigurable radios." [56] In an SDR radio components, traditionally implemented in

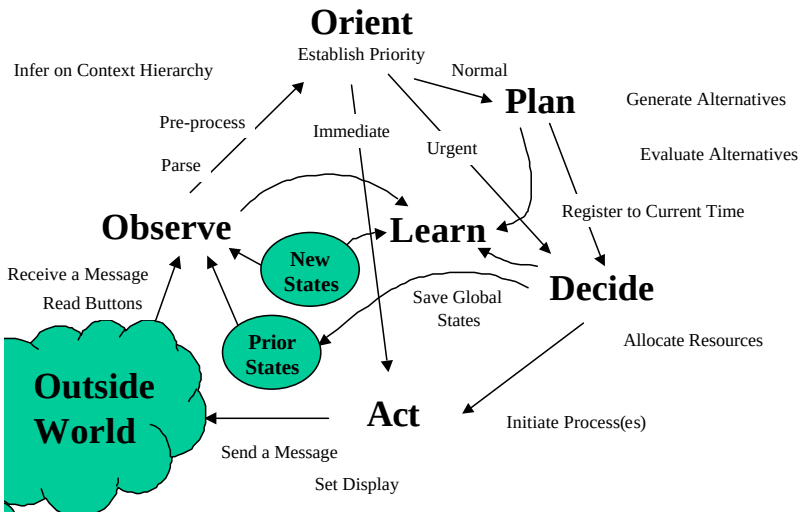

Figure 2.1: The Cognition Cycle, from [57]

fixed hardware, are moved into the software domain. The result is that protocols and waveforms are completely definable on-the-fly, which in a CR is exploited by the cognitive engine. *Software* in this context denotes algorithms, downloadable to a device, which can be changed during radio operations.

SDRs are an evolution of digital radio designs. In a digital radio receiver, pictured in Figure 2.2 compared to an analogue receiver, fixed digital hardware performs signal processing. The digital systems are implemented in nonprogrammable Application-Specific Integrated Circuit (ASIC) hardware or fixed processing architecture Digital Signal Processors (DSPs) designed to perform one operation quickly. The antenna signal is analogue and is digitized by an Analogue to Digital Converter (ADC). The ADC is not fast enough to directly sample the antenna signal at its radio frequency (RF), for example a 101.3MHz FM radio signal. An analogue RF frontend translates the RF frequencies to a lower intermediate frequency (IF) range within the Nyquist frequency range of the ADC. The remaining signal processing is executed in fixed digital hardware. The point at

which the waveform is digitized is known as the *digital access point* [55]. The portion of the transceiver chain where the signal exists in analogue format is termed the *hardware-defined subsystem* and, similarly, the portion where the signal is in digital format is referred to as the *software-defined subsystem* [85]. The receiver output is a bitstream of the signal centred at baseband (0Hz).

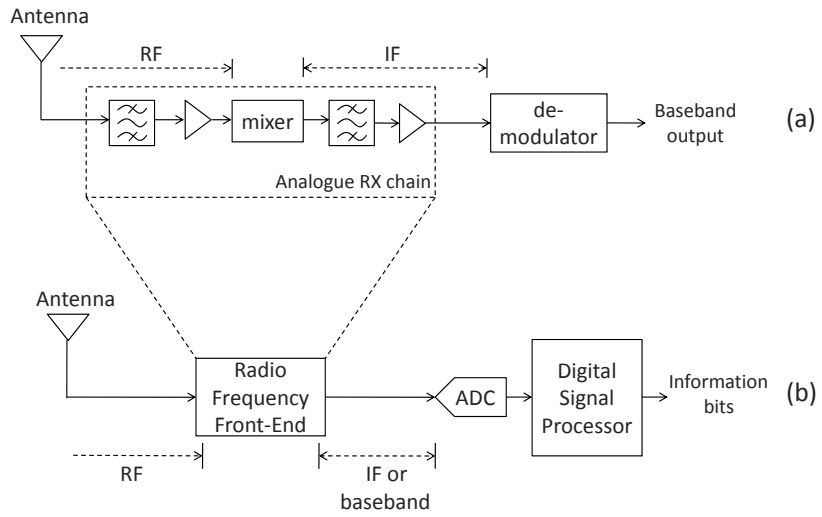

Figure 2.2: Analogue (a) and Digital (b) Hardware Receiver Chains, from [85]

A typical RF frontend structure is shown in Figure 2.3. The low pass filter cuts off frequencies above the RF signal of interest and the low noise amplifier amplifies the signal. The local oscillator is used to generate a sinusoidal wave with frequency RF-IF, which when multiplied with the signal in the mixer reproduces the signal now centred at IF and 2*RF-IF. The second low pass filter removes the higher frequency component, giving the translated signal at IF.

SDR is a digital radio where the digital hardware is reconfigurable in software. This is achieved by using Field Programmable Gate Array (FPGA), DSP and, slowest, power-hungry but most flexible, General Pur-

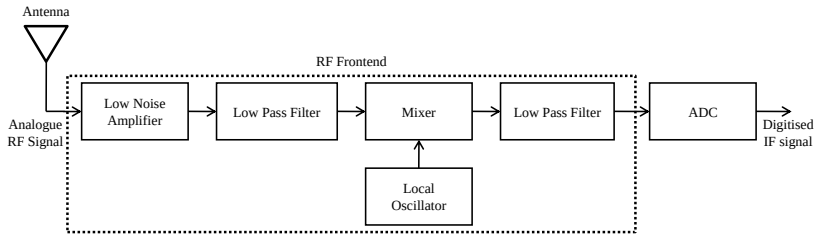

Figure 2.3: RF Receive Front End, based off a description in [1]

pose Processor (GPP) technology (i.e. PCs). The difference is illustrated in Figure 2.4, showing a block diagram of the SDR architecture. FPGAs consist of an array of Configurable Logic Blocks (CLBs), logic gates that are connected together. The interconnections are determined by RAM or Flash switches which can be toggled on or off by writing a bit to them. Digital radio subsystem algorithms can be mapped onto a direct logic gate implementation in the FPGA. The procedure requires a design description of the function written in a Hardware Description Language (HDL), e.g. Verilog or VHDL. The file is compiled into a bitstream that configures the switches to match the function logic. In DSPs and GPPs digital systems are implemented as software programs running on top of fixed hardware architectures, thus speed is inherently orders of magnitude slower than a FPGA hardware implementation. FPGAs represent a fast reconfigurable radio implementation platform. However, gate and interconnection constraints due to a finite number of gates limit the functionality that can be loaded onto the FPGA at one time.

SDRs potentially allow the user to specify in software any waveform to transmit or receive on the fly. Current applications are limited by its greater cost and the slow speed of signal processing in software ($\sim\mu$s) when compared to hardware ($\sim$ns). For example, Gilmore [37] attempted to implement an SDR HDTV receiver with a PC set up to perform frame processing. Unfortunately, as it took 40 seconds of processing for 1 second of data, realtime reception was not attainable.

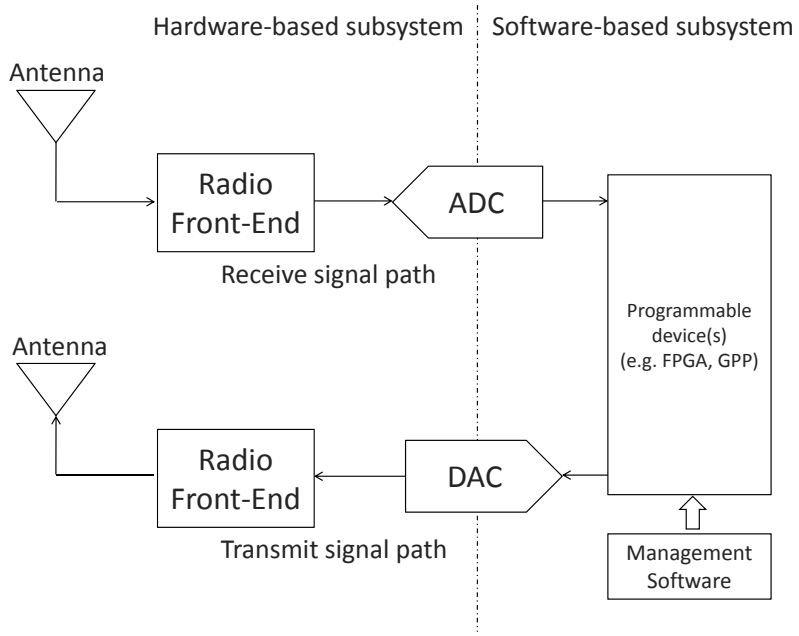

Figure 2.4: Block Diagram of a Typical SDR, from [85]

2.5 Spectrum Sensing

The role of spectrum sensing is to identify in which portions of the spectrum a PU is active and cannot be used (*spectrum occupancy*), and vice versa where an absence of PU activity (*spectrum hole*) or whitespace exists. Spectrum sensing can also establish the quality of the whitespace for communications.

Whitespace exists across multiple dimensions. For example, if a frequency is in use by a PU that is communicating with a certain code, the SU can adopt an orthogonal code and transmit without causing interference. Consequently to allow the CR to fully exploit the whitespace, spectrum sensing should "involve obtaining the spectrum usage characteristics across multiple dimensions such as time, space, frequency and code" [98]. Ideally, sensing should be able to classify the source of the signal. This is necessary to defend against a PU emulation (PUE) attack [22],

where a selfish user mimics a PU to gain a channel advantage over other SUs.

If PU receivers in the area are far enough from the SU not to be affected unduly by its transmissions, their location (*space*) should be confirmed by spectrum sensing so the band can be used. This idea is expressed in the concept of *interference temperature* [13], denoting the power at a receiver generated by other users and noise sources per unit bandwidth with units Kelvin. Regulatory bodies may define an interference temperature limit that unlicensed users in the area can use to transmit, as long as the temperature at licensed receivers does not exceed this value [24].

Different sensing architectures and techniques have been proposed, with those providing greater sensing capabilities, like signal classification, doing so at the expense of complexity.

2.5.1 Spectrum Sensing Architectures

A distinction is made between organizational and hardware architectures for spectrum sensing. Single-radio and dual-radio hardware architectures exist for performing sensing [76]. Radio communications and sensing are handled by the same device in single-radio architecture, with a time slot reserved in between transmission for sensing. A separate radio is dedicated for sensing in the dual-radio architecture so the two activities can proceed simultaneously without lost communications efficiency. The disadvantage is extra hardware. Sensing requires high-speed high dynamic range Analog-to-digital converters (ADCs) capable of observing a large Nyquist spectrum bandwidth and high-speed signal processors.

In non-cooperative organizational architectures, each radio samples the spectrum, processes the sensing results and independently arrives at a decision as to whether the channel is in use or is free. Compare this to cooperative sensing where spectrum sensing information is shared between radio agents in the network and occupancy determined from the collec-

tive data. This reduces the probability of misdetection, averts the *hidden PU* problem (where one radio is not able to detect a transmitting PU due to fading effects, but whose own communications will interfere with the receiving PU) and PU and SU transceiver locations may be discovered (by triangulation and advertising respectively).

Centralized cooperative sensing has all information relayed to a central device or *fusion center* [98], which combines the sensing data, decides which channels are free and builds up a spectrum map listing the frequency bands. This process is known as *decision fusion*. The occupancy map is then sent to the individual radios or, as is done in IEEE 802.22 [26], commands determining which channel to use are sent. This centralized architecture suffers from the common problems of a single point of failure and limited scalability due to increased reporting overhead with a large number of users.

In distributed schemes, radios still share sensing information but decision fusion is performed individually, reducing the messaging overhead. Improving the reliability of individual decision fusion while minimizing the data that needs to be transferred across has been a focus of research in the literature. One-bit hard agent decisions (the channel is occupied, or it is not) only are communicated in the scheme presented by Sun et al [82] and, if more than M-out-of-N declare for occupancy, then the radio decides the band is in use.

2.5.2 Spectrum Sensing Techniques

Various techniques have been explored for sensing licensed users. Methods in the literature can be categorized as energy/power detector based sensing, cyclostationary feature detection and matched filter based sensing, with techniques requiring a greater knowledge of a PU signal exploiting it to identify the signal at higher noise levels.

Sensing involves a hypothesis test. The sensor measurement of the

channel, α, is compared to a signal threshold γ to decide between the hypotheses:

H_a**:** $\alpha > \gamma$, the channel is occupied (a PU signal is present)

H_0**:** $\alpha \leq \gamma$, the channel is free (a PU signal is absent)

The probability of detection, P_d, is defined as the probability the alternative hypothesis is selected when a PU signal is truly present. Noise, typically modeled in the literature as a zero-mean additive white Gaussian noise (AWGN) random variable [98], will occasionally lead to a false positive. The probability that the hypothesis test incorrectly decides the channel is occupied when it is not is denoted as the probability of false alarm, P_{fa}, caused by setting the threshold too low. A plot of P_d vs P_{fa} for varying γ is known as a *receiver operating characteristic* (ROC) curve and is used to select the best threshold for the current spectrum environment. Increasing the threshold to lower P_{fa} minimizes the frequency with which the SU must abandon the channel even when no PU is present, thus improving *throughput*, but increases the likelihood of interference to PUs by reducing P_d, in what is referred to as the *sensing-throughput tradeoff* problem [49].

2.5.3 Power Detection

The presence of a signal can be determined by an increase in power in the channel medium. Power detection sensing measures the channel power spectral density (PSD). This sensing technique is simple to implement and has low computational complexity when using Welch's method to estimate the PSD.

The periodogram at frequency f is defined as

$$P_N(\omega) = \frac{|X_T[n]|^2}{N}, \tag{2.1}$$

where $X_T[n]$ is the time-limited Discrete Fourier Transform (DFT) of the channel, N is the number of frequency bins, T is the collection time and $\omega = 2\pi f$ is the angular frequency.

In the limit as N goes to infinity, the expected value of the periodogram equals the PSD [21]

$$S_x(\omega) = E\{\lim_{x\to\infty} P_N(\omega)\}. \tag{2.2}$$

Welch's method for estimating the PSD takes the average of several periodograms. The PSD estimate is given by

$$S_x(\omega) = \frac{1}{M}\sum_{M} P_N(\omega), \tag{2.3}$$

where each periodogram is calculated over M time blocks. The computation of the FFT requires $O(N \log N)$ complex multiplications.

Detectors are usually designed for a constant false alarm rate (CFAR). Assuming zero-mean AWGN noise, the PSD estimate threshold that will obtain a given P_{fa} is

$$\lambda = Q^{-1}(P_{fa})\sqrt{\frac{2\sigma_w^4}{MN^2}} + \frac{\sigma_w^2}{N}, \tag{2.4}$$

where σ_w^2 is the variance of the assumed stationary channel noise PSD that can be estimated from long-term observations. A derivation of (2.4) appears in Appendix B. Energy/power detection depends on maintaining up-to-date channel noise floor statistics since it is unlikely to be stationary. Guard bands can be sensed for the noise level. After sensing, Lehtomaki et al [46] assumes the I smallest bin samples in the N-size energy spectral density (ESD) FFT result as the noise.

Disadvantages of this sensing method are poor performance under low SNR conditions and vulnerability to producing false positives in response to noise spikes. Baek et al [18] discusses coming to a sensing decision from multiple ESD measurements. This was shown to improve the ROC characteristics but with a tradeoff in increased sensing time. Energy/power detection is unsuitable for detecting many novel signals that operate close to the noise floor, such as ultra-wideband (UWB) which transmits a low-power signal over a large bandwidth. Beyond trivial cases, power and energy alone do not give enough information to classify different user signals, making it susceptible to PUE attacks.

2.5.4 Cyclostationary Feature Detection

Signals differ in their power spectrum density and spectrum correlation function distributions. Cyclostationary features like carrier wave periodicity produce a response in the distributions. A sensor designed to look for these features can detect the signal of interest with greater noise immunity and in the presence of other transmissions.

In layman's terms, a cyclostationary stochastic process possesses non-obvious periodicity "hidden" in its statistical properties. A stochastic process $x(t)$ is defined to be first order cyclostationary with period T if [78]

$$P\{x(t)\} = P\{x(t+nT)\}, \tag{2.5}$$

for arbitrary x, t and arbitrary integer n, where $P\{x(t)\}$ is the probability density function. It follows that the mean of the signal is periodic with Fourier series representation

$$E\{x(t)\} = \sum_{k=-\infty}^{\infty} m_k e^{j2\pi kt/T} \tag{2.6}$$

and coefficients

$$m_k = \frac{1}{T}\int_0^T E\{x(t)\}e^{-j2\pi kt/T}\,dt \tag{2.7}$$

Similarly, the process is second order cyclostationary with period T if its joint probability density satisfies

$$P\{x(t_1), x(t_2)\} = P\{x(t_1+nT), x(t_2+nT)\}, \tag{2.8}$$

where n is an arbitrary integer and t_1 and t_2 are arbitrary times, with the result that the autocorrelation function R_{xx} of the signal is periodic

$$R_{xx}(t,\tau) = E\{x(t)x^*(t+\tau)\} = R_{xx}(t+nT,\tau) \tag{2.9}$$

and so likewise can be represented by a Fourier series expansion,

$$R_{xx}(t,\tau) = \sum_{\alpha=-\infty}^{\infty} R_x^{\alpha}(\tau)e^{j2\pi\alpha t}, \tag{2.10}$$

with coefficients originally derived by Gardner [34]

$$R_x^\alpha(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} R_{xx}(t,\tau) e^{-j2\pi\alpha t}\, dt \tag{2.11}$$

$$= \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} x(t+\tau/2)x(t-\tau/2) e^{-j2\pi\alpha t}\, dt,$$

where T is the observation time.

$R_x^\alpha(\tau)$ is known as the *cyclic autocorrelation function*. The *spectral correlation function* (SCF) is defined as the Fourier Transform of the cyclic autocorrelation function and is a two-dimensional function in α, the cyclic frequency, and f, the cross spectrum frequency

$$S_x^\alpha(f) = \int_{-\infty}^{\infty} R_x^\alpha(\tau) e^{-j2\pi f t}\, d\tau \tag{2.12}$$

$$= \lim_{\tau\to\infty} \lim_{\triangle t\to\infty} \frac{1}{\triangle t} \int_{-\triangle t/2}^{\triangle t/2} \frac{1}{\tau} X_\tau(t, f+\frac{\alpha}{2}) X_\tau^*(t, f-\frac{\alpha}{2})\, dt,$$

where $\triangle t$ is the length of the observation interval

$$X_\tau(t,f) = \lim_{\tau\to\infty} \frac{1}{\tau} \int_{t-\tau/2}^{t+\tau/2} x(u) e^{-j2\pi f u}\, du, \tag{2.13}$$

is the *short-term Fourier transform*, the spectral component of $x(t)$ at frequency f observed over τ, and τ is a dummy variable which can represent either $\triangle t$ or the Fourier transform frequency resolution. The second order cyclostationary process with period T has SCF components at $\alpha = 2/T, f = 0$ which are its cyclostationary features.

Analogous to the stationary definition, a process possessing both first and second order cyclostationarity is said to be *wide sense cyclostationary* (WSCS). WSCS is found in many common signal types including sine wave carrier modulated signals and those that use cyclic prefixes or repeated spreading. Real life signal message data (bitstream) is found to closely approximate a stationary random process [78] so modulation by a repeating carrier, for instance amplitude modulation (AM), forms a cyclostationary random process with period that of the carrier and SCF responses at twice the carrier frequency.

Cyclostationary feature detectors operate by measuring the channel SCF. A WSCS signal $s(t)$ with cyclic frequencies α_j transmitted over a channel $h(k)$ with AWGN noise $n(k)$ will be received as $x(t)$ with spectral components at [81]

$$S_x^{2\alpha_j}(f) = H(f + \frac{\alpha}{2})H^*(f - \frac{\alpha}{2})S_s^{\alpha}(f) \tag{2.14}$$
$$S_x^0(f) = |H(f)|^2 S_s^0(f) + S_n^0(f).$$

The component at $\alpha = 0$ is simply the signal power spectral density. Figure 2.5 shows the SCF of a binary phase shift keying (BPSK) modulated signal $s(t) = m(t)cos(\omega_c t)$, where the carrier frequency $f_c = 125$MHz, $\omega_c = 2\pi f_c$ and the modulated signal $m(t)$ alternates between 1. Two peaks at $\alpha = 2f_c = 250MHz$ and two at $\alpha = 0$ are clearly visible.

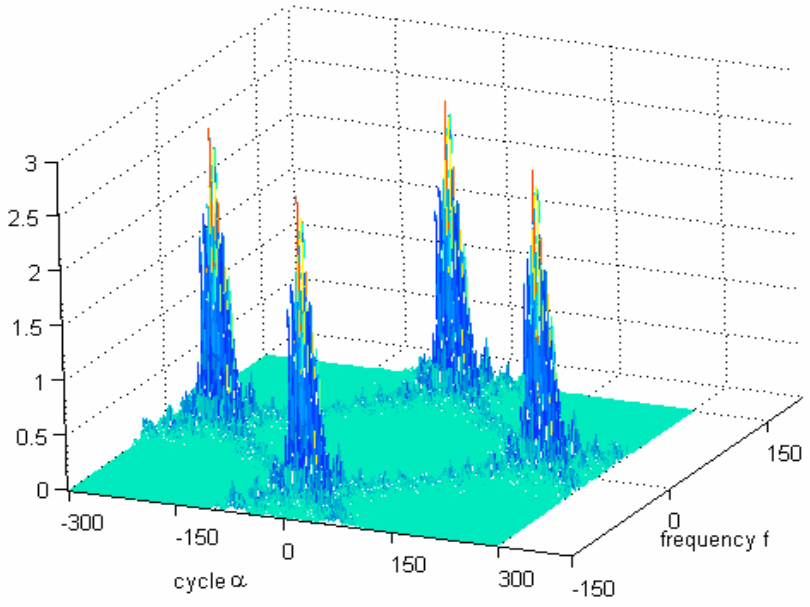

Figure 2.5: Spectrum Correlation Function distribution of a BPSK modulated signal, from [72]. Carrier at 125MHz, bandwidth 20MHz, square root raised cosine pulse shape with roll-off=0.25, sampling frequency 0.8GHz.

Depending on the cyclostationary features possessed, different WSCS signals will show distinct SCF patters, which can be exploited for signal

classification. If the PUs are known the sensing result can be filtered for its distinctive cyclostationary features, in the process reducing the contribution in the response from other signals. In [63], α-profiles, 2-D plots of the SCF at different values of α, from different signals are used to train a classifier block employing an artificial neural network. This raises the possibility for the radio to autonomously learn user signals it encounters. Also greater robustness against noise is achieved if the cyclostationary features are known, since white noise as a WSS process has no correlation and no response at the features. The presence of a generic signal with no features can still be detected by elevated levels at $\alpha = 0$, equivalent to the PSD.

A signal processing block diagram for measuring the SCF is shown in Figure 2.6. A disadvantage is that more computation is needed when compared to energy/power detection.

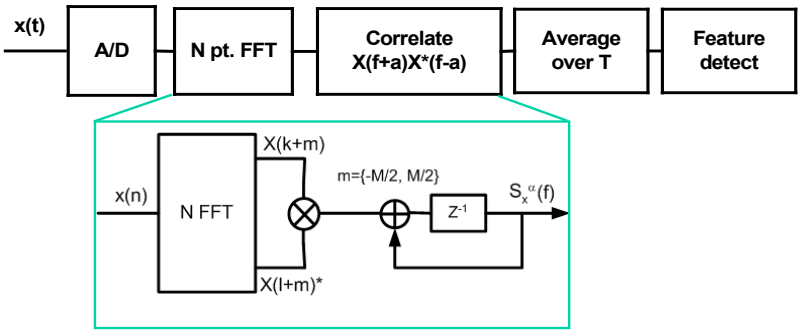

Figure 2.6: Cyclostationary Feature Detection Sensor Signal Processing Block Diagram (Frequency Smoothing Method Architecture) for measuring the Spectrum Correlation Function, from [72]. The interval $-M/2, M/2$ are the FFT bins of interest, T the collection time.

2.5.5 Matched Filter Based Sensing

For a known signal, the matched filter is a LTI (linear time-invariant) filter that maximizes the SNR, where the noise is assumed to be wide sense

stationary (WSS) zero-mean additive noise. Implementation of this approach requires explicit knowledge of the PU signal to derive the filter transfer function. However once this is obtained, the PU can be sensed with greatest P_d for given P_{fa}. Different signals are rejected by the filter, thus the approach is proof to misidentifying other PUs or malicious SUs. A disadvantage is a significant memory cost to storing individual transfer functions for every signal. If more than one signal uses the channel, an efficient scheme is required to decide which matched filter to apply when the channel is sensed. Generally, perfect knowledge of PU signals is not obtainable and matched filtering is not used in practise, although it sets an analytical benchmark.

2.5.6 Sensing Methods Summary

The various sensing techniques discussed have separate advantages and disadvantages, making them suited to different tasks. Energy/power detection is able to detect a generic set of signals operating above the noise floor without requiring information on their transmission characteristics. Cyclostationary feature and matched filter based detection are implemented to distinguish and detect a specific known PU with superior SNR.

2.5.7 Sensing Case Study: IEEE 802.22

Emerging CR protocols will often use a combination of sensing methods and the IEEE 802.22 WRAN standard for opportunistic TV channel access is presented as a case study. In the 802.22 architectures SUs, referred to as Consumer Premise Equipments (CPEs), are organized into a co-operative sensing network around a Base Station (BS). All users are required to regularly perform *fast sensing* employing energy/power detection or other simple, very fast ($\sim$1ms) algorithm for general signal detection [26]. The results of fast sensing are forwarded to the BS and if it suspects a signal

is present, it commands the CPE using the affected band to perform *fine sensing*. Fine sensing executes a longer (∼25ms) algorithm, such as signal classification, capable of verifying that indeed a licensed TV signal is using the band. IEEE 802.22 requires the SU to vacate the channel within 2 seconds of a PU appearing.

2.6 Summary

This chapter introduces background material on cognitive radio. Cognitive radio is framed as being motivated by the spectrum scarcity problem, which it is capable of addressing by combining adaptability, awareness and intelligence to opportunistically access whitespace. Software defined radio and spectrum sensing technologies are discussed as relevant to the radio implementation task of this thesis. The design of a cognitive radio for experimental validation of a Q-learning channel selection scheme on the GNU Radio SDR forms the content of Chapter 3.

Chapter 3

Radio Design and Implementation on GNU Radio

To answer the research question, the GNU Radio SDR platform is used to implement a real CR system. This section considers the design and implementation of spectrum sensing and *spectrum agile* functionality on the radio. CRs need to be able to transmit across different channels, new services which must be handled by the MAC and PHY layers.

This chapter is organized as follows. The different roles of the cognitive MAC and PHY layer are first discussed in Section 3.1. The design of the CR and the CSMA/CA-based cognitive MAC protocol are then described in Section 3.2. Finally, Section 3.3 presents the GNU Radio implementation, with the frame transmitter and receiver treated in Section 3.4, power detector in Section 3.5, and MAC layer implementation in Section 3.6.

3.1 Background

The Open System Interconnection (OSI) model for describing networks separates network functionality into layers. As the lowest layer, the Physical or PHY layer is responsible for the bitwise transmission of data between nodes. The modulation implementation and other signal process-

ing used, such as pulse shaping, to transmit bits across the physical link is defined in this layer. Residing above the PHY layer is the link layer which transfers data between adjacent nodes encapsulated in link-layer frames. This layer is further subdivided into the Logical Link Control (LLC), providing flow and error control, and Media Access Control (MAC) sublayers. Bit errors arising in the physical transmission are detected by the LLC layer and, in a reliable service, retransmission or error correction is performed to guarantee the original frame data is received.

In a shared medium, such as wireless, frames may be corrupted when several nodes transmit concurrently and it becomes necessary to use a MAC protocol in order to co-ordinate user access to prevent collisions. This is typically achieved in existing MAC protocols like CSMA/CA using handshaking, i.e. RTS/CTS, to announce when the medium is in use. DSAN networks require additional features from the MAC implementation. The protocol must handle the multichannel case and needs to distinguish in its response between other SUs, which can be co-ordinated with, and PUs, which must result in the spectrum being immediately vacated. Additionally using reconfigurable CRs, collisions could be potentially avoided by adjusting the physical transmission parameters, such as reducing the power level, so a channel could be shared without interfering with other users. Thus CRs usually see a close coupling of the PHY and MAC layers as identified in [30].

The responsibilities of a cognitive MAC protocol are described in [16]:

1. *Spectrum access*: Spectrum access needs to be coordinated to prevent collisions when multiple SUs select the same channel for communication. This constitutes the core function of classical MAC protocols like CSMA/CA and CSMA/CD, but in DSAN networks is complicated by SU access being spread across multiple channels. The protocol needs to arrange a synchronized period when all SUs meet at a *common control channel* to arrange which channel to transmit on and be aware of which other channels will be in use by SUs to avoid.

As CRs operate in licensed bands, the common control channel must be switchable since, when PU activity appears in the channel, SUs must vacate it as part of *spectrum mobility*. The C-MAC [27] protocol arranges a backup channel (BC) for nodes to switch to when a PU appears in the existing rendezvous channel (RC). The channel with the longest time between PU interruptions is selected for the RC. The common control channel is also used to exchange sensing results in co-operative sensing.

2. *Spectrum sensing*: The cognitive MAC protocol manages sensing for the presence of PUs. The protocol needs to arrange for quiet periods, when all SUs are silent, when sensing can be performed. The length of the sensing period needs to be chosen to balance protocol overheads with the likelihood a PU is correctly detected and identified. As discussed in Section 2.5.1 many architectural approaches to sensing are possible. Cooperative sensing improves the accuracy of the final PU spectrum occupancy map used for determining which channels are available for communications. Integration of results from different users in *decision fusion* can take place in a single node or distributed, with each user developing its own map.

3. *Spectrum allocation*: Spectrum allocation selects the channel for the node to communicate on from those not currently occupied by PUs. Intelligent/adaptive techniques are employed to select high-quality channels to improve performance, such as predictive channel selection as proposed in [41]. By selecting the data channel with the longest remaining idle time before a PU transmission, a 43% reduction was measured in [41] in the number of forced channel switches when compared to random channel selection for a SU operating among eight channels with periodic PU traffic.

4. *Transmitter-receiver handshake*: Otherwise known as *channel rendezvous*, a mechanism is needed to let the receiver know which data

channel the transmitter has selected for communications. In the CSMA-based MAC protocol proposed in [83], the channel is selected by the receiver. The protocol uses RTS-CTS-CONFIRM-DATA-ACK handshaking. RTS-CTS-CONFIRM is transmitted in the common control channel. RTS, sent by the transmitter, lists the available channels. The receiver chooses a channel from the list and sends this in the CTS. The sender broadcasts CONFIRM with the chosen channel before both nodes switch to the data channel. The advantage of this scheme is that PUs at both the transmitter and receiver ends are considered when arriving at the channel decision, averting the hidden user problem.

5. *Spectrum mobility*: SUs are required to immediately cease communications when a PU requires use of the occupied portion of the spectrum. Spectrum mobility techniques need to be defined for resuming communications in another channel with minimal interruption during the handoff period. Worsening channel conditions will also trigger spectrum mobility.

3.2 Radio Design

A single-radio CR is developed. Its logical architecture is depicted in Figure 3.1. Spectrum agile spectrum access is handled by the MAC protocol which controls when a frame should be transmitted and when to sense as part of the protocol. The Q-learning DCS scheme determines the data channel to use. This is passed to the frame handler which sets the PHY layer properties for the channel at the transmitter and receiver and breaks the MAC layer frame into individual bits that are sent across the channel by the transmitter, with the reverse being done for the received bits. A power detector is used for sensing. The distinction between single- and dual-radio architectures is defined in Section 2.5.1. In single-radio ar-

chitectures, data communications and sensing share one transceiver, thus both cannot proceed simultaneously.

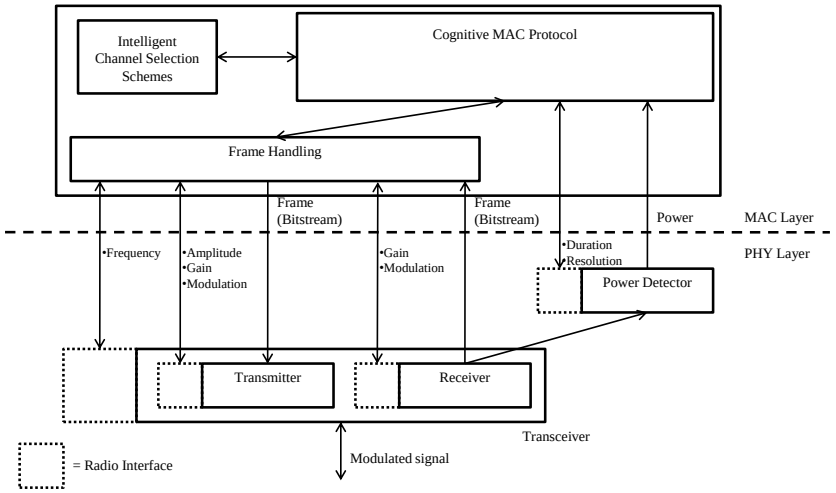

Figure 3.1: Single Transceiver Cognitive Radio Logical Architecture

3.2.1 MAC Layer Design

A single transceiver cognitive IEEE 802.11-based MAC protocol for ad hoc networks is designed. The protocol assumes higher link-layer issues such as neighbor discovery and address resolution have been handled and covers the interesting CR problem of transferring a data packet between a transmitter and receiver aware of each other on a licensed band of the transmitter's choosing. The implementation of intelligent *spectrum allocation* is not discussed in this section but in Chapter 4 where the reinforcement learning DCS scheme is introduced.

The protocol uses a modified form of CSMA/CA (carrier sense multiple access with collision avoidance). CSMA/CA is used by IEEE 802.11 and forms a natural starting point for a project of this type. Figure 3.2 shows the MAC frame format in the design. The IEEE 802.11 MAC frame format is retained with unused features stripped away, originally to keep

down the minimum transmission time during tests. Unfortunately as elaborated in Section 3.4 because of how the USRP USB interface works, packets are padded anyway to multiples of 512 bytes in excess of the frame header size so they can be sent to the USRP without waiting. Addresses 3

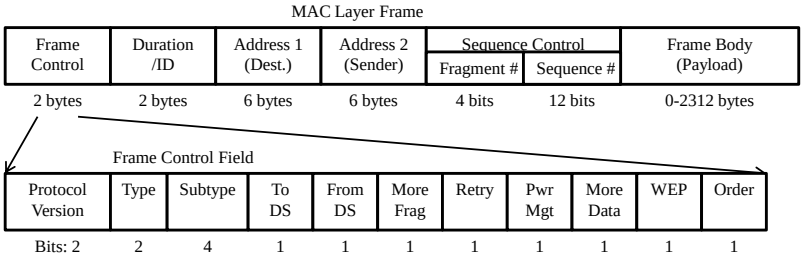

Figure 3.2: MAC Frame Format

and 4 have been removed, which in IEEE 802.11 are used for the base station ID and other infrastructure addressing modes, as the system is only used in ad-hoc mode. The frame fields are identical in purpose to IEEE 802.11 [20] and are summarized below:

- *Frame Control*: Broken up into subfields. The subfields *type* and *subtype* identify the frame's type. For instance a *type* value of `0b01`, *subtype* `0b1011` indicates an RTS frame. A retransmitted frame sets the *retry* bit. Data exceeding the payload size is fragmented and sent in separate frame transmissions. Subfield *More Frag* is set to 1 when there are more fragments belonging to the same frame following the current fragment.

- *Duration/ID*: This field indicates the time remaining in the transmission and is used to calculate the Network Allocation Vector (NAV).

- *Address 1/2*: 6-byte MAC addresses of the frame sender and destination.

- *Sequence Control*: used to detect missing or duplicated frames and fragments. The *Fragment Number* subfield is the number of fragments

the payload has been split up in. Subfield *Sequence Number* is incremented each time a frame is sent.

Our protocol uses a temporary common control channel (CCC) which all users tune to when not actively sending or receiving packets. The CCC is used to co-ordinate access between SUs to prevent collisions in the same licensed band, otherwise known as the *hidden multi-channel user problem* [95], and arrange channel rendezvous.

The CCC approach raises a number of issues. The DSA network premise is to solve the spectrum scarcity challenge by exploiting already allocated spectrum. A static CCC is unviable as the channel will be temporarily out of communications whenever it is used by a PU and licensing a dedicated CCC signifies the network is still dependent on free spectrum being available. The alternative, which we use, is for the channel to be temporary, with SUs dynamically switching to a different quiescent CC whenever a licensed user appears in the original channel. Previous studies have shown that it is highly likely there will be a number of common channels between neighbours [99], thus it is possible to find a new CC. For instance in the C-MAC [27] cognitive MAC protocol this is achieved by having nodes keep track of a backup channel to use if the CCC is occupied. The aim of the research presented in this thesis is to evaluate the data channel selection performance of a Q-learning scheme. To fix the effect CCC availability has on experimental results, for the purposes of this thesis the CC chosen is unused by licensed users in the timeframe of experiments considered, thus the protocol does not define a CC switching mechanism if a PU does appear in the channel.

The potential for the channel to be congested if there are too many users has been raised [87]. However it has been shown that a CCC can support 21 data channels with a high node density of 40 nodes [51]. Other approaches such as split-phase or hopping eliminate the need for a common control channel but requires node synchronization, not suitable in a completely ad hoc scenario. For these reasons, the cognitive MAC protocol

adopts the CCC approach.

Our MAC protocol's transmission timeline is shown in Figure 3.3. The sender and destination exchange RTS-CTS in the CCC after the channel has been free for a DIFS period. CCC transmissions use a common waveform known to all other users. The RTS payload contains the physical layer waveform parameters to use for the transmission, such as the channel frequency or modulation picked intelligently by the sender with the goal of maximizing performance and avoiding using a band currently occupied by PUs. The receiver echoes the waveform parameters in its CTS payload. Both sides use the CTS parameters to set their physical layer. Potentially the sender can pass a list of available channels and designed waveforms. The receiver can select from its own record of channels recently sensed unoccupied the best solution or propose a new one if it is using incompatible hardware, avoiding the *hidden PU problem* [100] where the transmitter selects a channel unaware of a PU at the receiver. Intelligent waveform design after Rondeau [71] and transmitter and receiver decision fusion are interesting topics for future work. In the initial protocol implementation set up for dynamic channel selection, the intelligent sender selects a frequency channel and the destination simply repeats it in its CTS.

