Considerations of Reinforcement Learning within Real-Time Wireless Communication Systems

Alyse M. Jones

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

William C. Headley, Chair

R. Michael Buehrer

Ryan K. Williams

June 15, 2022

Blacksburg, Virginia

Keywords: Wireless Communications, Reinforcement Learning,

Intelligent Radio, Spectrum Avoidance

Considerations of Reinforcement Learning within Real-Time Wireless Communication Systems

Alyse M. Jones

(ABSTRACT)

Afflicted heavily by spectrum congestion, the unpredictable, dynamic conditions of the radio frequency (RF) spectrum has increasingly become a major obstacle for devices today. More specifically, a significant threat existing within this kind of environment is interference caused by collisions, which is increasingly unavoidable in an overcrowded spectrum. Thus, these devices require a way to avoid such events. Cognitive radios (CR) were proposed as a solution through its transmission adaptability and decision-making capabilities within a radio. Through spectrum sensing, CRs are able to capture the current condition of the RF spectrum and based on its decision-making strategy, interpret these results to make an informed decision on what to do next to optimize its own communication. With the emergence of artificial intelligence, one such decision-making strategy CRs can utilize is Reinforcement Learning (RL). Unlike standard adaptive radios, CRs equipped with RL can predict the conditions of the RF spectrum, and using these predictions, understand what it must do in the future to operate optimally.

Recognizing the usefulness of RL in hard-to-predict environments, such as the RF spectrum, research of RL within CRs have become more popular over the past decade, especially for interference mitigation. However, the existing literature neglects to confront the possible limitations that pose a threat to the proper implementation of RL in RF systems. Therefore, this thesis is motivated to investigate what limitations in real-time communication systems can hinder the performance of

RL, and as a result of these limitations, emphasize the considerations that should be a focus in the design and implementation of radio frequency reinforcement learning (RFRL) systems. The effects of latency, power, wireless channel impairments, different transmission protocols, and different spectrum sensing detectors are among the possible limitations simulated and analyzed within this work that are not typically considered within simulation-based prior art. To perform this investigation, a representative real-time OFDM transmit/receive chain is implemented within the GNU Radio framework. The system, operating over-the-air through USRPs, leverages reinforcement learning, e.g. Q-Learning, in order to avoid interference with other spectrum users. Performance analysis of this representative system provides a systematic approach for helping to predict limiting factors within an implemented real-time system and thus, aim to provide guidance on how to design these systems with these practical limitations in mind.

Considerations of Reinforcement Learning within Real-Time Wireless Communication Systems

Alyse M. Jones

(GENERAL AUDIENCE ABSTRACT)

Because the space in which communication signals travel is congested with activity, collisions among signals, called interference, is becoming more of a problem in wireless communications. Therefore, to avoid such an occurrence, intelligent radios are used to adapt communication devices to operate optimally within this congested space. With the emergence of artificial intelligence, where devices can learn on their own how to adapt, one such way an intelligent radio can dynamically adapt to the congestion is through Reinforcement Learning (RL), which enables prediction of signal activity within the communication space over time. Intelligent radios equipped with RL learn through trial-and-error how to operate optimally within the communication space to avoid places within the communication space that are busy and congested.

Recognizing the usefulness of RL in hard-to-predict environments, research of RL within intelligent radios have become more popular over the past decade, especially for navigating a communication space that is congested and where collisions are common. However, existing literature neglects to confront the possible limitations that pose a threat to the proper implementation of RL in communication systems. Therefore, this thesis is motivated to investigate what limitations in real-world communication systems can hinder the performance of RL, and as a result of these limitations, emphasize the considerations that should be a focus in the design and implementation of communication systems equipped with RL. Effects, such as delays in the system, differences

in how the signal operates, and how the signal is affected while it is traveling, are among the possible limitations simulated and analyzed within this work that are not typically considered within prior art. To perform this investigation, a modern representative communication system was implemented within software-enabled radios. The system leverages reinforcement learning in order to avoid collisions with other signals in the communication space. Performance analysis of this representative system provides a systematic approach for helping to predict limiting factors within an implemented real-world communication system with RL and thus, aim to provide guidance on how to design these systems with these practical limitations in mind.

Dedication

To my family.

Acknowledgments

First, I would like to thank my advisor, Dr. Chris Headley, for his guidance, expertise, and patience. With his help, I was able to learn so much during these past two years and grow as a researcher in our field. I would also like to thank Dr. Daniel Jakubisin for his guidance as well, particularly for his expertise in 5G-based communication systems. I would like to thank Dr. Joseph Gaeddert, Dr. Bradley Davis, Dr. Daniel Jakubisin, and Megan Moore for their time and advice given during my practice defense. I am also thankful for Megan and Dr. Headley for reading my thesis and giving me feedback and advice on how to make it better. I would also like to thank my committee for all their support in the completion of my thesis. Lastly, I would like to thank my family, friends, and fiance for their support of me this past year. It was a stressful time, and I thank them for their patience, understanding, and love.

Contents

List of Figures

List of Tables

List of Abbreviations

5G NR 5G New Radio

AGC Automatic Gain Control

AWGN Additive White Gaussian Noise

CR Cognitive Radio

CRC Cyclic Redundancy Check

DM-RS Demodulation Reference Signals

DRL Deep Reinforcement Learning

DSA Dynamic Spectrum Access

DSSS Direct Sequence Spread Spectrum

FCC Federal Communications Commission

FEC Forward Error Correction

FH Frequency Hopping

FHSS Frequency Hopping Spread Spectrum

IFFT Inverse Fast Fourier Transform

IQL Independent Q-learning

IQR Interquartile Range

ISI Intersymbol Interference

ISM Industrial, Scientific, and Medical

JMAA Joint Multi-agent Anti-jamming Algorithm

LO Local Oscillator

MARL Multi-Agent Reinforcement Learning

MDP Markov Decision Process

MIMO Multiple Input Multiple Output

ML Machine Learning

OFDM Orthogonal Frequency-Division Multiplexing

OODA Observe-Orient-Decide-Act

OTA Over-the-Air

PAPR Peak-to-Average Power Ratio

PSD Power Spectral Density

RF Radio Frequency

RL Reinforcement Learning

RX Receiver

SDR Software Defined Radio

SINR Signal to Interference plus Noise Ratio

SNR Signal-to-Noise Ratio

TX Transmitter

UE User Equipment

WBSS Wideband Spectrum Sensing

Chapter 1

Introduction

With the ever growing amount of wireless devices, the radio frequency (RF) spectrum has become increasingly more congested [1]. In such an environment, it is becoming more difficult for devices to share the spectrum, where interference caused by collisions are inevitable and unavoidable for traditionally operated radios. This is especially true for tactical military communications, where there also exists malicious users who wish to exploit the spectrum for themselves and/or prevent others from using the spectrum [2]. With this RF environment, traditional radios are left vulnerable to interference and unfavorable spectrum conditions, with no way to avoid malicious users or to properly utilize the full spectrum. Proposed by Joseph Mitola [3], cognitive radios (CR) were presented as a hopeful solution to spectrum scarcity and congestion in an effort to make the RF spectrum a more accessible, efficient environment.

According to the Federal Communications Commission (FCC), CRs are devices "that

can change its transmitter parameters based on interaction with the environment in which it operates" [4]. Note that the FCC's definition does not explain the extent of the word *cognitive* in cognitive radio. Since Mitola proposed CRs in 1999 [3], the literature has made a cognitive radio more synonymous with dynamic spectrum access (DSA) technology, which requires only spectrum sensing and not necessarily any cognition, according to [5]. The authors in [5] explain further that a CR can actually have different levels of cognition. In this thesis, the terms *tunable*, *adaptive*, and *smart* are used to describe and differentiate the capabilities of the types of cognitive radios discussed in this work. For future reference, a *tunable* CR refers to a CR that can only adapt its transmission parameters, such as carrier frequency and modulation, based on some given, fixed strategy. An *adaptive* CR refers to a CR that can perform spectrum sensing, and based on those results, determine a suitable action. This kind of CR makes decisions based on its sensing results. Lastly, a *smart* CR refers to a CR that can predict and anticipate future requirements of the radio based on the changing conditions of the environment. Therefore, a *smart* CR can adjust to dynamic, unforeseen behaviors in the environment and learn to adjust the radio such that it performs optimally over time. This kind of CR is typically equipped with machine learning (ML) techniques to make these predictions.

At its most basic cognitive level (e.g. tunable), a CR user can better utilize the spectrum by manually adjusting its parameters to more efficiently operate within its channel. Often times, however, a CR must also switch to other channels if current channel conditions become poor and/or priority of that band is given to another user. To find other suitable channels,

it must also be able to find channels in good condition or channels that are empty (e.g. spectrum holes). An adaptive CR is able to do so through spectrum sensing, where signal detection methods are employed to detect channels with little to no user activity. As a result, the hope is that the spectrum becomes less challenging to navigate and results in more efficient communication for users who have these cognitive abilities. However, though tunable and adaptive CRs enable more efficient spectrum utilization, the spectrum can still be a harsh environment, with collisions still possible. Through spectrum sensing alone, an adaptive CR cannot avoid interference entirely because it cannot *predict* the movement of other users. Additionally, these radios cannot control the behavior of external factors, such as the malicious intent of other users and/or a drastic change in the RF environment. For an adaptive CR, a common fix is to hop channels to avoid operating on such a channel. However, an adaptive CR can still find itself on a channel with interference despite sensing at that moment in time that the channel was empty. For a spectrum with heavy traffic, this has become a more prevalent problem, regardless of the movement of other users causing interference is malicious or unintentional.

As a result, research has expanded to making CRs more protected against interference, particularly against a category of malicious users called jammers, as their efforts are often more directed and thus more harmful to a CR. For tunable and adaptive CRs, Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS) are classic methods to mitigating jammers [6] that these CRs can utilize. These methods can be implemented by radios as a countermeasure against malicious users and thus, may be employed

by a CR when it notices the presence of a jammer. Both FHSS and DSSS are efficient at mitigating unintentional narrowband interference and partial-band jammers. However, both methods have weaknesses against users that are versatile in their behaviors and/or have intelligent capabilities. Though an improvement, FHSS is still susceptible to accidental collisions, is energy inefficient from the hopping, and is vulnerable to users that can decode its hopping pattern [7]. Though better than FHSS, DSSS is not as effective against broadband interference [8]. A vulnerability to both methods, however, is their lack of intelligent adaptability in response to rapidly changing dynamics of the environment. Therefore, even with the option to employ these interference mitigation methods, these CRs still face the same dilemma discussed before, where interference is likely when the CR does not understand the behavioral patterns of external factors, such as jammers and consistently occurring poor channel conditions, that affect the RF environment. Thus, without the ability to *predict*, it will be hard for the CR to meet its performance objective. Therefore, a CR with a higher level of cognition is required for the CR to understand the dynamics and behavior of external factors affecting the environment. This desired level of cognition can be achieved through Reinforcement Learning (RL), making the CR a *smart* CR. RL enables predictive cognition in its functionality, allowing devices the opportunity to optimally operate in an unstable RF spectrum. For interference mitigation, this means RL can help CRs better learn the hard-to-predict behaviors of external factors causing interference within the RF spectrum that would otherwise be hidden to spectrum sensing. Spectrum sensing can only determine the *current* condition of the RF environment, but RL can predict the *future* con-

dition, even with limited knowledge. At the very least, the CR requires two things for the proper implementation of RL - spectrum sensing results identifying the presence of other users in the environment and action performance feedback from its receiver indicating if it has been interfered with. RL uses this information to infer the behavior patterns of other users and/or the conditions of the spectrum to *predict* areas that could be harmful to the CR. This enables the *smart* CR to *intelligently adapt* to the current environment, rather than being fixed to one implementation, such as is the case for FHSS and DSSS and CRs with lower cognition levels. Thus, through its prediction characteristic, RL can prove useful for resilient interference mitigation.

RL for interference mitigation has been researched more over the past decade, with a plethora of contributions made to the scientific community. This includes the formulation of a now well-known Markov Decision Process (MDP) framing for interference mitigation problems and its use in well-known RL algorithms, such as Q-learning ([9]–[11]). Additionally, researchers have proved that RL for spectrum avoidance works in a simulation setting, of which they focused primarily on improving the RL algorithms' performance using state-of-the-art methods. Recently, research has expanded to testing RL for interference mitigation over real hardware to prove RL can achieve convergent, optimal behavior over real radios and not just in computer simulations. However, existing literature does not consider any limitations that could affect the performance of RL in a real setting. In the presence of unavoidable limitations, such as latency and imperfect sensing, that plague real systems, there is yet to be an analysis of what sacrifices are made to realize a radio system with RL

such that RL works perfectly as intended.

Thus, this thesis is motivated to investigate how limitations inherent to real systems can affect the overall performance of the system's RL. To do so, an RL-based Orthogonal-Frequency Division Multiplexing (OFDM) transmitter-receiver framework was created in GNU Radio to test the effects of different limitations and how they effect RL. Additionally, by prioritizing convergent behavior obtained with minimal errors in sensing, and no delays in feedback, it can be determined what within the communication system is adversely affected by the implementation of RL, and as a result, how it can be mitigated and avoided. Of particular interest is the analysis of the accuracy of RL due to latency limitations in real-time systems and the inherent trade-offs in parameters such as goodput and duty cycle that occur from integrating RL into the system. More specifically, in this thesis, a real-time over-the-air (OTA) OFDM-based communication system designed with RL for spectrum avoidance to solve interference mitigation was implemented and tested over Universal Software Radio Peripheral (USRP) radios, over both wired and wireless channel mediums, to study these effects on the performance of RL. Through the designed framework, Q-learning performance was verified, and the system was then used to study the trade-offs a reinforcement learning system requires for proper operation.

Through testing and analysis, it was found that latency, wireless channel impairments, and limits to hardware memory can hinder the operation of RL in radio frequency reinforcement learning (RFRL) systems. For latency, a trade-space study was helpful in determining how latency can cause a problem and also helpful in setting the right parameters for the

system to ensure RL accuracy. For example, the trade-space found that to prioritize accuracy in RL, duty cycle and goodput of the transmitted OFDM signal must be lowered as the signal's sampling rate increases. For wireless channel impairments, because the designed framework was not implemented with multipath and Doppler shift mitigating techniques, the system was found vulnerable to propagation effects, as expected. However, if the transmission power is high enough, the effect of multipath can be lowered within this framework. Additionally, the framework was also able to handle low Doppler shifts due to the implemented duty cycle, which helped slow the effects of Doppler. Through the findings of this thesis, it was shown that this kind of analysis can help in predicting the practical limitations and considerations that occur in real-time RFRL systems. As a result, RF engineers can gain a better understanding of how radios with RL will behave in a real situation and how it affects the communication system. By predicting its weak points, they can then move closer to a system that can be instantly deployable, a trait that has long been desirable in the design of CRs [12].

1.1 Contributions

The contributions of this thesis are summarized as follows:

- In Chapters 3 to 7, the implementation of a real-time Orthogonal Frequency Division Multiplexing (OFDM) communication system equipped with RL for interference mitigation within GNU Radio for USRP operation is developed to test wired and wireless

(OTA) channel mediums. This framework was implemented with the flexibility of implementing different protocols and different spectrum sensing detectors, as was not considered in prior art. Additionally, the implemented duty cycle, which has not yet been considered in existing literature for RL-based spectrum avoidance, was added to measure and mitigate latency.

  - Presented in Chapter 4, the incorporation of comparative global communication protocols, such as IEEE 802.11a and 5G New Radio (5G NR), within the implemented framework.

  - Presented in Chapter 6, the incorporation of a polyphase channelizer detector as an option for spectrum sensing within the implemented framework. This detector has not yet been considered in prior art but is in this thesis because it is an efficient way to break up the wideband spectrum into sub-channels.

- Presented in Chapter 8, an analysis of different effects not typically considered, such as latency, power, distance, fading, and memory requirements, that can cause limitations to the optimal convergent performance of an RFRL system.

  - For latency, a trade-space analysis is formulated to predict the operating range of different transmission parameters, such as duty cycle and goodput, that result in optimal RL performance. It was found that the trade-space helped in predicting the operating range of parameters such that optimal RL performance is maintained. Here, the parameter was duty cycle, which was adjusted based on the

rate of individual components, such as signal sampling rate. To ensure accurate RL, duty cycle was required to be raised when faster signal sampling rates were used.

  - For power and distance between the transmitter and receiver, varying of the powers and distances is performed to understand the impact of noise on RL. It was found that multipath is present in the natural testing environment, causing poor RL performance for a low power, larger distance configuration. However, because the high power cases performed well, this issue can be mitigated by raising the transmit power. To fully mitigate the issue, multipath mitigation techniques should be implemented.
  - For mobile conditions, varying of the velocity of a device is performed to understand how speed of movement can affect RL. It was found that low Doppler shifts could be mitigated with the duty cycle, but high Doppler shifts require mitigating techniques, such as FEC coding and interleaving, to maintain the optimal performance of RL.

- Demonstrations in Chapter 8 showing how different protocols and different spectrum sensing methods can effect the trade-space. It was found that duty cycle must be raised when the rate of individual components increased. This happened in the case of the polyphase channelizer detector, which processed faster due to its increased efficiency. However, as a result, a lower duty cycle was required to account for system latency.

1.2 Thesis Outline

Chapter 2 contains the background information required to understand the motivation behind this thesis and its contributions. It starts with a more detailed overview of why CRs are needed, and proceeds in defining what a CR is and how it works by referencing the cognitive cycle. Why RL is often used in accomplishing the goals of cognitive radios and how it has helped optimize a CR's cognitive performance is discussed, specifically for interference management through spectrum avoidance. The effects of interference on a communication system is shown through a simple simulation, with solutions following. State-of-the-art traditional solutions, such as Frequency Hopping Spread Spectrum (FHSS) and Direct Sequence Spread Spectrum (DSSS), are first discussed. Following a brief analysis of their limitations, RL for interference management in cognitive radios is presented. An analysis of closely related work and their methods are summarized, with an emphasis on how this thesis improves upon existing research.

Chapter 3 begins by introducing the general system model used in this work to investigate the spectrum avoidance in the presence of interference scenario. The representative system model in GNU Radio is then presented, displaying an overview of major components involved in the implementation of the system, followed by an overview of the formulation of RL within GNU Radio. The implementation of each major component involved in the system model is then detailed to outline the design process within GNU Radio, where Chapter 5 discusses the OFDM transmitter, receiver, and channel. The two different protocols

used to format the transmitter and receiver were IEEE 802.11a and 5G NR, both of which are discussed in Chapter 4. Chapter 6 discusses the implementation of the two different spectrum sensing detectors used, the power spectral density (PSD) based detector and the polyphase channelizer detector. Lastly, Chapter 7 outlines the implementation of RL within GNU Radio, which concludes by verifying that the custom RL block created in GNU Radio performs as expected.

Chapter 8 tests the framework with different effects, such as the effect of latency, wireless channel impairments, and different spectrum sensing detectors, to see how they influence the performance of RL beyond what is typically considered. Additionally, the formulation of a trade-space for real-time systems with RL is presented to demonstrate how limitations can be predicted and mitigated. This thesis concludes with Chapter 9, which summarizes the results found in Chapter 8, highlights the challenges faced during implementation, outlines the limitations of the current work, and provides a summary of future directions that could improve upon this thesis.

1.3 Relevant Publications

- **Conference Proceedings:** A. Jones and W. C. Headley, "Considerations of Reinforcement Learning within Real-Time Wireless Communication Systems," *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*, 2022 (in review).

- **Tutorial:** A. Jones and W. C. Headley, "The implementation of an OFDM-based

Reinforcement Learning Wireless Communication System within GNU Radio," *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*, 2022 (to be submitted).

  - The framework will be open-sourced through GitHub.

- **Deliverables:** The work presented in this thesis is also part of an ongoing unclassified research project *5G++* sponsored by the Army and IAI, under the direction of Dr. Daniel Jakubisin and Dr. Chris Headley.

Chapter 2

Background

This chapter provides the background motivating this thesis. It begins with a brief overview of cognitive radios (CR) and how they can still be susceptible to interference, despite its cognitive capabilities. After highlighting existing vulnerabilities in typical state-of-the-art methods, reinforcement learning for interference mitigation is then presented as a promising solution, as corroborated by existing literature. Finally, existing literature is thoroughly examined to highlight the shortcomings by which this thesis aims to answer.

2.1 Cognitive Radios

As wireless communication technology advances, a growing problem has been spectrum scarcity, which worsens as the number of connected devices continues to rise, with an expected 13% increase by the year 2025 [1]. Consequently, there exists a high demand but little supply for additional spectrum bandwidth. However, with the spectrum being highly

regulated and also expensive, the pressing need is not necessarily more bandwidth but better utilization of the spectrum [13]. In 1999 [3], cognitive radios were proposed as a solution to improve spectrum utilization by using its re-configurable and cognitive capabilities to locate unused and/or under-utilized bands of the spectrum [14]. Its re-configurability is enabled by Software Defined Radio (SDR), which is equipped with field programmable gate arrays (FPGAs) that allow the reprogramming and adaptation of transmission parameters within the radio. Its cognitive capabilities are enabled through its cognitive cycle, enabling the radio to determine what needs to be adapted based on the current condition of the RF environment.

The cognitive cycle of a CR has three main components: spectrum sensing, spectrum analysis, and spectrum decision. A full description of each is given to demonstrate how a CR can use cognition to determine what needs to be reconfigured and as a result, better utilize the spectrum.

- **Spectrum sensing** uses the RF stimuli from the RF environment to observe the spectrum and its current condition. This part of the cognitive process enables spectrum access, where the available parts of the spectrum (e.g. spectrum holes) are identified, and spectrum avoidance, where the avoidance of other users becomes possible. The full capability of spectrum sensing includes determining whether other users are present or not through detection. Because cognitive radio's are mainly mobile devices that have medium-to-low computational capabilities and also energy constraints, a commonly used spectrum sensing method for detection of a signal in CR is energy detection due to its low complexity [14]. Though other detection methods are actively being

researched, it is not within the scope of this thesis.

- **Spectrum analysis**, through metrics, quantifies the performance of a transmission over a channel through feedback from either spectrum sensing or the receiver. This can be determined through performance metrics that can give insight on the quality of a transmission over a chosen channel. As an example, performance metrics can include bit-error-rate, channel capacity, delay, signal-to-noise ratio, path loss, interference, and packet error rate [14]. From these metrics, the cognitive radio can fully estimate and understand the current condition of the channel. For example, if the receiver calculates a high bit error rate, then the CR can infer in its decision making process that the channel contains poor conditions for transmission and/or contains interferers.

- **Spectrum decision** is the stage where the CR selects the most appropriate spectrum hole for transmission and/or adapts the transmission parameters to utilize the band more efficiently based on feedback from spectrum analysis. It is at this point in the cognitive cycle that a CR determines what needs to be re-configured.

2.2 Interference in Cognitive Radios

Interference in communication systems has long been an issue in the security of the wireless physical layer. For CRs, interference often occurs when transmitting over the same frequency band at the same instant in time, ultimately corrupting both itself as well as others. This is common in a congested RF spectrum and can usually be mitigated through spectrum

avoidance, where a CR uses its results from spectrum sensing and spectrum analysis to determine the bands with interference and avoid transmitting over them. However, with only a detector and basic adaptive capabilities, a CR, particularly ones with only tunable and adaptive cognitive capabilities, cannot fully avoid interference when it does not understand the *behavior* of the RF environment and its users. With its current capabilities, it can only understand the *condition* of the RF spectrum, but that is not enough, especially when malicious users are involved.

Jammers, generally characterized as a malicious user, can intentionally cause interference by injecting high noise signals into the same frequency band. When interference occurs, a user will know based on feedback from its receiver or even a lack of feedback altogether. For example, for an OFDM system, the receiver can provide both a packet error rate and a bit error rate. The packet error rate is feedback from its header, which tells the receiver whether it was able to detect the start of the transmitted signal or not. In an ideal system, this should be 0, but if the signal is corrupted, the start of the signal will be undetected, and the packet error rate will increase. The bit error rate, which is the number of bits that are wrong when compared to the transmitted bits, should be also be 0. In the presence of interference, both metrics increase significantly. Therefore, CRs must avoid and/or eliminate interference in an effort to protect the efficiency and operation of its own system.

FHSS and DSSS are the main state-of-the-art methods to jamming mitigation. Often utilized by Bluetooth communications, FHSS allows a CR to quickly hop across the spectrum in a pseudo-random fashion, providing better security against unintentional narrowband

interference and/or narrowband and partial-band jammers [7]. DSSS allows a CR to become harder to detect and thus harder to jam by spreading the signal out over a much wider bandwidth, pushing the signal closer to the noise floor. As a result, DSSS is efficient against interference caused by multiple narrowband signals [8]. However, both methods cannot handle dynamic behavior. Thus, even with classic interference methods, interference on some level is unavoidable. In the case of FHSS, there is no intelligence to the pseudo-random sequence, as it was not generated according to the behavior of the environment. Hence, accidental collisions are still an issue, and if an intelligent jammer figures out its hopping pattern, FHSS becomes ineffective [7]. For DSSS, though more protective against narrowband jamming, it can still fail under broadband jamming [8] and is also vulnerable to intelligent jammers. Thus, accidental collisions are also still possible. Therefore, even though a CR can be adaptive, such as is the case for a tunable and adaptive CR, it is not adaptive enough to fully mitigate interference due to a lack of predictive capabilities. However, if a CR gains the ability to learn the *behavior* of the environment and its users, such as is the case for smart CRs, mitigating hard-to-predict interference becomes possible. A CR can gain this ability through Reinforcement Learning (RL), and by formulating it specifically for spectrum avoidance, RL can help the CR become more efficient at mitigating interference.