Just as in IEEE 802.11, other SUs receiving the RTS or CTS, for hidden users, set their NAV vectors and avoid using the data band specified in the payload for the frame duration. Before this timer expires, they may still arrange to transmit in other bands. Binary exponential backoff (BEB) is retained from IEEE 802.11. After a successful transmission, a retransmission or if the CCC is sensed busy at the start of the transmission, the node waits a number of Slot Time (ST) intervals chosen randomly from 0 to 2^{i-1} before transmitting. Initially $i = 0$ and is incremented every time there is a collision until RTS is sent successfully. This prevents waiting users from broadcasting simultaneous RTS. l Following CTS, the active pair tune to the data channel they have agreed upon. The transmitter senses the band,

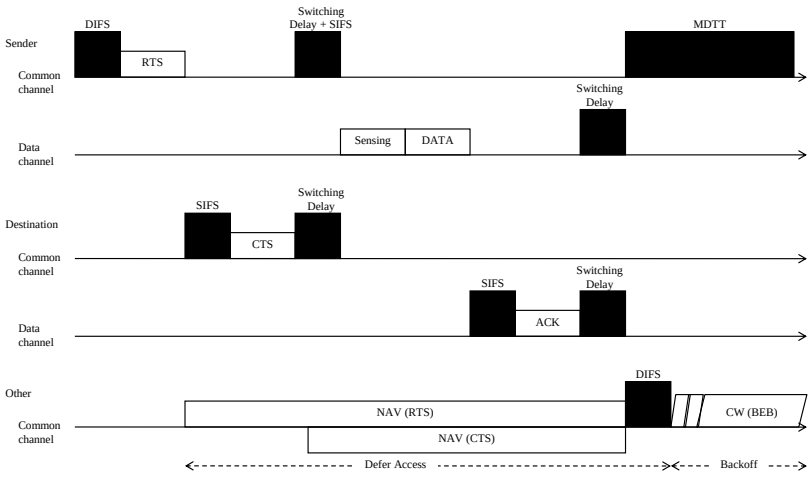

Figure 3.3: MAC Protocol Transmission Timeline

a final check to see if a PU or other SU is present. This approach is known as *Listen-before-Talking* sensing. If a licensed user is sensed by the power detector, the transmitter aborts, returns to the CCC and waits an MDTT (Maximum Data Transmission Time) interval defined as the maximum transmission cycle time and longer than the destination timeout before reattempting transmission. If the channel is unoccupied, the transmitter sends its DATA packet and receives an ACK packet signalling correct delivery. Both nodes then return to the CCC. A new transmission is not attempted until the MDTT period elapses. Thus, if no RTS or CTS has been received in MDTT, the node knows that when it can transmit again no other SU is transmitting on a channel it may wish to send on. The overhead ensuring this is a limitation of the CCC approach as found in other work [58].

The MAC protocol has a responsibility to manage spectrum sensing. Regular sensing needs to be arranged for when all SUs are not broadcasting and PU activity can be determined. Sensing must be regular or otherwise frequent and diversely spread across the licensed bands, so all SUs have a recent picture of which channels are available. Sensing is time

wasted that is not being spent transferring data and the overhead of any such scheme needs to be weighed accordingly. In the literature, protocols often assume the ability to synchronize users' activities. In IEEE 802.22, a Base Station (BS) instructs other users when and where to perform sensing and ensures the channel is quiet when this takes place [26]. The Cognitive MAC [27] (C-MAC) protocol has a split-phase setup. Time is split into a Beacon Period (BP) and Data Transfer Period (DTP). Nodes tune to a common Rendezvous Channel (RC) at the start of the BP and announce which channel they are going to transmit on. If no transmissions are planned the nodes on that channel perform sensing and broadcast the results of it during the next BP.

Our *Listen-before-Talking* sensing approach achieves a fully ad-hoc solution without synchronization overheads or hardware. Prior to sensing RTS-CTS reserve the channel, keeping it free of other SUs. Sensing occurs in-band in the channel the node pair has selected for communications. The protocol does not define a mechanism for regular sensing of all bands. This is given over to the reinforcement learning DCS scheme in Chapter 4. As part of its exploration policy, all channels are randomly selected for transmission giving frequent sensing of all channels that can be used to build up a PU occupancy map. Sensing of a channel can be explicitly requested through the RTS mechanism, which will be required if one side does not transmit often. Although transmission is aborted if a PU is detected, interference may still be caused if the user is hidden or appears after sensing. The second is unavoidable with a single transceiver design.

An improvement to the MAC protocol would be to reduce the individual sensing load by sending the ACK over the CCC. Waiting SUs can determine from a successful transmission the channel is not occupied by PUs and suitably extrapolate from a failed transmission, leading to sharing of sensing results. This approach has other advantages. Licensed users would not be interfered by the ACK, in particular, hidden users on the receiver side that are not detectable by *Listen-before-Talking* sensing. In this

protocol, the ACK is transmitted in the data channel. Previous work by Yau et al [96] uses data channel ACK and this feature has been adopted.

3.3 GNU Radio Implementation

The CR is implemented in GNU Radio. The GNU Radio software architecture for designing SDR lets high-performance PCs perform real-time signal processing. Once a waveform has been designed in software, the physical transmission of the signal is handled by a USRP RF frontend. A general overview of the architectures is given in Appendix A and should be referred to.

The CR software implementation is shown in Figure 3.4. Each radio makes use of:

- One Linux PC host computer radio backend, that performs PHY layer waveform processing and hosts the MAC layer protocol implementation.
- One USRP1 with USB2.0 connection to the computer.
- One RFX2400 2.3-2.9 GHz capable transceiver daughterboard or XCVR2450 2.4-25GHz and 4.9 to 5.85GHz dual-band transceiver daughterboard.
- One VERT2450 dual band 2400-2480MHz and 4.9-5.9GHz vertical antennas.

The USRP FPGA can be programmed, but developing this capability was outside the scope of this thesis. The default FPGA image is used that implements segments of four Digital Down Converters (DDCs) and Digital Up Converters (DUCs), as well as an IF (intermediate frequency)-baseband conversion implementation. This entails the entirely userspace-based implementation architecture with all remaining baseband signal processing handled in host computer flowgraphs.

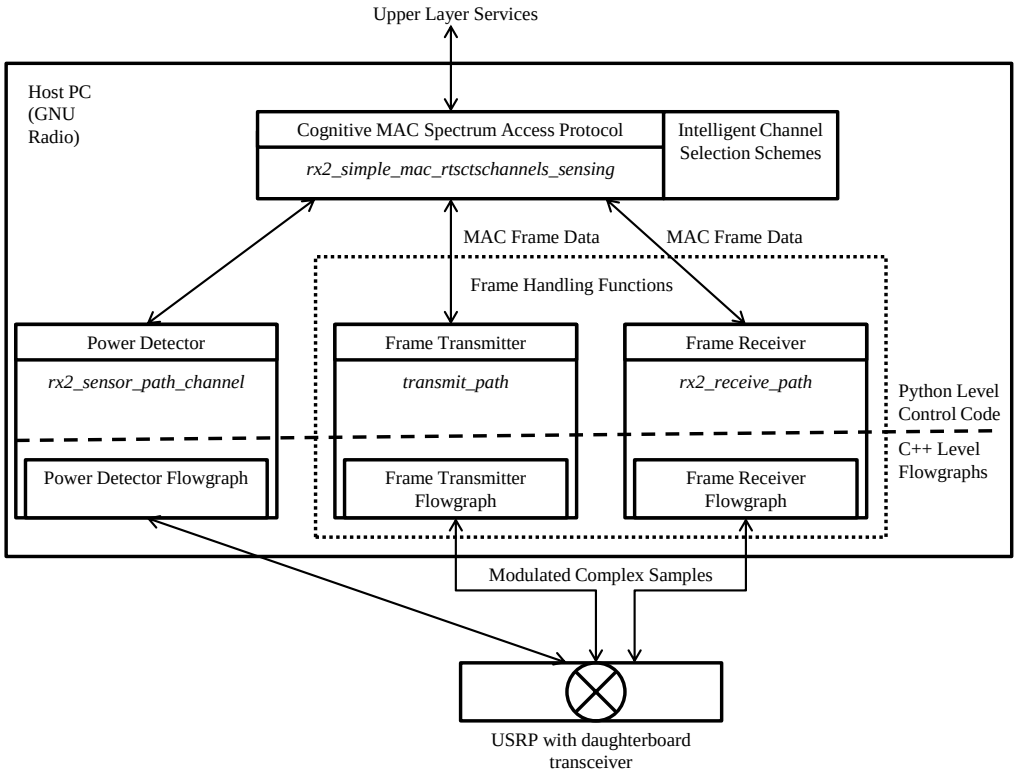

Figure 3.4: Cognitive Radio System Architecture

The power detector, frame transmitter and receiver from the radio design in Figure 3.1 are implemented in separate python classes. Each class, e.g. `transmit_path`, handles a GNU Radio flowgraph, performing the appropriate signal processing task and provides an interface of methods for modifying, sending and receiving flowgraph data. The MAC protocol class `rx2_simple_mac_rtsctschannels_sensing` creates an instance of each class and controls the protocol operation by accessing the class interfaces. Frame handling operations in Figure 3.1 are also handled by `transmit_path` and `receive_path`.

MAC frame data is passed down to the `transmit_path` frame transmitter class. The class encapsulates the frame with physical layer log-

ical information used for decoding, such as a CRC-32 error detection field. The encapsulated frame is then put on a flowgraph responsible for modulation and the resulting complex baseband signal is passed over the USB to the USRP for transmission. The inverse operation is performed by the `rx2_receive_path` frame receiver class. The `rx2_sensor_path_channel` power detector class creates and manages a flowgraph to measure the channel power. The python class then applies a hypothesis test on the output to determine the presence of a PU is indicated. The MAC protocol receives the boolean result.

3.4 Frame Handling

The role of the frame transmitter and receiver is to achieve bit-by-bit physical transmission and reception of MAC frames. This is implemented using the native GNU Radio packet transmitter and receiver examples, `gnuradio-examples/python/digital/benchmark_tx.py` and `benchmark_rx.py`. The design can be improved, but the existing contribution achieves what is required in terms of packet error rate for instance.

3.4.1 Frame Transmitter

The frame transmission process as implemented is summarized in Figure 3.5. The MAC protocol calls `transmit_path.send_pkt` to send `payload`, a MAC frame, which is passed in as a packed binary data string (python `struct.pack`). The MAC frame data is encapsulated with physical layer logical information fields as depicted in Figure 3.6. The preamble and access code are specifiable. The access code is shared between users in the network and marks the beginning of the frame for the receiver. The payload length field in fact consists of two duplicated two byte signed shorts, each giving the actual payload length. The size of these fields al-

lows for a payload between 0 and 4096 bytes. The CRC (cyclic redundancy check) value of the payload is calculated and used to determine if it has been correctly received.

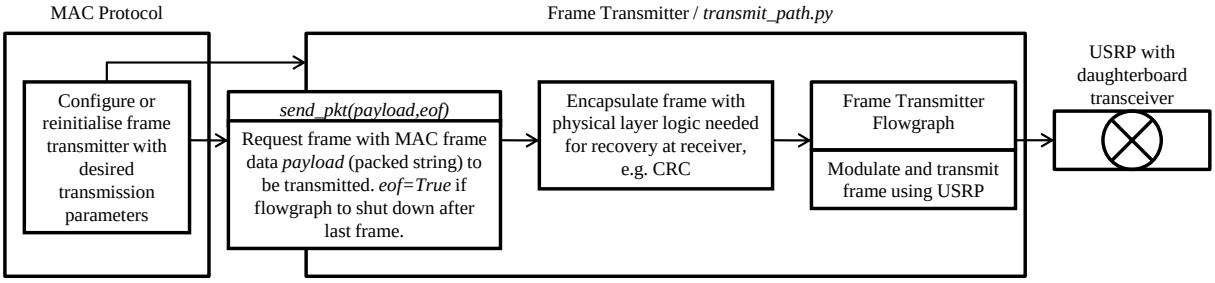

Figure 3.5: Frame Transmitter Operation

As noted in Section A.4.2, a problem with the USRP USB interface is that a packet is sent only after at least 512 bytes of data is available at the buffer. To ensure the frame is sent immediately to the USRP for transmission, the frame is padded so the modulated frame comes to a multiple of 512 bytes or 128 complex samples in size.

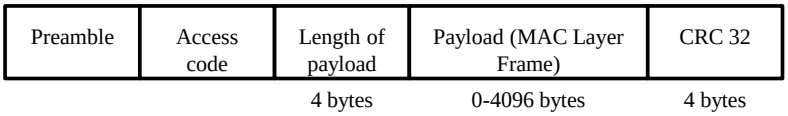

Figure 3.6: Frame Format

The frame is next whitened or scrambled by being `xor`'ed with a prearranged offset mask to prevent eavesdropping. The whitened frame is added to the message queue at the source block of the frame transmitter flowgraph, which is shown in Figure 3.7 and is responsible for modulating the frame and transmitting the physical layer signal over the USRP. The `_pkt_input` message source block dequeues and converts the frames into a bitstream. This is modulated by the next block which outputs

complex float I&Q sample denoting the equivalent baseband signal notation of the RF waveform. In the equivalent baseband signal notation, $Z(t) = I(t)+jQ(t)$ corresponds to $Re\{Z(t)e^{j\omega t}\} = I(t)cos(\omega t)-Q(t)sin(\omega t)$ at RF, where ω is the carrier frequency in rad/s and $I(t)$ and $Q(t)$ are known as the inphase and quadrature phase signals. The USRP block sends data to the device. Before being sent to the USRP for transmission the waveform values are scaled to a desired level (0-32768 max deflection for complex floats).

The USRP sink and source blocks (GNU Radio library `usrp.source_c` and `usrp.sink_c blocks`) transmit and receive baseband signal data to and from the device and allow the USRP and its daughterboards to be configured. The block creates a `subdev` object via which device parameters can be set or read by the flowgraph program such as the daughterboard to transmit or receive on, the daughterboard gain, RF frequency, decimation, interpolation and other IF and RF communication variables. The USRP source and sink decimation and interpolation set the rate at which data is processed by the flowgraph, hence the signal bandwidth.

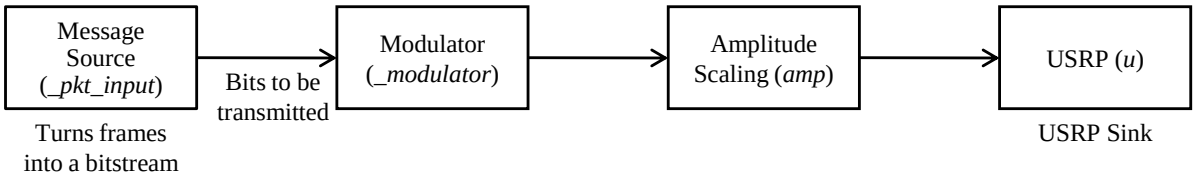

Figure 3.7: Frame Transmitter Flowgraph

A large number of modulator blocks have been written for GNU Radio implementing a wide variety of modulation types. A full listing can be found in the GNU Radio doxygen generated API [6]. Radio experiments in this thesis use GMSK (Gaussian Minimum-Shift Keying) and DBPSK (Differential Binary Phase Shift Keying) modulation, the implementation of which is discussed below.

In DPSK, bit patterns are represented by a change in the phase of the carrier. A binary '1' in DBPSK is denoted by adding a phase of 180° to the carrier, a '0' with no change in phase. More bits can be represented per symbol by reducing the phase difference, such as DQPSK which encodes 4 values per symbol separated by 90°.

The DBPSK modulator flowgraph is illustrated in Figure 3.8 and the structure is used for other m-ary DPSK modulation implementations. The `bytes2chunks` block regroups the incoming bytes into k-bit vectors, where $k = 1$ for DBPSK is the number of bits per symbol. The `symbol_mapper` block then Gray-codes each vector. A Gray code has the property that successive values differ in only one bit. For instance 00,01,11,10 is an example of a 2-bit Gray code. For PSK modulation of higher ordinality, e.g. 16-ary, `symbol_mapper` reduces the overall bit error rate. The most likely case of interference results in the symbol being mapped incorrectly to an adjacent PSK constellation point. This will only lead to one bit in the sequence being in error since successive Gray coded symbols differ in only one bit [32].

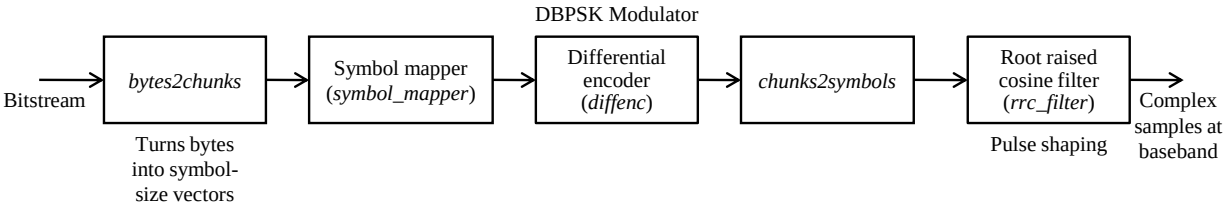

Figure 3.8: DBPSK Modulator Flowgraph

The differential encoder block, `diffenc`, carries out the following operation:

$$y_i = y_{i-1} \oplus x_i, \tag{3.1}$$

where y_i is the output bit to be actually transmitted, y_{i-1} the previous output bit, x_i the input Gray coded bit and $\oplus$ is modulo-2 addition. The input at `chunk2symbols` is mapped to the BPSK constellation and the modu-

lated complex sample is output. It can be seen then the differential encoder uses the past bit to map a '1' to a change of phase from the previous symbol of 180° and '0' to no change, in accordance with DBPSK modulation.

Finally, the symbol is filtered by a root-raised cosine (RRC) filter to reduce intersymbol interference (ISI). Effects such as multi-path fading or distortion in the channel can cause a symbol to be received with a delay or spread out, leading to ISI if it overlaps with subsequent symbols. A second matched RRC filter is implemented at the receiver, forming a raised-cosine (RC) filter with impulse response [66]

$$h(t) = \sin\left(\frac{t}{T}\right)\frac{\cos(\frac{\pi\beta t}{T})}{1-\frac{4\beta^2 t^2}{T^2}}, \tag{3.2}$$

where $0 \leq \beta \leq 1$ is the roll-off factor and T the sampling period. The impulse response is zero at all nT, $n \in Z/\{0\}$, known as the zero-ISI property, so if correctly sampled at the receiver, symbols are received without ISI.

3.4.2 Frame Receiver

The frame receiver and flowgraphs are depicted in Figures 3.9, 3.10 and 3.11. The receiver demodulates and recovers the MAC layer frame payload of the frames being transmitted. The receiver uses a callback arrangement. The MAC layer control code specifies a function handler satisfying the `phy_rx_callback` prototype to python class `rx2_receive_path`'s constructor. This is called whenever a frame has been received and decoded. The receiver does not attempt error correction, but checks the CRC physical logic field to determine if the MAC layer frame payload has been correctly received. The GNU Radio library DBPSK demodulator, shown in Figure 3.11, performs the inverse operation to the modulator. First, complex samples received from the USRP are scaled by `pre_scaler` to fit within the range [-1, 1]. The `agc` block applies a gain factor based on the maximum magnitude of the past 16 samples so the range is fully exploited. The signal is then filtered by `rrc_filter` implementing the matched re-

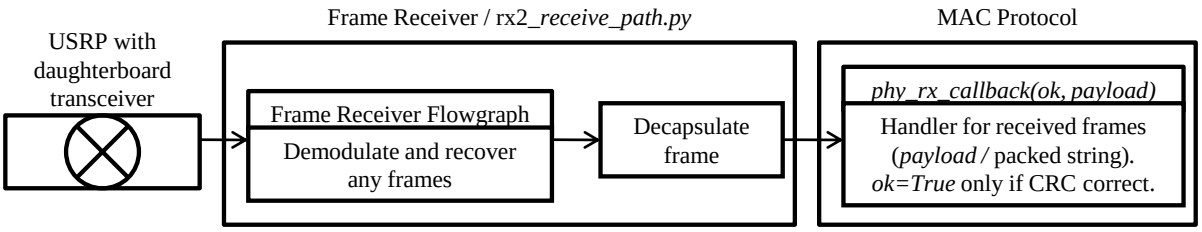

Figure 3.9: Frame Receiver

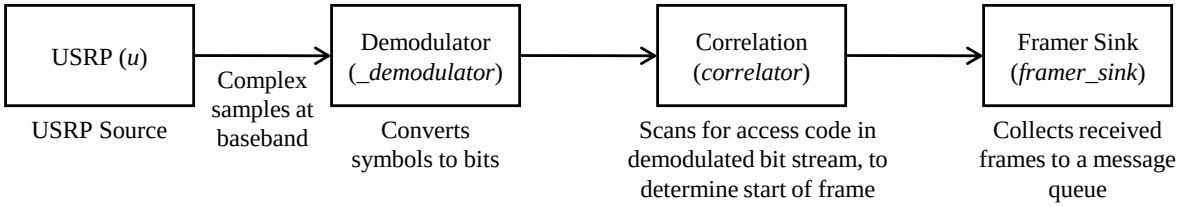

Figure 3.10: Frame Receiver Flowgraph

ceiver half of the raised cosine filter. Phase, frequency and symbol synchronization are performed by `receiver`. The differential decoder multiplies the current sample with the conjugated previous sample

$$y_i \overline{y_{i-1}}. \tag{3.3}$$

The `slicer` block matches the received sample with the closest BPSK constellation point, outputting the Gray encoded k-bit bit sequence (1-bit in

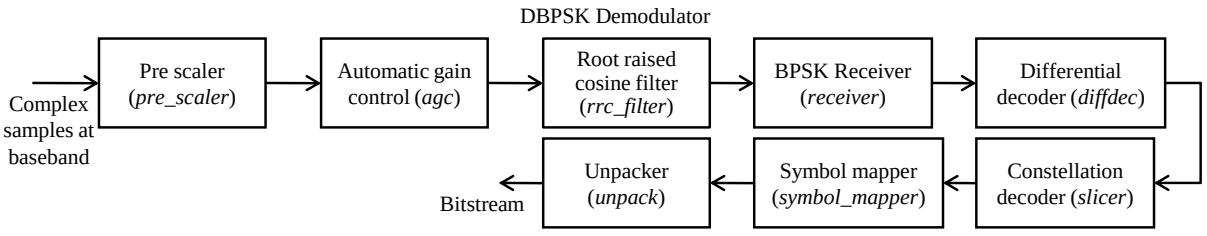

Figure 3.11: DBPSK Demodulator Flowgraph

the case of BPSK) in vector form. Finally, the Gray coded bits are mapped to binary by the `symbol_mapper` block and `unpacker` unpacks the vector sequence into a bit stream.

It remains for the frame to be recovered from the bit stream. The flowgraph `correlator` block scans for the access code. The Hamming distance between the bits and the known code is calculated, with the code recognized as being present if the distance is within a threshold (default 12). For each input bit it outputs alongside a flag bit. The flag is set to 1 if the corresponding data bit is the first bit after the access code and thus the start of the payload length field. The `framer_sink` block checks to see if the next two bytes are duplicated for confirmation that it is the payload length field. Recall from the physical layer packet logic, the field is the short length repeated. If not, the packet is dropped. Otherwise the whitened payload with the CRC is retrieved and put on a message queue structure. A queue watcher thread that has been blocking on the queue removes the frame, dewhitens it and checks the CRC against the MAC frame data payload. The payload is passed to the callback with `ok` set to `True` only if the CRC is correct.

3.4.3 Control Interface

The CR MAC layer requires access to a flexible PHY layer. It needs to be able to change the per packet physical transmission parameters to take advantage of new data channels. Enhancements are made to the transmitter and receiver to support this. Block parameters are exposed to the MAC protocol for control through accessor and mutator API methods. For instance the amplitude of the transmitter `scaler` block can be changed and the transmission/reception frequency is tuned using `transmit_path/rx2_receive_path.set_freq()` and `get_freq()`. This adjusts the setting in the GNU Radio library `usrp.source_c` or `usrp.sink_c` block which transmits and receives to and from the USRP.

The block configures the USRP and daughterboards via a `subdev` object and the gain, decimation, interpolation and board status can be changed as well at the request of the MAC layer. The source and sink set the rate at which data is processed in the flowgraph and hence the transmitted signal and received bandwidth, adjustable using the decimation and interpolation settings. Blocks with different functionality, such as another modulation technique, can be swapped out but this requires the flowgraph to be temporarily stopped. The current solution is for the MAC protocol to create a new transmitter or receiver. Python's `OptionParser` is used to pass in the many initialization settings.

3.5 Power Detector

The constant false alarm rate (CFAR) power detector calculates the average channel power spectral density and applies a hypothesis test to decide if a signal is present. Welch's method, as outlined in Section 2.5.3, is used to obtain the PSD estimate.

The power detector flowgraph shown in Figure 3.12 computes the squared magnitude FFT. The block `s2v` converts the stream of complex baseband samples from the USRP into a stream of N-size vectors. Next, block `fft` operates on each vector and computes the N-size FFT, the squared magnitude of which is then calculated by `c2mag`. A Hanning window is applied before the FFT to mitigate spectral leakage effects. Finally, `stats` computes

$$\overline{P_N} = \frac{1}{K} \sum_{n=-K/2}^{n=K/2} \frac{|X_T[n]|^2}{N}, \tag{3.4}$$

which is the average of all periodograms corresponding to frequency bins $n = [-K/2, K/2]$ which the channel spans. The FFT bandwidth is set by the USRP decimation rate and this extra step allows a channel to be singled out. The result is added to a message queue.

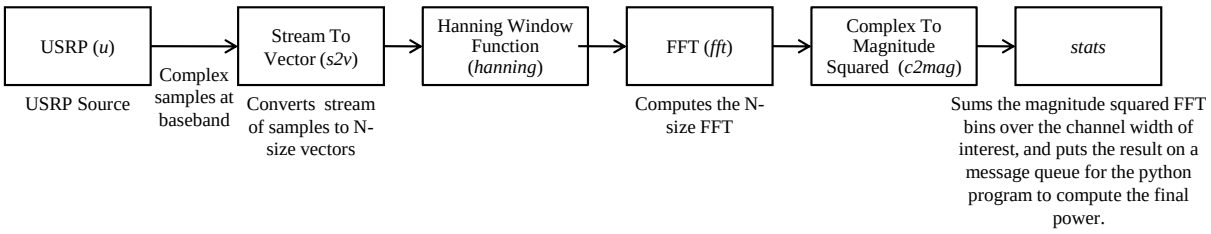

Figure 3.12: Power Detector Flowgraph

The flowgraph runs in the background. The power spectral density estimate is not calculated until the MAC protocol requests the channel be tested for a signal by calling `rx2_sensor_path_channel.is_pu_present()`. This empties the queue, blocks until M periodograms which are all up-to-date samples have been received and averages them to give the channel average PSD estimate

$$S_x(\omega) = \frac{1}{M} \sum_{M} \overline{P_N}. \tag{3.5}$$

The estimate is compared to the decision threshold λ maintained by python class `rx2_sensor_path_channel`. Calling `rx2_sensor_path_channel.is_pu_present()` returns `True` to a user signal being present in the channel if the PSD estimate exceeds the threshold, otherwise `False`.

H_a**:** $S_x(\omega) > \lambda$: `True`, the channel is occupied

H_0**:** $S_x(\omega) \leq \lambda$: `False`, the channel is free

The threshold value is set by `rx2_sensor_path_channel.get_set_channel_statistics(notrials)` to meet a constant false alarm rate, specified on initialization, according to Equations (B.7) and (B.8) derived in Appendix B. Number `notrials` $S_x(\omega)$ PSD estimates are measured in an unoccupied channel and the mean μ and standard deviation σ of the distribution derived. The CFAR threshold value is set

to

$$\lambda = \mu + \sigma Q^{-1}(P_{fa}), \tag{3.6}$$

where Q is the Q-function. The greater the number of periodograms averaged, M, in Equation 3.5 the more closely the distribution approximates the normal distribution according to the central limit theorem (CLT).

In our relatively short-term experiments, the common control channel employed by only SUs is sampled at the start of the experiment, when no users are transmitting, and the noise statistics are used for all data channels. This assumes the noise is stationary and there is no protocol for keeping statistics up-to-date.

3.5.1 Control Interface

The same control interface design philosophy is used for the power detector. The MAC protocol is provided accessor and mutator functions to control the bandwidth sensed, the FFT resolution and the central frequency sensed.

3.6 MAC Layer Implementation

Each radio implements the MAC layer defined in Section 3.2.1. An overview of the implementation is depicted in Figure 3.13. The transmission and reception of upper link layer packets using the MAC transmission protocol in Figure 3.3 is handled by an instance of python class `rx2_simple_mac_rtsctschannel_sensing`.

On creation, the class sets up the radio physical layer. The frame transmitter, receiver and power detector are instantiated and the flowgraphs started. The initial layer settings can be specified as additional arguments when `rx2_simple_mac_rtsctschannel_sensing` is created, which are read in using python's `OptionParser` utility. Auto transmit/receive (ATR) switching is enabled on the USRP daughterboards.

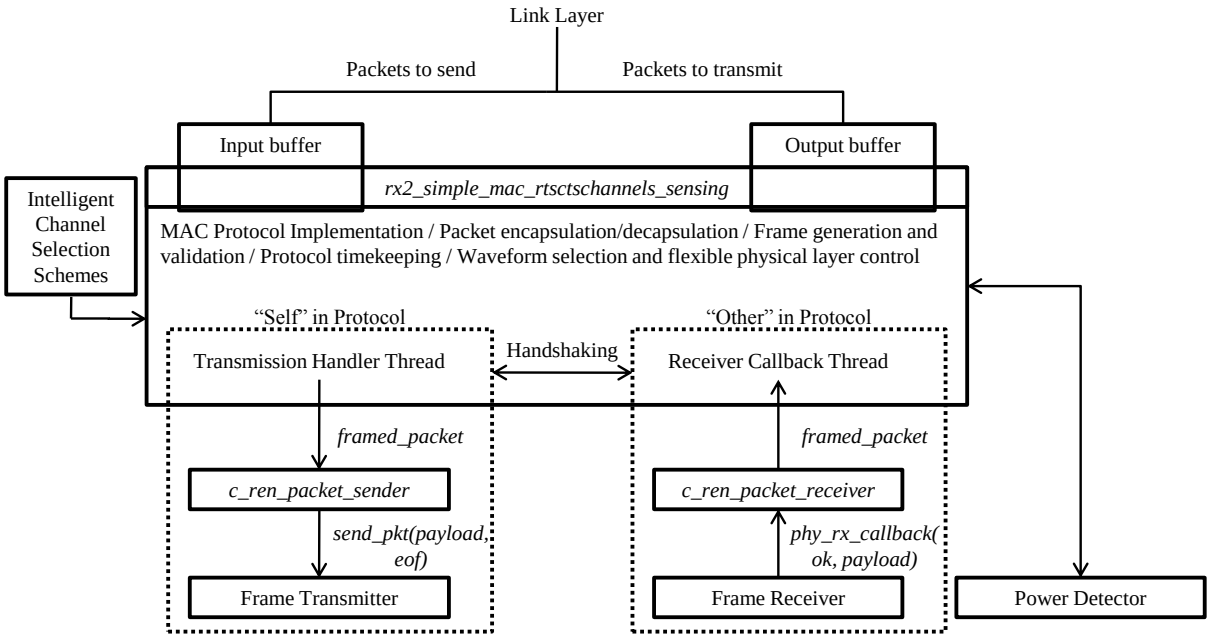

Figure 3.13: Python GNU Radio MAC Layer Implementation

In this mode, the transceiver defaults to receive mode up till data is present at the TX FIFO queue, referring to Figure A.4, at which point the USRP toggles the transceiver to transmit operation until the data has been sent. Since the design uses a single transceiver board, the frame receiver and power detector share the same `usrp.source_c` block. As a result `rx2_simple_mac_rtsctschannel_sensing` must reconfigure the block for *Listen-before-Talking* sensing and then restore it to the parameters used for reception.

The class reserves an input buffer and output buffer. Link layer packets to be transmitted are placed at the input buffer, which are then processed in order. Class `rx2_simple_mac_rtsctschannel_sensing` coordinates sensing, timing and the channel the physical layer is tuned to. MAC frames are represented as a `framed_packet` object containing fields for the sequence number, type and so on. The contribution of the ADROIT

project [5] is acknowledged. ADROIT released code for a GNU Radio implementation of an 802.11 receiver and our `framed_packet` class is derived from the MAC frame structure class of the same name in their 802.11 receiver code. Class `rx2_simple_mac_rtsctschannel_sensing` generates control frames (RTS, CTS, ACK) and encapsulates the payload at the buffer into a DATA frame. When in the protocol the frame is to be physically transmitted, the `framed_packet` object is passed to class `c_ren_packet_sender`. The class converts the frame to the `python.struct` packet binary string format used by the frame transmitter, which transmits it. The class `c_ren_packet_sender` also fragments the frame if it exceeds the 802.11 2312 byte total payload length limit and correctly sets the fragmentation number in the frame. Each fragment is sent to and is transmitted separately by the frame transmitter.

The frame receiver listens to frame responses from other SUs. Class `c_ren_packet_receiver` acts as the callback function for the receiver and converts received binary string MAC frames into `framed_packet` format. Further processing is handled by `rx2_simple_mac_rtsctschannel_sensing`. DATA packets addressed to the radio are decapsulated and added to the output buffer for the link layer to read.

The class `rx2_simple_mac_rtsctschannel_sensing` spawns a handler thread for the ongoing transmission, which is responsible for sending all frames the radio is required to by the MAC protocol. The transmission involves a two-way handshaking exchange of control frames with the node the radio is communicating with. Received frames are detected by the frame receiver's queue watcher thread (recall Section 3.4.2), which triggers the callback function `c_ren_packet_receiver`. The packed binary string payload is converted to a `framed_packet` object if it was not garbled (`ok=False`). If the frame has more fragments to follow, `c_ren_packet_receiver`, suppresses its output and waits for the remaining fragments, which are reassembled into a single `framed_packet`

object.

The queue watcher thread can be viewed as the source of frames produced by the node (*other*) with which the radio is communicating, while the handler thread is responsible for generating all frames transmitted by the radio (*self*) in the timeline. As frames are received are sent, each thread progresses the handshaking process. After transmitting its frame the handler thread waits on a python condition variable until protocol timeout occurs. The time is established by the python `time` function. If a frame is received before timeout this is put in a shared variable and the receiver wakes the handler. The first act of the handler is to validate the frame and if correct (e.g. type, sequence number), proceeds with the rest of the protocol. Timer expiration or reception of another or corrupt frame causes the handler to terminate. Existence of an ongoing transmission is determined by whether the reference to the handler thread is not null.

The procedures used in the implementation for receiving and transmission according the MAC protocol are described. The receiver, in between responding to an ongoing transmission, is tuned to the CCC. On receiving a frame, if it is not an RTS with this radio as the destination, the callback updates the NAV timers describing how long data channels are occupied. If it is, a handler thread for the upcoming transmission is spawned. The handler generates a CTS frame confirming the channel and other transmission parameters to be used and passes it to the transmitter for delivery. ATR automatically sets the transceiver into transmit operation, and returns it to receive mode once the frame has been sent. The receiver is then tuned by the handler to the data channel. The handler waits on a python condition until either timeout occurs or the receiver wakes the condition after receiving a frame. This frame is validated and if it is the DATA frame, the handler replies with the ACK frame and the decapsulated upper layer packet is added to `rx2_simple_mac_rtsctschannel_sensing`'s output buffer. If none, another or corrupt frame is received, the handler terminates.

Another thread is monitoring the input buffer for upper layer packets. Once there is no current ongoing transmission, a handler thread is spawned to transmit the packet at the head of the list. The thread decides on the channel to use. The strategy used for channel selection is the subject of the following chapter where a number are evaluated in a real-life demonstration. RTS is sent after the channel has been idle for DIFS following the NAV time having elapsed. CTS, DATA and ACK are received and transmitted using the condition handshaking model as demonstrated previously.

3.7 Summary

In this chapter the implementation of the physical CR system used in this thesis is recorded for reproducibility. The design of an ad hoc cognitive MAC protocol supporting channel switching to avoid PUs is presented. Its GNU Radio software implementation is then described with a basic supporting PHY layer comprising a power detector for spectrum sensing and transmission and DBPSK modulated data. The radio is used with the Q-learning channel selection scheme, presented in Chapter 4, the experimentally evaluate the effect on performance the intelligence has. The results in Chapter 7 form the research contribution of this thesis.

Chapter 4

Dynamic Channel Selection using Q-Learning

This chapter presents the ad-hoc Q-learning DCS scheme for improving the likelihood of successful packet transmission in CRs. In this chapter, we first examine related work on dynamic channel selection in Section 4.1. After presenting background material on reinforcement learning in Section 4.2, Q-learning is introduced in Sections 4.3 and 4.4 as an online model-free learning approach that simplifies design issues. Finally the Q-learning scheme and its implementation on our CR are described in Sections 4.5 and 4.6.

4.1 Dynamic Channel Selection

In Chapter 2, we described CR in terms of possessing reconfigurability, awareness and intelligence. CRs are defined by the ability to autonomously adapt transmissions to meet QoS performance requirements. A current research focus is in applying intelligence to dynamic channel selection so the SU is able to learn to transmit on the optimal channels maximizing its performance. A number of schemes have been considered in the literature with a common approach involving learning the perfor-

mance of different channels through trial and error selection, from which the estimated best behaviour is implemented.

In [60] learning is used so that the SU selects channels with reduced interference. At the start of each transmission cycle, the SUs sense the signal interference at each channel. The channels with the least cumulative interference over multiple cycles are selected with greater probability. Song et al [79] uses a stochastic channel selection algorithm. The SU keeps a running record of the probability of successful packet transmission per channel. The selection probability of each channel is increased if its successful transmission probability value is greater than the channel just used, otherwise decreased, thus the overall probability a channel is selected with a higher successful transmission probability increases. The algorithm is defined for a stationary system where the number of SUs and number and utilization of PUs are fixed. A simulation with one SU and five PUs shows the scheme converges so the channel with least utilization is exclusively selected. Song et al. [79] claim this is the optimal solution as any extra channel switching adds delays and degrades performance *when the least used channel has already been found*.

Instead of selecting the best quality channel per user, the scheme may consider maximizing the total utilization among all SUs in the network while minimizing potential interference to PUs, becoming a multi-objective channel allocation problem. Game theoretic approaches to channel allocation are presented in [99] and [68]. Nie and Comaniciu [60] formulate a game where the value placed on a channel is based on the sum of the total interference at the channel and the interference the user will cause to other SUs if it transmits on the channel. It is shown that the game converges to a Nash equilibrium solution where channels are allocated minimizing the total interference among SUs in the network. A protocol is proposed for exchanging interference information among users so allocation can proceed in a distributed manner.

The level of interference is enforced in policy based CRs [91]. A major

example is the xG Radio [54]. Licensed users accept opportunistic use of their bands and set hard bounds on the interruptions they will tolerate. Transmission requests generated by a policy based CR are vetted against a Policy Reasoner (PR) that checks policy conformance. Waveforms whose frequency or power exceeds what is regulated are rejected as illegal. In this thesis, we premise that the interference caused by the radio is managed by the MAC protocol being used. For instance, the IEEE 802.22 [26] protocol requires users to vacate a channel within 2s of a PU being detected. Our CR uses CSMA/CA to prevent collisions between SUs and performs sensing to avoid PUs before transmission, and creates a list of legal unoccupied data channels. Improvements to this area are suitable for future work, especially with respect to using co-operative sensing. Thus, our CR's dynamic channel selection task involves maximizing individual SUs' performance within the restrictions set by policy/protocol.