2.3 Introduction to Reinforcement Learning

Before formulating RL for spectrum avoidance, a brief introduction of RL and how it can be incorporated in a CR is given. Reinforcement learning is an area in machine learning where

intelligent agents use feedback information and past experience within an environment to optimize its performance in achieving some goal [15]. More specifically, RL allows a system to adapt to dynamically changing conditions in an environment through a reward system. At the beginning, decision-making is by trial-and-error, where decisions are decided randomly, but over time, it learns an optimal decision-making strategy based on wrong decisions it made during the trial-and-error process. To do so, there are five main parts of an RL algorithm that is required for the system to learn:

- An **agent** is a cognitive user acting within the RL environment. It is the one performing the actions that affect the environment's current state. Every agent must have "a goal or goals related to the state of the environment" [15]. The agent's mission is to sense the environment conditions to pass along to the algorithm and then act on the action the algorithm found. If operating correctly, over time, the actions chosen are optimal, meaning these actions meet the goals set by the system. *In reference to a cognitive radio's cognitive cycle, the agent is a user that can sense the RF spectrum environment, e.g. the agent is the CR.*

- The **environment** in a reinforcement learning context is the physical medium in which the agent operates. It is within the environment where an agent acts on a decision. *In reference to a cognitive radio's cognitive cycle, the environment is the RF spectrum.*

- The **state** represents the current condition of the environment. At every time step, the environment has a state based on some action that the agent performed. The condition

of the state is what helps the algorithm make decisions. The state is also often called an observation of the environment. *In reference to a cognitive radio's cognitive cycle, the state is found through spectrum sensing.*

- An **action** is the instruction for the agent and is also what dynamically changes the environment. *In reference to a cognitive radio's cognitive cycle, the action is found through spectrum decision, which utilizes an RL algorithm to obtain this action.* In this context, possible actions can include the center frequency, power, modulation scheme, etc.

- The **reward** is a metric that tells the algorithm if the chosen action the agent performed met the specified goal of the RL formulation. If the chosen action was wrong, a negative reward is typically given to tell the RL algorithm that the chosen action resulted in undesirable performance. Conversely, if the chosen action was right, a positive reward is typically given to reinforce to the RL algorithm that its doing well. Over time, if operating properly, the reward is maximized by the RL learning process. *In reference to a cognitive radio's cognitive cycle, the reward is found through spectrum analysis.* In this context, possible reward metrics can include a bit error rate, signal-to-noise ratio, channel capacity, etc. For example, if the goal of the agent was to avoid interference, then a low bit error indicates that the agent chose the right action.

2.3.1 Example of Reinforcement Learning in Cognitive Radios

For a CR, Fig. 2.1 shows how RL can naturally be integrated into its cognitive cycle. This figure merges the concepts of the cognitive cycle of a CR and basic RL theory, where the cognitive cycle is highlighted in blue. For this process, the agent is the CR. The RF environment is a wideband spectrum of N channels, each at a center frequency f_i with bandwidth B_i, where $i = 0, 1, ..., N-1$. The signal stays within a channel for T seconds. For a CR with RL, the process is as followed:

1. **Observe the Environment through Spectrum Sensing**. The CR first observes the RF environment through its spectrum sensing capability, which takes the RF stimuli of the environment and produces a raw state (or condition) of the environment through the CR's sensing algorithm. Also through the sensing algorithm is a threshold decision process, which discretizes the raw state into a processed state.

2. **Spectrum Analysis through the Receiver**. For spectrum analysis, the CR uses its receiver to determine whether the transmitter's action resulted in successful communication. As an example, the receiver provides packet loss information, which determines how successful communication was on a particular channel. If no packets were lost, then the communication was successful. If packets were lost, then it was not successful. This can then translate into a reward, which provides a metric on how well the agent chose the appropriate action.

3. **Spectrum Decision through RL**. The CR agent uses an RL algorithm, such as

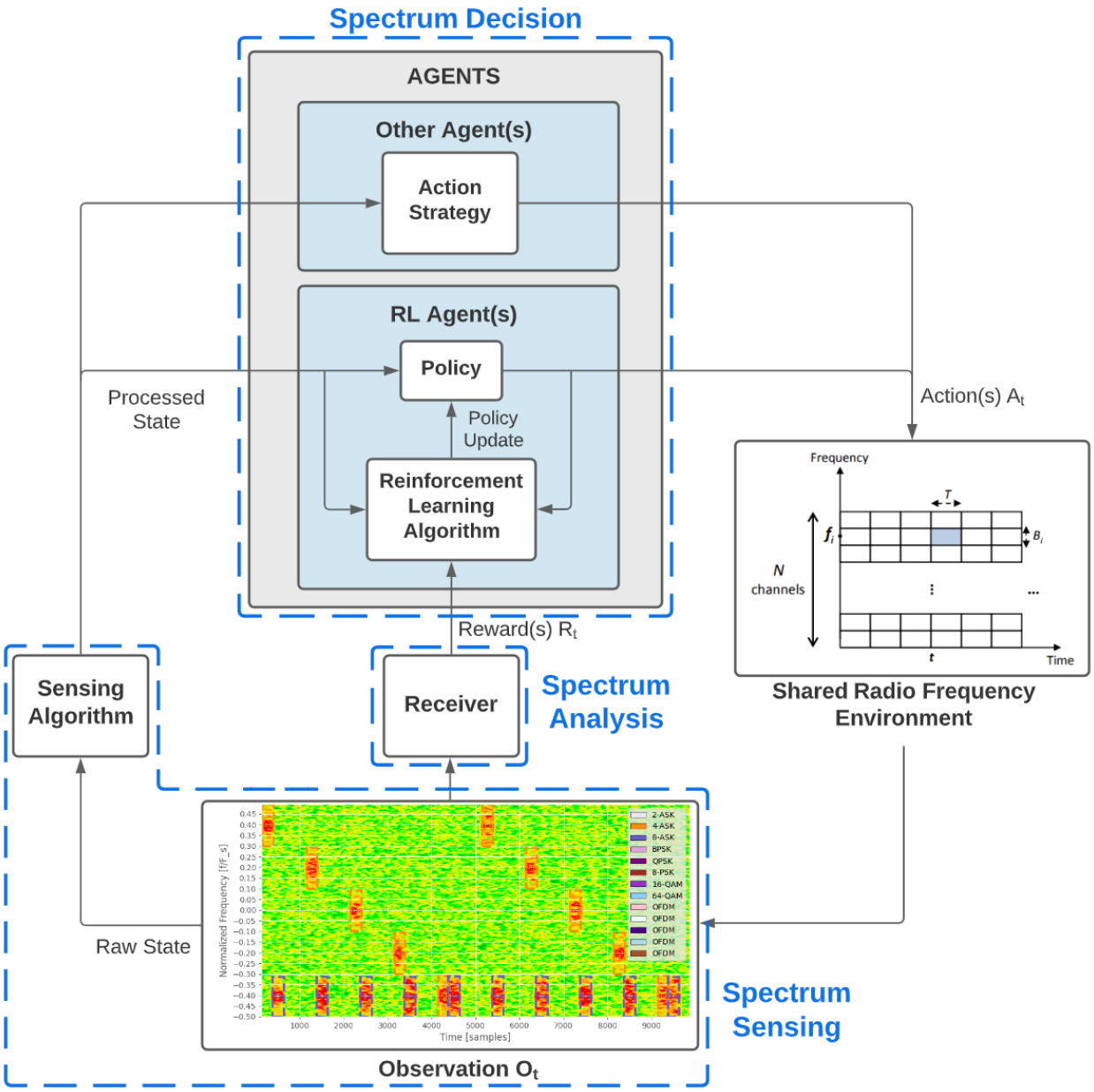


Figure 2.1: Integration of RL into a CR

Q-learning, for its spectrum decision process, which requires the processed state from spectrum sensing and the reward from spectrum analysis (through the receiver). Here, the next action for the transmitter to take is determined through the RL algorithm,

which uses the state and reward to infer the next best action to take. For a CR, the next action can be the next channel to transmit over and/or the transmission parameters that the CR must change / reconfigure for optimal communication and spectrum utilization.

The example shown in this section shows just one way of how RL is integrated into a CR system by using a CR's natural cognitive cycle to acquire a state and reward and then by implementing an RL algorithm as the spectrum decision strategy. As a result, RL is being heavily utilized in the development of cognitive radio for performance optimization [16]. This includes for interference mitigation, which is formulated and explained in the next section.

2.4 Reinforcement Learning for Spectrum Avoidance

This section formulates RL for spectrum avoidance, as used in a majority of related works and will serve as the scenario used in this thesis. Additionally, this section outlines the contributions of existing literature on this topic.

2.4.1 Formulation for Interference Mitigation

The formulation presented in this section for spectrum avoidance in the presence of interference is used in the current work. However, implementation will follow a model-free RL algorithm, where the probability transition matrix is not required, such as is the case for Q-learning. In a model-based environment, the probability transition matrix must be known.

However, in real environments, this is not ideal, and model-free algorithms are preferred. Both model-based algorithms, such as is the case for a standard Markov Decision Process (MDP), and model-free algorithms, such as Q-learning, are discussed in this section.

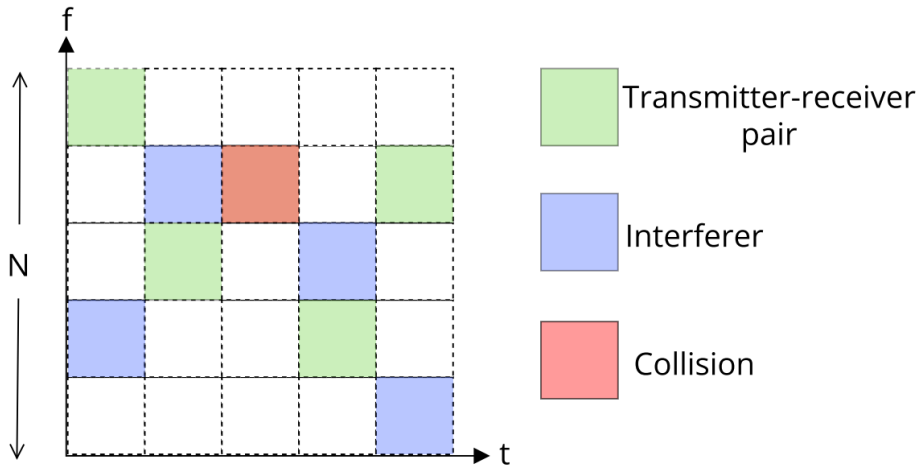


Figure 2.2: Illustration showing example movement of transmitter-receiver pair and interferer, including when a collision between the two occurs. Figure inspired by [17].

2.4.1.1 Spectrum Avoidance Scenario

Given N channels, suppose there is a transmitter-receiver pair transmitting across a channel. There also exists another user that causes interference to the transmitter-receiver pair. Whether the interference is intentional or unintentional is unknown. The goal of the transmitter-receiver pair, equipped with RL, is to avoid any channels that could have interference. Fig. 2.2 shows an example of the interaction between the intelligent transmitter-receiver pair and the interferer within the RF environment of N channels within a wideband

spectrum. RL is used to avoid collisions, such as the one shown in red.

2.4.1.2 Markov Decision Process (MDP)

The most basic scenario of RL for spectrum avoidance due to interference is through modeling the problem as a Markov Decision Process (MDP). An MDP is defined through its state and probability transition matrix, which gives the probability of an agent transitioning from one state to another. The probability transition matrix consists of all possible sequences of state transitions that can occur for each possible action and is defined as $P(s, s', a)$, where s is the current state at time t, s' the next state at time $t+1$, and a the action at time t. It has a dimension size of $S \times S' \times A$, where S represents the size of the set of all possible states, S' the size of the set of all possible future states, and A the size of the action set.

For spectrum avoidance, an MDP is used to represent the availability of sub-channels within the RF environment. Through its spectrum sensing capability, a CR agent can assign two values to each of the i-th sub-bands: a 0 when a sub-channel is un-occupied and a 1 when a sub-channel is occupied. Thus, the states for each sub-channel is $S_i[t] = [0, 1]$ for $i = [0, 1, ..., N]$. Therefore, if $N = 5$, a possible state at time t with two users on sub-channels f_0 (for the agent) and f_3 (for another fixed user) is $S[t] = [1, 0, 0, 1, 0]$, where $S_0[t] = 1$ and $S_3[t] = 1$. From this state, in RL, the agent will eventually learn to avoid sub-channel $i = 3$.

The state of each sub-channel transitions from state to state with probability $p^i(s, s^{'}) = Pr\{S_i[t+1] = s^{'} | S_i[t] = s\}$, where s denotes the current state at time t and $s^{'}$ the next state at time $t+1$ [18]. This is summarized in Fig. 2.3, which shows how the transition

probabilities are modeled for this scenario. The probability associated with the i-th sub-

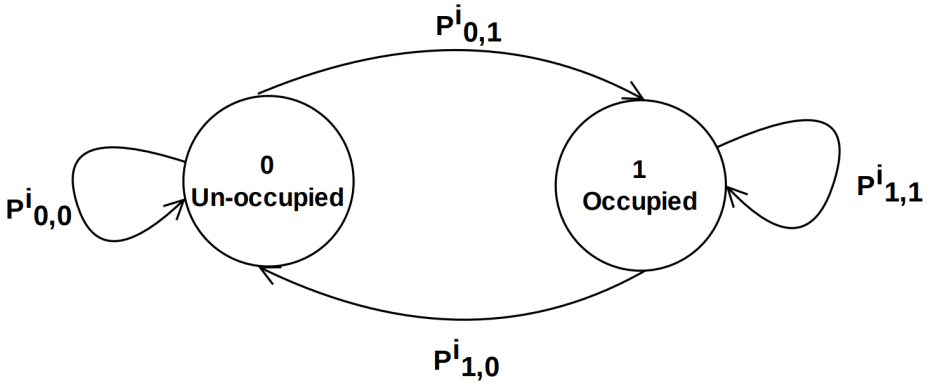


Figure 2.3: Markov chain for spectrum occupancy for a single sub-channel, as inspired by [18].

channel transitioning from states 0 to 0, 0 to 1, 1 to 0, and 0 to 0 given an action a are provided as $P_i[0,0,a]$, $P_i[0,1,a]$, $P_i[1,0,a]$, and $P_i[1,1,a]$, as summarized below.

a	$s_0^{'}$	$s_1^{'}$
s_0	$P[0,0,a]$	$P[0,1,a]$
s_1	$P[1,0,a]$	$P[1,1,a]$

For each time step, the appropriate probability is updated based on the number of occurrences of that state-action pair, as shown in Algorithm 1, where matrix O keeps track of all occurrences of the pair (s, s', a). P is updated using the pair (s, a), where the probability of the next state s' is updated by dividing its current probability by the sum of all future states for the (s, a) pair. For example, if the pair is $(s = 0, s' = 1)$ for action a and currently $O[s = 0, :, a] = [3, 1]$ and $P[s = 0, :, a] = [0.75, 0.25]$, then $P_i[0, 1, a]$ is updated to

$$P_i[0,1,a] = \frac{O[s=0,s'=1,a]}{\sum_{n=1}^{2} O[s=0,n,a]} = \frac{1+1}{5} = 0.4. \tag{2.1}$$

Algorithm 1 Process to Update the Transition Probability Matrix P

Result: Transition probability matrix P

1: s ← current state

2: s' ← next state

3: a ← chosen action

4: O[s,s',a] = O[s,s',a] + 1

5: Update P matrix by: P[s,s',a] = O[s,s',a] / sum(O[s,:,a])

Thus, for $(s = 0, a)$, the probabilities are updated to $P[s = 0, :, a] = [0.6, 0.4]$. Over time, as more actions are taken, each probability in the probability transition matrix is updated. These probabilities are utilized to update the Bellman equation, as represented by the value function in Equation 2.2 below

$$V^*(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} P(s, s', a) V^*(s) \right] \tag{2.2}$$

where $R(s, a)$ represents the reward given for the state-action pair (s, a) [15]. The value function $V^*(s)$ represents the expected objective value when following the optimal policy π^*, by which the system uses to make optimal decisions at each time step. The policy π^* "corresponds to the action which maximizes the value function in a particular state" [19]. Therefore, to get an optimal action at each time step, the policy can be used to retrieve the optimal action given the current state s, such that $a^* = \pi(s)$. This leads to the next state s', and the process repeats.

2.4.1.3 Q-Learning

The section before described the process for a model-based RL algorithm, but this section describes the model-free version, Q-learning, which uses the concepts of an MDP but eliminates the dependence on the probability transition matrix P. Q-learning is also the reinforcement learning algorithm used in this thesis.

Algorithm 2 Q-learning algorithm [20],[15]

Result: $Q(s, a)$ and the optimal policy π^*

Initialize Q

$\epsilon \leftarrow$ threshold that determines when to use the random vs. policy-based action

n $\leftarrow$ number used to compare ϵ with

for each iteration **do**
  Initialize state choice s
  for each step of iteration **do**
    $n \leftarrow$ random number between 0 and 1
    if $n < \epsilon$ **then**
      $a \leftarrow$ random action from action space
    else
      $a \leftarrow$ max $Q(s, :)$
    end if
    Take action a
    Simulate next state s' and reward r
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r+\gamma \max_{a'} Q(s', a')\text{-}Q(s, a)]$
    s $\leftarrow s'$
    $\pi^* \leftarrow$ max position of Q given s
  end for
end for

The Q-learning algorithm is outlined in Algorithm 2, where it utilizes the Bellman Equation as defined in Equation 2.3.

$$Q(s,a) = Q(s,a) + \alpha[r(s,a) + \gamma \max_{a'} Q(s',a) - Q(s,a)] \tag{2.3}$$

Notice that $P(s,s',a)$ in Equation 2.2 is replaced with the Q-table Q, which represents a look-up table for each state-action pair. Replacing P with Q transforms the RL algorithm from a model-based to model-free algorithm.

The Bellman Equation aims to maximize the state-action pairs that result in the highest average reward. It does so by updating the Q-table at each time step for a current state s and chosen action a. If the action chosen resulted in an undesired state, then the Q-value for that state-action pair will be decreased as time continues. Conversely, if the action was chosen correctly, then the Q-value for that state-action pair will stabilize to a maximum value determined by the learning rate α.

Algorithm 2 uses the epsilon-greedy policy to choose the action. This means, based on some ϵ, it chooses the action according to the policy π^* decided by the Q table when $n > \epsilon$. Else, it chooses the action randomly. The epsilon-greedy policy allows for proper exploration and exploitation within the system. The higher ϵ is, more exploration occurs. As a result, more random actions are taken, and the system is able to learn more of what actions are wrong. The lower ϵ is, more exploitation occurs. As a result, actions are chosen according to the learned policy from the Bellman Equation. If enough exploration has not occurred by this point, which is characterized by oscillations in convergent behavior, then there is a

chance that the system has not properly learned what actions are wrong given some state, thus resulting in a sub-optimal policy. Therefore, to reach an optimal policy, there must be a good balance between exploration and exploitation. At the beginning, more exploration and less exploitation should be utilized. This is because the learning should be maximized early on so that it can quickly learn from wrong actions and thus achieve a better policy. At the end, more exploitation and less exploration should be utilized. At this point, after an appropriate amount of exploration, the system should have obtained the optimal policy. In order to properly balance exploitation and exploration to achieve an optimal policy, given a minimum epsilon ϵ_{min}, maximum epsilon ϵ_{max}, decay rate r, and instantaneous epoch time t, a decay function using Equation 2.4 was applied to ϵ. Thus, over time, the system will transition from a policy containing randomness to an optimally learned policy.

$$\epsilon = \epsilon_{min} + [\epsilon_{max} - \epsilon_{min}]e^{rt} \tag{2.4}$$

To handle sudden changes in the environment, after an optimal policy has been found but needs to be updated, epsilon can be reset back to its maximum value, and the policy can be updated again through the epsilon decay process to adjust for those sudden changes.

Learning Rate α

Notice that Equation 2.3 contains the parameter α. This is the learning rate, which decides how much the new value or the old value is weighted. More specifically, it controls the rate at which the system learns. A higher α allows for faster learning, meaning it converges to a working policy faster, but the chances of a sub-optimal policy are greater. Conversely, a lower α learns much slower but produces more optimal policies. To start, in this system, a

higher α will be used, and if sub-optimal policies are observed, α will be lowered.

Discount Factor γ

Notice that Equation 2.3 contains the parameter γ. This is the discount factor, which decides how much the immediate or future reward is weighted. γ is a real number in the interval (0,1). It is used in the policy to help maximize the expected accumulated future reward. If γ is 0, then immediate rewards are valued more, whereas if γ is 1, then future rewards are more important. For large state spaces, typical values that lead to an optimal policy are $\gamma = [0.8, 0.99]$ [21], which were determined through extensive research in the RL field. However, in this system, due to memory constraints, the state space will remain small. Thus, a lower γ will be used so that the system is more sensitive to sudden dynamic behaviors in the environment.

2.4.2 Related Work

This section details current literature in RL for spectrum avoidance and how they led to the motivation of the current work.

2.4.2.1 Focus on the Advancement of RL Algorithm Performance

Current studies in RL-based spectrum avoidance solutions have thoroughly researched with respect to simulation-based algorithm development, with the main focus being the advancement and efficiency of the algorithms themselves, but with little consideration for actual real-time implementation. For example, [9]–[11], [22] improve upon the standard Q-learning

methods and highlight increases in algorithm performance. In [9], their proposed method of Q-learning was compared to a fixed and myopic policy, where myopic occurs when $\gamma = 0$, meaning the effect of the current action on future rewards is ignored. In their results, the reward for the proposed method resulted in a reward 15% greater than the myopic policy and 42% greater than the fixed strategy [9], demonstrating the increase in their proposed algorithm's performance. [10] expands upon the results in [9], comparing the proposed method in [9] to their own proposed method. In both studies, because a large state and action space can explode the dimensions of the Q-table, only at most 8 channels were available to users, four of which can be occupied by interferers at any time. The authors found that their proposed Q-learning method resulted in a higher average reward by about 65% more than in [9]. In [11], an ON-policy version of Q-learning was implemented, where at each time-slot, the CR followed a greedy strategy by maximizing the Q-value when choosing the action, rather than randomly selecting the action. In this study, only 4 channels were considered, with one intelligent CR and one interferer. Their results claim the CR was able to learn the interferer's pattern within one to four episodes of learning, depending on the initial state of the simulation. Thus, they conclude their proposed algorithm converges faster than the standard Q-learning algorithm.

[22] applies the concepts in [11] but uses feedback from the receiver to find hidden interferers within the spectrum. In this study, only 4 channels were considered and the signals were BPSK modulated. Their proposed method was compared to a fixed strategy, e.g. no intelligence or adaptive capabilities, and a sensing-based strategy, no intelligence but has

adaptive capabilities. In finding hidden interferers, the authors claim their proposed algorithm could find a fixed-sweeping interferer 85% of the time, compared to 66% for fixed and sensing-based methods. For a reactive interferer, where the interferer is not active all the time unless it senses other users, the authors claim their proposed algorithm could find this type of interferer 71% of the time, compared to 1% for fixed and sensing-based methods, showing the improvement of their proposed method to non-intelligence based capabilities.

To summarize, the studies mentioned in this section use Q-learning to avoid interference in the RF spectrum. Thus, this thesis will use Q-learning to accomplish the same thing. However, these studies only consider a simulation-based setting and do not consider realistic aspects of a CR, such as spectrum sensing, nor do they consider real-time implementation. Thus, this thesis will consider both, where spectrum sensing is used to determine the state of the RF environment and real radios are used to transmit and receive.

2.4.2.2 Expansion to Multi-Agent Reinforcement Learning (MARL)

[6], [18], [23], [24] expand the spectrum avoidance problem to multi-agent reinforcement learning, where two or more CR agents are involved. A simple example of MARL for spectrum avoidance is presented in [18], where there are at least two CR agents and one interferer across 5 available channels. The results in [18] demonstrated the increased performance of the system of MARL in comparison to a random policy, which performed about 35% worse.

[23] considers the mutual interference caused by competing CR users in the presence of a malicious interferer, with only 5 channels available to the users. This study, such as is

the case in [6] and [24], proposes using cooperation among CR users, where information is exchanged via a common control channel, establishing the opportunity for a more informed decision. Results showed that the proposed method achieved over 95% of the maximum performance, whereas a sensing-based method performed below 90%.