4.2 Reinforcement Learning

Any dynamic programming approach where the agent uses trial-and-error to learn, such as those considered, can be classified as a reinforcement learning problem. Reinforcement Learning (RL) is an online approach to learning where the optimal course of action is found by evaluating the scalar rewards, hence the name *reinforcement* learning, received in repeated interactions with the environment. RL DCS schemes for CRs are presented in [48] [42] [96] [97].

RL models the world as a Markovian finite-state discrete-time stochastic dynamic system as shown in Figure 4.1. In the system, at discrete time epoch t the agent selects an action $a_t \in A$ to execute while it is in one of a finite set of states $x_t \in X$. The action leads to the state of the system changing, which transitions to a new state $y = x_{t+1}$ with probability $P_{xy}(a)$,which is Markovian since it depends only on the present state. The agent is able to perceive the results of its actions, receiving a scalar reward

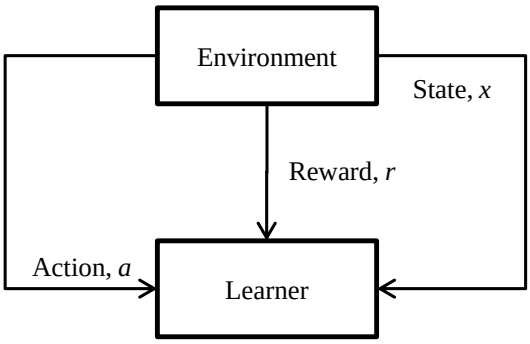

Figure 4.1: Finite-State Discrete-Time Stochastic Dynamical System Environment, from [59]

r.

The goal of RL is to discover an optimal policy π maximizing the long-term sum of rewards it receives, expressed by the value function

$$V^{\pi}(x) = E\{\sum_{t=0}^{\infty} \gamma^t r(x_t, \pi(x_t)|x_0 = x\}. \tag{4.1}$$

The policy $\pi(x)$ defines the action to be taken in state x, resulting in the reward $r(x_t, \pi(x_t))$. The value $0 \leq \gamma < 1$ is known as the *discount factor*, thus the value function represents the immediate reward plus the discounted reward of all future actions using the policy. The discount factor is used to tune the agent's reliance on future uncertain rewards and prevent the function from becoming infinitely equal.

This optimal policy is denoted π^* with value function

$$V^*(x) = \max_{\pi} V^{\pi}(x). \tag{4.2}$$

$V^{\pi}(x)$ can be rewritten as

$$V^{\pi}(x) = R(x, \pi(x)) + \gamma \sum_{y \in X} P_{xy}(\pi(x)) V^{\pi}(y). \tag{4.3}$$

In Equation 4.3, the expected discounted reward $V^{\pi}(x)$ is expressed as the sum of $R(x, \pi(x)) = E[r(x, \pi(x))]$, the expected reward for taking policy

action $\pi(x)$ in state x, and the expected discounted reward for following the policy π in the state transitioned into. This recursive form is known as a *Bellman equation*.

Rewriting the optimal policy equation in the same form produces

$$V^*(x) = V^{\pi^*}(x) = \max_{a \in A}[R(x,a) + \gamma \sum_{y \in X} P_{xy}(a)V^{\pi}(y)]. \tag{4.4}$$

Equation 4.4 is referred to as the *Bellman optimality equation*. It follows that the optimal policy's choice of action in state x is given by

$$\pi^*(x) = \arg\max_{a \in A}[R(x,a) + \gamma \sum_{y \in X} P_{xy}(a)V^{\pi}(y)]. \tag{4.5}$$

The significance is that if the optimal policy functions are known, the agent can maximize its cumulative discounted reward by selecting the action given in Equation 4.5.

4.3 Q-learning

Q-learning is a model-free reinforcement learning method. In Q-learning the optimal policy is learnt through a Q-value that is updated with the rewards of actions as the agent performs them.

The Q-learning algorithm maintains a table of Q-values, $Q_{(x,a)}(t)$, referred to as a Q-table. $Q_{(x,a)}(t)$ is the Q-value associated with state-action pair (x,a) at time t and is updated according to the rule

$$Q_{(x,a)}(t+1) = (1-\alpha)Q_{(x,a)}(t) + \alpha[r + \gamma \max_{b \in A} Q_{(y,b)}(t)] \tag{4.6}$$

whenever the agent performs action a in state x. In Equation 4.6, r is the reward received after the action was selected at time t, $Q_{(x,a)}(t+1)$ is the updated Q-value and $0 < \alpha \leq 1$ is the *learning rate*. The parameter α defines the emphasis the agent places on more recent reinforcement in learning over previous information. When $\alpha = 1$ the old Q-value is ignored in the

result, leading to one-shot learning. A greater learning rate improves the speed with which the agent is able to exploit changes in the environment but increases its susceptibility to noise.

So long as all actions are repeatedly sampled in all states, the Q-values will converge asymptotically to

$$Q^*(x,a) = R(x,a) + \gamma \sum_{y \in X} P_{xy}(a) V^{\pi^*}(y) \tag{4.7}$$

and the optimal policy followed by selecting

$$\pi^*(x) = \arg\max_{a \in A} Q^*(x,a) \tag{4.8}$$

which is equivalent to the definition in Equation 4.5. The proof of convergence is derived by Watkins and Dayan [90].

The agent needs to balance between greedily selecting the maximum Q-value, which is only its estimate for the optimal policy, and sampling other actions to ensure Q-value convergence. The two processes are respectively termed *exploitation* and *exploration* and form a *bandit problem* as is known in game theory. It is necessary to choose actions which are not the best so better ones may be discovered. Selecting the uncertain best action all the time, exploitation, will not improve the estimates of other actions.

Many ad-hoc exploration strategies have been proposed [96], such as the softmax approach. In our Q-learning DCS scheme, we retain the ϵ-greedy exploration used in [96] [97] so our results which are obtained in a real wireless test can be compared. In ϵ-greedy exploration the estimated optimal action(s), that is, possessing the highest Q-value, is chosen with probability $\frac{1-\epsilon}{l} + \frac{\epsilon}{n}$ where n is the number of available actions and l the number of best actions. $0 < \epsilon < 1$ is a small constant probability. Other actions are picked with probability $\frac{\epsilon}{n}$. In some strategies ϵ is slowly reduced if the system is stationary. However this does not make sense in channel selection when PU traffic patterns and noise characteristics are apt to

change, for instance in a mobile situation. Therefore in various CR situations we should not allow ϵ to reduce, or perhaps allow it to reduce to some preset limit or grow as the scenario changes.

A problem with the ϵ-greedy strategy is it is as equally likely to select a promising channel as it is to choose a channel with a track record of bad performance. In *Boltzmann exploration* channels are chosen probabilistically according to the distribution [84],

$$P_x(a) = \frac{e^{Q(x,a)/T}}{\sum_{b \in A} e^{Q(x,b)/T}} \tag{4.9}$$

which increases with the channel's Q-value estimate. $P_x(a)$ is the probability action a is selected in state x. The temperature T modifies the amount of exploration.

4.4 Evaluation of Q-Learning

Kaelbling et al [47] identifies three criteria for measuring the performance of a learning algorithm:

- Eventual convergence to optimal
- Speed of convergence to optimality
- Regret

Regret is the expected reward lost in learning compared to choosing the optimal behavior from the start. Two no-regret measures are widely defined in the literature [38]. The learning strategy is said to exhibit *no-external-regret* if the cumulative expected reward achieved by the actions selected by the learning algorithm is equal or superior to that obtained in the worst-case by any other fixed sequence of actions. If the expected rewards exceeds or is equal to that of any other strategy, obtained by replacing the

choice of action a made by the learning algorithm with a' for all actions a and a', the strategy shows *no-internal-regret*.

The advantage of Q-learning is that model-free online convergence to the optimal policy is guaranteed. Policy, value iteration and genetic programming require a model of the system with which to calculate the expected reward and fitness of potential behaviors. For instance based on the SINR sensed, if the bit error rate is used for the reward the resulting BER is established from an equation relating the two variables. This may be impractical given the complexity of the wireless environment. Sensing may be unable to capture all the data, such as fading and path loss effects, needed by the model and keeping the model up-to-date introduces severe sensing and computational overheads. Representing the model occupies memory. In contrast, given a reward scheme, Q-learning will converge to an optimal policy maximizing the expected reward within the actions available to it, in the limit that all behaviors are infinitely sampled. Using the current example, specifying BER as the cost metric, Q-learning will develop a strategy to minimize the BER. The policy is derived from trial and error association of the action with the BER at the time of sampling without requiring knowledge of the system mechanics of how BER is explicitly defined by the channel conditions. Q-learning is an online process and the individual Q-value update at the end of each action is not computationally complex. The algorithm does not need to keep a model of the environment in memory, only a Q-table equal to the number of states and actions.

Q-learning is known to converge unreliably when there are multiple agents [64]. A co-adaptation effect can be set up where agents continuously perceive a better strategy because of the effect of other agents' policies, never achieving a stable solution. An example is shown in Gomes and Kowalcyzk [70], where a game is set up with no Nash equilibrium. The Q-values of the agents is found to oscillate with time. Li [48] proves Q-learning will converge in the case of two non-communicating interfering

SUs choosing from two channels with fixed rewards and suggests using fictitious play to predict and compensate for other users' strategies.

Performance can be further differentiated by the speed of convergence. RL algorithms are slow to converge and in Q-learning Q-values converge exponentially quickly to their true values [90]. Convergence depends on all states and primitive actions being infinitely sampled. In MAC layer channel selection the agent has a fixed number of channels to explore but Q-learning does not scale well to waveform or packet design when *every* waveform parameter could be seen as needing to be sampled and decisions need to be made on continuous variables, such as the power setting. In this scenario RL requires human knowledge to make the problem tractable, such as state clustering to reduce the number of state-action pairs, or it may be necessary to model aspects of performance in place of actual exploration.

Despite the issue of convergence speed, Q-learning has been found in simulation to give significant performance improvements. Jiang et al [42] discuss a Q-learning DCS scheme. A successful packet transmission rewards a Q-value associated with the channel, and the channel to be used for the next transmission is selected based on the channel with the highest Q-value. The scheme is tested in simulation in a fixed system consisting only of SUs and the values learnt by the radios are retained from previous experiments. The authors show that by doing this 93% of users have found their optimal channel and do not need to move from it on next startup.

4.5 Q-Learning Dynamic Channel Selection Scheme

The aim of this thesis is to experimentally evaluate the performance of the Q-learning DCS scheme proposed by Yau et al [96] [97], which has only been tested in simulation. In this section, we present our modification of

the scheme for the experimental implementation. The Q-learning model is described in Table 5.1. Its goal is to learn the channel quality, which at the MAC layer is the proportion of packets that will be correctly received when sent using the channel. The responsibility of the MAC layer is the error-free adjacent-link transfer of frames. The scheme uses the learned results to select the best channel with the objective of maximizing the SU's probability of successful packet transmission.

Q-Learning Element	Description	Representation
Action	Action: data channel for packet transmission	$i \in \{1, 2, ..., N\}$
Reward	Constant positive reward value awarded to channel on successful packet transmission. Constant negative cost incurred on failure	$R = +RW$ (successful packet transmission), $-CT$ (failed packet transmission)

Table 4.1: Dynamic Channel Selection Scheme Q-Learning Model

The radio can choose from N data channels to transmit a packet using the cognitive MAC protocol defined in Figure 3.4. The data channels are denoted numerically and comprise the actions the agent can carry out The channel Q-value is updated at the end of the transmission attempt and receives a positive constant value reward $+RW$ on a successful packet transmission. As there is only one state $\gamma = 0$, thus the Q-value is updated according to

$$Q_i(t+1) = (1-\alpha)Q_i(t) + \alpha r, \tag{4.10}$$

where $i \in \{1, 2, ..., B\}$ was the channel used for transmission. The radio's MAC protocol always returns to the CCC after transmission. If there is an error in the data channel (interference causing either DATA or ACK to not

be received correctly) and the transmission attempt fails, a negative reinforcement cost with constant value $-CT$ is incurred. The aim is to learn which out of many heterogeneous channels subject to varying noise levels and PU utilizations is best suited for packet transmission, represented in the Q-value. Collisions in the CCC are not the fault of the data channel and do not receive any reinforcement.

CSMA/CA avoids interference between SUs in the data channel, hence the data channel packet transmission success rate is unaffected by other users' strategies. The multiagent co-adaptation effect is avoided and, from Equation 4.7, Q-value convergence is guaranteed to

$$Q_i^* = R(x, a) = (1 - P(A_i))CT + P(A_i)RW \tag{4.11}$$

on the condition that all channels end up infinitely sampled. Assuming fair link layer channel contention, the Q-learning convergence time should scale linearly with the number of SUs in the network. The Q-value is a weighted measure of $P(A_i)$, the successful transmission probability in channel i.

A simple analysis shows that after convergence has been achieved, the selection of the channel with the greatest Q-value estimate is a *no-external-regret* and *no-internal-regret* solution. The expected reward for substituting in a different channel or fixed selection of another channel is lower.

To ensure all channels are repeatedly visited, ϵ-greedy exploration is used as described in Section 4.3. The selection rule is as follows: the Q-table is consulted by the transmitter when it makes its channel decision prior to sending RTS. From the channels not currently in use by another SU, one of the l out of n overall channels with the greatest Q-value is selected with constant probability $\frac{1-\epsilon}{l} + \frac{\epsilon}{n}$, $0 < \epsilon < 1$. Other channels are chosen with probability $\frac{\epsilon}{n}$, thus the learning algorithm is no longer no-regret.

The scheme is selfish in the regard that the agent is seeking to maximize its packet transmission success rate and learning does not consider

potential interference to other users. Our scheme is designed to optimize SU performance within the collision management provided by the MAC protocol. The study of policy enforcement, restricting behavior so that effects such as interference are within regulation limits in policy based CRs, is a separate area of CR research and is an interesting problem for future work. The level of PU interference that arises during our scheme is studied in our experiments.

4.6 Cognitive Radio System Integration

This section describes the implementation of the Q-learning DCS scheme in the CR system developed in Chapter 3. A `channeltable` python class object, handling data structures involved with Q-learning, is instanced in `rx2_simple_mac_rtsctschannels_sensing`. The class contains fields for the Q-table, learning rate α and rewards. The Q-table is implemented as four arrays of the frequency and waveform parameters, Q-value and a channel integer reference number that collectively define a channel. The object is accessed through methods:

- `channeltable.getMaxQvalueChannel()`: returns the frequency, waveform and integer reference of a channel with maximum Q-value selected pseudo-randomly from the Q-table, using the python `random.randint` method.
- `channeltable.getRandomQvalueChannel()`: returns the frequency, waveform and integer reference of a channel selected pseudo-randomly from the Q-table, using the python `random.randint` method.
- `channeltable.increaseQvalue(channelId)`: updates the Q-value of the channel with integer reference `channelId` with reward RW in the case of a successful packet transmission according to the Q-learning scheme update rule.

- `channeltable.decreaseQvalue(channelId)`: updates the Q-value of the channel with integer reference `channelId` with cost CT in the case of a failed packet transmission according to the Q-learning scheme update rule.

Data channel selection occurs during the creation of the RTS frame by the transmitter in `rx2_simple_mac_rtsctschannels_sensing`. First a pseudorandom float in the interval [0,1] is generated using python `random.uniform`. If the float exceeds the exploration rate ϵ the protocol calls `channeltable.getMaxQvalueChannel()`, retrieving the channel with greatest Q-value, otherwise according to ϵ-greedy exploration `channeltable.getRandomQvalueChannel()` is called to obtain a random channel. The returned channel definition is passed in the RTS and used by both radios to set their data channels at the physical layer later in the protocol. At the end of the transmission, `rx2_simple_mac_rtsctschannels_sensing` calls `channeltable.increaseQvalue(channelId)` if it was successful or `channeltable.decreaseQvalue(channelId)` if it failed to update the Q-value.

4.7 Summary

This chapter has presented the Q-learning DCS scheme to be experimentally evaluated in this thesis. Our implementation of the Q-learning DCS has been shown to learn the channel expected proportion of packets that will be correctly received, expressed as a weighted sum in the Q-value, and selects the channel to maximize this. The advantages and disadvantages of Q-learning have also been discussed. The model-free Q-learning approach simplifies implementation, but only if performance is not impaired by the increased convergence time. We consider the evaluation of the presented Q-learning scheme on a real radio system a vital contribution for understanding whether the Q-learning approach is viable for CR.

If the performance impact of increased convergence time proves to be insignificant, Q-learning represents a possible solution to a truly *cognitive* CR capable of autonomously acting for itself without needing to specify *how* to achieve a task.

Chapter 5

Experimental Methods

This chapter discusses the wireless experiment developed to evaluate the Q-learning DCS scheme. A test scenario is formulated in Section 5.1, in which the performance of the channel selection scheme is measured through the progress of the CR in transferring data among licensed users. The thesis implements an experiment attempting to replicate an instance of the scenario, to within the capabilities of the GNU Radio and USRP hardware. The task of implementing the physical experiment is described in Section 5.2, which contains the design of GNU Radio PU nodes and necessary administration systems. The logging methodology is established in Section 5.2.4. Section 5.3 identifies the key performance metrics the experiment aims to evaluate. These may be referred to if the experiment needs to be reproduced and to determine how the results were obtained. Finally the wireless experimental setup is presented in Section 5.5.

5.1 Test Scenario

The wireless test scenario is designed to evaluate the Q-learning scheme's performance selecting among multiple data channels with different PU utilization. The scenario is summarized in Table 5.5, with experimental, timing and Q-learning parameters collated in Tables 5.6 and 5.7.

Category	Symbol	Details	Values
Scenario		Number of SUs	2
	n	Number of data channels	3
		Runtime	Scenario-specific (s)
		Interference model	Binary
Secondary users			
		SU traffic	Always backlogged
		Size of SU packet (DATA payload), l_{SU}	Fixed, scenario-specific (bits)
Data channels			
Primary users		PU traffic model	Stochastic channels with exponentially distributed ON and OFF times
	$t_{\text{PU},i}$	Packet duration	Fixed, scenario-specific (s)
	ρ_i	Utilization of each PU traffic	Scenario-specific
Channel quality	P_i^E	Packet error rate	Scenario-specific
Q-learning			
	α	Learning rate of Q-learning	0.2
	ϵ	Trade-off between exploration and exploitation	0.1
	γ	Discount factor	0
	RW	Reward	15
	CT	Cost	5

Table 5.1: Experiment Summary

The Q-learning scheme performance is measured by the progress of a pair of SUs in sending across a file over the experiment runtime. The SUs consist of the CRs we developed in Chapter 3, using the cognitive MAC protocol to exchange frames and utilizing the Q-learning DCS to select

from $n = 3$ channels in the system to transmit. The transmitting SU has data to send at all time, i.e. is always backlogged. The DATA frames sent each time are identical in size as link-layer considerations are beyond the scope of what the experiment is attempting to measure. The recipient SU has no packets to send. All channels are located in the 2.4-2.5GHz ISM band.

Each data channel i is shared with a GNU Radio device designed to represent a licensed user of the band. The packet-based PU traffic is modeled simplistically as an M/D/1 queueing process. Fixed-size packets arrive with exponentially distributed interarrival times, are enqueued for transmission and take a constant time to transmit based on the PU bitrate. The channel utilization is $\rho_i = t_{PU,i}\lambda_i$ where λ_i is the PU traffic arrival rate. There is no co-ordination between the PUs or SUs. Neither user is able to recognize each other's frame transmission protocols. Hidden users are avoided. All radios are located within one hop of each other and the signal generated at any one user interferes with the signal received at any other user, leading to both packets being lost according to this binary interference model. As the MAC transmission protocol and sensing are not under investigation and perfect sensing and an error free CCC are assumed, both of which end up being achievable in the experiments. Changing the reliability of the MAC protocol does not affect the relative performance of the scheme when compared to other DCS strategies, such as random channel selection.

The scenario thus consists of a stationary number of SUs and PUs, whose traffic utilization is also stationary, similar to [79] and [96]. We use the SU Q-learning parameters from [96] so that results can be compared. The test is conducted in a busy environment with other WiFi transmissions present. Each channel has an underlying packet error rate (PER) due to noise and interference effects originating from users sharing the ISM band, such as WLAN hotspots. All packets are sent without retransmissions and the outcome of the transmission attempt is final.

Experimental design involves balancing analytical tractability and resource availability with the generality of the result obtained given the assumptions or simplifications made. CRs are an emerging technology and the bands as well as licensed users with which they will be deployed have yet to be decided. This experiment provides performance results in a packet-based network, which are often modeled as Poisson processes [65] as is done here. Packet traffic at the network core has been found to slowly tend to Poisson [44], although the likely deployment scenario for CRs is little utilized bands at the network edge. Poisson processes are a classical approach for modeling voice networks and Kos and Bester [44] conclude that real packet systems can be reasonably modeled by equivalent Poisson packet traffic and improved by combining it with empirical traffic results. For TV or emergency frequencies, which are two bands of interest in SU network literature [26] [25], it is unclear how well this model applies. The usage level of a police dispatch radio would on inspection be expected to vary with the time of day and be prone to sudden bursts of activity. Wang and Salous' [88] analysis of the Global System for Mobile communications (GSM) band found that the spectrum occupancy pattern of an emergency radio band was best approximated by a seasonal autoregressive integrated moving average (ARIMA) model. Nonetheless, Poisson traffic models are widely used in CR spectrum management literature [89] [86].

A concern is whether the experimental results can provide any useful contribution given the small-scale of the network. The number of users makes maximum use of the restricted availability of GNU Radio devices and host computers, if one PU is to be available for each data channel. Generality is not lost by using a single agent. The multiagent case as shown previously affects the learning convergence time but not the stable solution. Analytical performance when there are many channels is derived in Chapter 6.

5.2 Scenario Implementation

5.2.1 Secondary Users

The existence of only one SU pair removes the need for the CCC to manage collision avoidance. In the experiment, the CR MAC protocol module `rx2_simple_mac_rtsctschannel_sensing` (refer to Section 3.6) is modified so RTS-CTS channel information is exchanged via sockets between transmitter and receiver processes. The DCS scheme performance results, in selecting data channels, are made independent of the quality of the CCC and particular MAC protocol used in the SDR implementation. Likewise MDTT and DIFS become unnecessary and are set to $t = 0$s to reduce the transmission cycle time, shortening the time required by the experiment.

The transmitting and receiving CRs run separate versions of `rx2_simple_mac_rtsctschannel_sensing`. The receiver is implemented by providing no packets at the `rx2_simple_mac_rtsctschannel_sensing` input buffer to send. To give backlogged packet traffic, the transmitter is set to loop transmitting a fixed link layer payload.

5.2.2 Primary Users

The PU is implemented in GNU Radio by making use of physical layer elements from the CR, under a new top-level python class `rx2_pu_tx` in place of `rx2_simple_mac_rtsctschannel_sensing` handling Poisson traffic generation. Class `rx2_pu_tx` initializes the existing frame receiver when it is instanced. The sensor and received are not used. Each PU makes use of a USRP mounting a single RFX2400 or XCVR2450 daughterboard for its RF frontend.

Class `rx2_pu_tx` loops, sleeping for an exponentially distributed time set by calling `random.expovariate` using the utilization interarrival

time parameters, with $\rho_i = t_{PU,i}\lambda_i$, then generates a MAC layer DATA packet which it adds to the frame transmitter input message queue. The CR frame transmitter reads the queue for data, which it transmits. All packets use a fixed payload and are generated as for a new MAC protocol transmission, with unique (incremented) sequence number.

In order to record the level of PU interference caused by the SU, a passive receiver is set up in each channel. It logs the PU packets it receives correctly using the convention in Section 5.2.4. The receiver is implemented in GNU Radio and uses a USRP frontend with either a single RFX2400 or XCVR2450 daughterboard. The top-level python class controlling the radio backend, `rx2_pu_rx`, creates an instance of the CR frame receiver and logs the received packets. So that PUs and SUs cannot recognize each other the two types of users implement different modulation schemes, with GMSK for PUs and DBPSK being used by the SUs. The simultaneous presence of a different user type appears as bit errors and signal interference.

5.2.3 Process Administration

To co-ordinate the system and automate parts of the experiment, a python program `master_server` was written. For convenience, `master_server` enables a batch of individually described runs to be described in code which it then automatically executes, for instance, to repeat the experiment with different SU utilizations or signal amplitude settings. The program uses `ssh` network protocol calls to each host computer address to start the SU and PU programs. The users do not begin transmitting but initialize all necessary flowgraphs and variables immediately up to this point. The user notifies `master_server` it is ready via a network socket provided by the program. Once all users have done so, `master_server` broadcasts a synchronized go order on all sockets. The SU transmitter records the time elapsed and once runtime is exceeded,

communicates this via socket exchange to `master_server` which then broadcasts stop orders to all programs.

5.2.4 Logging Methodology

The per packet transmission attempt results are logged to the hard disk by the SU transmitter and PU transmitter and receiver for experimental analysis. The SU's nth packet transmission attempt result is recorded in the logfile as a new data line,

t_1(n), t_2(n), outcome(n), ch_no(n), seq_num(n), qval(n), bytes_transmitted(n)

for future offline MATLAB processing. The description of the entries is as follows:

- t_1(n): time at which the DATA frame was enqueued at the frame transmitter.
- t_2(n): time at which the ACK frame was received (applicable to successful transmissions only)
- outcome(n): transmission outcome (0: unsuccessful because ACK was not received, 1: successful, 2: unsuccessful because signal was detected in *Listen-before-Talking* sensing).
- ch_no(n): Data channel number attempt was made on.
- seq_num(n): Sequence number of transmission.
- qval(n): Data channel Q-value at end of transmission.
- bytes_transmitted(n): Payload bytes if transmission was successful, otherwise zero.

The host machines are synchronized using the `ntpd` NTP (Network Time Protocol) client daemon. Additional metadata is generated at the head of the logfile. An example is shown below,

```
SUTX
Transmit Path:
Using TX d'board A: XCVR2450 Tx
Tx amplitude     24000.0
modulation:      gmsk_mod
bitrate:         250kb/s
samples/symbol:  2
interp:          256
Tx Frequency:    2.5G
Receive Path:
Using RX d'board A: XCVR2450 Rx
Rx gain:         0.0
modulation:      gmsk_demod
bitrate:         250kb/s
samples/symbol:  2
decim:           128
Initial Rx Frequency:    2.5G

Sensor Path:
Using RX d'board A: XCVR2450 Rx
gain:         0.0
decim:           128
Initial Rx Frequency:    2.5G

Power Sensor (Channel):
Channel Freq Width: 375k
N:     384
FFT Size:     512
```

```
;
01_50_33__07_02_2010
1265460555.9897289
350
0, 1, 2
2425000000.0, 2450000000.0, 2475000000.0
-1, -1, -1
;;


SUTX
Transmit Path:
Using TX d'board A: XCVR2450 Tx
Tx amplitude     24000.0
modulation:      gmsk_mod
bitrate:         250kb/s
samples/symbol:  2
interp:          256
Tx Frequency:    2.5G
Receive Path:
Using RX d'board A: XCVR2450 Rx
Rx gain:         0.0
modulation:      gmsk_demod
bitrate:         250kb/s
samples/symbol:  2
decim:           128
Initial Rx Frequency:    2.5G

Sensor Path:
Using RX d'board A: XCVR2450 Rx
gain:         0.0
```

```
decim:           128
Initial Rx Frequency:    2.5G

Power Sensor (Channel):
Channel Freq Width: 375k
N:     384
FFT Size:     512
```

In this first section, the physical layer transmission parameters in use are listed.

```
01_50_33__07_02_2010
1265460555.9897289
350
```

The next two lines record the date the run was begun. The second line is the UTC time since the Unix epoch, as generated by python `time.time()`, and t_1 and t_2 are offset from this value so they can be logged with higher precision. The third line comprises the run duration.

```
0, 1, 2
2425000000.0, 2450000000.0, 2475000000.0
-1, -1, -1
```

The final section of metadata contains a description of the data channel frequencies and the respective initial Q-values.

The logfile entries generated by the PUs follow the same format. Unlike the SUs, there is no protocol to tell the transmitter if its packet was successfully received. Instead, the transmitter logs when it sends its packet, which is treated as always successful, while separately the receiver logs the PU packets it correctly receives. The files are combined in MATLAB postprocessing to follow the SU standard, using the packet transmission and reception times for t_1 and t_2 and creating failed attempt entries for those packets with sequence numbers not recorded by the receiver.

5.3 Performance Metrics

This section defines the performance metrics used for analysis. In the Q-learning DCS, the strategy or scheme used for channel selection involves choosing the band with the greatest expected guarantee of packet transmission success. The Q-learning approach is utilized to learn the expected probability from experience. The analysis of the experiment results maintains the distinction.

5.3.1 Scheme Performance

The performance of the Q-learning scheme is measured by the SU packet transmission success probability, goodput and level of PU interference achieved in the experiment.

SU Packet Transmission Success Probability ($P(A)$)

The Q-learning scheme selects the channel with the estimated maximum expected probability any transmission attempt will be successful, unless exploration is required. The achieved probability comprises a performance metric, defined as the average SU packet transmission success probability across the logged results.

The average probability over log entries $[i, f]$ is calculated from the logfile by computing

$$\overline{P(A)} = \frac{|\{n \in [i, f] | \texttt{outcome(n)} = 1\}|}{|\{n \in [i, f]\}|}. \tag{5.1}$$

The average SU packet error rate is of course $1 - \overline{P(A)}$.

SU Goodput (G)

The goodput is defined as the useful payload bits correctly exchanged per unit time. The achieved experimental SU goodput is measured as the av-

erage goodput, computed from the logfile as

$$\overline{G} = \frac{8^* \sum_{n=i}^{n=f} \texttt{bytes\_transmitted(n)}}{t_2(f) - t_1(i)}. \tag{5.2}$$

The results will be specific to physical channel bitrates in the experiment. The Q-learning scheme considers only the MAC layer transmission success probability and will poorly optimize the goodput when channel bitrate is heterogeneous, for instance if the channel with low loss rate has lower bitrate. The metric is necessary to provide comparisons with DCS schemes that wait before transmitting.

PU Interference (I_j)

The PU interference caused by the SU is primarily determined not by the DCS but the sensing and collision avoidance techniques in the radio implementation. Useful conclusions can be drawn when evaluating the relative interference of other schemes.

We define I_j as the PU packet error rate in channel j due to SU interference. While this is not directly given by the logged outcomes, if the channel packet error rate due to noise P_j^E which is assumed to be independent is known, the average interference is given by,

$$\overline{I_j} = (\frac{|\{n \in [i,f] | outcome(n) \neq 1\}|}{|\{n \in [i,f]\}|} - P_j^E)/(1 - P_j^E) \tag{5.3}$$

from the PU logged data.

5.3.2 Q-Learning Performance

Kaelbling et al's [47] metrics for evaluating the performance of learning algorithms were stated in Section 4.4, where it was shown that Q-learning eventually converges to the optimal no-regret strategy within the scheme. The performance of the Q-learning algorithm is measured by the speed

of convergence. The online performance of the scheme and the change in scheme performance with time is analyzed to establish the speed of convergence to the optimal policy.

5.4 Alternative Strategies

The Q-learning scheme is analyzed based on its relative performance compared to other DCS schemes. Five alternative strategies are considered. Random channel selection, a rule-based DCS scheme and a no-regret Q-learning scheme are implemented on the radio by a simple modification of the `channeltable` selection method. Performance results are obtained by repeating the experiment with these algorithms. Two ideal strategies are formulated and set analytical upper bounds for the goodput. Their performance is inferred from the logged PU traffic.

5.4.1 Random Channel Selection

Random channel selection is used as a baseline DCS strategy. The transmitter selects the data channel uniformly at random from the set of available channels to use for the next packet transfer.

5.4.2 Rule-based Channel Selection

This rule-based strategy was analyzed in simulation alongside the Q-learning scheme in previous work [97] and was found to have significantly improved throughput over random channel selection. Channel selection in this scheme is determination by applications of the rules:

- The same data channel is used if the previous packet transfer is successful.
- Otherwise a channel is selected uniformly at random out of the other available channels.

5.4.3 No-Regret Q-learning (Best Channel) Channel Selection

This strategy selects the channel with the least utilization for transmission. The utilization is given to the scheme by the specification of the channel utilization in the scenario. If there is more than one channel with the least utilization either of these channels is selected uniformly at random. If the channel PER is negligible, the channel with the least utilization has the greatest expected successful transmission probability. The scheme gives the no-regret converged Q-learning strategy, except as the utilizations are known the Q-values can be derived from the start without requiring Q-learning to learn them during runtime or exploration.

5.4.4 Non-Deferred Ideal Strategy

This strategy assumes perfect knowledge of PUs' activities by the SU. If in the next transmission no PU is active while the SU is transmitting in the data channel, the scheme selects the first, ordered numerically, such channel. Otherwise a data channel is selected uniformly at randomly and the transmission attempt will fail due to the resulting interference. The scheme successfully exploits any whitespace intersecting with the next transmission.

The performance of this strategy is calculated based on the PU traffic observed in the equivalent experimental trial when run with the Q-learning DCS scheme. The traffic is reconstructed based on offline analysis of the PU logfiles, which give the time packets are enqueued at the frame transmitter, thus the times when the PU is transmitting is found by adding on an extra packet transmission time of activity starting from the enqueue time. A MATLAB function was written to step through applying the ideal DCS strategy. The total number of SU packets successfully transmitted is calculated from the sum of the number transmitted without SU interference in each channels j corrected for the independent PER by

multiplication with $1 - P_j^E$.

5.4.5 Deferred Ideal Strategy

This strategy is an extension of the non-deferred ideal strategy. If no available channel is found, the scheme suspends the packet transfer until a data channel is free from PU activity during the SU transmission time, rather than transmitting futilely. Consequently no PU interference arises from this strategy. Note that neither ideal strategy represents the theoretical upper limit to the number of packets that can be transmitted, which can be viewed as a packing problem to align the time packets are transmitted with the occurring whitespace.

5.5 Experimental Setup

In this section we describe the physical setup of the wireless experiment. The noise, SINR and physical layer parameters are recorded in enough detail for the experiment to be reproduced.

5.5.1 Overview

The test setup consists of five host computers which run GNU Radio 3.2 on NetBSD and are networked within the wider VUW ECS network. Computer specifications consist of 3.00GHz Intel Core 2 Duo CPU E8400 processors with 3.21GB RAM. The transmitting and receiving SUs are implemented on separate workstations. Each of the remaining three computers executes the PU receiver and transmitter processes for one of the three data channels and can be identified by having attached two USRP kits.

Initial design work and testing was performed with the radios in the setup arrangement connected over wireline. Signals were transmitted via RG858C/U 50Ω coaxial cable and two 1-8 2000-4200MHz ZB8PD-4-S+ power splitters were used to combine the user signals at the receivers.

Figure 5.1: Wireless Experimental Setup

Scarcity of daughterboards was an issue. For consistency the SUs and PU receiver USRPs mount XCVR2450 daughterboards, but because there was an insufficient number of either type, the PU transmitters use RFX2400 daughterboards. Even so there was an insufficient number of daughterboards to implement PU receivers in all three channels as well as implement the SUs and PUs, thus the 2.475GHz channel is not logged by any PU receiver device. Each USRP transmits and receives via a 3dBi VERT2450 dual band 2400-2480MHz and 4.9-5.9GHz vertical antenna.

5.5.2 Channel Setup

Three nominally 337.5kHz width data channels centred at 2.425GHz, 2.450GHz and 2.475GHz are arbitrarily chosen in the 2.4GHz ISM band. The common control channel is established at 2.400GHz. The main sources of wireless interference encountered in this band were 802.11 WLAN and Bluetooth devices. Thirteen 22MHz width channels in the 2.4000-

2.4835GHz band are defined for use by 802.11 Wi-Fi as illustrated in Figure 5.2. Channels {2,3,4,5}, {7,8,9,10} and {12,13} respectively overlap with the 2.425GHz, 2.450GHz and 2.475GHz data channels. The level of usage and channel allocation varies with the user.

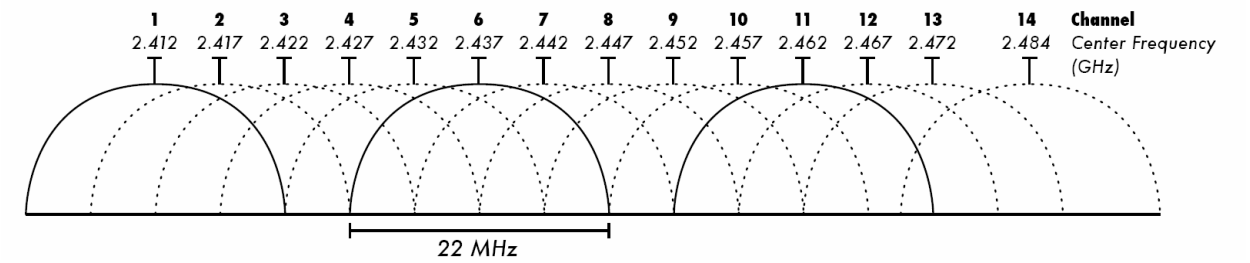

Figure 5.2: 2.4GHz band 802.11 Wi-Fi Channels, from [35]

The 2.4GHz wireless noise was observed over an interval of 24 hours using a Rohde&Schwarz FSL spectrum analyzer mounting a VERT2450 antenna. Figure 5.3 plots several traces including the average noise power. The *max hold* and *min hold* traces record the maximum and minimum observed power at each frequency over the time period. The *max hold* trace is dominated by multiple signals' profiles indicating there is significant band usage. These signals' SNR are in the order of 40dB as measured from peak to the average noise floor. The *single* trace is an instantaneous capture of the spectrum power and shows one WLAN channel in use. Using the spectrum analyzer's spectrogram function it was empirically found that other signals appeared as transient spikes, re-emerging within several hundred milliseconds.

In our experimental setup, SUs and PUs transmit respectively GMSK and DBPSK modulated packets with symbol rate $R_S = 500kS/s$ and bitrate $B = 250k$bit$/s$, corresponding to a USRP receiver decimation of 128 and transmitter interpolation of 256. Figures 5.4 and 5.5 show plots of the DBPSK and GMSK transmission in the 2.425GHz channel obtained off the spectrum analyzer. The RRC pulse shaping filter is used with roll-off factor $\alpha = 0.35$. This should theoretically give a signal baseband bandwidth

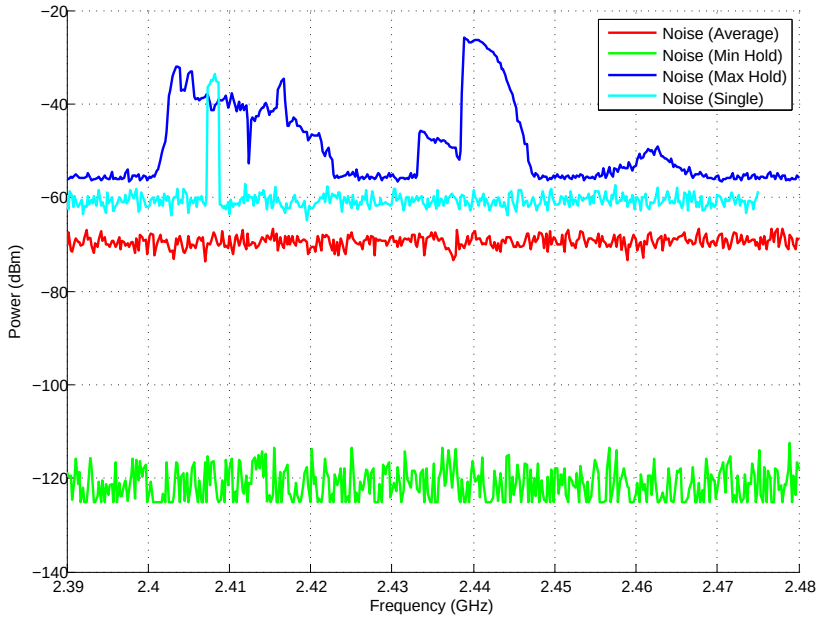

Figure 5.3: Channel Noise, observed over a 24 hour interval

W of

$$W = \frac{1+\alpha}{2} R_S = 337.5\text{kHz}. \tag{5.4}$$

In the introduction to GNU Radio it was stated that the maximum double-sided signal bandwidth that could be transmitted and received by the USRP was 8MHz. It was found that using a higher bitrate would cause frequent overruns in the PU, since the same computer must handle both receiver and transmitter processes and the receipt of a packet triggers writing to a logfile, thus the USB limit was not approached. Receiver gain is set to 0.