In [6], two CR users and one interferer over 5 channels is considered. This study considers using a joint Q-table that all users can learn from, in addition to their own independent Q-table. The proposed method JMAA, short for joint multi-agent anti-jamming algorithm, was compared to independent Q-learning (IQL), where the CR agents do not sooperate with each other, a frequency-hopping (FH) based method, where a fixed hopping pattern is used, and lastly, a sensing-based method, where only sensing is used to determine interference location. Their results showed that JMAA achieved 90% of the maximum performance, compared to the 70% for IQL, 65% for FH-based, and 60% for sensing-based, proving the advantage of cooperation among MARL agents. [24] expands upon the results in [6], comparing JMAA to other methods, such as IQL-ACK, where decisions are made on the user's indepedent Q-table but where the mutual interference can be determined through the joint Q-table. These studies proved that by making decisions through the joint Q-table, as is the case in JMMA, approximately 95% of the maximum performance can be achieved.

To summarize, these studies use MARL to better avoid interferers. However, these studies are still only simulation-based and do not expand to more realistic implementations that real radios would have to utilize.

2.4.2.3 Expansion to Deep Reinforcement Learning (DRL)

[25]–[30] apply deep reinforcement learning methods to handle more complex spectrum avoidance problems, such as handling fast changing users and intelligent interferers. In [31], a deep CNN-based Q-network was used to find the optimal action a. This is the most used DRL method in the existing literature for spectrum avoidance. The action set was the available channels and the state space a spectrum waterfall. The spectrum waterfall displays "the combination of sequential observations of spectrum within a fixed range of frequency and length of time" [31]. The x-axis shows what signals exist in which frequency where the y-axis shows the time at which the signal is in that frequency band. The different colors represent the energy of the signal. This spectrum waterfall is fed into a CNN that incorporates q-learning to optimally choose the next action.

In [25]–[27], [30], the focus was on using DRL for to learn the fast-changing patterns of users, specifically fast-changing jammers. In [25], their DRL implementation was compared to Q-learning for different jamming patterns. Their results showed that for simple patterns, such as sweeping and comb, DRL performed just as good as Q-learning. Hence, for simple jamming patterns, Q-learning is preferred since DRL is computationally intensive. However, for more dynamic and intelligent users, the authors found DRL performs better by 32%. In [30], the improvement of using DRL is also seen, where the proposed DRL method was compared to Q-learning and a random-based policy when in the presence of an intelligent jammer and 3 various types of interference signals, showing at least an 18% performance improvement over Q-learning and a faster convergence time.

Other work using DRL for spectrum avoidance focused more on algorithm development, such as in [27]–[29]. These works aim to implement a better performing DRL algorithm compared to existing state-of-the art DRL methods. This is the case in [27], where the authors compare their method to actor-critic and deep Q-network methods, which have become the standard for applications in DRL. Their proposed method performed approximately 20% better than the actor-critic method and 68% better than a deep Q-network.

The studies described in this section use DRL for interference mitigation through spectrum avoidance. However, again, they are not applicable to real world applications, as they are simulation based and are not deployable over real radios. Thus, the method presented in this thesis was designed to be deployable over real radios.

2.4.2.4 Incorporation of Spectrum Sensing

Among these studies, to make simulations more realistic for use in a real CR by utilizing its wideband spectrum sensing capability, some use spectrum sensing, as opposed to pre-defined simulation-based state environment definitions, to incorporate the sensing mechanism radios need to acquire an observation and/or reward for RL. Sensing methods include using energy detection ([22]), the power spectral density ([18], [25]–[27], [32]) and signal-to-noise ratio to capture the interference profile ([26], [28], [30]).

Power Spectral Density (PSD)

To capture the presence of users within the RF environment, the authors in [18], [32] demon-

strate a way to capture the state based on the PSD using thresholds. This method allows for the entire spectrum to be sensed, rather than a single channel. Additionally, this method of detecting signal presence requires the least amount of a priori information of other signals within the environment. Thus, less hard assumptions about other users are made and the sensing function of the system can function independent of other users.

In order to implement the PSD method in a system, equations 2.5 and 2.6 are required. The PSD of a sub-band signal can be found through

$$\hat{S}_y(F) = \frac{1}{N}\left|\sum_{n=0}^{N-1} y[n]e^{-j2\pi Fn}\right|^2 = \frac{1}{N}|Y(F)|^2 \tag{2.5}$$

where the smoothed estimate of the PSD is as follows

$$T(Y) = \frac{1}{LN}\sum_{l=-(L-1)/2}^{(L-1)/2} |Y[k+l]|^2 \tag{2.6}$$

L is the length of the rectangular smoothing window, and Y is the FFT of $y[n]$. To use this method, the PSD must be smoothed because of noise fluctuations. Otherwise, the noise may cause the estimator to exceed the detection threshold [32].

To find the state of each sub-band given the entire wideband spectrum's PSD, a threshold is applied. For example, [18] used a threshold based on bandwidth. The cognitive radio computes the maximum available bandwidth B, so that the state of a subband i, where $i = 0, 1, ..., N-1$ of an N channel wideband spectrum, is as follows

$$s_i = \begin{cases} 1 & B \geq \beta \\ 0 & B < \beta \end{cases} \tag{2.7}$$

where β is the minimum required bandwidth for transmission. In this simulation, if $B \geq \beta$, the sub-band is available, e.g. unoccupied. On the other hand, if $B < \beta$, the sub-band is not available, e.g. occupied.

Utilized in [32], another threshold that can be applied is the Neyman-Pearson threshold, which is determined by the noise floor estimate, given a fixed maximum tolerable false-alarm probability. The cognitive radio can then determine the location of signals in that channel based on that threshold.

Signal to Interference plus Noise Ratio (SINR)

The authors in [25], [27]–[30] create a system where the reward is based on whether or not there was a successful transmission, where SINR defines that success. SINR can be calculated using equation 2.8 below.

$$SINR = \frac{P_{U,t} h_U}{n + P_{J,t} h_J I(f_{J,t} = f_{U,t})} \tag{2.8}$$

$P_{U,t}$ and $P_{J,t}$ are the transmission powers of the legitimate user and the jammer, and h_U and h_J are the respective channel gains. n represents the noise power, and $I(x)$ is a function used to account for jammer-user collisions. If $f_{J,t}$ is equal to $f_{U,t}$, which are the frequency channels the jammer and user occupy at time t respectively, then $I(x) = 1$. Thus, when the channel of the user and jammer are the same, the jammer succeeds in degrading SINR. For the reward, a positive reward is given if the SINR is above a given SINR threshold, as shown

below in equation 2.9. Else, a 0 is given.

$$r_{SINR}(a_t) = \begin{cases} r_t & SINR_t \geq SINR_{\text{threshold}} \\ 0 & SINR_t < SINR_{\text{threshold}} \end{cases} \tag{2.9}$$

However, there is often a "cost" parameter factored in. For example, [28] adds in a switching cost c to minimize unnecessary energy consumption, and the transmit power cost C_p to achieve successful transmission with as low power consumption as possible. This expression is given in equation 2.10, where the action at time t is denoted as a_t.

$$r_t = r_{SINR}(a_t) - c(a_t) - C_p P_{U,t} \tag{2.10}$$

Energy Detection

The authors in [6], [22] use energy detection to gain knowledge about signal presence in the spectrum environment. It can be leveraged to find both the state and reward, as is the case for [22]. For the reward, the energy of a channel can be used to characterize where other users are, e.g. occupied channels will have high energy whereas unoccupied channels will have low energy. In this case, the metric used is as follows

$$R_f(S, S') = 1 - \frac{E(f)}{E_{\text{tot}}} \tag{2.11}$$

This metric takes the energy in the channel of interest and divides it by the total energy across the spectrum. An occupied or jammed channel will have high energy and thus low reward while an empty channel will have low energy and thus high reward.

Using energy detection for the state, it is a similar process as described for PSD, but more general in that it does not have to use formal threshold methods, such as Neyman-Pearson.

For instance, in this work, energy detection will be applied to each subband after applying a polyphase-channelizer filter bank or the PSD. Two different detectors are used to analyze how computational complexity of using different spectrum sensing methods can affect the system, particularly in terms of latency.

2.4.2.5 Incorporation of Theoretical Timing in Real Systems

Because simulation timing is often ignored in simulation-based prior art, [6], [23], [24], [33], [34] even consider the theoretical timing needed in a real system to properly perform reinforcement learning. The beginning stages of this process mainly considered only the time required for transmission, sensing, and learning (in this order). Over time, more detailed versions emerged, which contained the following, in addition to the time for transmission, sensing, and learning: the timing in which the states and actions are available to the RL algorithm and the time needed for feedback - T_{ACK}, which considers when the new action is conveyed to the system. In this work, these timing concepts were used but with small modifications to handle any existing latency, as will be discussed in Chapter 5.

2.4.2.6 Incorporation of Universal Software Radio Peripherals

[32], [33], [35]–[37] expand these systems to real-time implementation over Universal Software Radio Peripherals (USRPs), which are SDRs used for RF applications. To implement RL in such a system, USRPs are used as the transmitter and receiver, which operate over a wireless channel. On a computer, instructions are given via a software program for RF, such

as LabVIEW or GNU Radio, which controls the interaction between the transmitter and receiver as well as the RL (denoted as the cognitive engine). For example, in [32], [33], the cognitive engine gives the transmitter the next action to take. The transmitter then makes that change and transmits over the wireless channel. The receiver receives the signal and uses spectrum sensing to determine the state of the RF environment. The sensing results from the receiver are then fed back to the cognitive engine, and the cycle repeats. A similar process is used in the current work, but with sensing occurring at the transmitter for reasons that are explained in Chapter 5.

[32], [35], [37] mainly focused on the performance of RL over USRPs. All found convergent behavior and had a working platform. However, timing was exaggerated to handle latency that naturally occurs in hardware. In particular, [37] states that a potential weakness of the entire exchange from software to the USRP is USB latency and process scheduling time. To handle it, [37] exaggerates the timing to handle any delays. *However, this thesis aims to understand limitations caused by latency, specifically by measuring the minimum amount of latency possible for the system to operate properly. Additionally, how this latency affects the accuracy of RL decision making is analyzed and discussed in Chapter 8.*

2.4.2.7 Motivation for a Feasibility Analysis of RL in Real-Time Systems

The literature discussed in this chapter helped formulate the motivation behind this thesis. After thoroughly reviewing the existing literature presented in this chapter, the contributions of these studies include the following:

- Prove that RL for spectrum avoidance works in a simulation setting, such as in Sections 2.4.2.1, 2.4.2.2, and 2.4.2.3.
- Improve upon state-of-the-art RL methods for spectrum avoidance to achieve optimal performance, as shown in Sections 2.4.2.1 and 2.4.2.3.
- Address specific problems related to spectrum avoidance, such as mutual interference among users, as investigated in Section 2.4.2.2, and fast-changing users, as discussed in Section 2.4.2.3.
- Prove that RL for spectrum avoidance can be applicable in a real-time setting by employing spectrum sensing (Section 2.4.2.4) and theoretical timing of operations within the RL process (Section 2.4.2.5).
- Prove that RL for spectrum avoidance does in fact work in a real-time setting over radio hardware (USRPs), as discussed in Section 2.4.2.6.

Serving as motivation for the current work, what is missing is an in-depth analysis of what could cause a real-time RL system for spectrum avoidance to perhaps fail and/or perform sub-optimally in how it learns, ultimately shifting the focus from RL algorithm advancement to practical feasibility of RL in real systems. Intuitively, one way RL could fail is through its algorithm, particularly when the state and reward do not accurately reflect the current environment. This can happen in two ways. The first is through an imperfect representation of the state and reward, where the imperfections are so great that it cannot accurately represent the environment. The second is delayed feedback, where the state and reward may

have represented the environment multiple time steps ago but do not at the current time step. Hence, without a well-defined state and reward, RL is crippled. Thus, it begs the question, "How does a real environment and a real agent with real hardware affect the state and reward?" In a real situation, imperfections within the system are present everywhere. Therefore, at what point do the imperfections make RL unfeasible in a real system? Thus, this thesis is motivated to answer the following questions and/or concerns:

- What limitations within real systems could disrupt the accuracy of the RL process?
- Can the effect these limitations have be mitigated and/or avoided?
- How much imperfection / error can exist within the system for RL to work properly?
- For a system that runs continuously, what design decisions must be made to maintain good RL performance?
- Because effects caused by these limitations are often heightened in real hardware, how can the limitations be mitigated such that the transition from software to hardware is a much smoother process?