The XCVR2450 receiver sensitivity to XCVR2450 GMSK-modulated (SU) and RFX2400 DBPSK-modulated (PU) signals was measured. One thousand 944-byte packets were transmitted in the 2.425GHz channel and the bit error rate calculated, where the receiver and transmitter are cabled together to isolate the radios from outside interference. The transmit amplitude is varied across the full-scale range 0-32768 to adjust the SNR. Fig-

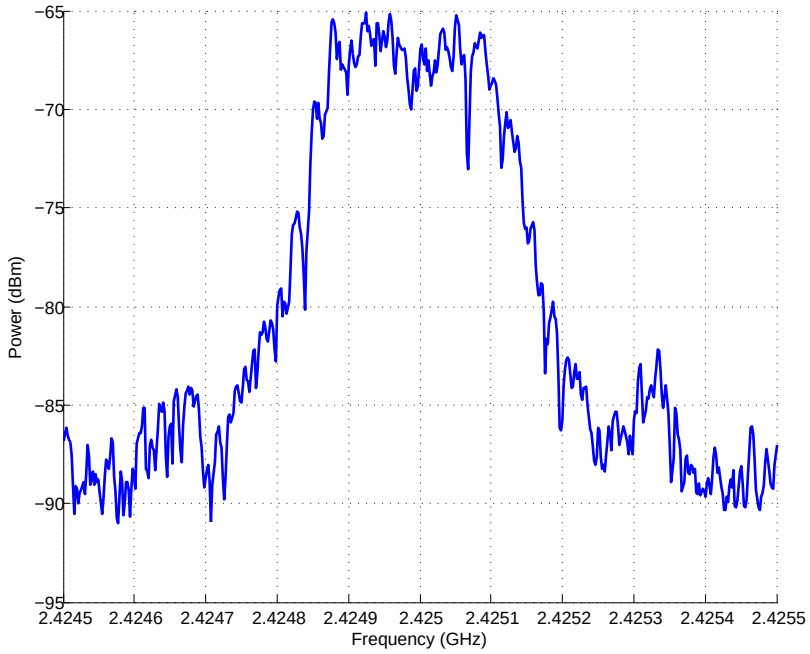

Figure 5.4: DBPSK modulated signal spectrum plot

ure 5.6 plots bit error rate (BER) against SNR, defined as the difference in the mean power of the signal in a 338kHz band centred around the channel frequency from that of the same band when the signal is absent, i.e. the level of the noise floor. The power is computed as the mean of the spectrum analyzer average power trace spanning the 338kHz band.

Referring to Figure 5.6, the PU signal bit error rate rises if the SNR is too high, which occurs because at higher amplitudes the USRP DAC goes into gain compression causing nonlinearities in the output signal reducing the effective SNR for correct demodulation [29]. The experiment signal amplitude values are chosen so the SNR inside the setup lies in the 30dB region where bit error rate is negligible. The average noise floor was observed to be flat and consistent. There was no significant difference in the noise floor between data channels or, later at the time of the experiments, when sampled at the beginning of separate runs. The SU transmit amplitude

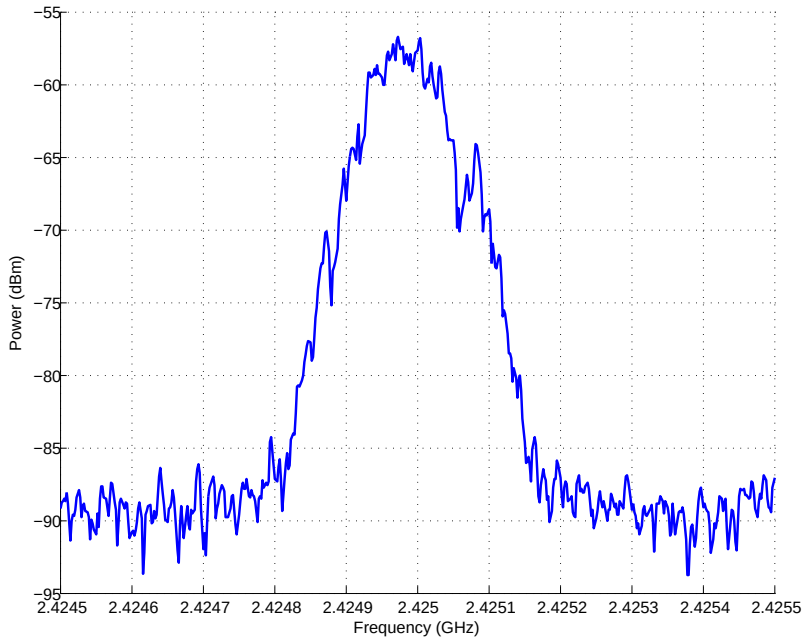

Figure 5.5: GMSK modulated signal spectrum plot

is set throughout to 9000 and 24000 for the PU transmit amplitude, with Table 5.2 showing the SNRs inside the setup and Table 5.3 the wireless bit error rate due to noise.

Transmitter	**Receiver**	**SNR(dBm)**
SU (TX)	SU (RX)	33.6 ± 1.5
SU (TX)	PU (RX)	29.8 ± 1.6
SU (RX)	PU (RX)	29.3 ± 0.5
PU (TX)	PU (RX)	33 ± 2
PU (TX)	SU (TX)	34.0 ± 0.7
PU (TX)	SU (RX)	31.9 ± 0.5

Table 5.2: Wireless Setup SNR Settings

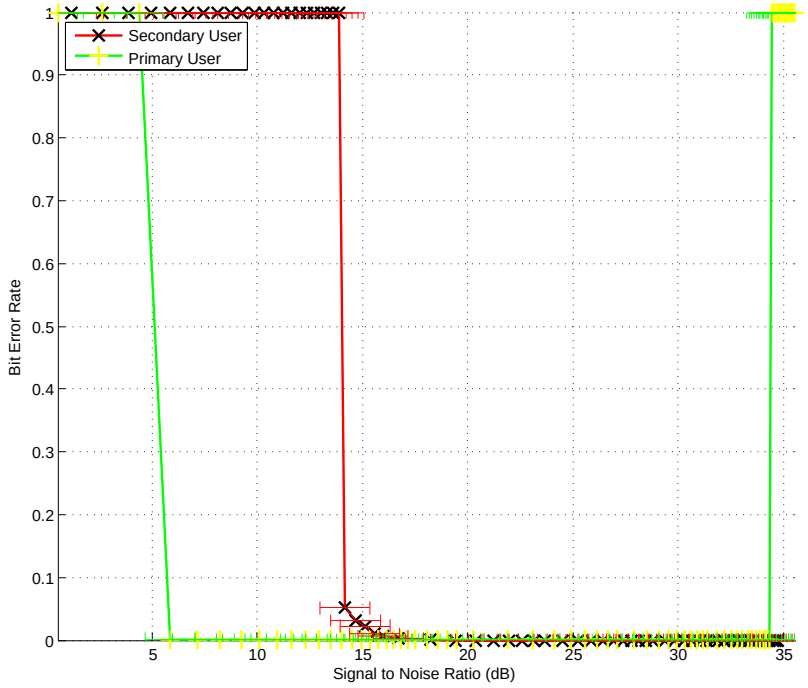

Figure 5.6: 2.425GHz SUTX-SURX and PUTX-PURX BER against SNR

Transmitter	**Channel (GHz)**	**BER(10^{-8})**
SU	2.400	20.81
SU	2.425	20.79
SU	2.450	20.80
SU	2.475	20.82
PU	2.400	2.081
PU	2.425	2.080
PU	2.450	2.079
PU	2.475	2.083

Table 5.3: Wireless Setup BER Settings

Under these SNR conditions, it was found that perfect detection could

be achieved simultaneously with negligible power detector false alarm probability. The power detector settings in the setup are described in Table 5.4 and are expressed in terms of the variables in Section 3.5. These were arrived iteratively. The value of $Q^{-1}(P_{fa})$ was increased until zero false alarm results were received after 10,000 sensing operations when no user signal was present. The values of $N \geq 512$, M were reduced while this standard was maintained to minimize sensing time and complexity without affecting performance. In this case the minimum values of M, N were obtained without the false alarm rate increasing. The value of $Q^{-1}(P_{fa})$ was then multiplied by a safety factor of 10. The probability of detection is 10,000 positive readings after 10,000 sensing operations when each PU signal was present.

Parameter	**Value**
USRP Decimation	128
N	512
K	384
M	1
$Q^{-1}(P_{fa})$	99
P_{fa}	$(0.0 \pm 0.5) \times 10^{-5}$
P_d	1.0
Sensing time, t_{SENS}(ms)	23
FFT double-sided bandwidth (kHz)	500
K-slice bandwidth (kHz)	375

Table 5.4: Wireless Setup Power Detector Settings

Regarding the notation in use when referring to channels by index, such as the packet error in channel i, P_i^E, the channels are numbered so the data channel at 2.425GHz is referred to as data channel 1, the data channel at 2.450GHz as data channel 2 and the data channel at 2.475GHz as data channel 3. This carries through when describing the

settings in use for the wireless scenario. "The experimental trial was run with PU channel utilizations [0.2,0.5,0.6]", and similar, should be taken as [ρ_1,ρ_2,ρ_3]=[0.2,0.5,0.6].

5.5.3 Timing Setup

Without programming the USRP FPGA, the userspace architecture used by the CR is unable to replicate the strict timing requirements of common wireless protocols such as IEEE 802.11. USB latency introduces a delay when a block of signal data is transferred across the interface given by [75]

$$\triangle_{\text{USBwait}} = \frac{f(512, \text{fusb\_nblocks}^* \text{fusb\_block\_size})}{\text{sample\_size}^* f_s}, \tag{5.5}$$

where $f(x, y)$ is the size of data at the USB buffer before a packet is sent, x being the least and y the most. The denominator gives the rate at which data is being accumulated at the buffer, with f_s the sampling frequency. The USRP Cypress FX2 controller requires at least 512 bytes before it will transmit a packet. The maximum is user specifiable. GNU Radio recommends `fusb_block_size`=1024 and `fusb_nblocks`=16 for realtime operations which we use. Further reductions produce unacceptable losses in the achievable data rate due to increased USB protocol overhead. Substituting in the decimation rate of 128 in use and `sample_size`=2*16 bits=4 bytes for the size of a complex float gives a sampling frequency of $f_s = (64MS/s)/(128) = 500kS/s$. It follows the minimum USB latency in the setup is 256μs. Schmid et al [75] found that the latency tended towards the maximum specified, in which case this would be 8.192ms. For comparison 802.11 SIFS (Short Inter Frame Space) interval, at 28μs, is two orders of magnitude smaller. Interprocess scheduling also introduces significant jitter. This matter is covered fully in the discussion on GNU Radio, Section A.4.2 in particular. Useful comparative performance results can still be obtained, even if the numerical goodput and other metrics achieved by the Q-learning, rule-based and random channel selection schemes do not reflect industry speeds.

The packet transfer times in the setup, as defined in the revised MAC protocol transmission timeline in Figure 5.7, are collated in Tables 5.5,5.6 and 5.7, which also provide a general summary of the experiment parameters. There is a significant delay, $t_{\text{SENS\_DATA}} = 16ms$, between when the transmitter completes *Listen-before-Talking* sensing and when the start of the DATA packet is physically transmitted by the radio. The time was measured by logging when the transmitter finishes sensing and the receiver decodes the packet, then subtracting from the difference the DATA packet transfer time. The delay is attributed to USRP latency and processing delays at the host computer. Similarly $t_{\text{DATA\_ACK}}$ is the time between receipt of the DATA packet and start of the physical transmission of the ACK packet by the receiving SU. The interval $t_{\text{DATA\_ACK}}$ was found to be 26ms with SIFS set to 1ms. These latencies are not negligible mandating their inclusion in Figure 5.7. The average delay observed over 3000 measurements is quoted.

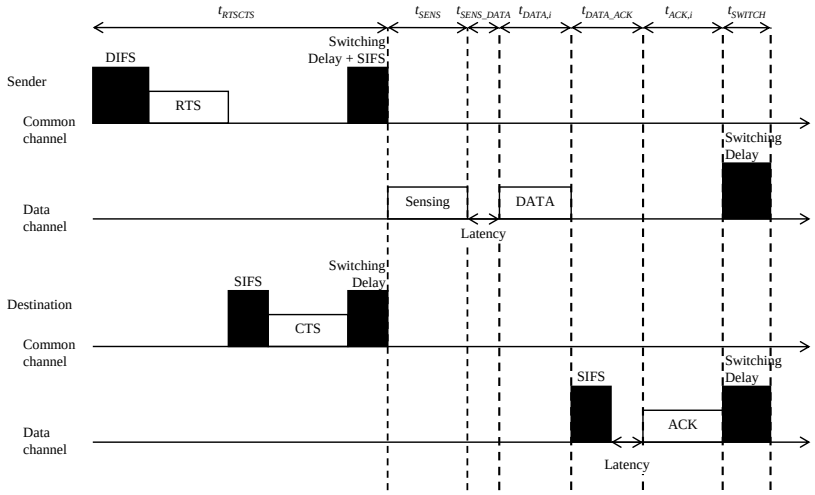

Figure 5.7: Cognitive MAC Protocol Transmission Timeline in Channel i

Symbol	**Details**	**Values**
	Number of SUs	2
n	Number of data channels	3
	Common control channel	2.400GHz
	Available data channels	[2.425GHz, 2.450GHz, 2.475GHz]
	Runtime	350s
	Interference model	Binary

Table 5.5: Experimental Setup

Symbol	Details	Values
Secondary users		
	SU traffic	Always backlogged
SIFS	SIFS duration	1ms
t_{RTSCTS}	RTS-CTS exchange duration	<1ms
t_{SWITCH}	Channel switching duration	12ms
t_{SENS}	Sensing duration	23ms
$t_{\text{SENS\_DATA}}$	Latency	16ms
$t_{\text{DATA\_ACK}}$	Latency	2.6ms
$t_{\text{ACK\_EXP}}$	ACK expiration timer	10ms
$t_{\text{SENS\_EXP}}$	Sensing abort expiration timer	35ms
B	Bitrate	250kb/s
l_{SU}	Size of SU DATA packet	944 bytes
	Size of SU ACK packet	40 bytes
$t_{\text{DATA},i}$	DATA packet duration	30.2ms
$t_{\text{DATA\_PADDED},i}$	Padded DATA packet duration	33ms
$t_{\text{ACK},i}$	ACK packet duration	1.3ms
$t_{\text{ACK\_PADDED},i}$	Padded ACK packet duration	16ms
$t_{A,i}$	Successful transmission cycle duration	110ms
$t_{B_1,i}$	Failed (data transfer) transmission cycle duration	191ms
$t_{B_2,i}$	Failed (aborted sensing) transmission cycle duration	191ms
Primary users		
	PU traffic model	Stochastic channels with exponentially distributed ON and OFF times
	Packet size	9376 bytes
$t_{\text{PU},i}$	Packet duration	300.0ms
$t_{\text{PU\_PADDED},i}$	Padded packet duration	311.3ms
ρ_i	Utilization of each PU traffic	[0.1,0.9]
Data channels		
$P_i^E(\text{DATA})$	SU DATA PER	0.0016
$P_i^E(\text{ACK})$	SU ACK PER	6.7×10^{-5}
$P_i^E(\text{PU})$	PU PER	6.7×0.0016

Table 5.6: Experiment Notations, Parameters and Timing Values

Symbol	Details	Values
α	Learning rate	0.2
ϵ	Exploration rate	0.1
γ	Discount factor	0
RW	Reward	15
CT	Cost	5

Table 5.7: Experimental Q-Learning Parameters

The experimental SU and PU data payloads are chosen to exaggerate the transfer times so the proportion of time spent transmitting against time wasted due to protocol overheads is realistic. An important distinction needs to be noted. The frame transmitter pads the PLCP encapsulated MAC frame to the nearest multiple of 512 bytes to avoid waiting at the USB interface. Intervals $t_{\text{DATA},i}$, $t_{\text{ACK},i}$ and $t_{\text{SU},i}$ are the times required to transfer at B=250kbit/s the SU DATA, ACK and PU frames, of size 944, 40 and 9376 bytes. The maximum 802.11 MAC frame payload is 2312 bytes and the PU frame is fragmented when it is transmitted. The frame data is padded by the transmitter to a total size of 1024, 512 and 9728 bytes, from before. Any other radios will suffer interference in the channels during the time need to transfer the padded frame size $t_{\text{DATA\_PADDED},i}$, $t_{\text{ACK\_PADDED},i}$ and $t_{\text{SU\_PADDED},i}$, although the receiver will already have obtained all of the frame if interference occurs during the padded section after the end. The PU packet transfer time is set an order magnitude greater than the SU DATA frame transfer time to be consistent with CR literature, which emphasize that SU keep licensed band transmissions short in order to quickly respond to the arrival of a PU.

The increased packet transfer times require longer run durations to observe the behavior of the DCS schemes. Various measures described were implemented to lower the transmission time. As there is only one SU pair, CCC co-ordination is redundant. MDTT and DIFS are dispensed with and

RTS-CTS is carried out via reliable socket information exchange, with the interval t_{RTSCTS} corresponding to the time required. This improves the generality of the performance results obtained for the DCS schemes: it is independent of the quality of the CCC which is now perfect. The times required for the different SU transmission outcomes in the setup shown in Table 5.6 are the median observed durations. The times exhibited a high degree of variability suggestive of the effect of process scheduling, following the delay measurement conclusions made by Nychis et al [61]. Table 5.8 gives the median, standard deviation, minimum and maximum transmission time measurements observed over 3000 repetitions.

Outcome		**Median**	**Std dev**	**Min**	**Max**
$t_{A,i}$	Transmission successful in channel i (*ms*)	110	52.0	78.35	493.4
$t_{B_1,i}$	Transmission fails in channel i - DATA or ACK packet not received correctly (*ms*)	191	5.0	179.4	208.3
$t_{B_2,i}$	Transmission fails in channel i - aborts in sensing because a PU is detected (*ms*)	190.8	5.7	179.0	210.1

Table 5.8: Transmission Outcome Duration Measurements

5.6 Summary

This chapter described the experimental setup used by this thesis to evaluate the Q-learning channel selection scheme. A wireless scenario replicating a possible network deployment of SUs was developed and suitable performance metrics defined. The logging methodology used to reliably record the results and implementation of other support systems were ex-

plained. Finally the specific timing and other experimental parameters were stated for reproducibility.

Chapter 6

Experimental Model

The performance of the SU when employing Q-learning channel selection in the wireless scenario, defined in Section 5.1, is derived analytically for a general number of data channels of different PU utilizations.

After establishing various preliminary results in Section 6.1, Markov chain analysis is used to find expressions for the SU's performance using Q-learning channel selection in Section 6.2 in terms of the long term packet transmission success rate and goodput. Although not necessary, the level of PU interference caused by the SU is calculated. Markov chains are a familiar tool in analysis and, while the present derivation is careful to explain the terms being used, a useful online reference can be found here [39]. In Section 6.2.4, we develop a discrete-time model of the SU's behavior and derive upper and lower bounds for the Q-learning convergence time within the scenario. Using the same metrics, the performance is derived for random channel selection in Section 6.3. Finally, in Section 6.4, setup implementation non-idealities are discussed that lead to the analytical performance being different to what is observed experimentally.

6.1 Preliminary Results

In this section, variables used in the analysis are defined and preliminary results for the SU packet transmission outcome in each data channel are derived. The scenario is defined in Section 5.1. We use the notation in Figure 5.7 and Table 5.6 for the scenario parameters, such as the MAC protocol SU packet transmission timeline. In the scenario, the backlogged SU transmits fixed size DATA and ACK packets, with transfer times $t_{\text{DATA},i}$ and $t_{\text{ACK},i}$ in data channel i depending on the bitrate. Intervals $t_{\text{SENS\_DATA}}$ and $t_{\text{DATA\_ACK}}$ represent small processing and timing delays in between. The PUs also broadcast only fixed length packets, lasting duration $t_{\text{PU},i}$ in channel i, which arrive at the input queue with exponentially distributed interarrival times. Our analysis assumes $t_{\text{PU},i}$ is longer than both $t_{\text{SENS\_DATA}} + t_{\text{DATA},i}$ and $t_{\text{DATA\_ACK}} + t_{\text{ACK},i}$, which is adhered to in the experiment, in order to simplify interference calculations. This is a valid generalization since secondary usage policy is to keep communications short to be able to quickly cease activities once a PU moves into the band.

In the scenario, the transmission attempt in channel i can have three possible outcomes with resulting durations:

- Transmission successful in channel i, A_i: $t_{A_i} = t_{\text{RTSCTS}} + t_{\text{SENS}} + t_{\text{SENS\_DATA}} + t_{\text{DATA},i} + t_{\text{DATA\_ACK}} + t_{\text{ACK},i} + t_{\text{SWITCH}} + t_{\text{MDTT}}$
- Transmission fails in channel i - DATA or ACK packet not received correctly, $B_{1,i}$: $t_{B_{1,i}} = t_{\text{RTSCTS}} + t_{\text{SENS}} + t_{\text{SENS\_DATA}} + t_{\text{DATA,i}} + t_{\text{ACK\_EXP}} + t_{\text{SWITCH}} + t_{\text{MDTT}}$
- Transmission fails in channel i - aborts in sensing because a PU is detected, $B_{2,i}$: $t_{B_{2,i}} = t_{\text{RTSCTS}} + t_{\text{SENS}} + t_{\text{SENS\_EXP}} + t_{\text{SWITCH}} + t_{\text{MDTT}}$

The interval t_{RTSCTS} is constant since with only one SU transmitter no additional time beyond DIFS is incurred in carrier sensing the CCC. The data channel is selected by the scheme and after RTS-CTS the radio senses the

channel. If a PU is detected the transmission is aborted, leading to outcome $B_{2,i}$. The radio waits sensing expiration timer $t_{\text{SENS\_EXP}}$ for the receiver to reset, switches back to the CCC and listens for t_{MDTT} for all other SU transmissions to finish before commencing another attempt. If both the ACK and DATA packets are received correctly, the transmission is successful (outcome A_i) and the transmitter returns to the CCC and waits t_{MDTT}. Otherwise, in outcome $B_{1,i}$ the transmitter waits expiration timer $t_{\text{ACK\_EXP}}$ for the ACK packet that never arrives before returning again.

The probability of each outcome is derived assuming the SU and PU states at the time of transmission are independent, i.e. there has been sufficient time in between a prior attempt on the channel so that whether the PU has a packet to send in its queue is unknown. This is more likely to hold the greater the number of channels to select from. Binary interference is assumed. Both SU and PU transmitting at the same time in the same channel results in both users' packets being lost.

The probability the data channel is sensed as clear, $P(s_i)$, requires there to be no ongoing PU packet transmissions prior to sensing. The PU is an M/D/1 system and this is equivalent to the probability there are zero packets in the system [40]

$$P_{0,i} = 1 - \rho_i. \tag{6.1}$$

Following this no new SU packets must arrive within the sensing period t_{SENS}. The probability there are no arrivals in time t is easily found for a Poisson process

$$F_i(0,t) = e^{-\lambda_i t}. \tag{6.2}$$

Thus

$$P(s_i) = P_{0,i} F_i(0, t_{\text{SENS}}). \tag{6.3}$$

The SU's DATA and ACK packet are sent on the condition sensing passes. The probability both are correctly received depends on no new PU packet

arriving and no independent packet errors occurring

$$P(p_i|s_i) = F_i(0, t_{\text{SENS\_DATA}} + t_{\text{DATA},i} + t_{\text{DATA\_ACK}} + t_{\text{ACK},i})(1 - P_i^E(\text{DATA}))(1 - P_i^E(\text{ACK})), \tag{6.4}$$

where $P_i^E(\text{DATA})$ and $P_i^E(\text{ACK})$ are the channel DATA and ACK packet PERs.

The probability of each outcome is thus

$$P(A_i) = P(p_i|s_i)P(s_i) \tag{6.5}$$

$$P(B_{1,i}) = (1 - P(p_i|s_i))P(s_i) \tag{6.6}$$

$$P(B_{2,i}) = P(p_i|s_i)(1 - P(s_i)) \tag{6.7}$$

6.2 Q-Learning Channel Selection Scheme

We proceed with the long term performance analysis of the Q-learning DCS scheme. The selection of channels can be described by a Markov chain with $n \times n$ transition matrix **M**, where entry m_{ij} is the probability channel j is selected for the current transmission that follows directly on the previous attempt being in channel i.

The Q-value for channel i converges to

$$Q_i^* = R(x, a) = (1 - P(A_i))CT + P(A_i)RW. \tag{6.8}$$

The long term performance is derived assuming the average channel Q-value is equal to the convergence goal, thus the performance models the SUs as having steady-state Q-values equal to the converged Q-values. Denote the set of l channels with the greatest converged Q-value as K. The transition matrix **M** of the SU is given by the rules for ϵ-greedy exploration

$$m_{kk} = m_{ik} = \frac{1 - \epsilon}{l} + \frac{\epsilon}{n} \tag{6.9}$$

$$m_{ij} = \frac{\epsilon}{n}, (k \in K, i, j \notin K). \tag{6.10}$$

The Markov chain is regular since the transition matrix has no zeroes, meaning it is possible to move directly from one state to any another state. The theory of regular Markov chains [39] states that the probability channel or state j is selected for transmission in the long-run approaches the value w_j, which is independent of the starting state. The unique $1 \times n$ vector

$$\mathbf{w} = (w_1, w_2, ..., w_n) \tag{6.11}$$

is known as the common row of the limiting matrix of M and can be found from the properties

$$w_1 + w_2 + ... + w_n = 1 \tag{6.12}$$

$$\mathbf{wM} = \mathbf{w}, \tag{6.13}$$

which give $n + 1$ equations in n unknowns.

For the Q-learning scheme Equation 6.13 gives

$$\left(\frac{1-\epsilon}{l} + \frac{\epsilon}{n}\right) \sum_{p=1}^{p=n} w_p = w_k, k \in K \tag{6.14}$$

$$\frac{\epsilon}{n} \sum_{p=1}^{p=n} w_p = w_j, j \notin K. \tag{6.15}$$

Thus

$$w_k = w_j \left(1 + \frac{n}{\epsilon} \frac{1-\epsilon}{l}\right). \tag{6.16}$$

Substituting into (7.2) gives

$$w_k = 1 + \epsilon \left(\frac{1}{n} - \frac{1}{l}\right) \tag{6.17}$$

6.2.1 SU Packet Transmission Success Rate

The vector **w** gives the steady-state channel selection probabilities of the scheme. It follows then in the steady-state that the general probability a transmission attempt is successful is given by

$$P(A) = \sum_{i=1}^{i=n} P(A_i) w_i. \tag{6.18}$$

6.2.2 SU Goodput

The goodput is defined as the useful payload bits delivered per unit time. The average transmission cycle time is

$$t_{\text{cycle}} = \sum_{i=1}^{i=n} P(A_i) t_{A_i} w_i + \sum_{i=1}^{i=n} P(B_{1,i}) t_{B_{1,i}} w_i + \sum_{i=1}^{i=n} P(B_{2,i}) t_{B_{2,i}} w_i. \quad (6.19)$$

The analytical steady-state SU goodput in bits per second is thus

$$G_{SU} = \frac{P(A) l_{\text{SU}}}{t_{\text{cycle}}}. \quad (6.20)$$

6.2.3 PU Interference (Packet Based)

The level of PU interference caused by the SU is derived. Define $B_{3,i}$ as the outcomes where a transmission in channel i fails with the DATA or ACK packet not being received correctly as a result of interference with the PU

$$P(B_{3,i}) = P(s_i)(P(d_i) + P(a_i)), \quad (6.21)$$

where

$$P(d_i) = 1 - F_i(0, t_{\text{SENS\_DATA}} + t_{\text{DATA},i}) \quad (6.22)$$

is the probability the DATA packet is not received correctly due to PU interference and the probability the ACK packet is not received correctly due to PU interference is given by

$$P(a_i) = (1 - P(d_i))(1 - P_i^E(\text{DATA}))(1 - F_i(0, t_{\text{DATA\_ACK}} + t_{\text{ACK},i})), \quad (6.23)$$

since the packet is not sent unless the DATA packet is correctly received.

The probability any SU transmission attempt interferes with the PU in channel i is given by

$$w_i P(B_{3,i}). \quad (6.24)$$

The attempt interferes with one PU packet, that being the earliest PU packet arrival during $t_{\text{SENS\_DATA}} + t_{\text{DATA},i}$ or $t_{\text{DATA\_ACK}} + t_{\text{ACK},i} < t_{\text{PU},i}$. The

SU will abort during sensing if the PU is still transmitting in a subsequent attempt, thus each interfered packet counted by the equation is distinct. Without the inequality, the number of interfered and overlapping PU packets is nontrivial since although he expected number of PU packets transmitted during any time interval t in channel i is

$$\frac{t\rho_i}{t_{\text{PU},i}}, \tag{6.25}$$

the PU is known to have no packets to send at the beginning.

The steady-state proportion of PU packets in channel i interfered with is therefore the expected number of packets lost due to SU interference in the channel per SU transmission cycle time, divided by the expected number of PU packets transmitted in the attempt duration, lasting t_{cycle},

$$P(I_{i,\text{pkt}}) = (w_i P(B_{3,i}))/\left(\frac{t_{\text{cycle}}\rho_i}{t_{\text{PU},i}}\right) = \frac{w_i P(B_{3,i}) t_{\text{PU},i}}{t_{\text{cycle}}\rho_i}. \tag{6.26}$$

Note that Equations 6.18, 6.20 and 6.26 are general expressions and only the steady-state vector $\mathbf{w}$ is specific to the scheme.

6.2.4 Speed of Convergence

The convergence speed is a prime concern when considering RL algorithms, which have a reputation for being slow. In this section we derive upper and lower bounds for the convergence time of the Q-learning scheme.

The expected discrete-time behavior of Q-values can be modeled by a system of difference equations according to Gomes et al [70]. The Q-learning update rule

$$Q_{a_i}(k+1) = (1-\alpha)Q_{a_i}(k) + \alpha r_i \tag{6.27}$$

can be rewritten as

$$Q_{a_i}(k+1) - Q_{a_i}(k) = \alpha(r_i - Q_{a_i}(k)). \tag{6.28}$$

The speed with which the Q-value for action a_i, corresponding to channel i, is updated depends on the frequency the channel is selected. Let $x_i(k)$ be the probability channel i is selected for transmission attempt k. As ϵ-greedy exploration is followed, $x_i(k)$ is given by

$$x_i(k) = (1-\epsilon) + \frac{\epsilon}{n}, \text{ if } Q_{a_i}(k) = \max_a Q_a(k) \tag{6.29}$$

$$x_i(k) = \frac{\epsilon}{n}, \text{ otherwise.} \tag{6.30}$$

The expected change in Q-values is directly affected by $x_i(k)$, which determines the rate of update, and is given by

$$Q_{a_i}(k+1) - Q_{a_i}(k) = x_i(k)\alpha(E(r_i) - Q_{a_i}(k)), \tag{6.31}$$

where $E(r_i) = (1 - P(A_i))CT + P(A_i)RW$ is the expected constant reward for the transmission attempt in channel i. The evolution in Q-values will differ between individual scenario runs due to stochastic variation in which particular action is selected per epoch. The trend observed over many runs is modeled.

The expected behavior of the Q-values is therefore modeled by the system of equations:

$$\begin{aligned} Q_{a_i}(k+1) &= Q_{a_i}(k) + x_i(k)\alpha(E(r_i) - Q_{a_i}(k)) \\ E(r_i) &= (1 - P(A_i))CT + P(A_i)RW \\ x_i(k) &= (1-\epsilon) + \frac{\epsilon}{n}, \text{ if } Q_{a_i}(k) = \max_a Q_a(k) \\ x_i(k) &= \frac{\epsilon}{n}, \text{ otherwise.} \end{aligned} \tag{6.32}$$

The probability $x_i = x_i(k)$ is constant until a different channel emerges with the highest Q-value. During this time Equation 6.32 is a first-order

linear difference equation which as well known has explicit form,

$$\begin{aligned} Q_{a_i}(m) &= (1 - x_i\alpha)Q_{a_i}(m-1) + x_i\alpha E(r_i) \\ &= (1 - x_i\alpha)^2 Q_{a_i}(m-2) + (1 - x_i\alpha)x_i\alpha E(r_i) + x_i\alpha E(r_i) \\ &... \\ &= (1 - x_i\alpha)^m Q_{a_i}(0) + x_i\alpha E(r_i)(1 + (1 - x_i\alpha) + (1 - x_i\alpha)^2 + ... + (1 - x_i\alpha)^{m-1}) \\ &= (1 - x_i\alpha)^m Q_{a_i}(0) + x_i\alpha E(r_i)\frac{1 - (1 - x_i\alpha)^m}{1 - (1 - x_i\alpha)} \\ &= A_i^m Q_{a_i}(0) + E(r_i)(1 - A_i^m), \end{aligned} \tag{6.33}$$

where $A_i = 1 - x_i\alpha < 1$, $B_i = x_i\alpha$, so

$$\lim_{m\to\infty} Q_{a_i}(m) = E(r_i)(1 - 0) = E(r_i), \tag{6.34}$$

i.e. the expected reward.

As shown by Equation 6.33 the Q-values converge in a piecewise exponential fashion. The rate of exponential convergence, A_i, is constant until points of intersection where the channel Q-value decreases or increases past the Q-value of the channel with the largest Q-value. This causes the value of x_i to change between the channel selection probability for exploration or exploitation, which determines A_i. The limit of the new difference equation remains the expected reward.

Convergence time is set by A_i. In the worst-case scenario $Q_{a_i}(k)$ is never the highest and the channel is shunned outside of exploration, thus $x_i(k) = \frac{\epsilon}{n}$ is constant. In this case the expected number of action epochs required for the contribution of the converged Q-value expected reward term to rise to proportion p and conversely the contribution of the starting Q-value to decay to proportion $(1 - p)$ is

$$\begin{aligned} A^m Q_{a_i}(0) + E(r_i)(1 - A^m) &= (1 - \rho)Q_{a_i}(0) + \rho E(r_i) \\ t_{\text{conv\_upp}}(p)\ln\left(1 - \frac{\alpha\epsilon}{n}\right) &= \ln(1 - \rho) \\ t_{\text{conv\_upp}}(p) &= \frac{\ln(1 - \rho)}{\ln(1 - \frac{\alpha\epsilon}{n})}, \end{aligned} \tag{6.35}$$

which defines an upper bound for the convergence time.

In the best-case scenario, the channel Q-value is always the largest, in which case the corresponding lower bound for the convergence time is

$$t_{\text{conv\_low}}(p) = \frac{\ln(1-\rho)}{\ln\left(1-\alpha\left(1-\frac{(n-1)\epsilon}{n}\right)\right)}. \tag{6.36}$$

The convergence time is shortened by increasing the learning rate, α. However, this reduces the stability of the Q-value, visible as the jagged sections in Figure 6.1, since the outcome of each individual action induces a greater response. The convergence time should scale linearly with the number of SUs assuming fair link layer channel contention.

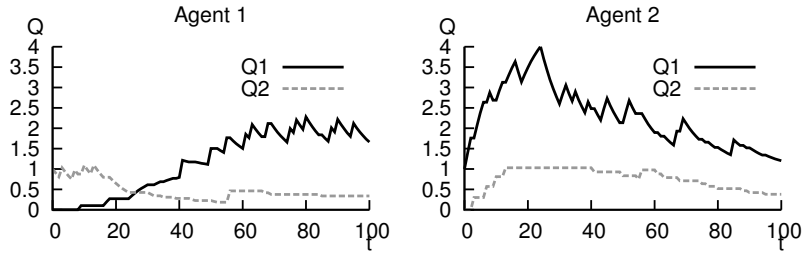

Figure 6.1: Example of the change in Q-values in an individual run applying Q-learning to the Prisoners Dilemma, from [70]

6.3 Random Channel Selection Scheme

The SU performance using the random DCS scheme is derived. In this scheme, data channels are selected with equal preference, thus the transition matrix M is still regular and is now given by

$$m_{ij} = \frac{1}{n}, (i, j \in \{1, 2, ..., n\}). \tag{6.37}$$

The $1 \times n$ common row of the limiting matrix of M, w, trivially evaluates to

$$w_i = \frac{1}{n}, (i \in \{1, 2, ..., n\}) \tag{6.38}$$

and the packet transmission success probability, goodput and interference found by substituting $\mathbf{w}$ into the general expressions given in Equations 6.18, 6.20 and 6.26.

6.4 Experimental Non-Idealities

The analytical model is derived for the scenario given in Section 5.1. Benchmarking of the experimental setup showed however that it was not able to perfectly realize the scenario. In particular at higher utilizations the actual PU channel utilization was less than the desired value, e.g. 0.9, and no longer approached a Poisson traffic distribution.

The traffic of the PUs was logged over 1000s at nominal set utilizations of 0.1,0.2,...,0.9. Figures 6.2 and 6.3 plot the packet interarrival and waiting time distributions in the 2.425GHz channel. The waiting time is the time a packet spends in the queue before being transmitted, and is calculated by subtracting the constant packet transfer time from the logged reception and enqueue time. The theoretical M/D/1 system distributions are plotted alongside, substituting in for the utilization value the actual utilization calculated from the number of PU packets generated at the transmitter during the 1000s interval. The value used is indicated in the legend. The analytical waiting time distribution for an M/D/1 system is given by Crommelin (1932) as

$$P(W \leq x) = P_0 \sum_{k=0}^{m} \frac{\{-\lambda(x-mD)\}^k}{k!} e^{-\lambda(x-mD)}, mD \leq x \leq (m+1)D \quad (6.39)$$

where is λ the traffic arrival rate, x the waiting time, D the service time ($t_{\mathrm{PU},i}$ in our case) and $P_0 = 1 - \rho$ the stationary probability of the system containing no more than 0 customers.

While the interarrival time between packets is exponentially distributed as expected the waiting time deviates from the M/D/1 model. In 6.3, the experimental waiting time is longer at higher utilizations. This

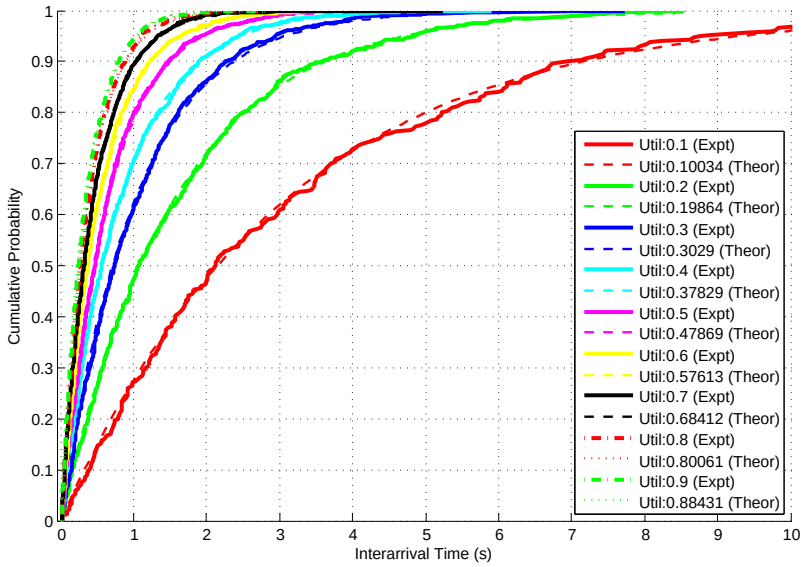

Figure 6.2: CDF of observed and theoretical PU packet arrival times

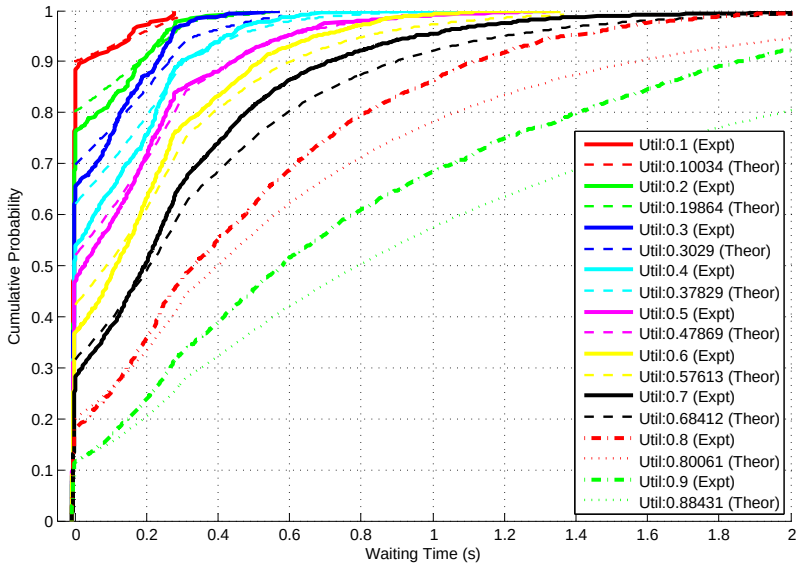

Figure 6.3: CDF of observed and theoretical PU waiting times

could not be corrected in our userspace radio implementation. As shown in Table 5.8 inherent latency between packet transmissions is introduced

by the USB interface and interprocessing delays, which would contribute to the waiting time distribution being non-ideal. An effort was made to reduce other delay sources. Various optimizations were made to the python code controlling the MAC layer, such as removing error checking code in the final version used in the experiment and use of more efficient library data structures. However no significant change to the waiting time distribution was observed.

This required changes to the analytical model. The M/D/1 system assumption is used to develop the preliminary results in Section 6.1, where a Poisson interarrival rate and fixed MAC protocol timing requirements are employed to derive the SU transmission outcome probabilities in each channel, $P(A_i)$, $P(B_{1,i})$, $P(B_{2,i})$. Otherwise the Q-learning channel selection scheme SU packet transmission success rate, goodput and interference in Section 6.2 and dynamic model in Section 6.2.4 are found from a pure Markov chain and discrete-time analysis. $P(A_i)$, $P(B_{1,i})$, $P(B_{2,i})$ and $P(B_{3,i})$ are used as terms to populate the Markov chain and the equations, such as the expected reward, in these sections without any underlying requirement that the system is M/D/1.

The values of $P(A_i)$, $P(B_{1,i})$ and $P(B_{2,i})$ for the setup were found experimentally. With the SU set to only being able to select from one data channel, the scenario was run for 1000s. This was repeated for each channel, with the PU utilization of the channel raised in steps of 0.05 from 0 to 1.0. Figures 6.4, 6.5 and 6.6 plot the observed success ($P(A_i)$), failure ($P(B_{1,i})$) and abort ($P(B_{2,i})$) probabilities in each channel against PU utilization alongside what is expected analytically in Section 6.1. The experimental results are graphed against the actual utilization during the run based on the number of PU packets logged as being transmitted, rather than that specified in python. Compared to an M/D/1 system the probability of failed transmissions is higher throughout compared, while the successful transmission probability is less. Interestingly, the data channels exhibit identical probabilities so the channel quality is homogeneous for

SU packet transmission, which is confirmation of the measured channel BERs in Table 5.3 and that PU-SU interference is binary in all channels.

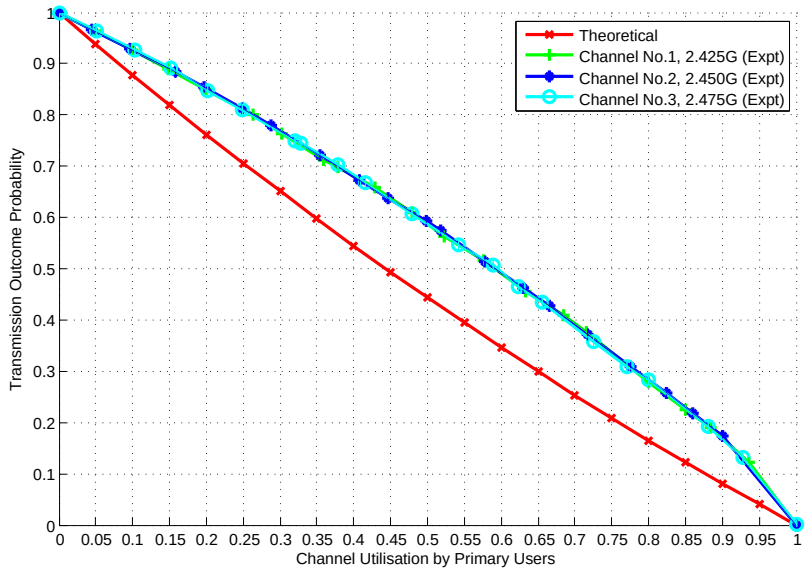

Figure 6.4: Per channel SU successful transmission probability, $P(A_i)$, against PU channel utilization

The empirical probabilities are used to derive the analytical results in Chapter 7. For instance the expected reward supplied to the Markov chain for selecting channel 1 with utilization 0.1 is calculated using the plotted probability of success in Figure **??**, which is representative of the setup implementation.

6.5 Summary

Two models have been derived in this chapter. The channel selection behaviour of the Q-learning scheme was represented as a Markov chain. Using this expressions for the long-term goodput and packet transmission success rate of the SU were found for a general number of users in the experimental setup. A second discrete-time model of the change in Q-values

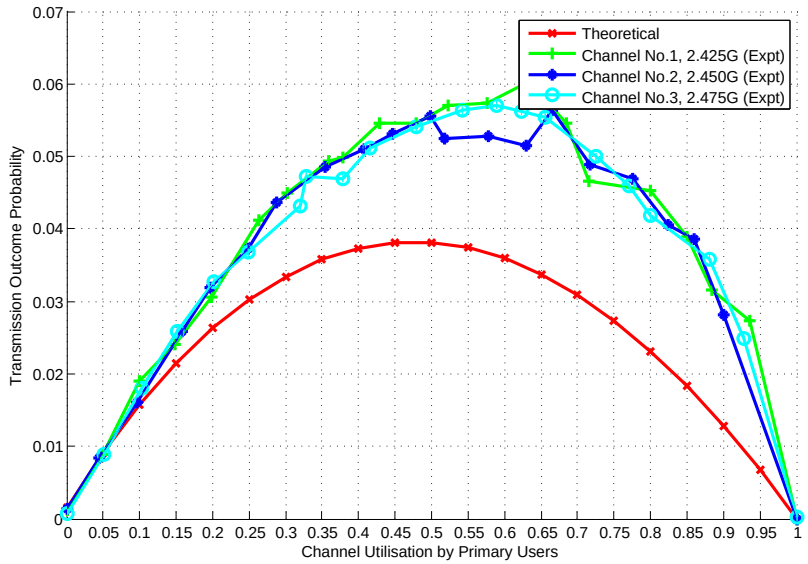

Figure 6.5: Per channel SU failed (data) transmission probability, $P(B_{1,i})$, against PU channel utilization

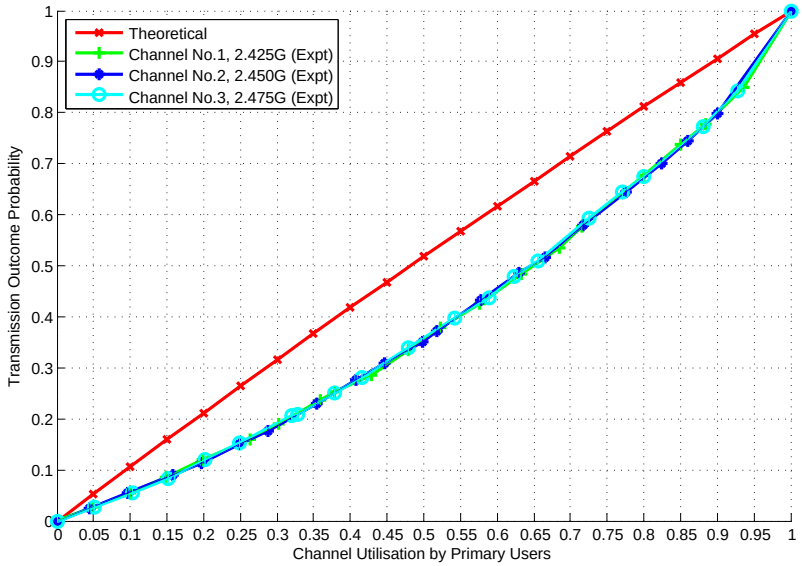

Figure 6.6: Per channel SU aborted (sensing) transmission probability, $P(B_{2,i})$, against PU channel utilization

in the scheme over time is developed and which predicts the convergence time of the scheme. Experimental results are used to validate the analytical models in Chapter 7.

Chapter 7

Results

This chapter presents the results of the wireless experiments. This chapter has two main goals. Firstly, the quality of learning of the Q-learning DCS scheme is assessed. Based on Kaelbling's [47] criteria given in Section 5.3.1, the speed of convergence of the Q-learning algorithm and performance within a given time is measured in experiment. Secondly, the scheme is compared against ideal and alternative random and rule-based DCS strategies in terms of packet transmission success probability, SU goodput and PU interference, which are identified as the key testable performance metrics in Chapter 4. The predictions of the analytical model are tested against the experimental results, in order to verify its validity and whether it can be used to estimate how the scheme behaves in a larger realistic network scenario.

The Q-learning scheme performance is measured with the parameters in Table 5.6 in use. The effect of changes in the Q-learning settings on the performance of the DCS scheme is studied in [97] in simulation. Repeating this examination here would not produce novel results, besides verifying reproducibility on a physical system. Collecting performance when different values of RW and CT and learning and exploration rates α and ϵ are used is quicker to achieve by adjusting the parameters in a simulation.

This chapter is structured as follows. Single runs with different utiliza-

tions are used to evaluate the dynamic behavior of the Q-learning scheme in Section 7.1 for the scenario with three data channels. The time for the strategy to converge is found to be six transmissions based on the successful packet transmission rise time, but this is long in comparison to the expected three transmissions required by rule-based selection to randomly sample all channels for whitespace and if found, exploit it. A number of results are found for the transmission success, goodput and interference performance of the various schemes in Section 7.2. Taking the average result from running the scenario using all valid combinations of individual channel utilizations in the set $\{0.1,0.2,0.3,\ldots,0.9\}$, averaging to $\{0.1,0.2,0.3,\ldots,0.9\}$, the Q-learning approach enhances the proportion of SU successful packet transmissions by 39.9% and goodput by 56% compared to random selection of channels. Performance is significantly inferior to both ideal bounds and the rule-based scheme. *Listen-before-Talking* sensing is unable to prevent packets being lost by the PU due to secondary usage and up to a 39.09% loss rate is observed. In Section 7.3 the experience of working with the GNU Radio platform is recounted.

7.1 Q-Learning Performance

7.1.1 Speed of Convergence

The convergence behavior of Q-learning in the DCS scheme was examined. The wireless scenario was run with PU utilizations [0.9,0.7,0.2] and the observed change in SU Q-values compared to analytical results. A discrete-time model is derived in Chapter 6 for the expected behavior of channel Q-values from initial starting Q-values, but lacks verification. The experiment was repeated 50 times and the median Q-value per channel at each action epoch plotted. This is to reduce the effect of the random selection of channels which varies between trials per epoch, or *noise*, and observe the trend in the Q-values. The median is employed as it

is less sensitive to outlier values in comparison to the mean for example. The starting Q-values are initialized with distinct values so the initial choice of channel is biased and not random which would require more results to be aggregated. Three combinations of initial Q-values are used, $Q_0 = [0, 5, 10], [0, 10, 5], [10, 5, 0]$ corresponding respectively to in, partially and out of order of the ranking of the predicted final expected channel Q-values, for a total of 150 wireless scenario runs each of 350s duration summing to an execution time of 14.6 hours.

Figures 7.1,7.2 and 7.3 plot the 50-run median Q-values against the SU transmission attempt, or *action epoch*, for each of the three cases. Graphed alongside are the analytical expected Q-values found by solving for the difference equation given in Equation 6.32. As predicted by the model, the experimental Q-values converge exponentially and asymptotically to constant final values. There is no significant difference within the noise uncertainty between the experimental asymptotic Q-values and the expected reward for transmitting on the channel, $E(r) = [-3.38, -0.08, 10.24]$ according to Equation 6.8, which should analytically be the convergence targets.

Figure 7.4 shows a typical example of the change in Q-values during a single run using $Q_0 = [10, 5, 0]$. The picture is radically different to the expected asymptotic exponential convergence in the previous figures. Instead the Q-values fluctuate wildly. It appears though that the Q-values end up oscillating about the expected reward, which is confirmed by the median results, thus the frequency of positive Q-value changes due to the rewards from successful transmissions and the frequency of negative changes from failed transmissions reaches equilibrium based on the channel utilization. Smooth convergence to constant values could be achieved by modulating the oscillations by steadily reducing the learning rate α with time.

The channel the SU has selected at a certain action epoch, for the same single run, is plotted alongside the channel Q-values in Figure 7.5. A point

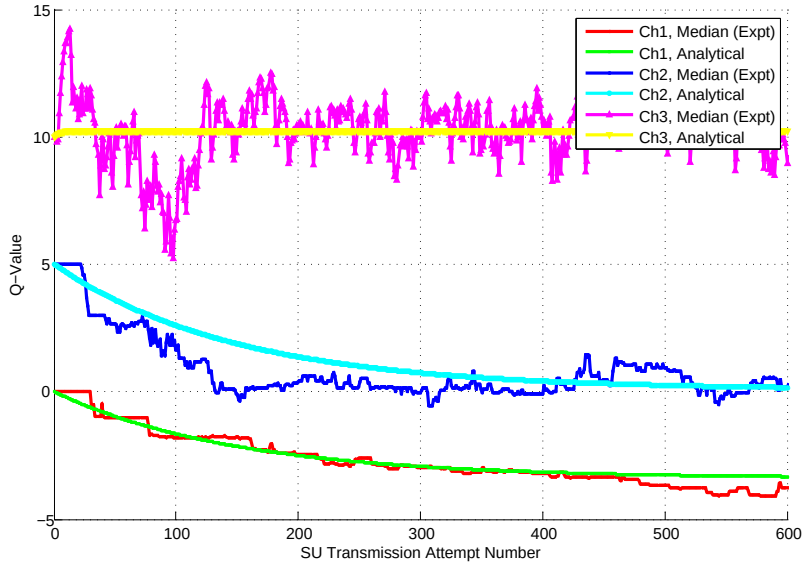

Figure 7.1: Median Q-values against the SU transmission attempt number, for channel utilizations: $[0.9, 0.7, 0.2]$, $Q_0 = [0, 5, 10]$

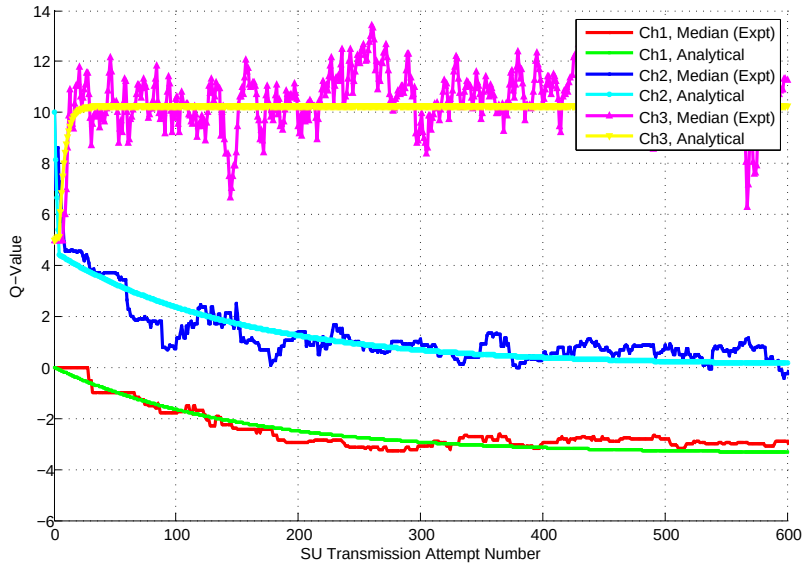

Figure 7.2: Median Q-values against the SU transmission attempt number, for channel utilizations: $[0.9, 0.7, 0.2]$, $Q_0 = [0, 10, 5]$

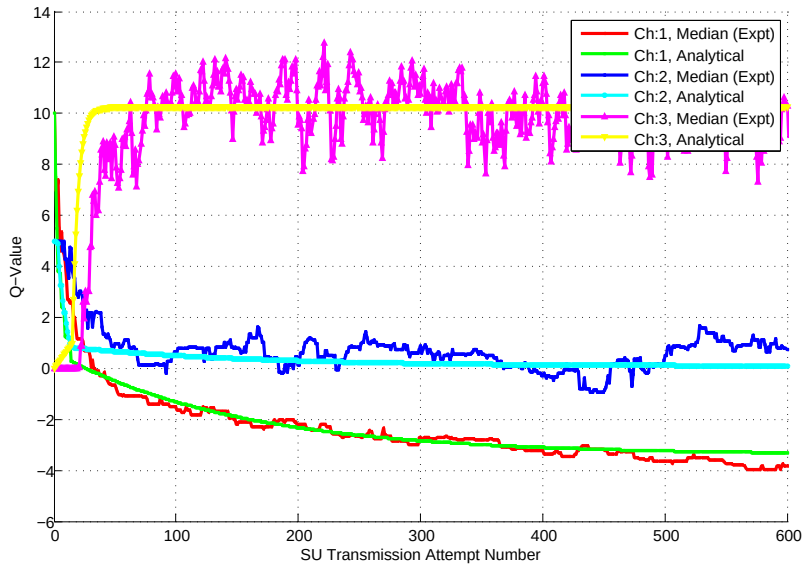

Figure 7.3: Median Q-values against the SU transmission attempt number, for channel utilizations: $[0.9, 0.7, 0.2]$, $Q_0 = [10, 5, 0]$

is marked at the predicted converged Q-value level of the channel, if the SU selected the channel for the transmission attempt. The channel being used can be indirectly found from the plot of the Q-values, in which case it is indicated by the value corresponding to the channel changing after the time it was selected. However by directly plotting the selected channel, Figure 7.5 provides a clearer analysis. The Q-values of channels 2 and 3 are often constant for extended intervals since, having lower Q-values their only opportunity to be selected is through random exploration with probability 0.033.

Figure 7.5 indicates the Q-learning DCS scheme is not effective at exploiting correlation between transmission attempts. This is shown in the figure at epoch 347 where the radio continues to use channel 1 for 7 consecutive attempts despite each try being unsuccessful as, although its Q-value estimate is degraded at each epoch, it still exceeds the other channels'. The scheme should recognize the channel is under sustained use by the PU, i.e.

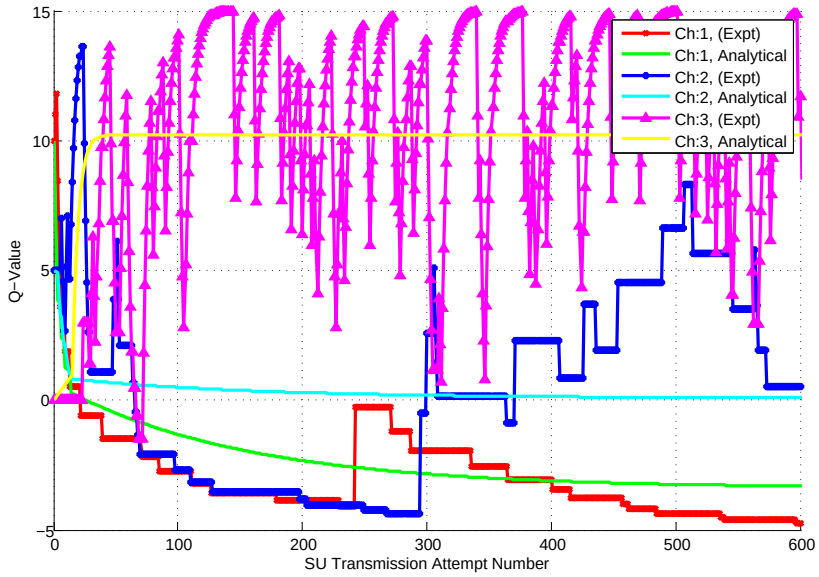

Figure 7.4: Single run Q-values against the SU transmission attempt number, using channel utilizations: $[0.9, 0.7, 0.2]$, $Q_0 = [10, 5, 0]$

is transmitting a packet train, and that exploitation will be fruitless during this time. Similarly, exploration of other channels sometimes produces a successful result but the scheme makes no further attempts to exploit it when the channel is now known to be free. Instead, the SU is most likely to revert to channel 1 with probability 0.93. Increasing the learning rate would reduce the time before the radio turns to an alternate channel if a long period of interference is experienced. The greater the learning rate, the more emphasis is placed on recent occupancy or quiescence within the channel in the Q-value rather than the long-term average channel utilization. Typically, in real packet based wireless system noise and activity occur in busts, making this ability to react to local variations in channel conditions essential.

The discrete time model is used to derive the convergence speed of the Q-learning algorithm. This is determined by how often channels are selected in ϵ-greedy exploration giving two convergence time bounds, which

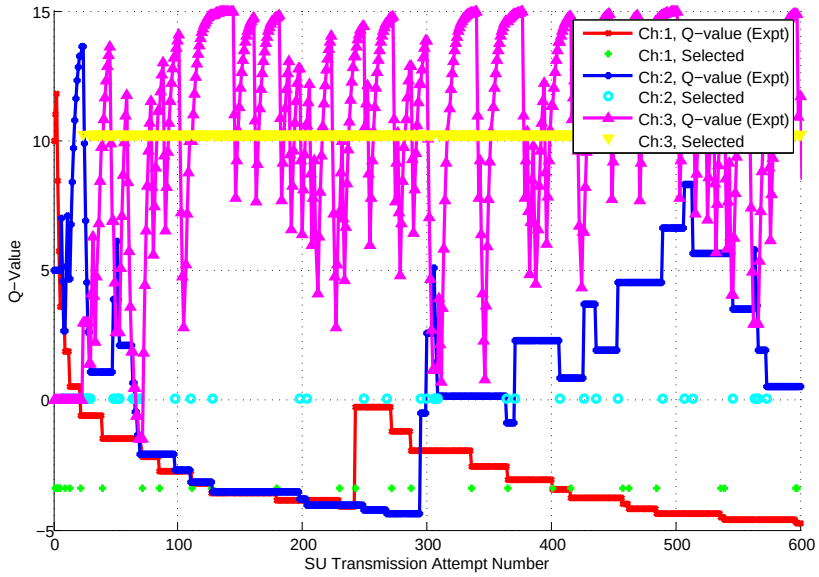

Figure 7.5: Single run data channel selected by the SU against the SU transmission attempt number, using channel utilizations: $[0.9, 0.7, 0.2]$, $Q_0 = [10, 5, 0]$

are given in Section 6.2.4. Substituting in the Q-learning parameters in use, stated in Table 5.6, 95% convergence will occur in the best-case within $t_{\text{conv\_low}}(0.95) = 14.5$ transmission attempts if the channel is selected for exploitation throughout, or equivalently has the highest Q-value throughout. In the worst-case the channel is selected only randomly during exploration and 95% convergence takes place after $t_{\text{conv\_upp}}(0.95) = 447$ transmissions attempts and sets an upper convergence bound.

The channel selection policy is decided by the ranking of channels by their Q-value estimates. The expected converged channel selection policy is to exploit the channel with least average utilization, which is learned as soon as its Q-value channel estimate exceeds the other channels'. In the scenario, Equation 6.33 predicts the Q-value of channel will rise above those of the other channels after four transmission attempts when $Q_0 = [0, 10, 5]$ and 14 attempts when $Q_0 = [10, 5, 0]$. The intersec-

tion point where the Q-value of channel 1 exceeds the other channels' is observed after attempt numbers 7.5 and 27 respectively in the experimental median results. Its Q-value then quickly rises according to the lower convergence bound. This indicates the performance of the scheme reaches steady-state far faster than the 447-attempt worst-case learning convergence bound suggests.

7.1.2 Online Performance

The speed of convergence of the Q-values affects the online performance of the Q-learning DCS scheme. This is analyzed by measuring the performance of the scheme within a given time after initialization. The SU median running successful transmission probability, $P(A)$, is plotted against action epoch (transmission attempt number) in Figure 7.6 for the same data obtained in Section 7.1.1. Each data value is the aggregated median successful transmission probability calculated from the record of all SU transmission attempts up to the current number from the start of the runs. As expected, the successful transmission probabilities converge to experimentally identical values.

Table 7.1 records the settling time, overshoot and rise time of transmission probability trend. The percentage overshoot is the amount the performance exceeds the final value, expressed as percentage of the final value. Whether the plot overshoots or not tends to be determined by the initial Q-value estimates. The traces for $Q_0 = [10, 5, 0], [0, 10, 5]$ overshoot since the successful transmission probability at the start most likely originate from sampling the greatest Q-value channel, with utilization 0.2 or 0.7, before the result is attenuated by later selection of the channel with worst utilization, 0.9. The plot for $Q_0 = [0, 5, 10]$ slowly rises, with the median performance initially being skewed by failed transmission exploiting the worst utilization channel.

The settling time is defined as the time required for the running packet

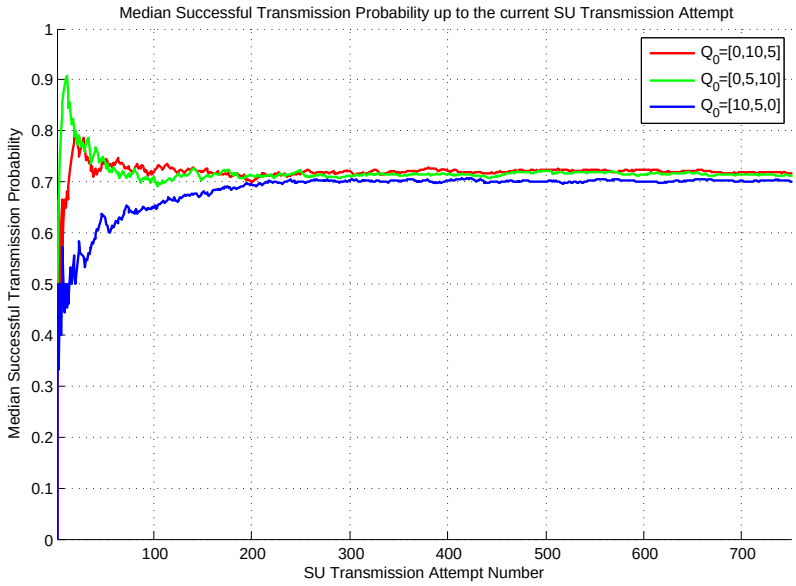

Figure 7.6: Median running successful transmission probability, for channel utilizations: $[0.9, 0.7, 0.2]$ and different Q_0

transmission success probability to settle to within 5% of its final value. The rise time in PID control theory is the time the success probability takes to go from 10% to 90% of the steady-state value. If the channels' Q-values begin undifferentiated, the initial selection policy will be random. Alternatively the rise time can be defined as the time for the success probability to reach 95% of the difference between the Q-learning and random channel selection steady-state probability. The rise and settling times provide measures for the performance convergence time.

Initial Q-values (Q_0)	**Settling Time (Epochs)**	**% Overshoot**	**Rise Time (Epochs)**	**Rise Time (95% Random / Epochs)**
[0,10,5]	31	12.192	3	13
[0,5,10]	42	27.084	3	4
[10,5,0]	125	0.317	70	164

Table 7.1: Median Successful Transmission Probability with Time, for Channel Utilizations: $[0.9, 0.7, 0.2]$

The rise and settling times when $Q_0 = [0, 5, 10]$ are significantly slower than in the other cases. A 26-point moving average of the median successful transmission probability for this scenario is plotted in Figure 7.7. The probability settles within the first 35 transmissions after an initial sharp rise. This demonstrates the final probability value is achieved much faster, with the delayed response in the running probability due to the contribution of the initial poor choice of channels when Q-values were biased to the worst channel with utilization 0.7.

After 400 attempts the probability oscillates 0.22 ± 0.02 peak-peak about 0.70. This is a concern if strict QoS requirements exist for a minimum level of throughput. A possible approach is to use receiver buffering to average the throughput. However, the oscillations show that, even after being smoothed with a 26-point moving average, significant variations in the performance are still present from transmission interruptions due to PU activity, which should be considered when it comes to setting the buffer size. Figures 7.4 and7.5 already indicate the MAC protocol cannot rely exclusively on the DCS strategy for *spectrum handoff* as the Q-learning scheme is shown to retry transmitting on the same channel that is occupied several times until it is free or its Q-value falls below that of other channels.

The experiment is repeated to determine if these results are general.

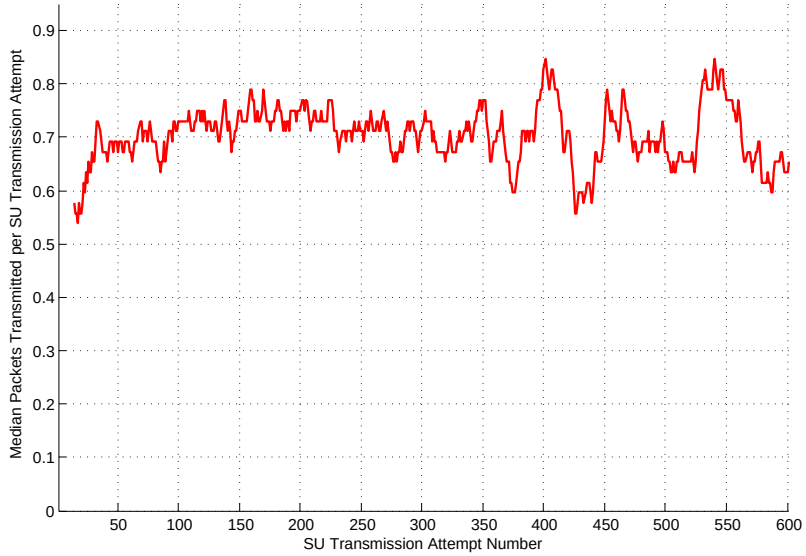

Figure 7.7: 26-term moving average of the successful transmission probability, for channel utilizations: $[0.9, 0.7, 0.2]$ and $Q_0 = [10, 5, 0]$

The wireless scenario was run for all PU channel utilization combinations such that the overall utilization across all data channels averages to 0.5, where each PU is able to select from an utilization in $\{0.1, 0.2, 0.3, , 0.9\}$. Figure 7.8 plots the SU median running successful transmission probability against action epoch. The initial Q-values used are so the rise and settling times summarized in Table 7.2 are based on Q-learning having no set preference for any particular channel at the start. The results are similar to Figure 7.6 and Table 7.1. After an initial period of uncertainty, the transmission success probability reaches a stable value. The median rise time is four and six attempts, using the new definition.

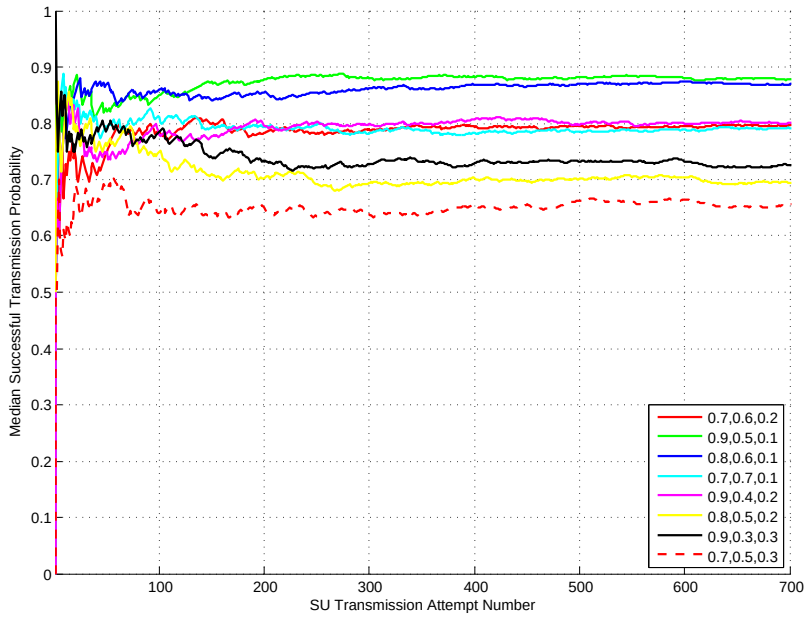

Figure 7.8: Median running successful transmission probability, for channel utilization combinations averaging to 0.5

Channel Utilization Combination	**Settling Time (Epochs)**	**% Overshoot**	**Rise Time (Epochs)**	**Rise Time (95% Random) / Epochs)**
[0.5,0.5,0.5]	183	32.166	5	6
[0.4,0.5,0.6]	58	7.808	4	5
[0.3,0.6,0.6]	107	3.598	3	14
[0.4,0.4,0.7]	512	49.716	1	1
[0.3,0.5,0.7]	87	23.973	2	2
[0.2,0.6,0.7]	49	2.494	4	54
[0.3,0.4,0.8]	106	36.835	1	1
[0.2,0.5,0.8]	68	6.128	8	9
[0.1,0.7,0.7]	40	1.371	5	44
[0.3,0.3,0.9]	34	11.545	4	4
[0.2,0.4,0.9]	15	11.614	4	5
[0.1,0.6,0.8]	18	0.202	5	16
[0.1,0.5,0.9]	55	1.594	3	13
Median:	58	7.808	4	6

Table 7.2: Median Successful Transmission Probability with Time for All Channel Utilization Combinations averaging to 0.5

The rise time is significantly faster than the 447-attempt worst-case time bound for Q-learning convergence given in Section 7.1.1. This is explained by analysis of the Q-learning scheme. The expected converged channel selection behavior rather than the Q-value behavior is set as soon as the channel with least utilization is top ranked by Q-value. This channel is then selected whenever outside of exploration and remains so as the Q-value estimates converge to their final values. The current top-ranked Q-value changes according to the best-case time bound for exploitation. For 95% convergence in the scenario this bound is 14.5 transmissions. Q-value intersection with the least utilization channel varies inside this bound depending on the initial Q-values and for the three cases in Section 7.1.1 occurs after 7.5 and 27 attempts. Based on the similarity we conclude final performance is achieved by the time of the best-case convergence time bound. Moreover this time is constant with the number of data channel as defined in Equation 6.36 in the limit of large n.

The time in real terms is given. The timing in the scenario exaggerated to handle excessive USRP latency. The maximum time for 6 transmission attempts is 1.15s. Substituting in a typical network packet transfer time of 500μs final performance is attained in real terms within only 3ms irrespective of the number of data channels. Assuming fair contention among SUs the convergence time scales linearly with the number of CRs within the network.

7.2 Q-Learning Channel Selection Scheme Performance

Our second goal is to obtain experimental results on the performance of the Q-learning DCS scheme, simulated in previous work [97]. The experiment is designed to investigate the performance that the scheme will achieve at different levels of PU utilization of the network, as measured

by the SU packet transmission success probability, goodput and the interference caused to the PUs.

The experiment is run for every permutation of PU channel utilizations averaging to an overall utilization across all data channels in the set $\{0.1, 0.2, 0.3, \ldots, 0.9\}$, where each individual PU utilization is selected from the set $\{0.1, 0.2, 0.3, \ldots, 0.9\}$. For instance, utilizations of [0.7,0.7,0.7], [0.5,0.7,0.9] and [0.7,0.5,0.9] are valid unique permutations for an overall utilization of 0.7. Channel Q-values are unset at the beginning, thus $Q_0 = [0, 0, 0]$. Each run lasts 350 seconds, representing a minimum of 1860 SU transmissions occurring in the timeframe if all attempts fail with time taken 191ms. Earlier, we showed final performance is reached within a median of six epochs so the average packet transmission success rate and other metrics obtained from the record reflect the long-term performance of the DCS scheme. A full characterization involves 243 runs each of 350 seconds of the scenario giving a total of 24 hours execution time.

The analysis of the results is based on the comparative performance of Q-learning with the other random, rule-based, no-regret, deferred ideal and non-deferred ideal schemes, outlined in Section 5.4. The experiment is repeated with the SU employing random channel selection, which sets a baseline level of performance, then the rule-based and no-regret schemes. The ideal performance determined as the deferred and non-deferred ideal schemes is derived from the measurement of the PU queue found in the Q-learning scheme PU logfiles.

The set of runs for each scheme is repeated to establish the uncertainty and accuracy of the performance results at the different PU utilizations. For statistical significance, it is generally desirable for measurements to be repeated twenty times or more. Such time was not available with the laboratory setup, but three repetitions of a full characterization of each scheme were managed, totaling approximately 288 hours (12 days). Initially the intention was to perform the experiment on the wired and wireless setups. Eight repetitions of the wired setup were completed before wireless

results were deemed more useful. The wired results are not included here as they are of less interest. Figures are plotted against the observed utilizations. The error bars used in figures indicate one standard deviation of uncertainty either side of the reported value.