To answer these questions, a real-time implementation of a system with RL for spectrum avoidance was created, with GNU Radio as the main toolkit. The current work expands beyond simple linear modulation schemes, as was mostly the case in the discussed literature, and instead implements a modern communication format, Orthogonal Frequency-Division Multiplexing (OFDM), so that it is comparable with modern systems, like 5G. As a re-

sult, the conclusions made in Chapter 8 will be more applicable to existing and emerging communication systems. Additionally, though existing literature has considered practical discretization of the state through spectrum sensing methods, the current work acquires the state observation through a polyphase channelizer, which is discussed in Chapter 6. Through literature review, the understanding is that this method for spectrum sensing has not yet been considered in this context and would serve as an interesting comparison to other methods, such as the PSD method discussed in Section 2.4.2.4. Through this framework, an in-depth analysis was performed to understand the possible trade-offs that must be considered as a result of existing limitations. Trade-offs of interest include the balance between duty cycle, goodput, and reinforcement learning accuracy. Additionally, the trade-space analysis resulting from simulations can serve as an example to researchers in this area how to determine the system's limitations and use their results to design a more practical RF system with RL.

Existing literature closest to the content of the current work is [36]. The authors in [36] present a universal GNU Radio framework for RL meant for use to further studies in RL over radios within academia. Note that in [36], the communication system in GNU Radio was separate from the reinforcement learning, which was supplied through Open AI Gym. Their work makes the GNU Radio environment a custom environment within OpenAI Gym and then uses the step() function of OpenAI Gym to retrieve the state and reward required for RL computation from the GNU Radio environment. As such, the nature of this process occurs within a distributed fashion, with the radios only having access to the decisions from

RL and not RL itself. *However, in the current work, a custom block was implemented within GNU Radio for reinforcement learning to work continuously with the communication system, meaning the cognitive engine is deployed on the radios and the radios themselves perform RL.*

Additionally, the trade-off analysis presented in this paper was inspired by conclusions made in [36]. Among their findings was the effect a real channel had on RL. They analyzed what they call the "miss ratio", which is the percentage of time that the incorrect action for RL is taken. From their observation, this behavior is a result of latency, causing outdated observations and delayed execution of actions within the reinforcement learning process. They concluded that the latency introduced by the system transmitting over a real channel must be taken into account when designing wireless systems with reinforcement learning but had no solution as to how the latency could be mitigated. *Based on these findings, this thesis was motivated to simulate the effects of latency through a duty cycle in an effort to understand at what point is latency no longer an issue and also how much latency RL can handle within a real system.* To do so, in Chapter 8, the duty cycle was varied and decreased until outdated observations no longer occurred. As a result, the effect of latency and its causes are also summarized in Chapter 8. To capture a full grasp of possible limitations to RL performance, simulations were performed under varying circumstances, which include the following:

- Implementation of different protocols, such as IEEE 802.11a (WLAN) and 5G numerology 0, to investigate limitations in relevant modern-day communication systems.

- Comparison of different spectrum sensing algorithms to investigate its effect on latency and state imperfections.
- Simulation of different channel effects, such as physical distance between the transmitter and receiver, to investigate channel limitations.

All simulations were performed through the framework discussed in Chapter 3, which outlines the formulation of RL for spectrum avoidance used in this work, the main components involved, and how they were implemented in GNU Radio.

Chapter 3

System Model of an OFDM-based Reinforcement Learning Wireless Communication System

This chapter discusses the formulation of the reinforcement learning (RL) for spectrum avoidance used within the OFDM-based RL communication system implemented within this thesis. This includes an overview of the designed system, how the reinforcement learning problem was formulated, and what components were necessary to realize a complete reinforcement learning cycle.

3.1 System Model

Displayed in Fig. 3.1, where feedback is characterized by the dotted lines and communication / computation by the solid lines, there exists a communication system consisting of an

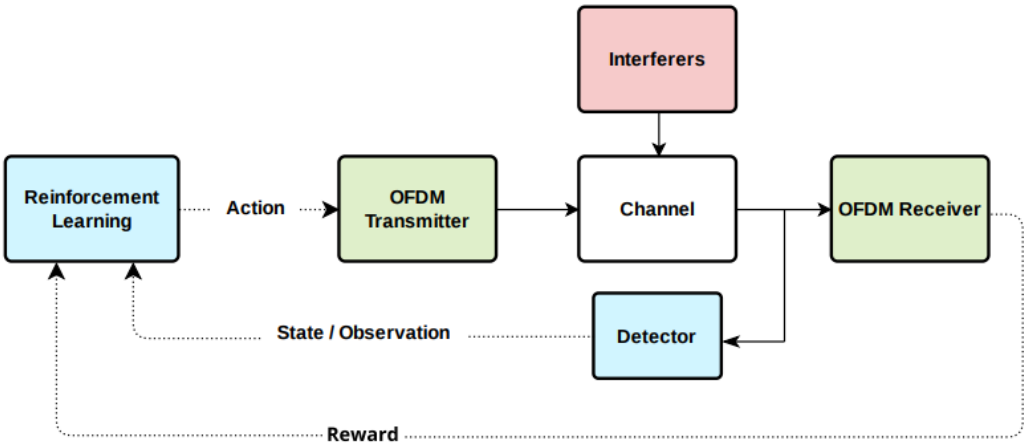


Figure 3.1: Considered representative system model of an OFDM-based RL-enabled wireless communication system. Here, required reinforcement learning feedback is shown by dotted lines and data communication by solid lines.

intelligent OFDM link equipped with reinforcement learning transmitting across a fixed wideband channel, of which consists of N equally spaced sub-channels. Its goal is to transmit and receive across a channel without interfering with other users' transmissions. Unknown to the user, there also exists an interferer who, whether intentionally or unintentionally, denies the user successful communication at its receiver. From a reinforcement learning perspective, for the intelligent OFDM-based link (consisting of a transmitter-receiver pair), the goal is to choose the best sub-channel by avoiding any sub-channel with interference. The assumption is that both the intelligent user and the interferer can communicate across one sub-channel for each epoch, which consists of 100 individual transmissions. Additionally, there are at least two users at all times. However, to simplify the problem, the number of users U are constrained to not exceed N, such that $U <= N$. Furthermore, the following assumptions

are made:

- Sensing can occur at either the transmitter or receiver. Thus, the transmitter and receiver sense the same interference environment, and so the hidden node problem is not an issue.
- Feedback from the receiver and spectrum sensing detector is lossless.
- There is perfect knowledge of the chosen action between the transmitter and receiver, such that the receiver can properly demodulate the received signal.
- When interference occurs, the assumption in this thesis is that the entire signal, both the header and payload, are corrupted.

The process presented in Fig. 3.1 is as follows. First, given an action by the reinforcement learning algorithm, the OFDM transmitter transmits across a channel according to that action, where the action is a center frequency chosen among the available N sub-channels. At the beginning of the process, the action chosen is random. However, as learning continues over time, the action becomes more intelligently chosen. The receiver, also having knowledge of the action, receives the signal and attempts to demodulate the data to see if the signal has been corrupted by interference. The receiver sends its reward to the reinforcement learning block. Simultaneously, the transmitter's detector applies a sensing algorithm to determine the signal presence across all N sub-channels and sends its result, e.g. state / observation, to the reinforcement learning block. Once the RL block has both the state and reward, it

calculates the best action for the intelligent user to take next. This process repeats until the simulation is over. The GNU Radio implementation of this process is discussed in the next section.

3.1.1 System Model in GNU Radio

Modeled after Fig. 3.1, Fig. 3.2 displays the different components implemented within GNU Radio for the system to operate as discussed. The main users include two interferers, highlighted in orange, and one intelligent OFDM user, whose transmitter and receiver are highlighted in red. Additionally, the transmitter and receiver use a real channel enabled by the use of USRPs to communicate. The blocks used to access the hardware is highlighted in green. Also, the necessary components for reinforcement learning are identified in red. They are the following: RL Agent through the transmitter, reward through the receiver, state through the transmitter's detector, and the cognitive engine that uses Q-learning to predict the next action. A detailed formulation of reinforcement learning is described below.

3.1.2 Reinforcement Learning Formulation

The OFDM user has a cognitive engine equipped with Q-learning. Its goal is to evade unknown interference within the radio frequency environment. It will know if it succeeded based on the packet information received from its receiver. The Q-learning algorithm will use this feedback to adjust its decision making. The following discusses the details on how the reinforcement learning was modeled, which follows a Markov Decision Process since

decisions follow the Markov property, where the future state only depends on the current state. Therefore, there is no memory, e.g. past outcomes are not recorded.

3.1.2.1 Action Space

The actions that the OFDM user can choose are the center frequencies of the N sub-channels in the radio frequency environment. Within a given wideband channel, there are N sub-channels at a frequency f_i with a bandwidth B_i, where $i = 0, 1, ..., N-1$, indicating the specific sub-channel. The N sub-channels are specified at the beginning and formulates the action space. Thus, the total number of actions A_n in the action space is N. For N sub-channels, the user can choose from the action set below:

$$A = [a_1, a_2, ..., a_N] \tag{3.1}$$

In GNU Radio, as shown in Fig. 3.2, the action is decided by the custom Reinforcement Learning block, which uses the Q-learning algorithm. The action is performed by the RL agent.

3.1.2.2 State Space

The state space S is defined as the set of all possible states that can occur. State values in this problem can either be 0 or 1, where 0 represents an unoccupied sub-channel while 1 represents an occupied sub-channel. Thus, the number of states in the state space is $S_n = 2^N$. Therefore, all possible combinations of how the state can be filled with 0s and 1s

formulates the state space. If $N = 5$, the state space would be

$$S = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.2}$$

where the columns represent the occupancy state of the ith sub-channel, which can either be 0 for unoccupied or 1 for occupied. The rows represent one possible state that the environment can experience. However, not every one of these states are valid to the considered environment. For example, because the number of users U was specified to be less than N, the last state [1 1 1 1 1] is not a valid state because it will never be realized by the algorithm. Extending the above example, where $N = 5$, if there is a fixed user at channel f_0 and a frequency hopper sweeping across at every time step starting at f_4, the valid states, with and without interference, would be

$$S = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.3}$$

In the GNU Radio implementation, as highlighted in Fig. 3.2, the state is obtained through a spectrum sensing detector, which is fully detailed in Chapter 6.

3.1.2.3 Reward

In general, the reward is formulated such that it maximizes the goal of the user, such that it quantifies for each time step how close the agent is to its goal. For this user, it prioritizes successful communication, which can be measured from the packet header information from the receiver. If the packet header p is received (e.g. True), a reward of 1 is sent. Else, a reward of 0 is sent. This is summarized below in Equation 4.8.

$$R[s,a] = \begin{cases} 1 & p = True \\ 0 & p = False \end{cases} \tag{3.4}$$

In GNU Radio, the reward is retrieved by the receiver, as shown in Fig. 3.2. Details of how the reward was obtained are described in Chapter 5.

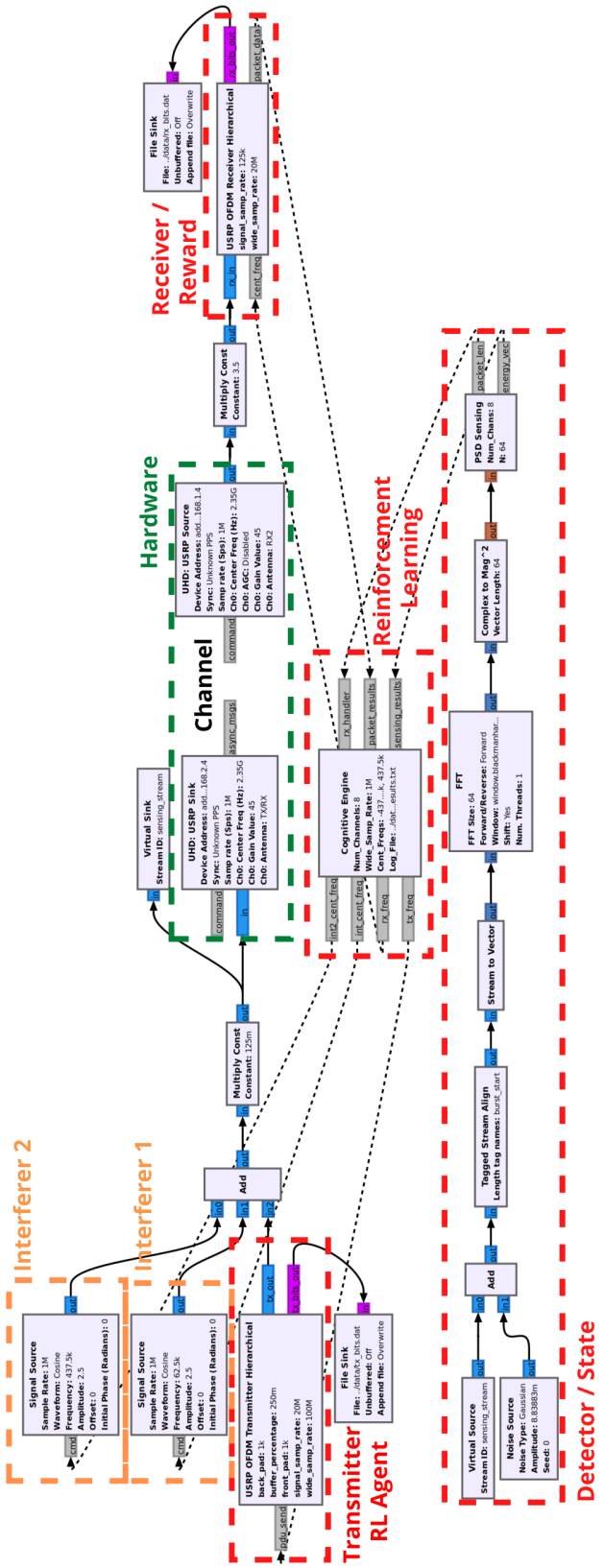


Figure 3.2: System model represented in GNU Radio with blocks specifying the different users and reinforcement learning components

Chapter 4

Transmission Protocols and Interference Definitions

This chapter discusses how the physical waveform of the signal was formatted with different frame structures according to different protocols, such as IEEE 802.11a and 5G New Radio (5G NR). Additionally, the interference is defined in this chapter, in addition to how they were implemented within GNU Radio.

4.1 Formatting the Signal based on Transmission Protocol

In this section, the frame structure of the transmitted signal of the RL agent is formatted according to the IEEE 802.lla and 5G NR protocols. Note that the full protocol standards are not implemented, but the OFDM symbol closely follows the OFDM physical waveform

structure of these protocols.

4.1.1 IEEE 802.11a Protocol Standard for OFDM Burst

The transmitted signal was formatted following the IEEE 802.11a OFDM burst format, as shown in Fig. 4.1. This frame structure consists of a preamble, header, and data payload. The preamble consists of two sync words of a set pattern for channel estimation and equalization, timing synchronization, and frequency offset estimation. The header, labeled as "signal" in Fig. 4.1, contains the packet information, such as the length and data rate, required for burst communication. The receiver uses the header to detect the end of the burst to properly retrieve the data payload, which follows the preamble and header. The length of the data payload can vary.

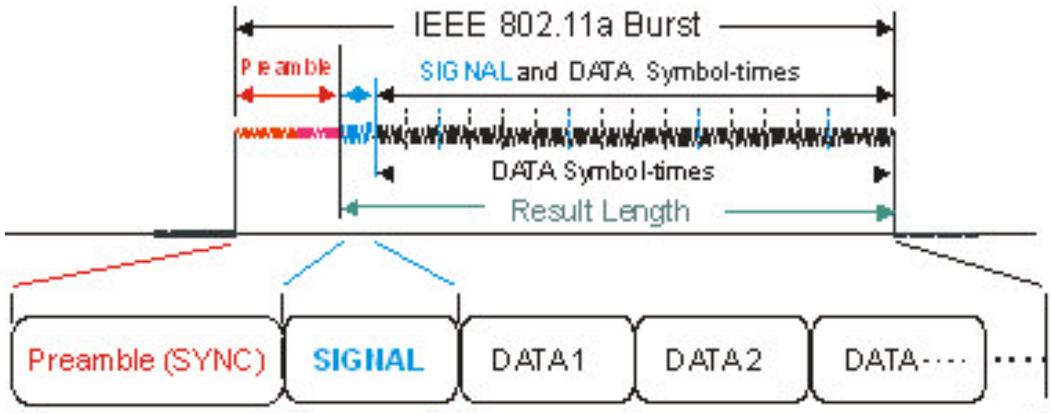


Figure 4.1: OFDM burst frame structure based on the standard protocol IEEE 802.lla as described in [38]. Determined as Fair Use, as explained in Appendix B.

To implement the protocol displayed in Fig. 4.1 in GNU Radio, a number of design

decisions were made. They are as follows.

- Determine the total number of OFDM symbols, including the number of preamble symbols, number of header symbols, and number of data payload symbols. This formats the frame structure.
- Determine OFDM subcarrier values and allocation, including allocation sets for the guard band, pilot carriers, and occupied carriers.
- Configure the preamble, header, and data symbols.

4.1.1.1 Frame Structure

First, the number of possible OFDM symbols that can be transmitted was determined. Through trial and error, it was found that GNU Radio can only handle at most 15 symbols. Therefore, reserving two for the sync words and one for the header (as per the IEEE 802.11a protocol standard), the system can generate at most 12 OFDM data symbols before the internal memory buffer fails. To be conservative, the number of OFDM data symbols generated was 7, resulting in a total number of 10 symbols for an entire OFDM burst. Table 4.1 summarizes the numbers used to create the frame of the OFDM burst. Fig. 4.2 visually describes the symbol setup of 1 OFDM burst.

Symbol Type	Number
Sync Word Symbols	$N_{sync} = 2$
Header Symbols	$N_H = 1$
Payload Symbols	$N_P = 7$
Total OFDM symbols	$N_{symb} = 10$

Table 4.1: Number of symbols for different symbol types in 1 OFDM burst

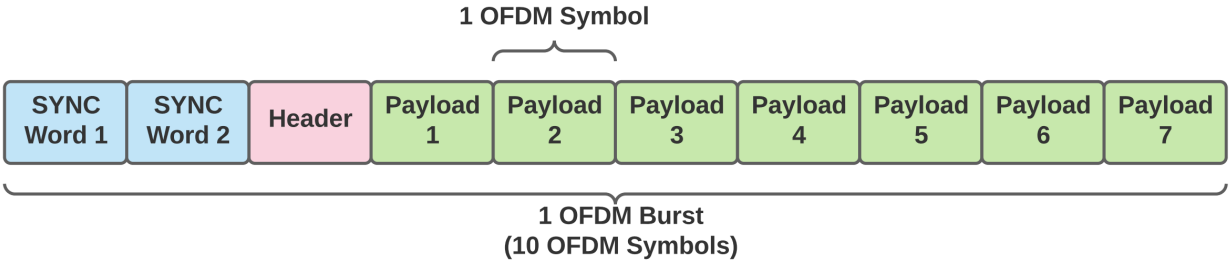


Figure 4.2: Visual demonstration of implemented OFDM frame structure

4.1.1.2 Subcarrier Allocation

To implement the frame structure discussed above, the individual OFDM symbol must be configured appropriately. Every OFDM symbol is generated from a set of subcarriers, typically consisting of a guard band, pilot subcarriers, a null subcarrier for the DC offset, and data subcarriers. For IEEE 802.11a protocol, the total number of subcarriers N_{SC} is 64, e.g. the FFT length is also 64. A discussion of the allocation of the subcarriers is discussed below. Note that GNU Radio centers the subcarrier set at index 0. Therefore, the subcarrier set goes from indices -32 to 31.

- *Guard Band.* The guard band helps mitigate non-negligible adjacent channel inter-

ference. A typical guard band consists of approximately 20% of the total number of subcarriers, with 10% of the null subcarriers distributed to the front and 10% to the back. 20% of 64 is approximately 12 subcarriers, leaving 52 subcarriers in total for the pilots and data subcarrier allocation. Because GNU Radio accounts for DC offsets, one of the 12 null subcarriers is dedicated to subcarrier 0, leaving 11 null subcarriers for the guard band with 6 null subcarriers at the front and 5 at the back. Therefore, its allocation set includes indices (-32 to -27, 27 to 31), totalling 11 subcarriers used for the guard band. For all indices, the value of the null subcarrier is 0.

- *Pilot Subcarrier Allocation.* Pilot subcarriers make the system more robust against frequency offsets and phase noise that can occur during channel estimation. Pilots typically have a set pattern and are known to the receiver. IEEE 802.11a protocol requires 4 equi-spaced and equi-powered pilot symbols. Its allocation set includes indices (-21, -7, 7, 21), totalling 4 subcarriers used for the pilots. Because the pilots are equi-powered, the pilot symbol set with an amplitude of 1 for one OFDM symbol is (1, 1, 1, 1).

- *Data Subcarrier Allocation*: Otherwise known as "occupied carriers", data subcarriers carry the information that will be sent to the receiver. Allocation of the data subcarriers are usually decided after the pilots and guard band have been set, meaning it is a set excluding the subcarriers used for the pilots and null subcarriers. Thus, its allocation set includes indices (-26 to -22, -20 to -8, -6 to -1, 1 to 6, 8 to 20, 22 to 26), totalling 48 subcarriers used for the payload.

Table 4.2 summarizes the allocation structure for a single OFDM symbol, and Fig. 4.3 outlines the overall subcarrier allocation structure, with indices labeled following the nomenclature in Table 4.2.

Subcarrier Allocation Variables	**Value**
FFT Length	$L_{FFT} = 64$
Number of Subcarriers	$N_{sc} = 64$
Number of Null Subcarriers	$N_{null} = 12$
Number of Guard Subcarriers	$N_{guard} = 11$
Number of Pilot Subcarriers	$N_{pilots} = 4$
Number of Data Subcarriers	$N_{data} = 48$
Number of Occupied Subcarriers	$N_{occupied} = 52$
Pilot Carrier Indices	(-21, -7, 7, 21)
Data Carrier Indices	(-26 to -22, -20 to -8, -6 to -1, 1 to 6, 8 to 20, 22 to 26)
Pilot Symbols	(1, 1, 1, 1)

Table 4.2: Subcarrier number and allocation indices breakdown for a single OFDM symbol [39]

Fig. 4.4 displays the OFDM signal in the frequency domain as generated in GNU Radio. For demonstration purposes, the pilot amplitude is exaggerated to display their location among the spectrum. Following the subcarrier structure discussed, Fig. 4.4 shows the pilots approximately equi-spaced and equi-powered. In between the pilots is the actual data being transmitted. On the sides is the guard band.

Note that these subcarrier allocations are designed for a single OFDM symbol. When creating multiple symbols, the allocation set is repeated. For example, for a transmission

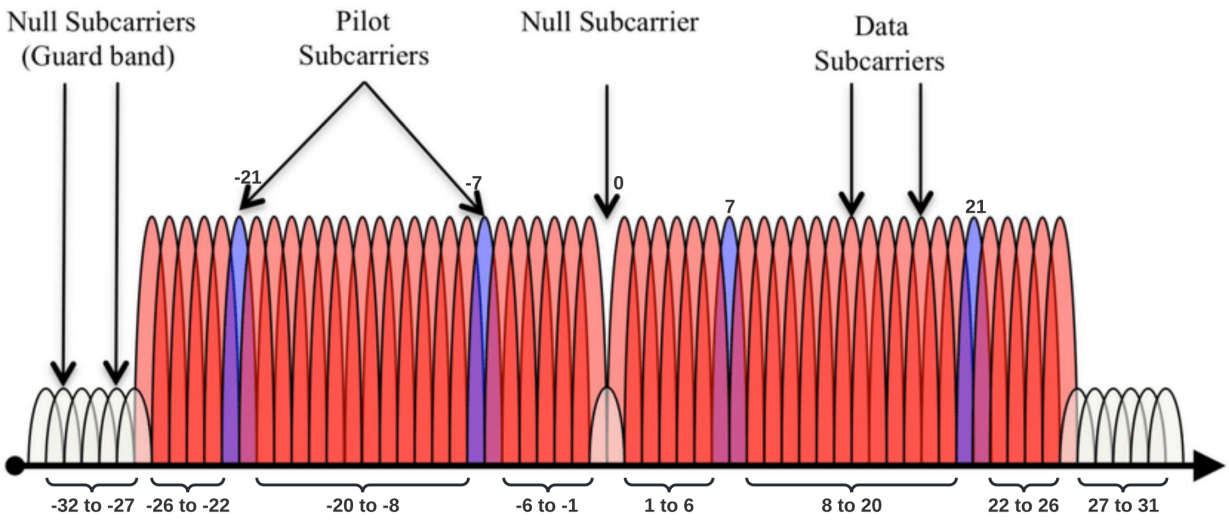


Figure 4.3: Visual demonstration showing index locations of OFDM subcarriers [40]. Determined as Fair Use, as explained in Appendix B.