7.2.1 Packet Transmission Success Probability, $P(A)$

The goal of the Q-learning DCS scheme is to maximize the proportion of SU packet transmissions that are successfully received. Figure 7.9 plots the average SU packet transmission success probability, $P(A)$, achieved in the experiment against the average PU channel utilization. The value of $P(A)$ is calculated from the total number of successful transmissions divided by the total number of transmissions that took place during the run. Each point in the graph represents the mean of the probabilities of all runs where the PU channel utilization permutation averages to the channel utilization. Thus the point at utilization corresponding to 0.5, for example, includes runs with data channel utilizations [0.5,0.5,0.5] and [0.1,0.7,0.7]. The individual Q-learning runs are plotted on the same graph to show the real $P(A)$, with points colored by the variance of the channel PU utilizations. The graph shows the non-idealities effect where the PU is not able to keep up with the packets to be transmitted at higher utilizations, causing the observed downward drift and increased spread in the actual overall and individual utilizations.

The Q-learning scheme consistently outperforms random channel selection. The observed transmission success rate is 1.60 times greater than the random scheme at a nominal overall utilization of 0.6 and 1.58 times at 0.8, but the improvement falls at lower utilizations to as little as 1.04 at a PU utilization of 0.1. The average improvement across all utilizations $[0.1, 0.2, , 0.9]$ is 39.9%. The Q-learning scheme shows similar performance to the no-regret best channel scheme. Performance is not significantly different for mean utilizations 0.2 to 0.6 but exceeds the success-

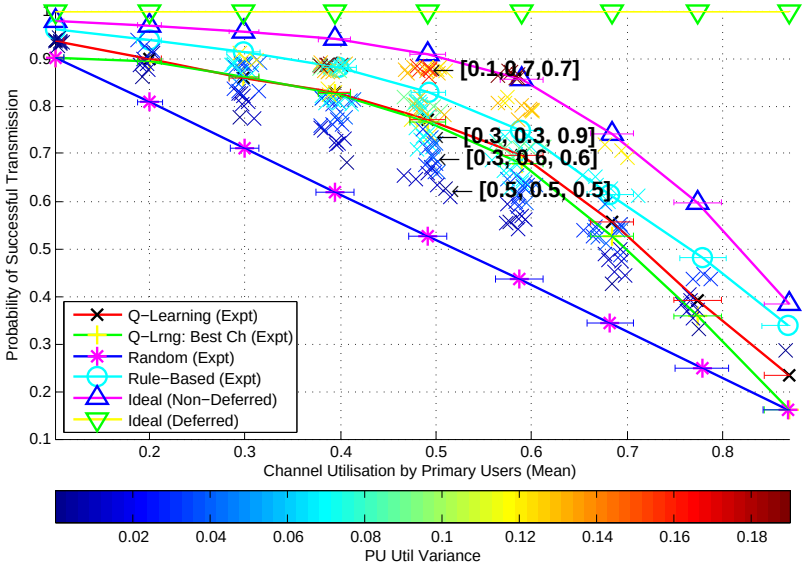

Figure 7.9: SU successful transmission probability, $P(A)$, against mean of the PU channel utilizations

ful packet transmission proportion elsewhere. At the lower and upper end of the utilizations where the channel utilizations are equal, namely [0.1,0.1,0.1] and [0.9,0.9,0.9], the no-regret scheme behavior then becomes identical to uniform random channel selection. Averaging all points, the Q-learning scheme packet transmission success probability is greater by a factor of 1.07. Performance is slightly worse but comparable to the rule-based scheme, with the Q-learning scheme closely following its performance. The probability $P(A)$ on average across all utilizations is smaller by a factor of 0.91.

Equation 4.11 shows the Q-learning scheme will exploit the channel with the least utilization, which is confirmed by the experiment results. Utilizations [0.1,0.7,0.7] and [0.3,0.3,0.9] exhibit similar performance to when the channel utilizations are [0.1,0.1,0.1] and [0.3,0.3,0.3]. The proportional improvement over the random DCS at higher utilizations is explained by the greater difference in whitespace between channels, since

a 0.1 reduction in channel utilization 0.9 represents a larger fractional increase in the exploitable whitespace than if the channel utilization was 0.2. To illustrate, channel 1 in [0.6,0.9,0.9]=0.8 (mean) is free four times as often as the other channels, which are of greater utilization, compared to 1.5 times for channel 1 in [0.1,0.4,0.4]=0.3 (mean). Thus, selecting the channel with least utilization by the Q-learning scheme has a greater effect on performance.

The analytical transmission success rate is plotted alongside the experimental results in Figure 7.10. The predicted performance of the random DCS is identical to that observed, to within measurement uncertainties. While they follow a similar trend, the experimental successful transmission probability for the Q-learning scheme is higher. The R^2 correlation is 0.9999 between analytical and observed random channel selection results and 0.9647 for the Q-learning scheme. This indicates the scheme successfully exploits local variations in PU traffic (*noise*), whereas the Markov chain analysis assumes steady-state converged Q-values for the system. This is seen at utilizations [0.1,0.1,0.1]=0.1 and [0.9,0.9,0.9]=0.9 where the experimental $P(A)$ is found to be greater than in random channel selection despite the fact the Q-value convergence rule should lead to uniform sampling of each channel. The results show the analytical model is indicative of the performance of the Q-learning scheme. The ability of the DCS to add to the real success probability by learning and reacting, for instance to a brief quiescent period in a channel, is influenced by the actual traffic, exploration policy and learning rate.

The distribution of transmissions which were not successful is relevant. The probability of transmission attempts ending in failure or aborting in sensing is plotted in Figures 7.11 and 7.12. The channel packet error rate in the scenario is negligible so failed transmissions correspond to losses caused by PU interference.

At all points, the majority of unsuccessful transmissions experimentally observed in the Q-learning channel selection scheme consist of those

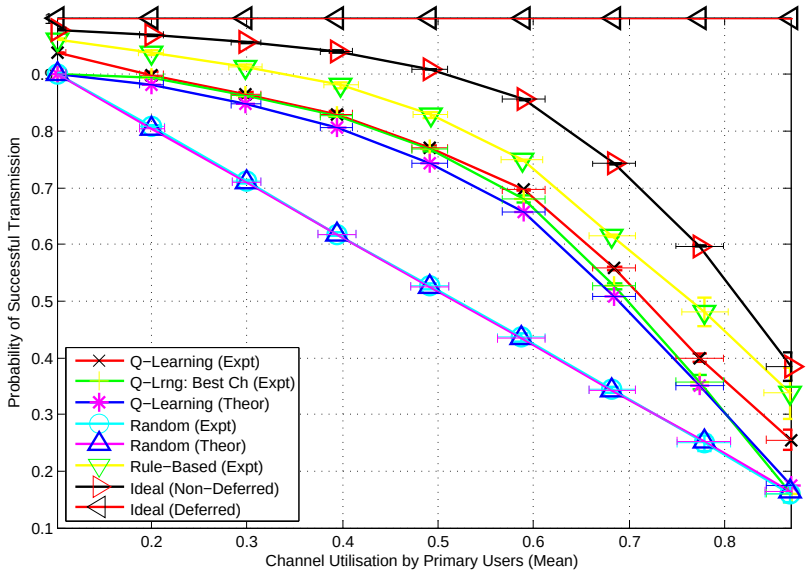

Figure 7.10: Mean SU successful transmission probability, $P(A)$, against mean of the PU channel utilizations

aborted without harming the transmitting PU in the channel. Failed transmissions are less frequent, up to a maximum at utilization 0.8 where fully 5.75% of SU transmissions cause interference. The proportion of interfering attempts against all unsuccessful transmissions increases the lower the utilization on the plot is. At a utilization of 0.9 interfering attempts comprise 5.94% of all unsuccessful attempts but at 0.1 this rises to 29.1%.

Sensing is reliable so interference can only occur if a PU becomes active during the SU transmission, as otherwise prior *Listen-before-Talking* sensing will have aborted the transmission. Consecutive PU packets in a packet train, which are more likely to arise at higher utilizations, are safe from interference. We infer from the results that extra sensing mechanisms are needed to prevent interference to burst traffic and isolated packets. A solution would be for the CR to exploit upper network layer knowledge of the PU's behavior. If the licensed user network also co-ordinates multiple access, the CR should have the capability to decode preambles being ex-

changed, such as RTS, and use it to predict when the channel will become occupied. The ability to autonomously develop network protocols to use was originally envisaged by Mitola when he conceptualized CRs [57]. Interference can be avoided entirely if PUs preface all transmissions with an initial redundant section and, as a commensurate measure, SUs reduce the time taken to transmit to be inside this duration. However, current licensees are unlikely to tolerate having this requested of them with the necessary architectural changes. Future licensing agreements could mandate the use of SU friendly control mechanisms to give greater spectrum efficiency.

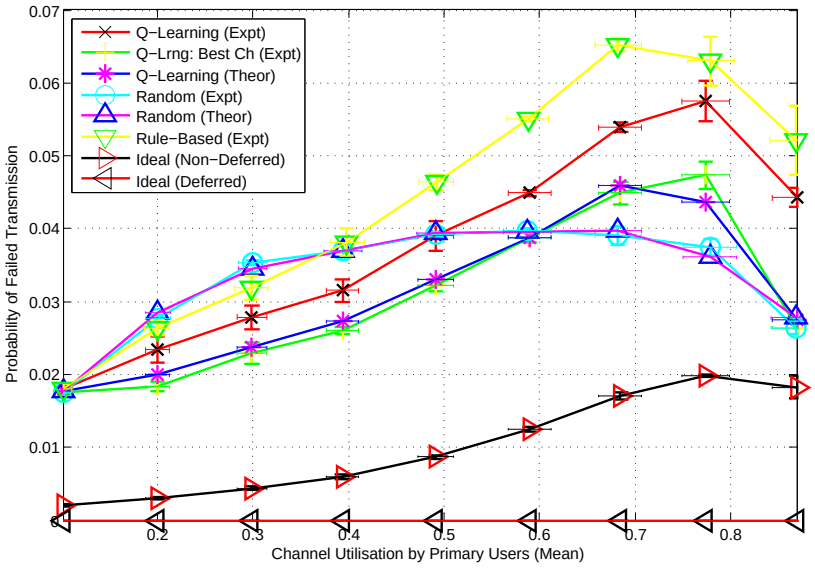

Figure 7.11: Mean SU failed (data) transmission probability, $P(B_1)$, against mean of the PU channel utilizations

Similar results for the packet transmission success probability are shown in Figure 7.13 where the experimental $P(A)$ is plotted when the first 225 transmission attempts are considered only. The time is limited to half the Q-learning worst-case convergence time bound of 447 attempts in order to investigate whether the level of performance during conver-

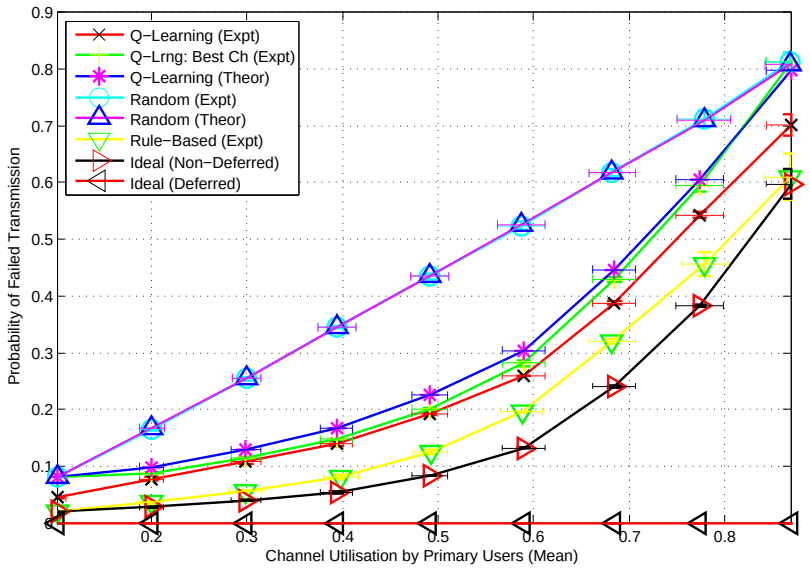

Figure 7.12: Mean SU aborted (sensing) transmission probability, $P(B_2)$, against mean of the PU channel utilizations

gence is different. The relative performance in terms of rankings of all schemes is unchanged. The experimental packet transmission success rate using the Q-learning DCS is on average across all utilizations is now 45.5% greater than the analytical performance given for random channel selection, which is in fact higher than the 39.9% when previously considering the entirety of the runs. We conclude the Q-learning scheme adapts rapidly, which is consistent with the findings in Section 7.1.2 that actual performance settles far sooner than the time required for Q-value convergence, within 6 attempts according to the results.

The results, especially at higher utilizations are more uncertain and it would have been preferable if more repetitions had been carried out. The rule-based scheme probability at mean utilization of 0.9 exhibits a standard deviation uncertainty equal to 53.6% of the mean value, indicating its performance is highly dependent on short-term PU traffic pattern that occurred.

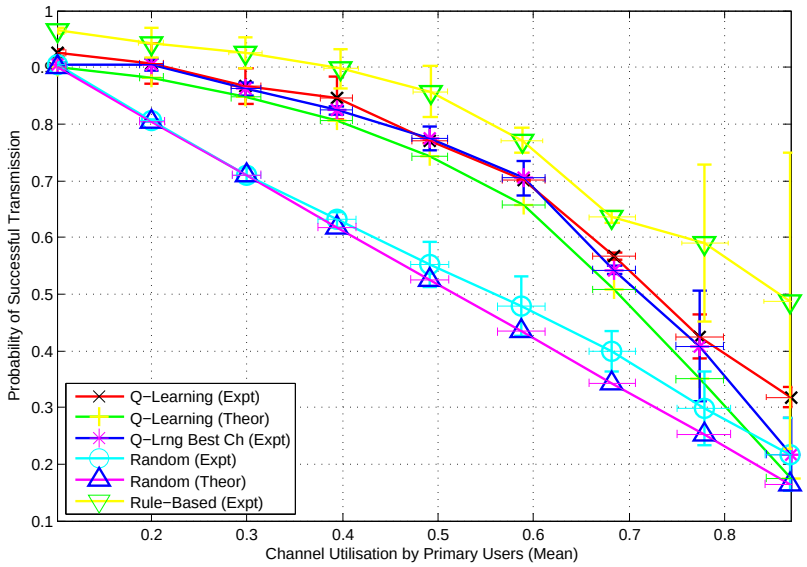

Figure 7.13: Mean SU successful transmission probability, $P(A)$, against mean of the PU channel utilizations, within the first 225 transmission attempts

7.2.2 SU Goodput, G

The goodput measures the useful payload bits transmitted per unit time. Figures 7.14 and 7.15 plot the average goodput in bits per second as measured during the entirety of each run using the same format as Figures 7.9 and 7.10 respectively.

The different schemes' performances are similar in form to that observed in the packet transmission success probability results, which is expected since the goodput is directly related to how many packets are transmitted correctly. The Q-learning DCS scheme outperforms random channel selection at all points. The goodput is on average 56% greater although again the factor of difference reduces at the lower utilizations. The Q-learning scheme has slightly increased goodput compared to the no-regret scheme, on average 9% greater, and is somewhat outperformed by the rule-based scheme, by a factor of 11.6% on average.

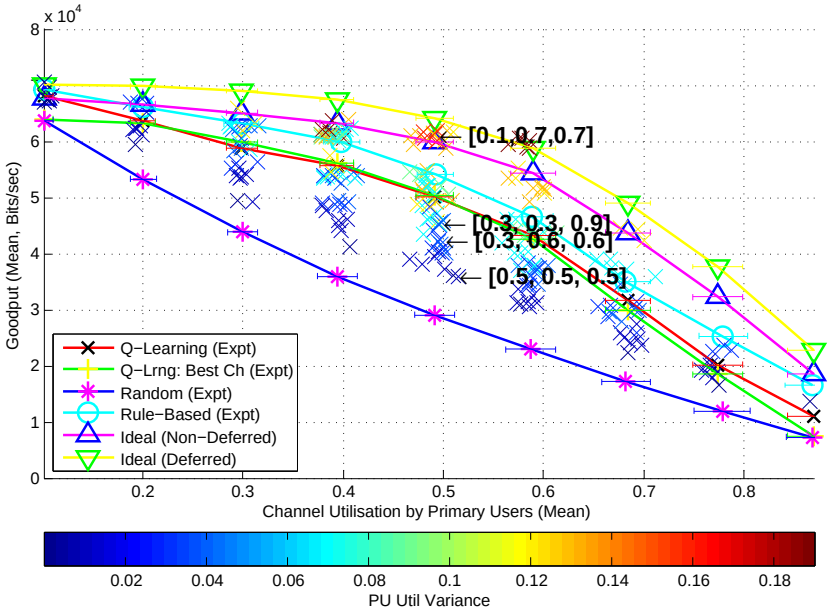

Figure 7.14: SU goodput against mean of the PU channel utilizations

Significantly, measuring the goodput allows the deferred ideal scheme to be judged. This strategy's successful transmission proportion is plotted in Section 7.2.1 for completeness but the results are not comparable as the scheme waits until it can transmit successfully in a channel, thereby guaranteeing perfect transmission, whereas all other schemes transmit immediately. Despite idling the deferred ideal scheme is consistently capable of greater goodput than the non-deferred ideal scheme, which is able to predict the outcome of transmitting on each channel and will select the first channel if it exists that results in the packet being received correctly. The minimum, average and maximum improvement is 3.66%, 9.79% and 23.0%. Both ideal schemes significantly outperform the other realizable schemes that do not require perfect foreknowledge of how PUs will transmit.

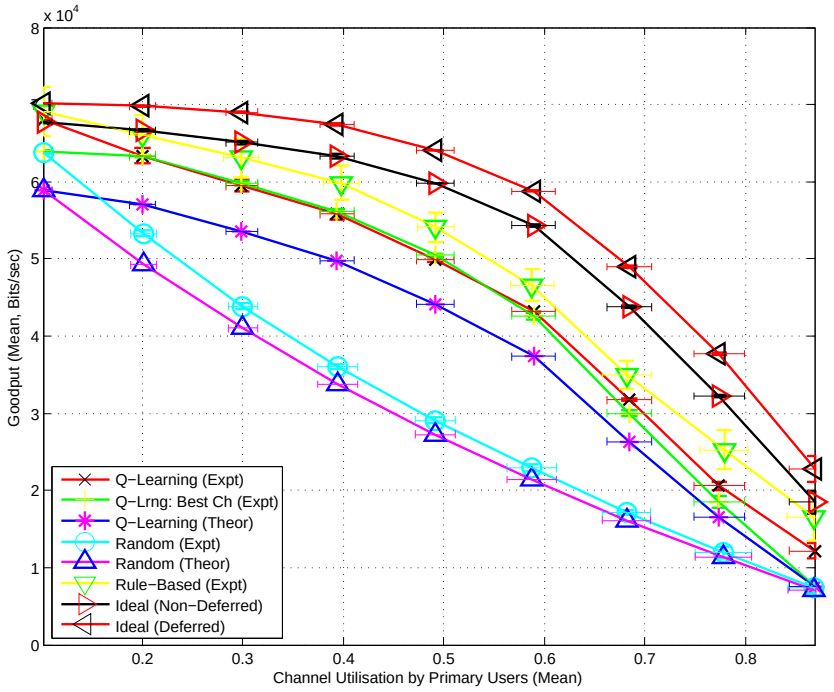

Figure 7.15: Mean SU goodput against mean of the PU channel utilizations

7.2.3 PU Interference, I_j

The average analytical and experimental PU interference is plotted in Figure 7.16. Each point in the graph is the mean of the PU interference in channels 1 and 2 during the experiment, plotted against the average PU utilization across all data channels. The PU interference in channel 3 is not included because, as mentioned in Section 5.5.1 due to equipment constraints no receiver could be implemented to log the PU. Since the results comprise running the scenario for each channel utilization permutation and as all channels and users are homogeneous, it is justified to interpret the figures as also the average interference caused to each PU or all PUs.

Outside of a mean utilization of 0.1 the interference caused by Q-learning channel selection is significantly greater than random channel selection. The interference rate is on average 46% greater and the absolute

value decreases with increased mean utilization for random, rule-based and Q-learning channel selection. This is consistent with our interpretation of the results regarding *Listen-before-Talking* sensing in Section 7.2.1. To repeat, *Listen-before-Talking* sensing cannot prevent the SU interfering with a packet broadcast while it is transmitting. However, subsequent packets in a packet train will be avoided which are more likely to occur at higher utilizations. The Q-learning scheme preferentially selects channels with a past history of guaranteeing successful packet transmission and is more likely to use those channels with least utilization. Isolated packets are more likely to be transmitted in this channel, resulting in the higher interference observed when compared to random channel selection where lower utilized channels are selected equally with other channels.

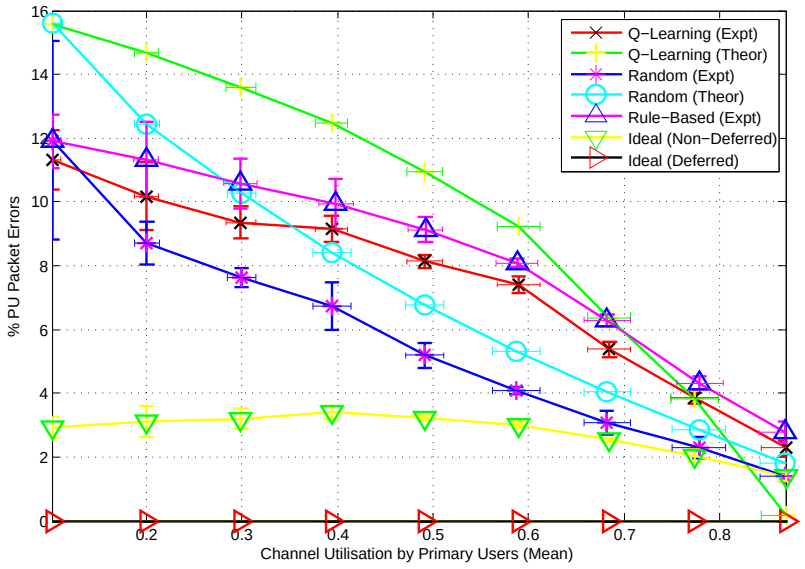

Figure 7.16: Mean PU interference against mean of the PU channel utilizations

Figure 7.17 graphs the average experimental interference. Additionally the interferences caused to each PU in each Q-learning run are plotted individually. Points are colored by the PU utilization and show interference

is greater in channels with lower utilization, with the same conclusion as for Figure 7.16. The interferences at points 0.1 and 0.9 on the x-axis, aggregating the performance when the channel utilization is [0.1,0.1,0.1] and [0.9,0.9,0.9], differs from interference elsewhere where the PU utilization is identical but the utilization among all channels is heterogeneous. The per channel interference when the utilization is [0.1,0.1,0.1] is 11.4±0.9% but the interference to a channel with the same 0.1 utilization can reach up to 39.09%, which occurs when the nominal mean utilization is 0.6 in the figure. Similarly when the utilization is [0.9,0.9,0.9] the interference is 2.2±0.3%, but this drops to 0.10% for an identical channel with 0.9 utilization at mean utilization 0.7. This shows Q-learning channel selection increases interference to the channel with the least utilization in the network but reduces harmful effects caused to other channels. This is expected since Q-learning favors selection of the least utilization channels, thus increasing the number of PU packets lost in the channel due to greater usage. The maximum interference caused to a channel with utilization 0.1 is 343% greater as compared to [0.1,0.1,0.1]. The top channel, ranked by Q-value, is selected with probability 0.93, whereas if all channels have equal Q-values this is 0.33 so there is a 280% increase in usage. Channels with lower Q-values are selected for exploration with probability 0.033, thus there is less SU interference.

The analytical results in Figure 7.16 significantly deviate from the observed interference. The R^2 correlation between the analytical and observed random channel selection results is 0.5381 while it is -0.1449 for the Q-learning scheme. The R^2 values indicate the analysis does not fit the experimental results. Theory correctly predicts that interference will trend downwards with increased mean utilization and the Q-learning DCS scheme will have raised interference over random, both trivial results. However the analysis poorly models the trend shape and significantly overestimates the packets that will be lost. The steady-state model cannot account for the Q-learning scheme's dynamic adaptation to local vari-

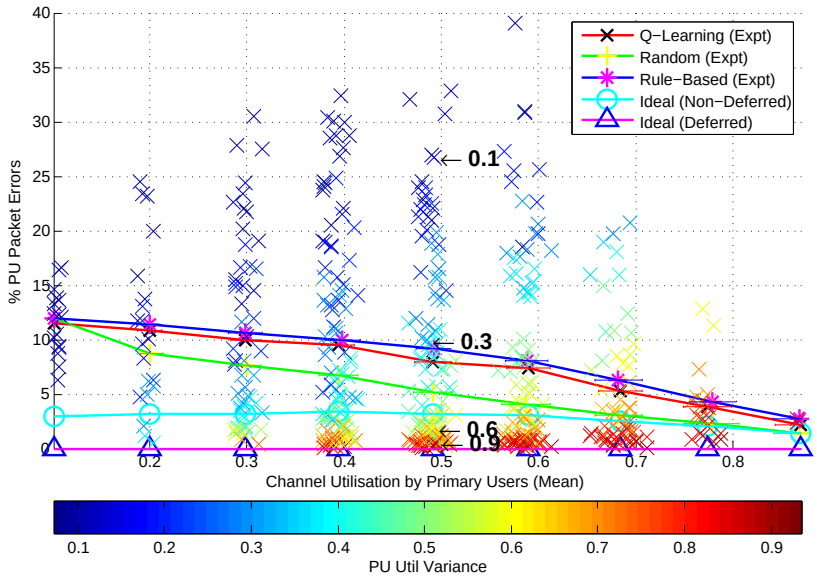

Figure 7.17: PU interference against mean of the PU channel utilizations

ations in channel occupancy which would lessen the impact the SU has on the PU, but this does not explain the discrepancy between random channel selection analytical and experimental results.

The analytical goodput accurately models the downward shape of the experimental goodput with utilization in Figure 7.15 but underestimates the experimental values. The R^2 correlation between analytical and observed random channel selection results is 0.9798 and 0.8935 for the Q-learning scheme, which is lower than the values for packet transmission success correlation in Figure 7.10.

The interference and goodput are derived from the analytical SU transmission outcome probabilities. It is likely the increased discrepancy is due to the propagated error in using the analytical result for the SU transmission outcome probability added to simplifications of the scenario made by the model. As discussed in Section 6.4 the model's assumptions that PU traffic is perfectly M/D/1 and timing is distribution-less, which in particular is used in finding the goodput and interference, are not realizable in

the GNU Radio implementation. The accurate fit in the successful transmission probability in Figure 7.10 is achieved only after substituting into the model the empirical channel transmission outcome probabilities measured at different PU utilizations.

7.2.4 Further Results

Thus far, in Figures 7.9-7.17, the performance of the different schemes has been plotted as the mean performance against the average PU channel utilization. This style is employed so the results can be cross-referenced with previous work in simulation [97], which presents their results using this manner. A problem is it is unclear from the graphs what the performance of a single run is, for instance what the real goodput is when the SU observes a channel with [0.9,0.3,0.3] utilization. Also the shape of the graphs is highly influenced by the runs contributing to each point, which are arbitrarily set to consist of all valid channel utilization combinations of [0.1,0.2,.,0.9] averaging to the x-axis value. The distribution of channel utilizations within all runs being averaged at a point and the number of runs is inconsistent, affecting the uncertainty and statistical significance of the result. Point 0.1 consists only of runs with the single utilization [0.1,0.1,0.1]. For 0.5 the corresponding set is comprised of 61 permutations, and because each run is average run [0.5,0.5,0.5] is considered equally to [0.3,0.3,0.9] despite the variation in utilizations. For the Q-learning scheme this issue is reduced by plotting the individual results identifiable by color, such as in Figure 7.9. However the scheme is compared to the alternative strategies by the averaged results, thus it is unclear if the SU goodput under the rule-based scheme in Figure 7.15 is greater or only so because of the distribution of utilization permutations in the averaged runs. The individual logfiles have been stored and are available as a resource for analysis.

Figures 7.18 and 7.19 plot a subset of the measurements used to generate Figures 7.10 and 7.15. The performance of the runs where the utiliza-

tions of all PUs are equal is graphed. So points on the x-axis at 0.2 in Figure 7.19 correspond to the goodput when the scenario channel utilizations are [0.2,0.2,0.2]. The performance trends are significantly different to what is recorded in Figures 7.10 and 7.15, showing the effect altering the distribution of channel utilizations in the runs considered at each point has. The performance of the no-regret scheme is identical to within uncertainties to random channel selection, which is to be expected as the strategy selects uniformly from all channels which are now of the same lowest equal utilization.

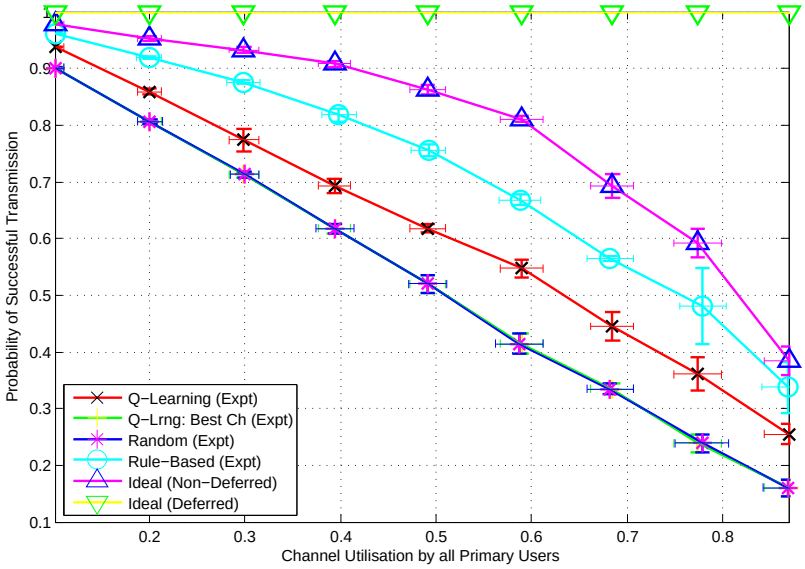

Figure 7.18: Mean SU successful transmission probability, $P(A)$, against the PU utilization in all channels

Again the rule-based scheme consistently outperforms the Q-learning DCS. The Q-learning scheme outperforms random channel selection, thus we conclude Q-learning is able to learn and respond to local decreases in channel utilization even if the average utilization across channels is identical. The successful transmission probability and goodput are on average 16.7% and 20.0% greater respectively. The average SU successful trans-

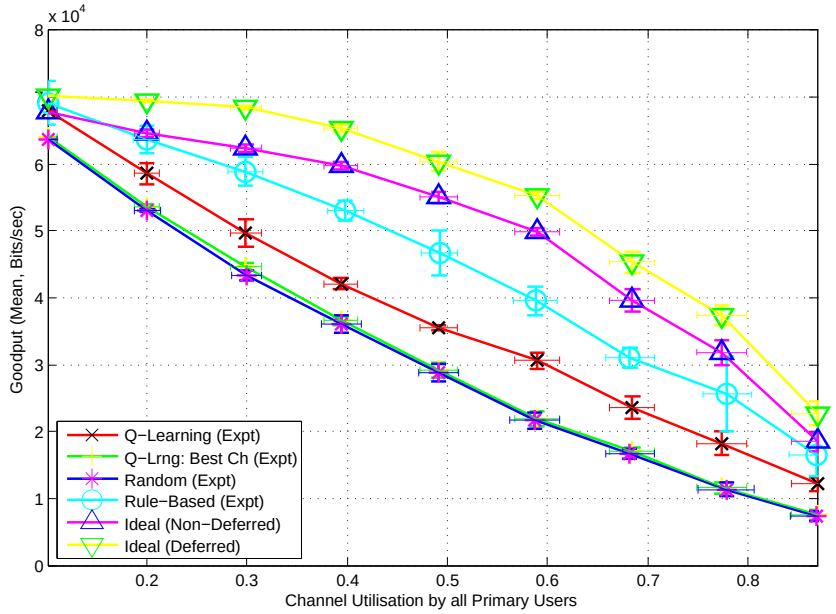

Figure 7.19: Mean SU goodput against the PU utilization in all channels

mission probability under the rule-based scheme is 19.80% greater than the Q-learning scheme, compared to 9.89% in Figure 7.10. The average goodput is increased by 24.88%. Since the improvement is less when all runs are averaged this suggests the difference in performance is reduced and the Q-learning scheme may outperform rule-based selection when the channel utilization is heterogeneous. The deferred ideal scheme results show that by employing a predictive approach to fully exploit the whitespace, significant usage can be obtained even when all channels are highly utilized. When the average utilization occurring in all bands is 0.9 the SU goodput has fallen only 46.47% from when the utilization is 0.1, despite the utilization increasing by 0.8. Instead of performance falling linearly with increasing utilization, as experienced by random channel selection, for the ideal schemes the drop off is reduced between low utilizations. With more data channels secondary usage should improve as it is more likely at least one data channel will be temporarily free for the ideal schemes to exploit.

The other extreme is plotted in Figures 7.20 and 7.21. At each point, the runs with the maximum PU channel utilization variance from the subset of all runs where the PU channel utilization averages to the utilization at the point. The average performance is plotted at the point. For example at utilization 0.8 the plotted performance is averaged from runs [0.9,0.9,0.6],[0.6,0.9,0.9],[0.9,0.6,0.9]. The results show that when there is a high spread in the channel utilization the performance of the Q-learning channel selection improves relative to the ideal and rule-based schemes, although both schemes still consistently outperform it. The average SU successful transmission probability and goodput achieved by the rule-based scheme is now only 4.04% and 5.48% greater. The no-regret scheme goodput is superior to the rule-based scheme at utilizations 0.4-0.6 where the channel utilization variance ranges from 0.16-0.19.

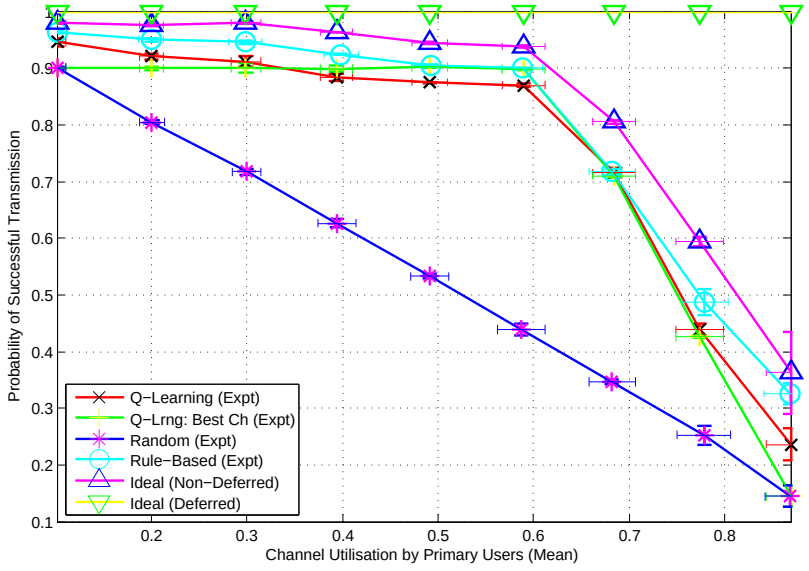

Figure 7.20: Mean SU successful transmission probability, $P(A)$, against mean of the PU channel utilizations

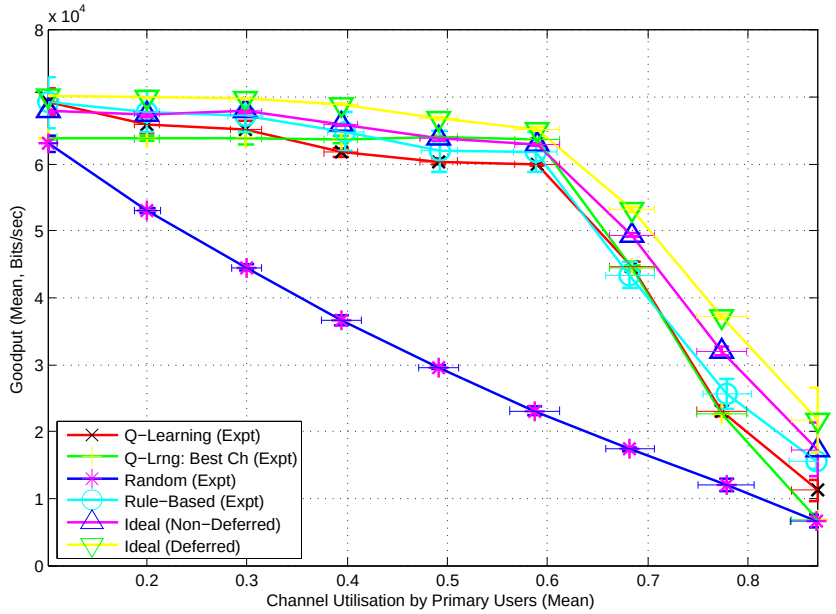

Figure 7.21: Mean SU goodput against mean of the PU channel utilizations

7.2.5 Discussion

The Q-learning DCS scheme has been found to significantly improve the SU's chance of success on transmission and goodput in the scenario over undifferentiated random selection of channels. In the experiment, by learning the channel with the least utilization, the Q-learning DCS provides an average 39.9% and 56% increase in the successful transmission probability and goodput respectively over random channel selection calculated across all mean utilizations. However performance falls well short of the maximum limits formalized by the ideal strategies. From Figures 7.10 and 7.15 the average packet transmission success probability can be improved by 23.4% and the goodput by 24.4% in following the non-deferred ideal scheme. The figure for the increase in goodput using the deferred ideal scheme is 37.6%. Performance is also reduced in comparison to the rule-based scheme, which exhibits 10.8% greater average probability of packet transmission success and 11.6% increased average goodput

over the Q-learning DCS.

The issue lies in the speed of learning. The Q-learning scheme takes a number of transmissions to respond to the appearance of whitespace in a channel. In the discussion of Figure 7.3 it was observed Q-learning does not switch to exploit this empty channel until the cumulative penalty of repeated interfered unsuccessful channels in the current occupied top-ranked by Q-value channel causes it to decrease below the Q-value estimate of the unused channel. The difference closes according to Equation 6.31. The exact point of intersection depends on the initial Q-values however the number of transmissions needed for the exploited channel to fall a certain percentage to its final Q-value of CT, for being occupied, is given by the best-time convergence bound Equation 6.36. The required time for the empty channel to rise a certain percentage to its final Q-value of RW is given by the worst-time convergence bound Equation 6.35. The contribution of this term is small in comparison to the top-ranked Q-value channel, which being sampled most frequently changes the most rapidly. Thus the best-time convergence bound can be used as an approximate measure for the transmission time required. Using the scenario Q-learning parameters the best-case time bound for 95% convergence is 14.5 SU transmissions. In physical terms this means the Q-learning scheme in the experiment is unable to take advantage of whitespace lasting less than 14.5 transmissions. The speed of convergence is increased by raising the learning rate, α, to a minimum 95% time of 1.1 transmissions when α is set to its maximal value of 1.

The non-deferred ideal scheme demonstrates that significant performance gains are possible if instant convergence were obtained. In the definition of the scheme, channel selection is based on foreknowledge of the outcome the choice will have. Either the transmission will abort or fail due to PU interference or the channel will be unoccupied during the duration of the data transfer. If such a channel exists, the scheme selects the first channel numerically. Otherwise a channel is selected at random for

transmission with invariant failure due to PU interference. Rephrasing this in terms of response time, the scheme requires zero transmissions to learn and adapt to whitespace in other channels. Current whitespace is always exploitable. The scheme could possibly be implemented if there were some form of PU negotiation or capability to predict PU traffic.

The rule-based DCS results are notable since it shows the truly cognitive approach to channel selection using Q-learning possesses inferior performance to what is a merely an adaptive selection algorithm. Two simple fixed rules decide which channel is to be used in this scheme. The scheme selects only the extant channel being used if the previous transmission on it proved successful. Otherwise a channel is selected uniformly at random from the set of other channels. The scheme is said to be adaptive rather than cognitive since no learning is involved. The results agree with the earlier hypothesis that performance is improved by reducing the time for whitespace to be located. When the scheme is randomly selecting among channels $\frac{n}{2} = 1.5$ in this case is the expected delay in transmissions from the time a channel becomes empty to being sampled, located and exploited by the algorithm, compared to 14.5 derived as an approximate guide for the Q-learning scheme.

It needs to be investigated whether this superiority in performance holds when there are more channels such as will occur in a realistic network-based scenario. The required time increases linearly with n, the number of channels, for the rule-based scheme but Equation 6.36 states that for the Q-learning scheme the time is constant in the limit of large $n >> \alpha\epsilon$. For the convergence time of the top-ranked Q-value, which is its original definition, this is true but it is incorrect in this context. The time until the channel where whitespace is occurring is top-ranked by Q-value depends on the number of data channels and its ranking by Q-value. In the worst case, if the channel has the lowest Q-value every other channel will first need to be tried unsuccessfully until their Q-values are reduced below that of the empty channel for this channel to be sampled, barring that it is

selected through exploration, which is worse in time by at least a factor of two compared to random selection to find the whitespace. The probability a channel is selected through random exploration also reduces with increased number of channels. However the alternative channels have a record of being less utilized and so are more likely to be ready for use than those channels trialled via random channel selection.

Reducing the convergence time by increasing α, the learning rate, in the limit to 1 produces almost identical behavior to the rule-based scheme. This leads to one-shot learning where only the immediate packet transmission outcome determines the Q-value. When $\alpha = 1$ a success sets the channel Q-value directly to the reward value, RW. A failure sets it to the cost, CT. If there is no exploration the Q-value at all other channels outside the one currently being used is equal to this cost as either the last transmission attempt in the channel was successful, in which case it will most likely be selected for the next epoch as it has the highest Q-value with RW, or the attempt failed and a channel is selected uniformly randomly from all available data channels which now have equal Q-values CT.

Figures 7.10 and 7.15 indicate the no-regret scheme achieves similar performance compared to the Q-learning scheme. The successful packet transmission proportion and SU goodput is smaller when the nominal mean utilization is 0.1 or exceeds 0.7 but elsewhere is identical within uncertainties. The average successful transmission proportion and goodput is 6.78% and 6.55% less, and against the rule-based scheme 14.66% and 14.89% less respectively. The no-regret strategy selects equally between the channels with the least scenario designated utilization. Consequently it models the behavior of the Q-learning DCS when $\alpha \to 0$ in the limit of infinite transmission epochs elapsing, guaranteeing that the Q-values have eventually converged to Equation 6.8 and as $\alpha \to 0$ remain fixed there unperturbed by stochastic variations in channel whitespace. Among the converged Q-values, those belonging to the channels with the least average utilization will be the greatest, causing their exploitation for channel

selection which is replicated by the no-regret strategy.

As stated, it is not within the scope of this thesis to determine exactly how the Q-learning scheme will perform at all values of the Q-learning variables, such as RW, CT or ϵ. However, the results gathered give an idea how varying the learning rate, α, will affect the scheme's performance. The no-regret scheme produces the steady-state behavior expected of the scheme when $\alpha \to 0$ and has lower goodput or likelihood of transmissions being successful compared to the rule-based scheme, approximating the behavior at the other extreme when $\alpha = 1$. Both schemes ignore exploration so conclusions are independent of the particular exploration policy in use. The performance of the Q-learning scheme where $\alpha = 0.2$ lies in between these two extremes. The comparative performance shows that spectrum agility and the ability to rapidly locate and relocate to occurring whitespace in any channel is more valuable than selecting the channel with the least utilization over time. The best goodput performance achieved by varying the learning rate is by the rule-based scheme, roughly equivalent to setting $\alpha = 1$. Thus, in fact, knowledge of the past utilizations is unnecessary since better performance is accomplished using one-shot learning which discards all but the previous result for use in decision-making.

This is interesting as the opposite result is reported by [97] for the same Q-learning and rule-based DCS schemes. A similar scenario is set up in OMNET++ to the one described in this thesis, involving a backlogged SU and 3 data channels occupied by PUs transmitting at exponentially distributed intervals. The Q-learning scheme simulation results are presented in the manner of Figure 7.10 where the average performance is plotted against the mean utilization of the channels and show it to consistently outperform the SU goodput of the rule-based scheme at all utilizations. The performance of the rule-based scheme used for the comparison is derived analytically from Markov chain analysis. Markov chain analysis of the Q-learning scheme presented here was unable to predict significant

performance improvement would be obtained from local whitespace in the experiments. It is possible this is neglected in [97] leading to the discrepancy. Also different scenario timing parameters were employed, preventing a true comparison to our results. Identical Q-learning parameters are used but the interframe and packet transfer times are in line with those expected of a network, for instance SIFS=10μs rather than in the order of milliseconds, This was not achievable by us in the real radio hardware when compared to evaluating performance in simulation environment.

Most significantly in [97] the SU and PU packet data packet durations are both 5.44ms. In this thesis the PU data packet duration is approximately an order of magnitude longer at 311.3ms compared to the SU DATA packet, which have a packet transfer time of 30.2ms. The cycle time for the SU to complete a successful transmission is 110ms and 191ms if the transmission fails, so there is a high correlation that using the same channel to transfer a packet after the previous attempt has failed due to PU interference will abort because the same PU packet is still being broadcast. If the past history of the channel was sufficiently underutilized the Q-learning scheme will repeat the use of this channel regardless. However the rule-based scheme will randomly select another channel where there is a chance the band will be free.

The single-state model of the system used by Q-learning is too simplistic to handle this dependency. A suggestion would be to represent the choice of the same channel after the previous attempt was unsuccessful as a separate action. This introduces $n = 3$ new states. The failed attempt causes the agent to transition to one of these states. The allowable actions consist of normal transmission on the other two channels and transmission, with the knowledge the last attempt failed, on the previous channel. If the attempt is successful the agent returns to the state consisting of normal transmission on all channels. The Q-values for repeating the use of the same channel become learnable and the action avoidable, which is combined with the performance improvement of learning to select the channel

with the least utilization. However, introducing new states lengthens the Q-value convergence time. If all actions are treated as unique, the number of actions added, 9 in this scenario, needing to be sampled increases exponentially with the number of channels.

Figures 7.20 and 7.21 show that if significant differences exist in the channel utilizations, then above a threshold the converged learned solution of selecting the channel with least utilization given by the no-regret scheme will have superior goodput performance over the rule-based scheme. This result is trivial since if transmission success is in the limit of being guaranteed in the channel with least utilization, while in other channels the high utilization prevents all transmissions being successful, exploration of the other channels is wasted. The threshold is measured as a utilization variance of 0.12 in the scenario. Even so in the results obtained with the maximum channel utilization variance of 0.19, corresponding to PU utilization permutations of [0.9,0.1,0.8], the difference is minimal, with goodput improved by 2.9%. When the channel utilizations are similar reducing the convergence time to exploit local whitespace improves performance as indicated by Figures 7.18 and 7.19, where the rule-based scheme has 24.88% increased goodput over the Q-learning scheme.

Figure 7.16 shows that the Q-learning scheme causes significantly increased interference to PUs over random channel selection or either ideal strategies. The average proportion of PU packets lost is 46% greater than the random scheme. The selection of the channel with the least utilization is detrimental with *Listen-before-Talking* sensing in use, since it is unable to prevent collisions with PU packets appearing mid-transmission but will avert the transmission if the packet is in the middle of a packet train which is more likely to occur in higher utilization traffic. The individual channel interference can be high. A maximum figure of fully 39.09% of all PU packets transmitted being lost due to SU interference was recorded in Figure 7.17. Predictive and more sophisticated techniques for sensing are needed for exponentially distributed traffic.

This level of interference to licensees is unacceptable. However this does not invalidate the concept of dynamic spectrum access networks and spectrum agile radios for making best efficient use of spectral resources. The deferred ideal scheme achieves superior goodput to all other strategies trialled and shows that by fully exploiting the whitespace significant unlicensed usage is obtainable in busy channels without hindering PU activity at all.

The central aim of this thesis is to answer the research question, *"what performance does the Q-learning channel selection scheme achieve in a real deployment scenario?"* and although many new results have been discovered regarding the performance of the Q-learning DCS scheme, as described in this section, the experimental scenario does not reflect a real network environment. The scenario considers the performance obtained in the case of only one pair of communicating SUs and three data channels, whereas a realistic network could comprise dozens of SU and PU nodes and channels over multiple hops. For comparison, IEEE 802.11 uses 13 channels in the 2.4000-2.4835GHz band.

It was recognized at the beginning that, due to equipment constraints, the experiment would be limited to a small-scale. The intention was that the research question would be answered by examining the performance of the scheme in the setup and by deriving an analytical model to predict the performance in the large-scale multiagent case. The theoretical analysis would be verified by comparing to the experimental results. The results show a poor fit with the analytical predictions. The average experimental successful transmission proportion is 8.72% higher than what the analysis estimates in Figure 7.10, with a R^2 correlation of 0.9647. The difference is greater for the goodput and PU interference in Figures 7.15 and 7.16 where the R^2 match between the Q-learning experimental and analytical results is 0.8935 and -0.1449 respectively. The model is invalid in assuming the average performance is given by the selection policy using static converged Q-values. Figure 7.3 shows the median Q-values reach

steady-state values but Figure 7.4 indicates the single-trial Q-values fluctuate. The model does not account for the Q-learning scheme responding to stochastic variations in the traffic, thus exploiting local whitespace leading to an increase in the actual obtained performance. In addition there are non-idealities in the scenario implementation. Interprocessing delays lead to significant distribution in timing which is modeled as exact. Modeling the DCS scheme and scenario in a discrete event time network simulator such as OMNeT++ or NS-2 would be able to reproduce the Q-learning scheme's dynamic behavior.

7.3 Experience Report

This section discusses the experiences of working with the GNU Radio platform to implement a CR testbed. After working with GNU Radio, the platform showed itself to be an effective environment for rapid prototyping. The open source license and strong contributor base were useful here. The extensive library of physical layer block implementations meant that different physical radio designs could be rapidly developed in this thesis by assembling flowgraphs from existing blocks or, where new functionality is needed, writing a new block that is inserted into the rest of the flowgraph. Cognitive functionality at the MAC layer is then added as python flowgraph control code. The CR in this thesis shows this legacy, making use of modified versions of the existing GNU Radio frame modulator and demodulator. This also means the design of the radio could be rapidly reconfigured. GNU Radio also had a helpful and responsive user base. Most questions could be resolved by asking on the GNU Radio mailing list, `discuss-gnuradio@gnu.org`, often through developer postings, which was invaluable for a first-time user with nothing previously to go on.

However shortcomings were found at the hardware and software levels. The USRP RF frontend performance is suboptimal. With the USB the

device can only transfer a maximum Nyquist bandwidth of 4MHz signal data to the host computer for processing. This has implications for the CR techniques that can be investigated using the system, effectively precluding realistic experiments on OFDM spectrum agile access or sensing across multiple channels, as well as hampering the ability for GNU Radio to be used for practical demonstrations, such as an 802.11 receiver which employs 13MHz channels. The USB also adds unacceptable latency when transmitting or receiving packets. Achieving strict timing was found to be problematic. Interprocessing delays caused a large spread in the transmission cycle times, shown in Table 5.8. The issue is the GNU Radio flowgraph scheduler which operates by polling of every flowgraph block, which if data is present at the input buffer and there is sufficient output space processes the data. The time data will actually be transmitted to the USRP is unpredictable and there is no support for priorities or explicit scheduling of the time the data will be processed. This severely affected the validity of the experimental results. It is unclear whether the large discrepancy observed between the analytical and experimental goodput and interference results is partly due to timing being modeled as exact in the analysis and the PU traffic implementation deviating from M/D/1 due to processing delays. The system developed in this thesis does not satisfy reproducibility or repeatability requirements, since the results are peculiar to the non-ideal implemented scenario, thus another experiment following the same scenario may experience different results. Case in point [97] which recreates the scenario in simulation comes to the opposite conclusion that the Q-learning scheme outperforms the rule-based selection strategy. Thus our experience shows that a direct GNU Radio implementation of a CR is unsuitable as a testbed.

7.4 Summary

The results of the experiment in Chapter 5 were presented in this chapter and compared with the analysis in Chapter 6. From observations of the SU Q-values, the Q-learning scheme converges reliably to the expected utilization-based reward found analytically. The SU goodput and successful transmission probability achieved under the Q-learning scheme is significantly superior to random channel selection. On average the successful transmission probability is 39.9

Chapter 8

Conclusions

The goal of this research is to answer the question, *"what performance does the Q-learning channel selection scheme achieve in a real deployment scenario?"* This study is important given the challenges of spectrum scarcity. Cognitive spectrum agile radios open up the possibility of shared use of licensed channels as long as PUs are not interfered with, allowing for a greater proliferation of wireless services than the currently overcrowded ISM bands support. Development of dynamic channel selection strategies to exploit high-quality whitespace is critical for bringing about spectrum agile radio. In previous work a Q-learning DCS scheme was proposed that maximizes the MAC layer packet transmission success probability by learning to transmit on the channel with least utilization. Initial simulation shows it to give promising results. The purpose of this research is to determine if the improved performance extends to real implementation in a realistic scenario environment.

To accomplish this, the performance of the Q-learning DCS scheme was studied empirically in a wireless network experiment. This involved the implementation of a CR system in the GNU Radio platform and the development of co-ordination tools and tools to log and analyze the experimental results. The single-hop performance of the SU using the scheme was considered among three channels occupied by PUs with exponen-

tially distributed packet interarrival times, with results able to be generalized to the multiagent SU case. The Poisson distribution was chosen as it models packet-based networks. An analytical model was derived using steady-state Markov chain analysis to predict the realistic large-scale network performance, which is not practicably achievable in the physical setup.

The SU packet transmission success probability, goodput and PU interference were measured across all stationary PU channel utilizations in the set $\{0.1, 0.2, \ldots, 0.9\}$. The Q-learning DCS scheme was found to significantly outperform the baseline performance set by random channel selection, with 39.9% and 56% increased successful transmission probability and goodput calculated over all utilizations. Performance improved when the learning rate was increased and the results of previous learning are less significant. A rule-based scheme similar to the one-shot learning case, selecting the previous channel if the last transmission was successful otherwise randomly transmitting on one of the other channels, resulted in 10.8% and 11.6% increased packet transmission success probability and goodput.

The Q-learning scheme convergence time was measured. The successful transmission probability was observed to rise to within 95% of its final running value by the time required for six transmission attempts, taking the median across all utilizations. The time corresponds to the number of attempts given by the analytical discrete-time model needed before updates to the least utilization channel Q-value causes it to rise above the others. Defined as the best-time convergence bound, this time is independent of the number of channels and decreases with increasing learning rate. After this point the final strategy has been reached, with convergence of the remaining Q-values having a negligible effect on the selection policy. Using Q-learning an empty channel is not identified until after six transmissions, whereas random channel selection by the rule-based scheme will begin exploiting the channel within an expected 1.5 attempts

with three channels. However above a threshold, if the variance between the channel utilizations is 0.12 or over, the Q-learning scheme outperforms one-shot learning thus validating learning to select the channel with least stationary utilization.

The channel Q-values responded to local variations in the utilization, thus the analytical model assumption that performance with steady-state converged Q-values could be used was invalid and the prediction results proved to be inaccurate.

The conclusion of the work is that using the Q-learning DCS scheme, SU goodput and MAC layer packet transmission success probability is increased in the real scenario. The analysis show that in dynamic channel selection, knowledge of the channel with least long-term utilization is less useful than the speed transient whitespace is located.

8.1 Contributions

The key contributions of this thesis include:

1. Empirical performance analysis of the Q-learning DCS scheme in a real system. The research significantly advances the knowledge that the community has of the scheme's behaviour, with a previous evaluation in simulation only. Novel results for the PU interference and time dynamic performance of the scheme have been obtained, metrics not considered in previous work. A more comprehensive comparitive approach is taken. The relative performance of the Q-learning scheme with respect to the ideal and rule-based schemes indicates shortcomings with the intelligent DCS not discernable when comparing the scheme only to random channel selection.

2. Derivation of an analytical model using Markov chain theory for the performance of the Q-learning scheme in the test scenario. The model serves to generalize the expected results for larger networks,

which is more likely in realistic deployments. The model has been validated with experimental results, though not fully, and the existence of several discrepancies in the observed results need to be investigated in future work.

3. In addition, ideal performance bounds were derived assuming the CR had perfect knowledge of licensed band usage to fully exploit the whitespace. The results give significant credence to the SU concept. The non-deferred ideal scheme gives superior performance to the Q-learning scheme is achieved while maintaining zero interference to PUs. Performance is comparable to licensed usage when the channels are low-medium utilized. In the experiment, the ideal SU goodput in Figure 7.19 when the channel utilization in all three channels is 0.5 is only 13.9% less compared to when the utilizations are 0.1.

8.2 Accomplishments

Although not a research contribution, the implementation of a CR on the GNU Radio platform was a substantial accomplishment. GNU Radio has been widely used for prototyping CR techniques, such as in [63] [74] [71]. However the focus elsewhere has been on testing physical layer techniques, such as signal classification, or demonstrating flexibility can be achieved by toggling between empty channels. Thus far implementing a *cognition cycle* at the MAC layer in GNU Radio to learn and select channels for transmission, which here we have done using Q-learning to learn channel occupancy history, is a novel application.

8.3 Future Work

Future work should take two main directions. Firstly, as identified in Section 7.3, GNU Radio hardware and software limitations impose sever performance penalties on the CR system as it stands now, which need resolving. Secondly, improvements to the Q-learning DCS scheme are suggested.

8.3.1 Testbed Development

The CR system suffers from lack of control in packet timing, which is critical if it is used to evaluate CR MAC protocol techniques. Needed improvements are discussed.

USB

The USB2.0 buffer adds latency to data being sent to or from the USRP. As described in Section 5.5.3 a minimum and maximum delay of 256μs and 8.192ms is incurred in the experiment due to the USRP USB controller requiring a multiple of 512 bytes of data to transfer across. This is small in comparison to timing variation due to interprocess scheduling, which in Table 5.8 leads to the same transmission operation being executed hundreds of milliseconds later. The 32MB/s USB2.0 bandwidth is the major problem which acts as a performance bottleneck. If complex float samples are used it allows a maximum 4MHz Nyquist bandwidth to be observed or broadcast, limiting the ability to test MIMO or OFDM applications for example. The most direct solution would be to modify USRP to use a different bus technology. For example, Ethernet supports a 1Gb/s transfer rate while PCI-Express is capable of 8GB/s. However the increased throughput must be considered against the latency of the bus protocol. This could also necessitate a platform change.

USRP-based Code

The USB bottleneck can be mitigated by transferring early stage host computer signal processing code to the USRP FPGA. The SPAN project [80] is a successful example of this. In their implementation of an 802.11 receiver the FPGA is programmed to perform the Barker dispreading. The 2MHz output of this, down from the 22MHz raw channel sample data, is passed to the host computer. Implementing the task in hardware would also lead it to be calculated more efficiently and eliminate interprocessing delays compared to handling it in software on the host computer. This would require familiarization with the Verilog HDL.

m-blocks

In Section A.4.1 we discussed message blocks or *m-blocks*, currently in development for GNU Radio. m-blocks greatly enhance GNU Radio's packet timing control support that is lacking in the thesis' CR implementation. Currently when a packet is processed in GNU Radio is left up to the flowgraph scheduler, which uses a crude polling mechanism on data streams. Hence interprocess scheduling leads to large variations in processing times. Using m-blocks data is processed as packets containing the signal data and metadata, which can be specified with a scheduler priority and timestamps for when the packet should be processed at each flowgraph block. The priority-based quasi-real-time scheduler ensures these times are met. Future work should look to porting over this design to use m-blocks.

Future Experiments

Further analysis of the Q-learning DCS scheme is needed. Its behavior needs to be modeled with more users and beyond the single-hop case. More complex testbed systems will be required. In the multihop scenario *hidden user* problems become an issue requiring co-operative sensing tech-

niques. The Q-learning scheme could be extended to learn not just the past history of PU activity per channel but the history of activity per PU. This necessitates the ability to classify signals, thus future work could look to implementing cyclostationary feature detection in place of the power detector. Additionally assuming a fixed CCC is unrealistic and the protocol for moving between temporary rendezvous channels needs to be formalized. Such large-scale experiments may be better suited to simulation which is cheaper, gives greater control to the designer and is less time-consuming.

8.3.2 Algorithm Development

Possible improvements to the Q-learning DCS scheme are discussed. The current reward strategy only measures whether a packet was successfully transmitted or not. This gives good results when all channels are homogeneous but if the bitrate is heterogeneous then selecting an underutilized but low capacity channel will give poor goodput. The scheme needs to be reformulated to emphasize the time-wise performance.

Using Q-learning, the scheme accumulates knowledge of the channel quality as expressed by the packet transmission success rate. The scheme should seek not only to exploit high quality channels but also optimize its transmissions within each channel, based on the learned characteristics. For instance, if the packet transmission success probability is low because of noise then the power level should be increased to compensate. In the opposite case the SU could take advantage of the low PER by transmitting larger size packets with greater protocol efficiency. This could be a separate cognitive extension to the radio.

A problem with the ϵ-greedy exploration policy in use is that, besides the top-ranked by Q-value channel, all other channels are selected equally during exploration even if a significant variation in channel utilization exists between them. An immediate improvement to the Q-learning scheme

would be to implement a continuous exploration technique such as Boltzmann exploration (Equation 4.9) so a channel that has the next lowest utilization should be selected more often than a fully utilized channel.

Appendix A

GNU Radio

This appendix serves as an introductory guide to GNU Radio and USRP. The core architecture, concepts and capabilities of GNU Radio and USRP are described in Sections A.1 and A.2, which may be safely skipped by readers already familiar with the platforms. Platform limitations with implications for this thesis implementation are then noted in Section A.4. Finally in Section A.5 related work using GNU Radio in CR research concludes this appendix.

A.1 GNU Radio

GNU Radio[1] [4] is a free open source software framework for implementing SDR on Linux systems. GNU Radio makes use of discrete signal processing blocks written in C++ on a host PC. Taking advantage of its open source license, over time enthusiasts have contributed hundreds of prewritten blocks now bundled with GNU Radio, providing such functionality as Phase Shift Keying (PSK) modulation and trellis encoding. Blocks are defined by the number of input and output ports they possess and the data stream data type accepted or produced at each port. The output

[1]GNU Radio v3.2, the current version as of June, 2009, was used for the work in this thesis. The thesis text references this version when describing GNU Radio.

port of a block can be connected to the input port of another block, if the data types are compatible, leading to the second block processing the data stream output by the first. A radio is created by wiring together different C++ signal processing blocks into a *GNU Radio flowgraph*, analogous to how hardware radios are built by cascading physical radio frequency (RF) components. For example an FM receiver can be made by joining together an FIR filter followed by a demodulator, as depicted in Figure A.1. A top-level python program is written to create the connections and handles the running of the flowgraph, such as scheduling and data handling. The signal processing blocks are written in C++ for high performance [19]. Python at the top-level simplifies modifying code since it is an interpretive language that does not need to be compiled between changes, but this causes it to run slower.

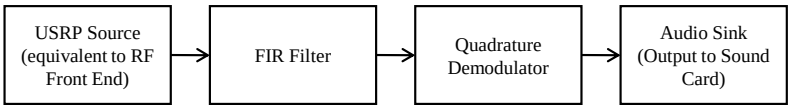

Figure A.1: GNU Radio flowgraph for a simple FM receiver

A block with no outputs is known as a *sink*. In the example flowgraph the FM signal terminates at an audio sink, which plays the waveform to the computer sound card. Other sinks implemented include a spectrum analyzer display sink. A *source* block has no inputs and generates the data used, such as from a file. Figure A.1 shows the signal originates from a USRP source.

GNU Radio is designed to work with the Universal Software Radio Peripheral (USRP) hardware peripheral manufactured and sold by Ettus Research LCC [2]. The USRP, pictured in Figure A.2, handles up- and down-conversion of signals from baseband to RF. The USRP is built around an Altera Cyclone EP1C12 FPGA with high-speed ADCs/DACs performing digital up/down conversion and is capable of mounting up to four

analogue RF frontend *daughterboards*. Various daughterboards exist to receive or transmit anywhere between 0-5.8GHz depending on the daughterboard. This is part of the design philosophy behind GNU Radio, which is to have the high speed general purpose operations like digital down conversion done in hardware. The flowgraph signal processing only handles the lower speed waveform specific processing, such as modulation, thus the GNU Radio SDR architecture can be thought of as a USRP frontend with a host computer backend. If speed is required some flowgraph processing operations can be programmed on to the FPGA with Verilog, the use of which is beyond the scope of this thesis. The USRP and USB exchange baseband signal data via a USB2.0 interface. The USRP also recognizes control information to change, for instance, the RF frequency with which the daughterboard is receiving. These are provided as methods for the python program by the USRP sink and source blocks.

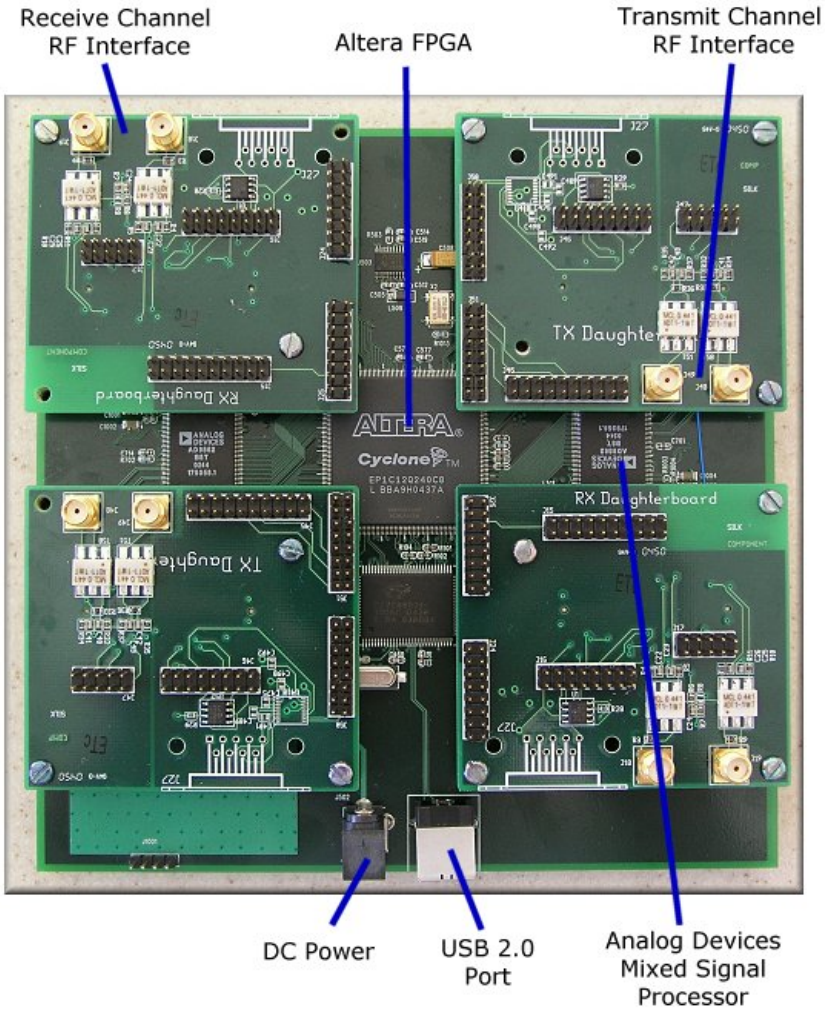

Figure A.2: Universal Software Radio Peripheral (USRP) Board mounting 4 BasicRX/TX Daughterboards, from [33]

The layered GNU Radio framework architecture is depicted in Table A.1. The C++ signal processing blocks have been previously discussed as has the role of the a top-level python program to connect together the blocks into a flowgraph and during runtime provide data for processing, act on received data and tune operation, such as change the USRP frequency through the sink and source. Scheduling within the flowgraph is

handled automatically by the GNU Radio scheduler, which is also written in python but is separate from the top-level python program. The SWIG [9] (the Simplified Wrapper and Interface Generator) tool interfaces between python and C++. Each block is associated with a SWIG `.i` wrapper definition file. The python program can access non-private C++ block methods or variables through this wrapper, although again doing this is slow.

Python Flow Graph
SWIG
C++ Signal Processing Blocks
USB2.0 Connection
USRP

Table A.1: GNU Radio Framework Architecture

A.2 Universal Software Radio Peripheral (USRP)

The USRP is a dedicated hardware frontend for handling the RF-IF and IF-baseband communication stages for GNU Radio. The USRP interfaces with the host computer backend over a USB2.0 link which it sends 16-bit I/Q complex float samples of the baseband signal for software processing. The USRP motherboard comprises an Altera Cyclone EP1C12 FPGA, 4 high-speed 12-bit 64MS/s (M sample/sec) ADCs and 14-bit 128MS/s DACs implemented on the AD9862 CODEC chip and a Cypress RX2 USB2.0 controller. 20dB programmable gain amplifiers (PGAs) are present before and after the ADCs and DACs respectively. Up to four daughterboards, analogue RF frontends, can be mounted onto the USRP. Each daughterboard slot is assigned two of the DACs or ADCs, which digitize the daughterboard signals at IF. The default FPGA image implements

multiplexed Digital Up Converters (DUCs) and Digital Down Converters (DDCs) that take the digital signal to baseband format for transmission over the USB link. Additionally, the USRP has auxiliary analog and digital I/O ports, some of which are used to control daughterboard operation. The USRP is fully duplex, although individual daughterboards may not be. The rate of data being simultaneously transmitted and received is limited by the USB interface.

A block diagram of the USRP is given in Figure A.3 and Figure A.4 shows the ADC/DAC and default FPGA image sections of the USRP.

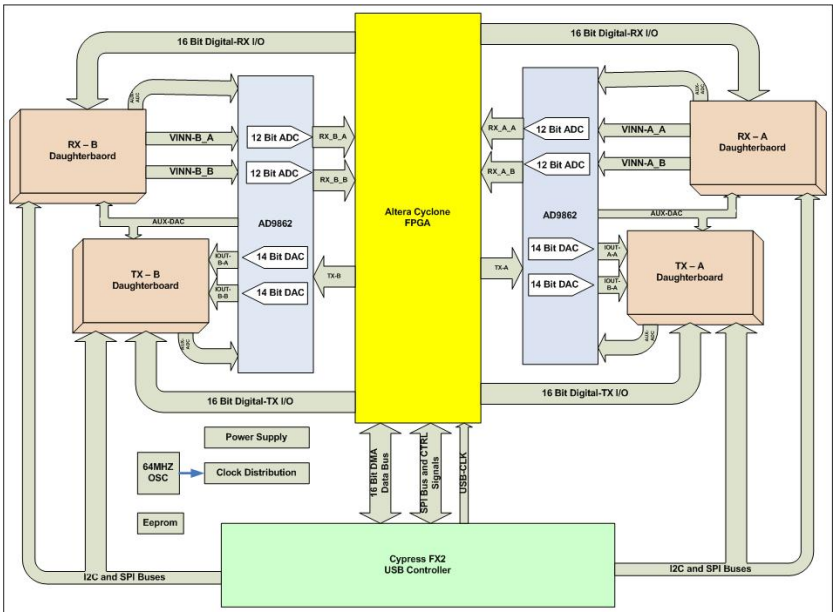

Figure A.3: USRP Block Diagram, from [3]

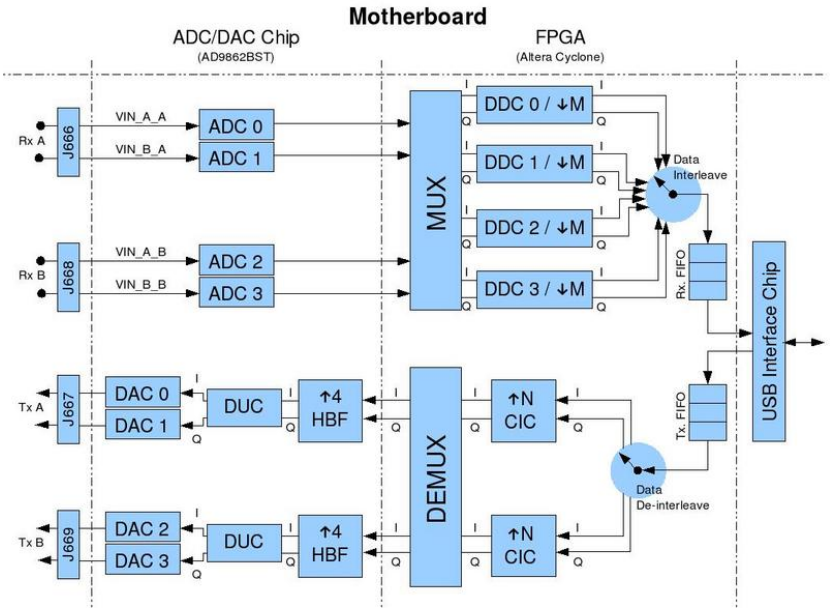

Figure A.4: ADC/DAC and FPGA sections of the USRP, from [3]

A number of daughterboards have been made available by Ettus Research LLC for working at different frequencies between 0-5.9GHz. In common with the rest of the open source GNU Radio project, most boards have schematics downloadable from the GNU Radio subversion repository [11]. This thesis uses the RFX2400 and XCVR2450 transceiver boards to work in the 2.4GHz Wi-Fi bands. The boards are:

- RFX2400: The RFX2400 transceiver has an operating frequency range of 2.3 to 2.9GHz with maximum transmit power 50mW (17dBm). The board can operate in full duplex mode as long as it is not transmitting and receiving simultaneously on the same frequency. The RFX2400 has a 20MHz TX/RX bandwidth, so concurrent transmission and reception can occur within any other two points inside this width [10].

- XCVR2450: The XCVR2450 transceiver can transmit up to 100mW (20dBm) and has a frequency range of 2.4 to 2.5GHz and 4.9 to

5.9GHz. The XCVR2450 is single-duplex only.

Figure A.5 shows a USRP kit mounting a RFX2400 daughterboard. Daughterboard inputs/outputs are brought out to SMA headers.

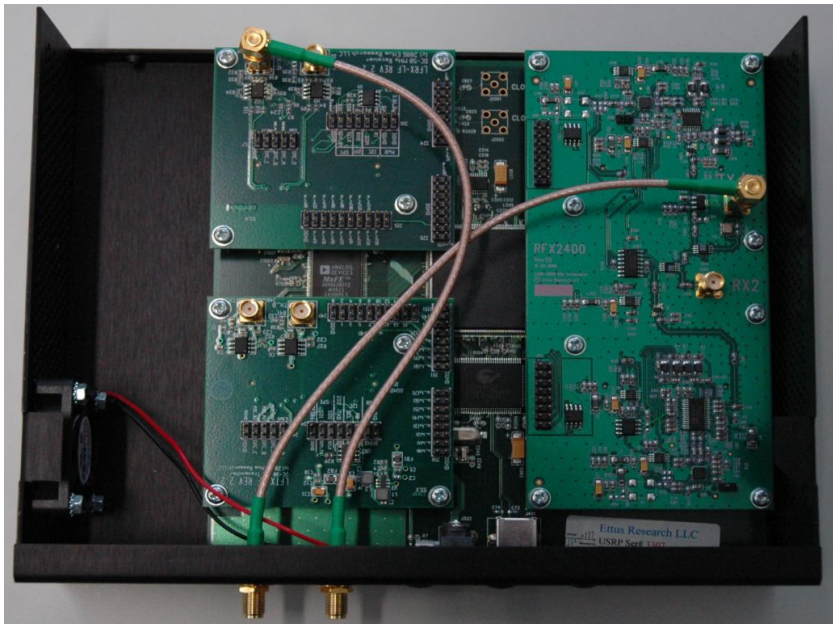

Figure A.5: USRP mounting RFX2400, LFTX and LFRX daughterboards

The received IF signals from the daughterboard are down converted to baseband by the CORDIC numerically controlled oscillator (NCO) mixer in the USRP DDC, which is fully implemented on the FPGA and depicted in Figure A.6. The signal is digitized at 64MS/s by the ADC for a total data rate of 256MB/s with 16-bit I& Q samples. The USB2.0 link has a specified data transfer rate of 480Mbit/s but in practice considering protocol overheads can only sustain 32MB/s [11], giving a maximum observable double-sided bandwidth of 8MHz. Even this cannot be achieved without overruns when a less powerful computer is playing host. An important role of the DDC is to decimate the signal to fit across the link. This is achieved using a cascade integrator-comb (CIC) filter. The 31-tap half-

band filters cascaded after are for out of band signal rejection. The DDC is configurable from GNU Radio for decimation by a factor of [8, 256].

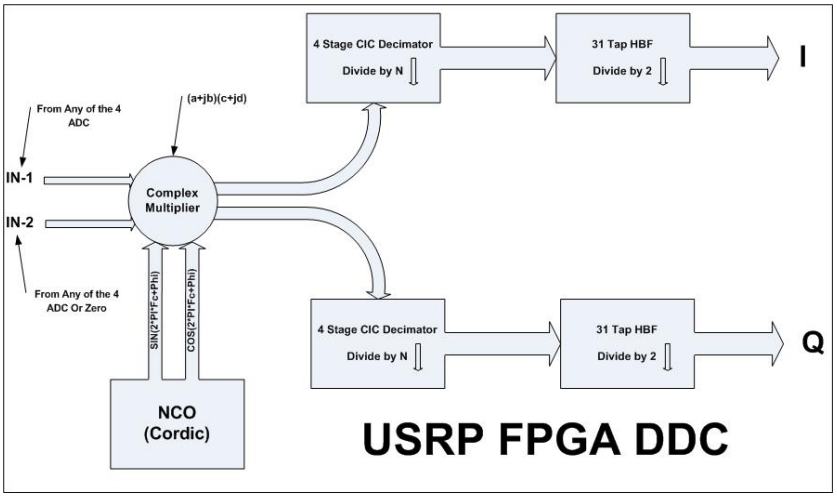

Figure A.6: USRP Digital Down Converter, from [3]

The DUCs, shown in Figure A.7, are implemented in the AD9862 ADC/DAC chips. Only CIC interpolators are programmed onto the FPGA to upconvert the computer signal to 32MS/s before the remaining on-chip stages take this up to the ADC 128MS/s. DUC interpolation by a factor of [16, 512] can be set by GNU Radio.

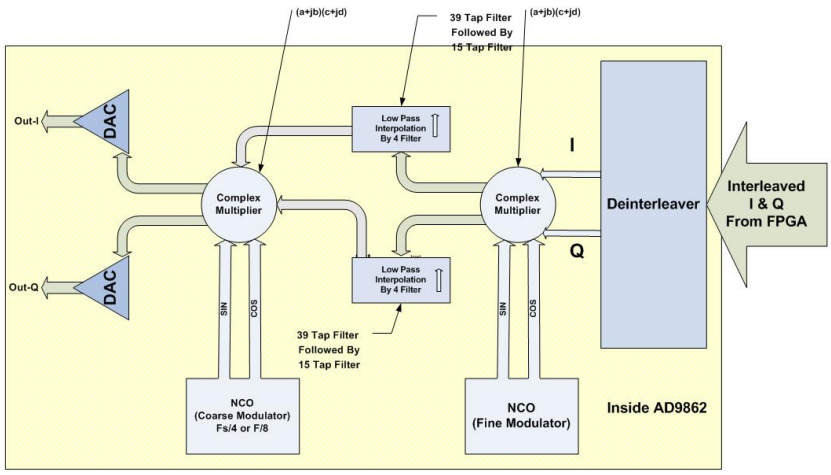

Figure A.7: USRP Digital Up Converter, from [3]

A.3 Further Reading

The purpose of this thesis is not to serve as a reference guide into the USRP and GNU Radio platform, and this brief introduction will have to suffice. A number of academic papers have been written going into the technical details of the devices and how to install, code and run a basic test program on GNU Radio. Manicka's [52] "GNU Radio Testbed" and Jones' [43] "Channel Sounding with Software Defined Radio" are good examples. Plentiful learning resources exist online such as Shen's [77] excellent tutorial series, which covers everything from step by step installation to writing custom GNU Radio blocks, and the official GNU Radio wiki [4] is there to be consulted.

A.4 Limitations of GNU Radio

The GNU Radio platform offers many advantages for SDR and CR research. It is open source and has a strong community base. Different groups have added to a large library of blocks providing capabilities rang-

ing from DBPSK modulation, FFT, FIR and Hilbert filters. Newcomers to the platform have many implemented signal processing functions to work from. The sheer number of blocks is daunting. The official website [4] links to a doxygen-generated list of all bundled blocks but the best approach to understanding what each block does is still to read the source code. If questions remain they can be posted on the GNU Radio mailing list, which unhappily goes by the motto ”we're helpful people, and we expect you to try to help yourself”, that is also to be found on the GNU Radio homepage.

At the same time there are a number of issues with GNU Radio. DSA using CRs is being proposed as a next step in wireless access. The major DARPA xG CR project is working with an 802.16 modem [54], using it to send out packets in free parts of the spectrum, and the IEEE 802.22 packet radio standard [26] is being authored for WRAN access of the TV spectrum. A problem is that GNU Radio is not designed for packet radio applications. The platform was originally created for processing of continuous streams of signal data and has been used with success in such applications, for example FM and AM radios. There is only rudimentary packet support. The limited bandwidth of the USRP is an additional concern and is general to the entire class of SDRs.

A.4.1 Software Limitations

No packet concept exists within GNU Radio flowgraph block architecture. In the flowgraph data is processed as streams of homogeneous items. Each block declares a `work()` function that takes a fixed size input unit and processes it to get the output. For instance, an N-size FFT block will require an N-size array of complex float samples at its input. The GNU Radio scheduler, a python thread, handles the memory management and running of the flowgraph. The scheduler iterates over the flowgraph blocks. If the input item(s) are available and there is sufficient room at the output buffers,

the scheduler invokes the block's `work()` function. The operation of the scheduler can be summarized by the pseudo-code algorithm in Figure A.8.

```
while enabled and nalive > 0 do
    for i = 1,2,..., blocks do
        if sufficient room in output port buffers for block i
        and data at input ports of block i
            then invoke work() on block i
        end if
    end for
end while
```

Figure A.8: GNU Radio Scheduling Loop Algorithm, from [19]

GNU Radio defines `message` source and sink blocks for sending and receiving discrete packets of data. The GNU Radio native implementation of a packet transmitter, `benchmark_tx.py`, defines a packet of data at the level of the top layer python program. This is inserted into a message queue in a message source flowgraph block. This block converts the packet into a stream of data samples which are carried on to modulation blocks and subsequently transmitted by the USRP. The receiver implementation, `benchmark_rx.py`, demodulates a stream of complex baseband samples from the USRP source block. A framer sink block looks for a header containing the payload length. The identified length of bytes are assembled with the header into a packet and inserted into a message queue. A separate python queue watcher thread blocks on the queue, waiting to take the receiver packet at the head and forward it to a callback function in the main python program for link layer processing.

Packets can be defined at the python top layer program, but flowgraph blocks only recognize streams of simple types such as floats. There is no interface for blocks to signal each other. The GNU Radio scheduler does not support priorities and explicit scheduling of block tasks is not possible,

rather it is handled automatically. Block input and output buffer sizes are set automatically by the buffer based on the block data type and ratio of input data items produced per output item, which is ill-suited to handling frames of different sizes. The implication is that once a packet has been queued on to the transmitter flowgraph it is impossible to know when it will be actually transmitted or to apply different block signal processing to a specific packet. Dhar et al [31] were unable to implement the MAC protocol they required on GNU Radio for these reasons: the lack of timers made it problematic recovering from errors and difficulty synchronizing flowgraphs (cited was forcing a transmit immediately after a packet is received). In response, the authors ported the GNU Radio blocks to Click, a software architecture for building routers, to take advantage of its explicit scheduling capabilities.

BBN Technologies Corp. has proposed a number of extensions to GNU Radio for handling of packet-based data. Chief among these are a priority scheduler and message blocks or *m-blocks* [19]. In *m-blocks* data is exchanged between blocks as arbitrarily sized *messages* consisting of two sections, the sample data and associated *metadata*. The metadata defines a message priority for the scheduler and a signal event name describing if the data should be processed differently by other m-blocks, which is useful in a MAC protocol where different behavior is required to handle different types of packets. The metadata also lists block entry/exit timestamps controlling when the packet should be processed at other blocks in the flowgraph. Signal metadata can be exchanged between blocks for status reporting and passing control information. Development code can be evaluated in the GNU Radio trunk and *m-block* features will be integrated into the bundled block library for the next release of GNU Radio (v3.3) [28]. The greater packet control possible comes too late for this thesis.

A.4.2 Hardware Limitations

The USB2.0 connection between the USRP and host computer backed is an important performance bottleneck. The USRP ADC and DAC process 4-byte complex float samples at 64MS/s and 128MS/s respectively, but this cannot be fully exploited. Received signals must be decimated down to fit across the USB2.0 link and subsampled versions of waveforms to be transmitted must be sent across, that are then interpolated up. USB2.0 has a raw data transfer rate of 480Mbit/s or 60MB/s. Cypress FX2 controller signalling protocol overheads mean that 32MB/s is in practice the maximum sustainable rate. A received signal must be decimated by at least a factor of 8 by the USRP FPGA DDC and a transmitted signal interpolated by a factor of at least 16. The maximum Nyquist bandwidth observable using the USRP is $32\text{MB/s}/(4^*2\text{B}) = 4\text{MHz}$ set by the USB.

The restrictive bandwidth limits how GNU Radio can be used. For example, the IEEE 802.11 WLAN standard uses a direct spread spectrum signalling (DSSS) method at its physical layer with 22MHz Barker spread channels [62], far in excess of the imposed USB bandwidth limit. BBN Technologies Corp's [19] implementation of a GNU Radio 802.11b receiver as part of the DARPA ADROIT project subsamples the channel when it reduces the signal bandwidth it receives to 4MHz. The resulting system can receive from an 802.11 station but only at ¡1Mbps and with very low SNR. Hamed Firooz [80] implementation reconfigured the Altera FPGA to perform the Barker dispreading. Thus the full 22MHz signal could be reduced to 2MHz when sent over the USB link and the full signal received with improved performance.

Schmid et al [75] notes that using USB introduces significant latency in addition to the slow data rate. A USB packet is sent to or from the USRP only after a sufficient amount of data is collected at the buffer. The delay introduced by the USB is given by

$$\triangle_{\text{USBwait}} = \frac{f(512, \text{fusb\_nblocks}^* \text{fusb\_block\_size})}{\text{sample\_size}^* f_s}, \tag{A.1}$$

where $f(x, y)$ is the size of data at the buffer before a packet is sent, x being the least and y the most with fusb_nblocks and fusb_block_size specifiable. The smallest allowed packet is 512 bytes long. The denominator gives the rate at which data is being accumulated at the buffer. f_s is the sampling frequency and for complex float samples sample_size $= 2^*16$bits $= 4$bytes. Setting fusb_nblocks*fusb_block_size to 512 bytes gives the least latency but the larger it is the greater the achievable data rate as protocol overhead is reduced. Schmid et al settled on values of fusb_nblocks $= 8$, fusb_block_size $= 2048$ for a good tradeoff between USB protocol overhead and package payload. They measured a USB latency of 1ms using a sampling rate of 8MS/s up to 30ms when sampling at 250kS/s. The USB tended to wait until the maximum payload size was reached before transmitting. These latencies are too large to implement current wireless protocol standards such as IEEE 802.11 where timing is needed in the microseconds. The 802.11 SIFS (Short Inter Frame Space) interval is 28μs for comparison [20].

Extensive GNU Radio delay measurements were conducted by Nychis et al [61]. Table A.2 summarizes the round-trip time (RTT) of a ping request by GNU Radio to the USRP, broken up into stages. For the minimum fusb_nblocks*fusb_block_size $= 512$bytes the average time for the kernel user ping request to be sent across the USB to the USRP, and the reply sent back to the kernel was measured at 148μs. It was concluded that there is a significant USB setup time since the 512-byte USB latency should come out to as low as 16μs according to A.1. Process scheduling introduced high jitter in the kernel to user process (GNU Radio) time. The maximum observed time was 7000μs with an average time of 27μs.

	Avg	SDev	Min	Max
User->Kernel (μs)	24	10	22	213
Kernel->User (μs)	27	89	13	7000
4096 Kernel<->FPGA (μs)	291	62	204	360
512 Kernel<->FPGA (μs)	148	35	90	193
GNU Radio<->FPGA (μs)	612	789	289	9000

Table A.2: GNU Radio Kernel Level Delay Measurements, from [61]

The successor to the USRP, the USRP2 introduced 25 May, 2009, works to overcome this issue. It replaces the USB with an Ethernet connection (1Gb/s) and can look at 50MHz of RF data [12]. Other SDR designs are available on the market with superior transfer latency and bandwidth characteristics. The flexComm SDR-2000 [8] uses a PCI interface (500MB/s) with Direct Memory Access (DMA).

The issue of USB latency can be alleviated by moving functionality away from the host and to the FPGA. This approach is studied by Nychis et al [61]. In his *host-PHY* CSMA/CA MAC protocol architecture, matched filters are implemented on the USRP for detecting a DATA packet. This triggers the fast transmit of a premodulated ACK packet stored by the FPGA.

In the process of compiling this thesis document in January it was discovered Auras [17] had developed an optimization of the Cypress FX2 firmware raising the achievable receive only USB bandwidth to 45MB/s and removing the restriction of being able to transfer only multiples of 512 bytes. This occurred too late to be taken advantage of in our work.

A.5 Related Work using GNU Radio

GNU Radio is a common platform used in the scientific literature for CR research. Two centers of research employing GNU Radio are Carnegie

Mellon University's Departments of Computer Science and Electrical and Computer Engineering co-developing the CogNet project and the Center for Wireless Telecommunications, Virginia Tech. Work is also proceeding at multiple other locations prototyping physical layer CR capabilities on GNU Radio, for example spectrum sensing or demonstrations of SDRs with basic cognitive capabilities. At the same time little work has been published on implementing intelligence on the platform and exploring the quantified benefits on learning in a practical system.

The Center for Wireless Telecommunications, Virginia Tech have developed a control architecture for implementing CR functionality named "Cognitive Engine" (CE) [45]. The CE is built around a central cognitive controller that calls methods on attached modules to execute the cognition cycle. Examples of modules include a decision maker module and a knowledge base database. GNU Radio and USRP are being used for the adaptive radio and sensing modules. Modular separation simplifies research collaboration, updating and experimentation. A working Public Safety Cognitive Radio, capable of finding and use of public safety radio bands in the area without the user having to intervene, was implemented on the CE by Ge et al [36]. Rondeau [71] developed an intelligent secondary usage waveform synthesizer on the CE. It employs a genetic algorithm to design a transmission waveform optimizing performance metrics, such as bit error rate, depending on the spectrum environment. The GNU Radio platform would then adapt to meet the waveform specifications, including tuning the modulation type, transmit power, symbol rate, pulse shaping, frequency and packet size. Tests showed that a waveform optimal for the environment could be found within 400 GA generations.

Carnegie Mellon, Rutger and the University of Kansas are working on CogNet, a CR protocol stack. CogNet's unique feature is that it will specify a cognitive network layer, incorporating concepts such as forwarding incentives and cross-layer aware routing. Most current implementation research focuses on physical- and link-layer issues such as co-operative

spectrum sensing. At node-level the CogNet architecture is split into a data plane and global control plane (GCP) [69]. The data plane contains the physical layer radio elements for handling transmissions. The GCP controls and adapts the data plane through an API and co-ordinates with GCPs at other nodes. The protocol is being prototyped with GNU Radio implementing PHY layer functionality and the project is expected to finish late 2009.

Interest in prototyping with GNU Radio has led to academic publications evaluating the platform for this purpose and their experiences with it. Already mentioned was Manicka's thesis "GNU Radio Testbed" [52], which explores testing MAC protocols on GNU Radio and is useful as an introduction to the system. Platform latency is characterized by Schmid et al [75]. Nychis et al [61] considers the implications of the high latency in implementing a realistic MAC protocol and proposes FPGA workarounds. Rachel Dhar et al [31] describes their team's partially successful efforts getting an 802.11-esque RTS-CTS MAC protocol working on GNU Radio and, as a reaction to needing explicit scheduling, porting of the signal processing blocks to the Click software router architecture. Another extension to GNU Radio is added by Scaperoth [73] who wrote an XML interface overlay to store and pass physical layer configuration details to the platform.

O'Shea [63] explores spectrum sensing on GNU Radio. He implemented a cyclostationary feature detector-based sensor with the capability to distinguish between signals. In his design, a classifier block was trained using an artificial neural network on the spectral correlation density alpha-profiles of differently modulated signal received by a GNU Radio radio. Tests showed that the classifier was able to correctly recognize signals of each modulation type but when the signal frequency was changed the system would misidentify signals due to different frequency offsets slightly altering the SCD alpha-profile, which supplied the training data.

The implementation of cognitive dynamic spectrum access radios on GNU Radio is explored by Crohas [74] and Yan et al [92]. Crohas imple-

ments a GNU Radio transmitter and separate receiver to send and receive music packets on unused channels. The transmitter has a power detector for determining whether another user is present out of four 1MHz channels to choose for communications. The receiver is not equipped with a spectrum sensor. There is no explicit channel rendezvous protocol. Instead, when the transmitter vacates a channel because it is occupied, after waiting a timeout period in which no packets are received the receiver cycles between the four possible 1MHz channels in search of the transmitter. Data packets are sent at a greater bitrate than the music sampling rate so playback is uninterrupted during the handoff period when switching channels. The SUs implemented by Yan et al use OFDM transmissions, divided into five subcarriers across 2448-2452MHz. Before transmission the opportunistic user senses using energy detection and if a PU is detected, the overlapping subcarriers are not used in the transmission.

Appendix B

Power Detector Analysis

Results leading to the derivation of Equation 2.4, giving the design threshold for a constant false alarm rate (CFAR) power detector, are presented in this appendix.

A power detector measures the channel power spectral density. Using Welch's method, the PSD estimate is found as:

$$S_x(\omega) = \frac{1}{M} \sum_{M} P_N(\omega), \tag{B.1}$$

, where

$$P_N(\omega) = \frac{|Y[n]|^2}{N}, \tag{B.2}$$

is the periodogram, M the number of periodograms averaged, Y the DFT (Discrete Fourier Transform) of the signal at the detector and N the size of the DFT taken.

The power detector needs to distinguish between when the channel is free, which assuming an AWGN channel gives the signal at the sensor as

$$Y[n] = X[n], \tag{B.3}$$

or occupied by a signal

$$Y[n] = X[n] + W[n], \tag{B.4}$$

where X is the transmitted PU signal and W noise. W is real zero-mean AWGN noise with variance σ_w^2. The channel assumption is justified as in the WLAN 2.4GHz frequency range the primary source of noise is thermal noise generated at the receiver, to which the AWGN model applies [67]. In non-coherent detection [50], X can be considered a real zero-mean Gaussian random process with variance σ_x^2.

The PU signal is declared present if the PSD estimate exceeds a threshold γ. If the signal is not present but the threshold is exceeded by noise effects, this results in a *false alarm*. The probability the threshold test incorrectly decides the channel is occupied when it is not is denoted P_{fa}, the probability of *false alarm*. The *probability of detection*, P_d, is defined as the probability that the test detects a PU signal in the band when one truly is present.

It is known that for a normally distributed variable L, such as W, with mean μ and variance σ, L^2 has a non-central chi-square distribution with mean $\sigma^2 + \mu^2$ and variance $2\sigma^4 + 4\mu^2\sigma^2$. Thus, when the channel is free $P_N(\omega) = \frac{|W[n]|^2}{N}$ has mean σ_w^2/N and variance $2\sigma_w^4 + 4\mu^2\sigma^2$. The variable $X[n]+W[n]$ is also normally distributed with mean 0 and variance $\sigma_w^2+\sigma_x^2$, thus $P_N(\omega) = \frac{|Y[n]|^2}{N} = \frac{|X[n]+W[n]|^2}{N}$, giving the value of the periodogram when the channel is occupied, has mean $\frac{(\sigma_w^2+\sigma_x^2)^2}{N}$ and variance $\frac{2(\sigma_w^2+\sigma_x^2)^4}{N^2}$.

If the number of periodograms averaged, M, is large enough, the PSD estimate approaches a normal distribution according to the central limit theorem (CLT)

$S_x(f) \sim Normal(\frac{\sigma_w^2}{N}, \frac{2\sigma_w^4}{MN^2})$, channel free / PU signal absent

$S_x(f) \sim Normal(\frac{(\sigma_w^2+\sigma_x^2)^2}{N}, \frac{2(\sigma_w^2+\sigma_x^2)^4}{MN^2})$, channel occupied / PU signal present.

The detection and false alarm probabilities are then given by

$$P_d = Q(\frac{\lambda - \mu_{\text{occupied}}}{\sigma_{\text{occupied}}}) = Q(\frac{\lambda - \frac{(\sigma_w^2+\sigma_x^2)^2}{N}}{\sqrt{\frac{2(\sigma_w^2+\sigma_x^2)^4}{MN^2}}}) \tag{B.5}$$

$$P_{fa} = Q(\frac{\lambda - \frac{\sigma_w^2}{N}}{\sqrt{\frac{2\sigma_w^4}{MN^2}}}), \tag{B.6}$$

where Q is the normal distribution Q-function.

Rewriting Equation B.6, the threshold required to achieve a given false alarm probability is

$$\lambda = Q^{-1}(P_{fa})\sqrt{\frac{2\sigma_w^4}{MN^2}} + \frac{\sigma_w^2}{N}, \tag{B.7}$$

or simply

$$\lambda = \mu + \sigma Q^{-1}(P_{fa}), \tag{B.8}$$

where μ, σ are the mean and standard deviation of the PSD estimate when the PU signal is absent.

Bibliography

[1] Entering the World of GNU Radio. https://radioware.nd.edu/documentation/basic-gnuradio/entering-the-world-of-gnu-software-radio.

[2] Ettus Research LLC. http://www.ettus.com/.

[3] FPGA. http://gnuradio.org/redmine/wiki/gnuradio/UsrpFAQIntroFPGA.

[4] GNU Radio. http://gnuradio.org/redmine/wiki/gnuradio/.

[5] GNU Radio - BBN Technologies Internetwork Research ADROIT Project. http://gnuradio.org/redmine/wiki/1/BBN_Technologies_Internetwork_Research_ADROIT_Project.

[6] GNU Radio 3.2svn C++ API Documentation. http://gnuradio.org/doc/doxygen/index.html.

[7] Radio Spectrum Allocations in New Zealand. http://www.rsm.govt.nz/cms/pdf-library/policy-and-planning/spectrum_chart.pdf.

[8] SDR-2000 PCI/PCI-X Software Defined Radio Platform. http://www.spectrumsignal.com/products/pci-x/sdr_2000.asp.

[9] Simplified Wrapper and Interface Generator. `http://www.swig.org/`.

[10] USRP - Frequently Asked Questions. `http://gnuradio.org/redmine/wiki/gnuradio/UsrpFAQ`.

[11] USRP Docs. `http://gnuradio.org/redmine/wiki/gnuradio/USRP`.

[12] USRP family brochure. `http://www.ettus.com/downloads/er_broch_trifold_v5b.pdf`.

[13] Establishment of interference temperature metric to quantify and manage interference and to expand available unlicensed operation in certain fixed mobile and satellite frequency bands. ET Docket 03-289, Notice of Inquiry and Proposed Rulemaking, 2003.

[14] Notice of Proposed Rule Making and Order. ET Docket No 03-222, December 2003.

[15] Notice of proposed rule-making and order: Facilitating opportunities for flexible, efficient, and reliable spectrum use employing cognitive radio technologies. ET Docket No. 03-108, February 2005.

[16] AKYILDIZ, I. F., LEE, W.-Y., VURAN, M. C., AND MOHANTY, S. NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer Networks* (2006), 2127 – 2159.

[17] AURAS, D. Improved FX2 firmware for USRP1 supporting higher USB bandwidth. `http://www.dominikauras.de/gnuradio/usrp_fx2.html`, September 2009.

[18] BAEK, J.-H., OH, H.-J., AND HWANG, S.-H. Improved Reliability of Spectrum Sensing using Energy Detector in Cognitive Radio System. In *ICACT 2008: 10th International Conference on Advanced Communication Technology* (February 2008), pp. 575 –578.

[19] BBN TECHNOLOGIES CORP. GNU Radio Architectural Changes. http://acert.ir.bbn.com/downloads/adroit/gnuradio-architectural-enhancements-3.pdf, 2006.

[20] BRENNER, P. A Technical Tutorial on the IEEE 802.11 Protocol. www.sss-mag.com/pdf/802_11tut.pdf, July 1996.

[21] CENTER FOR COMPUTER RESEARCH IN MUSIC AND ACOUSTICS (CCRMA), STANFORD UNIVERSITY. The Periodogram. https://ccrma.stanford.edu/~jos/sasp/Periodogram.html.

[22] CHEN, R., AND PARK, J.-M. Ensuring trustworthy spectrum sensing in cognitive radio networks. In *Proc. IEEE Workshop on Networking Technologies for Software Defined Radio Networks (held in conjunction with IEEE SECON 2006)* (September 2006).

[23] CI, S., AND SONNENBERG, J. A Cognitive Cross-Layer Architecture for Next-Generation Tactical Networks. In *MILCOM 2007: IEEE Military Communications Conference* (October 2007), pp. 1 –6.

[24] CLANCY, T. Achievable capacity under the interference temperature model. In *Proc. IEEE INFOCOM* (Anchorage, AK, May 2007), pp. 794–802.

[25] COOLEY, J. E. A Day in the Life of the RF Spectrum. MSc thesis, Massachusetts Institute of Technology, September 2005.

[26] CORDEIRO, C., CHALLAPALI, K., BIRRU, D., AND SHANKAR, S. IEEE 802.22: An Introduction to the First Wireless Standard based on Cognitive Radios. *Journal of communications* (2006), 38–47.

[27] CORDEIRO, C., AND CHALLPALI, K. C-MAC: A cognitive MAC protocol for multichannel wireless networks. In *Proc. IEEE DySPAN* (April 2007), pp. 147–157.

[28] CORGAN, J. GNU Radio Release 3.2 available for download or binary installation. `http://www.ruby-forum.com/topic/187710#new`, Forum contribution, May 2009.

[29] CORGAN, J. High packet error and reception problems with XCVR 2450. `http://www.ruby-forum.com/topic/104146#new`, Forum contribution, June 2009.

[30] CORMIO, C., AND CHOWDHURY, K. R. A survey on MAC protocols for cognitive radio networks. *Ad Hoc Netw.* (2009), 1315–1329.

[31] DHAR, R., GEORGE, G., MALANI, A., AND STEENKISTE, P. Supporting Integrated MAC and PHY Software Development for the USRP SDR. In *Proc. First IEEE Workshop on Networking Technologies for Software Defined Radio (SDR) Networks (Held in Conjunction with IEEE SECON 2006)* (September 2006).

[32] DIXON, R. C. *Radio receiver design*. CRC Press, 1998, ch. Appendix 3 Gray Codes, p. 397.

[33] ETTUS, M. USRP User's and Developer's Guide. `http://home.ettus.com/usrp/usrp_guide.html`.

[34] GARDNER, W. A. *Statistical spectral analysis: a nonprobabilistic theory*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1986, pp. 355–380.

[35] GAUTHIER, M. Wireless Networking in the Developing World. `http://wndw.net`, November 2009.

[36] GE, F., CHEN, Q., WANG, Y., BOSTIAN, C., RONDEAU, T., AND LE, B. Cognitive Radio: From Spectrum Sharing to Adaptive Learning and Reconfiguration. In *IEEE Aerospace Conference* (March 2008), pp. 1 –10.

[37] GILMORE, J. HDTV and the USRP. `http://lists.gnu.org/archive/html/discuss-gnuradio/2004-07/msg00034.html`, July 2004.

[38] GREENWALD, A., JAFARI, A., AND MARKS, C. A General Class of No-Regret Learning Algorithms and Game-Theoretic Equilibria. In *Proc. Computational Learning Theory Conference* (2003), pp. 1–11.

[39] GRINSTEAD, C. M., AND SNELL, J. L. *Introduction to Probability*, second revised ed. American Mathematical Society, 1997, p. 438.

[40] HILLIER, F. S., AND LIEBERMAN, G. *Introduction to Operations Research*, eighth ed. McGraw-Hill, 2005, p. 797.

[41] HOYHTYA, M. AND POLLIN, S. AND MAMMELA, A. Performance improvement with predictive channel selection for cognitive radios. In *CogART 2008: First International Workshop on Cognitive Radio and Advanced Spectrum Management* (February 2008), pp. 1 –5.

[42] JIANG, T., GRACE, D., AND LIU, Y. Performance of cognitive radio reinforcement spectrum sharing using different weighting factors. In *ChinaCom 2008: Third International Conference on Communications and Networking in China* (August 2008), pp. 1195 –1199.

[43] JONES, H. W. H. Channel Sounding with Software Defined Radio. Honors project, Victoria University of Wellington, October 2009.

[44] KOS, A., HOMAN, P., SLIVNIK, T., AND BESTER, J. Performance Evaluation of a Synchronous Bulk Packet Switch under Real Traffic Conditions. *Computer Networks* (1998), 2309–2326.

[45] LE, B., RODRIGUEZ, F. A. G., CHEN, Q., LI, B. P., GE, F., NAINAY, M. E., RONDEAU, T. W., AND BOSTIAN, C. W. A Public Safety Cognitive Radio Node. In *Proc. SDR Forum Technical Conference* (2007).

[46] LEHTOMAKI, J., VARTIAINEN, J., JUNTTI, M., AND SAARNISAARI, H. Spectrum Sensing with Forward Methods. *MILCOM* (2006), 1–7.

[47] LESLIE PACK KAELBLING, MICHAEL LITTMAN, A. M. Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research* (1996), 237–285.

[48] LI, H. Multi-agent Q-learning of channel selection in multi-user cognitive radio systems: a two by two case. In *SMC'09: Proceedings of the 2009 IEEE international conference on Systems, Man and Cybernetics* (Piscataway, NJ, USA, 2009), IEEE Press, pp. 1893–1898.

[49] LIANG, Y.-C., ZENG, Y., PEH, E., AND HOANG, A. T. Sensing-Throughput Tradeoff for Cognitive Radio Networks. In *ICC '07: IEEE International Conference on Communications* (June 2007), pp. 5330 –5335.

[50] LIN, W., AND ZHANG, Q. A design of energy detector in cognitive radio under noise uncertainty. In *Proc. ICCS* (Nov. 2008), pp. 213–217.

[51] LUO, T., MOTANI, M., AND SRINIVASAN, V. CAM-MAC: A cooperative asynchronous multi-channel MAC protocol for ad hoc networks. *3rd Intl. Conference on Broadband Communications, Networks and Systems (BROADNETS06)* (October 2006).

[52] MANICKA, N. GNU Radio Testbed. Msc thesis, University of Delaware, 2007.

[53] MCHENRY, M. NSF Spectrum occupancy measurements project summary. Subcontract FY2004-013, Shared Spectrum Company, 1595 Spring Hill Road, Suite 110, Vienna, VA 22182, August 2005.

[54] MCHENRY, M., STEADMAN, K., LEU, A., AND MELICK, E. XG DSA Radio System. In *DySPAN 2008: 3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks* (October 2008), pp. 1 –11.

[55] MITOLA, J. *Cognitive radio: an integrated agent architecture for software defined radio*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2000.

[56] MITOLA, J. AND MAGUIRE, G. Q. NTC-92. In *Telesystems Conference* (Washington D.C, USA, May 1992), pp. 13/15–13/25.

[57] MITOLA, J. AND MAGUIRE, G. Q. Cognitive radio: making software radios more personal. *Personal Communications, IEEE* (1999), 13–18.

[58] N. CHOI, Y. SEOK, Y. C. Multi-channel MAC protocol for mobile ad hoc networks. *Vehicular Technology Conference* (October 2003), 1379–1382.

[59] NIE, J., AND HAYKIN, S. A Dynamic Channel Assignment Policy through Q-Learning. *IEEE Transactions on Neural Networks* (1999), 1443–1455.

[60] NIE, N., AND COMANICIU, C. Adaptive channel allocation spectrum etiquette for cognitive radio networks. *Mob. Netw. Appl.* (2006), 779–797.

[61] NYCHIS, G., HOTTELIER, T., YANG, Z., SESHAN, S., AND STEENKISTE, P. Enabling MAC protocol implementations on software-defined radios. In *NSDI'09: Proceedings of the 6th USENIX symposium on Networked systems design and implementation* (Berkeley, CA, USA, 2009), USENIX Association, pp. 91–105.

[62] OHRTMAN, F., AND ROEDER, K. *Wi-Fi handbook: building 802.11b wireless networks*. McGraw-Hill, 2003, pp. 16–17.

[63] O'SHEA, T., CLANCY, T., AND EBEID, H. Practical Signal Detection and Classification in GNUradio. In *SDR Forum Technical Conference* (November 2007).

[64] PANAIT, L., AND LUKE, S. Cooperative Multi-Agent Learning: The State of the Art. *Autonomous Agents and Multi-Agent Systems* (2005), 387–434.

[65] PAXSON, V., AND FLOYD, S. Wide-area traffic: the failure of Poisson modeling. In *SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications* (New York, NY, USA, 1994), ACM, pp. 257–268.

[66] PROAKIS, J. *Digital Communications*, fifth ed. McGraw-Hill Inc., 2007, pp. 607–609.

[67] PROAKIS, J. *Digital Communications*, fifth ed. McGraw-Hill Inc., 2007, pp. 8–10.

[68] QIN, H., WANG, H., AND ZHOU, H. A Selfish Game-Theoretic Approach for Cognitive Radio Networks with Dynamic Spectrum Sharing. In *CSSE '08: Proceedings of the 2008 International Conference on Computer Science and Software Engineering* (Washington, DC, USA, 2008), IEEE Computer Society, pp. 1105–1109.

[69] RAYCHAUDHURI, D., MANDAYAM, N. B., EVANS, J. B., EWY, B. J., SESHAN, S., AND STEENKISTE, P. CogNet: an architectural foundation for experimental cognitive radio networks within the future internet. In *MobiArch '06: Proceedings of first ACM/IEEE international workshop on Mobility in the evolving internet architecture* (New York, NY, USA, 2006), ACM, pp. 11–16.

[70] RODRIGUES GOMES, E., AND KOWALCZYK, R. Dynamic analysis of multiagent Q-learning with ϵ-greedy exploration. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning* (New York, NY, USA, 2009), ACM, pp. 369–376.

[71] RONDEAU, T. W. *Application of Artificial Intelligence to Wireless Communication*. PhD thesis, Virginia Polytechnic Institute and State University, Virginia, September 2007.

[72] SAHAI, A., AND CABRIC, D. Cyclostationary feature detection. `http://www.eecs.berkeley.edu/sahai/Presentations/DySPAN05part2.ppt`, Tutorial presented at the IEEE DySPAN 2005 (Part II), November 2005.

[73] SCAPEROTH, D. A. Configurable SDR Operation for Cognitive Radio Applications using GNU Radio and the Universal Software Peripheral. Msc thesis, Virginia Polytechnic Institute and State University, Virginia, May 2007.

[74] SCAPEROTH, D. A. Practical Implementation of a Cognitive Radio System for Dynamic Spectrum Access. Msc thesis, University of Notre Dame, Indiana, July 2008.

[75] SCHMID, T., SEKKAT, O., AND SRIVASTAVA, M. B. An experimental study of network performance impact of increased latency in software defined radios. In *WinTECH '07: Proceedings of the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization* (New York, NY, USA, 2007), ACM, pp. 59–66.

[76] SHANKAR, N. S., CORDEIRO, C., AND CHALLAPALI, K. Spectrum agile radios: utilization and sensing architectures. In *DySPAN 2005: First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks* (2005), pp. 160–169.

[77] SHEN, D. GNU Radio Tutorial Series. `http://www.snowymtn.ca/GNURadio/GNURAdioDoc-1.pdf`, May 2006.

[78] SKINNER, B., INGELS, F., AND DONOHOE, J. Stationary and cyclostationary random process models. In *Proc. IEEE Southeastcon '94.*

'Creative Technology Transfer - A Global Affair'. (April 1994), pp. 450 –454.

[79] SONG, Y., FANG, Y., AND ZHANG, Y. Stochastic Channel Selection in Cognitive Radio Networks. In *IEEE Globecom'07* (2007).

[80] SPAN LAB AT THE UNIVERSITY OF UTAH. Implementation of Full-Bandwidth 802.11b Receiver. `http://span.ece.utah.edu/pmwiki/pmwiki.php?n=Main.80211bReceiver`.

[81] STOIANOVICI, V., POPESCU, V., AND MURRONI, M. A Survey on Spectrum Sensing Techniques for Cognitive Radio. `http://vega.unitbv.ro/~popescu/university20bulletin202008_final_VI.pdf`.

[82] SUN, C., ZHANG, W., AND LETAIEF, K. Cooperative Spectrum Sensing for Cognitive Radios under Bandwidth Constraints. In *WCNC 2007: IEEE Wireless Communications and Networking Conference* (March 2007), pp. 1 –5.

[83] THOPPIAN, M., VENKATESAN, S., AND PRAKASH, R. CSMA-Based MAC Protocol For Cognitive Radio Networks. *International Symposium on a World of Wireless, Mobile and Multimedia Networks* (2007), 1–8.

[84] TUYLS, K., VERBEECK, K., AND LENAERTS, T. A selection-mutation model for q-learning in multi-agent systems. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems* (New York, NY, USA, 2003), ACM, pp. 693–700.

[85] VALERIO, D. Open Source Software-Defined Radio: A survey on GNUradio and its applications. Tech. Rep. FTW-TR-2008-002, Telecommunications Research Center Vienna (ftw.), August 2008.

[86] WANG, B., JI, Z., LIU, K. J. R., AND CLANCY, T. C. Primary-prioritized Markov approach for dynamic spectrum allocation. *IEEE Trans. Wireless. Comm.* (2009), 1854–1865.

[87] WANG, J. C.-P., ABOLHASAN, M., SAFAEI, F., AND FRANKLIN, D. A Survey on Control Separation Techniques in Multi-Radio Multi-channel MAC Protocols. In *ISCIT 2007: 7th International Symposium on Communications and Information Technologies* (October 2007), Research Online.

[88] WANG, Z., AND SALOUS, S. Time series arima model of spectrum occupancy for cognitive radio. In *IET Seminar on Cognitive Radio and Software Defined Radios: Technologies and Techniques* (September 2008), pp. 1 –4.

[89] WATANABE, K., ISHIBASHI, K., AND KOHNO, R. An Analysis of Spectrum Management for Coexistence of Fixed and Cognitive Radio Systems. In *2006 Software Defined Radio Technical Conference* (November 2006).

[90] WATKINS, C. J., AND DAYAN2, P. Q-Learning. *Machine Learning* (1992), 279–292.

[91] WILKINS, D., DENKER, G., STEHR, M.-O., ELENIUS, D., SENANAYAKE, R., AND TALCOTT, C. Policy-Based Cognitive Radios. *Wireless Communications, IEEE* (August 2007), 41 –46.

[92] YAN, Z., MA, Z., CAO, H., LI, G., AND WANG, W. Spectrum Sensing, Access and Coexistence Testbed for Cognitive Radio using USRP. In *ICCSC 2008: 4th IEEE International Conference on Circuits and Systems for Communications.* (May 2008), pp. 270 –274.

[93] YANG, L., CAO, L., AND ZHENG, H. Proactive channel access in dynamic spectrum networks. *Physical Communication* (2008), 103 – 111.

[94] YAU, A.-L., KOMISARCZUK, P., AND TEAL, P. C2net: A Cross-Layer Quality of Service (QoS) Architecture for Cognitive Wireless Ad Hoc Networks. In *ATNAC 2008: Australasian Telecommunication Networks and Applications Conference* (December 2008), pp. 306 –311.

[95] YAU, A.-L., KOMISARCZUK, P., AND TEAL, P. A Survey on Multi-channel Medium Access Control (MAC) Protocols: A Cognitive Radio Perspective. In *NZCSRSC'09: 7th New Zealand Computer Science Research Student Conference* (April 2009).

[96] YAU, K.-L., KOMISARCZUK, P., AND TEAL, P. A context-aware and Intelligent Dynamic Channel Selection scheme for cognitive radio networks. In *CROWNCOM '09: 4th International Conference on Cognitive Radio Oriented Wireless Networks and Communications* (June 2009), pp. 1 –6.

[97] YAU, K.-L., KOMISARCZUK, P., AND TEAL, P. Performance analysis of Reinforcement Learning for achieving context-awareness and intelligence in Cognitive Radio networks. In *LCN 2009: IEEE 34th Conference on Local Computer Networks* (October 2009), pp. 1046 –1053.

[98] YUCEK, T., AND ARSLAN, H. A survey of spectrum sensing algorithms for cognitive radio applications. *Communications Surveys & Tutorials, IEEE* (March 2009), 116–130.

[99] ZHAO, J., ZHENG, H., AND YANG, G.-H. Distributed coordination in dynamic spectrum allocation networks. In *DySPAN 2005: First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks* (November 2005), pp. 259 –268.

[100] ZHENG, K., LI, H., DJOUADI, S. M., AND WANG, J. Spectrum Sensing in Low SNR Regime via Stochastic Resonance. *CoRR* (2009).