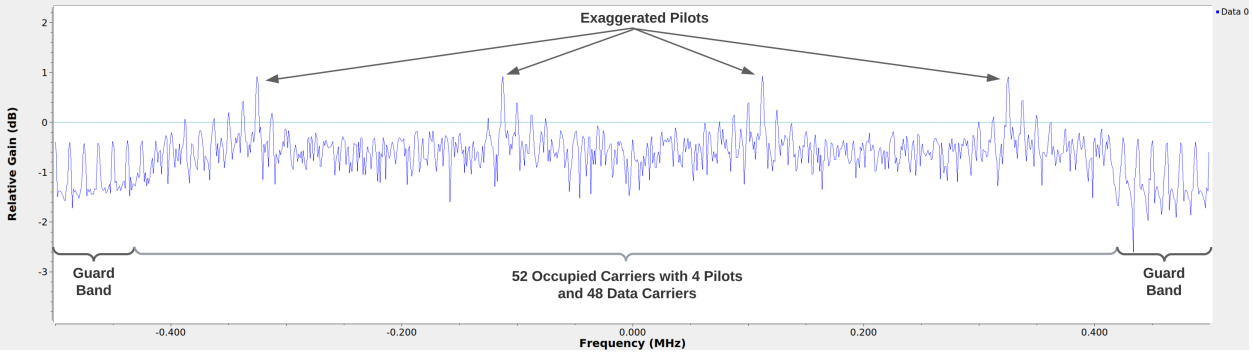


Figure 4.4: Example OFDM signal in the frequency domain displaying subcarrier allocation structure

with 1 header and 7 data symbols, the total pilot carrier set includes 8 sets of (-21, -7, 7, 21). The same applies for the occupied carrier set and pilot symbol set.

4.1.1.3 Configuration of the Preamble, Header, and Data Symbols

After the allocation sets have been determined, the preamble, header, and data symbols need to be further configured according to the IEEE 802.11a protocol. Further details of the configuration of these symbols are discussed below.

- *Preamble - Sync Word 1.* The first sync word of the preamble performs fine and coarse frequency offset and timing estimation. As per GNU Radio documentation [41], this symbol must be configured with zeros on alternating carriers, not including the subcarriers used for the guard band. Thus, this configuration rule applies to indices (-26 to 31). Following IEEE 802.11a protocol, this symbol is BPSK modulated, with a mapping of $(0 : \sqrt{2}, 1 : -\sqrt{2})$ [39]. Therefore, the values used in this symbol's allocation set alternates zeros with $\pm\sqrt{2}$.

- *Preamble - Sync Word 2.* The second sync word of the preamble performs coarse frequency offset and channel estimation. As per GNU Radio documentation [41], this symbol must be filled completely, not including the subcarriers used for the guard band. Thus, this configuration rule also only applies to indices (-26 to 31). Following IEEE 802.11a protocol, this symbol is BPSK modulated, with a mapping of $(0 : 1, 1 : -1)$ [39]. Therefore, the values used in this symbol's allocation set randomly alternates ± 1.

- *Data Payload.* This payload contains the actual data transmitted. In GNU Radio, the 7 data symbols are QPSK modulated using the embedded *digital* library through the command digital.constellation_qpsk(). This command creates a QPSK modulated

object, labeled as *payload_mod* in GNU Radio, with a mapping of $\left(0 : \frac{-1-j}{\sqrt{2}}, 1 : 0 : \frac{-1+j}{\sqrt{2}}, 2 : \frac{1+j}{\sqrt{2}}, 3 : \frac{1-j}{\sqrt{2}}\right)$.

- *Header*. The header provides frame information to the receiver, including the modulation, packet length, and frame length. Through a header, the receiver better knows what to expect when receiving a burst. In GNU Radio, the 1 symbol header is BPSK modulated using the embedded *digital* library through the command digital.constellation_bpsk(). This command creates a BPSK modulated object, labeled as *header_mod* in GNU Radio, with a mapping of $(0 : -1, 1 : 1)$. In addition, the header carries necessary information about the burst according to how the header bits are formatted. The default format is displayed in Fig. 4.5. The first 12 bits are used to store the length of the payload through the packet length tag (e.g. *packet_length_tag_key*), and the next 12 are used to store the length of the frame through the frame length tag (e.g. *len_tag_key*). The last 8 bits are used to store the CRC (Cyclic Redundancy Check) bits. Note that a 32-bit CRC translates to a 4-byte CRC, such that $N_{CRC} =$ 4 bytes. In GNU Radio, the header is formatted through the *packet_header_ofdm*

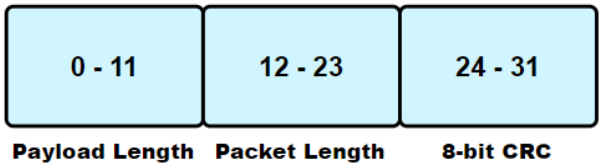


Figure 4.5: How the header uses bits to pass information to the receiver

function in the *digital* library. This function takes in a number of items, as expressed

Function Arguments	Input	Description
occupied_carriers	[occupied_carriers]	Contains a list describing which subcarriers carry data (refer to Table 4.2)
n_syms	N_H	Number of header symbols to format ($N_H = 1$)
len_tag_key	packet_length_tag_key	Label of the tag containing the packet length (80 bytes)
frame_len_tag_key	length_tag_key	Label of the tag containing the frame length ($N_P = 7$)
bits_per_header	header_mod.bits_per_symbol()	Number of bits represented by the modulation scheme. For BPSK, should be 1.
bits_per_payload	payload_mod.bits_per_symbol()	Number of bits represented by the modulation scheme. For QPSK, should be 2.
scramble_header	False	Scrambles the header bits to reduce PAPR (Peak-to-average power ratio) spikes

Table 4.3: Input to digital.packet_header_ofdm() for formatting the header in GNU Radio

and discussed in Table 4.3. Note that the value *packet_length_tag_key* stores the value of *packet_len_bytes*, which was determined using Equation 4.1 below.

$$packet\_len\_bytes = N_P \left[\frac{N_{data} \times log_2(M)}{8} \right] - N_{CRC} \tag{4.1}$$

For IEEE 802.11a with 7 payload symbols QPSK modulated ($M = 4$), this results in

$$packet\_len\_bytes = 7 \left[\frac{48 \times log_2(4)}{8} \right] - 4 = 80 \text{ bytes} \tag{4.2}$$

4.1.1.4 Timing Structure of the Transmitted Signal

Next is how long and how often each transmitted burst is sent, as displayed in Fig. 4.6, which breaks down the timing structure of one time step.

Beginning at t_0, the user transmits a 10 symbol OFDM burst for T_D seconds. As the burst is transmitted, the start and end of the burst is tagged in GNU Radio, making up the "tag region". These tags are shown in Fig. 4.7, where the start and end of the burst are labeled with the tags "burst_start" and "burst_end". They are helpful in telling the

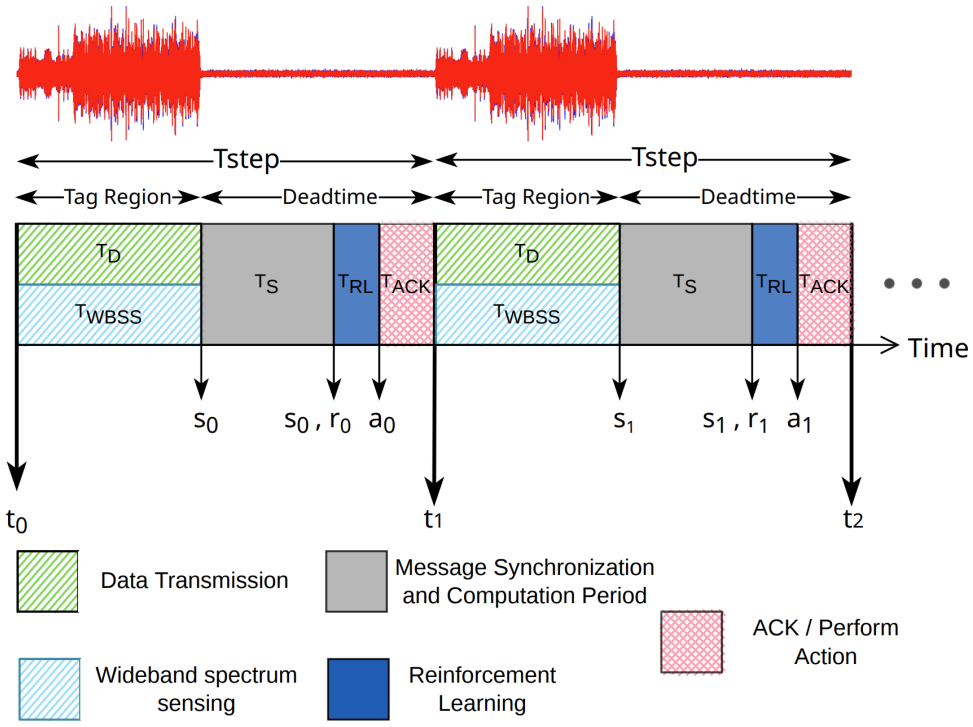


Figure 4.6: Timing structure of an entire time step, demonstrating the timing of transmission, spectrum sensing, state and reward feedback, and processing. This figure was inspired by existing timing diagrams in [33], [34], [23], [6], [24] but with added features specific to this work.

detector when to start and stop sensing for each time step. They force the detector to sense only the burst (not any whitespace) and process associated samples within this region as they are transmitted, hence the sensing time T_{WBSS} is the length of T_D. At the end of the burst, the detector provides the current environment state s_0, which is required for RL, for the current time step.

Next, the stream enters a period of "deadtime", which is a stream of 0s padded to the burst to give extra time for latency within the system. Deadtime was necessary for the

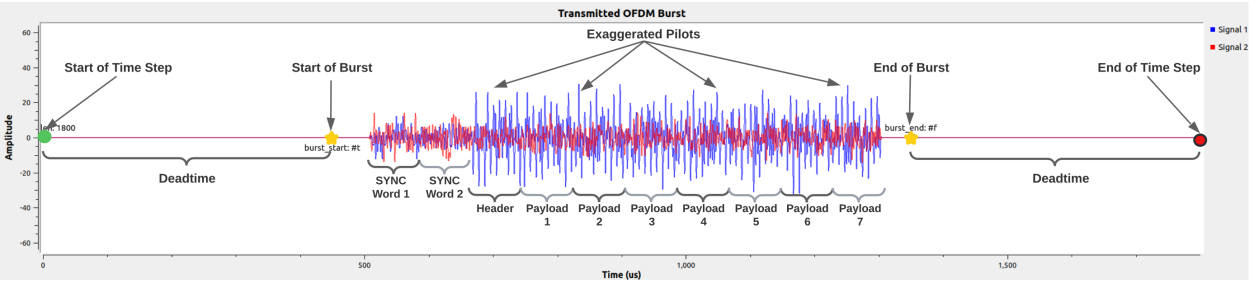


Figure 4.7: Transmitted burst as genered in GNU Radio showing symbol structure with the 10 OFDM symbols, consisting of the 2 synch words, 1 header, and data payload symbols. To demonstrate the timing of the signal, periods of deadtime, the start and end of burst, and the start and end of one epoch are labeled and identified. Note that the pilots are exaggerated to identify the header and data symbols.

following reasons:

- To use as a metric to determine the latency within the system. Latency could be caused by additional processing and computation time and through feedback delays in message passing. This latency is fully investigated through this deadtime (otherwise known as the duty cycle) in Chapter 8.
- To allow time for the state from the detector and reward from the receiver to reach the custom RL block before the next burst is transmitted.
- To enforce a continuous streaming system within GNU Radio so that the RL runs with the program, rather than outside it.
- To prevent having to start and stop the USRPs per transmission.

During the "deadtime" region, three main operations occur. First, is the "Message Syn-

chronization and Computation Period", labeled as T_S. It accounts for the following processes. After the transmitted signal travels through the channel, it is directed to both the receiver and the detector. From the receiver, a reward will be passed as a message to the RL block, and from the detector, the state will also be passed as a message to the RL block. However, there is no guarantee that the state and reward messages will arrive at the same time or even within the same time step, as blocks within GNU Radio all have different processing speeds. T_S attempts to account for the latency created by this feedback. The goal is for at the end of T_S, both the current state s_0 and the current reward r_0 will be sent to the RL block. Note that there are trade-offs with implementing T_S. If T_S is too short, there could be detector feedback but not receiver feedback and vice versa. Conversely, if it's too long, goodput and the duty cycle can suffer. Thus, an element of this research also studies the trade-offs between goodput, duty cycle, and the accuracy of state and reward information for reinforcement learning. This is further investigated in Chapter 8.

The second region of the "deadtime" T_{RL} is for reinforcement learning computation. At this point, through the proper setting of T_S, the custom RL block has both s_0 and r_0 ready for Q-learning. Due to the low complexity of Q-learning, T_{RL} is nearly negligible, as also defined in [33], [34], [23], [6], and [24]. At the end of T_{RL}, the next action a_0 for the next time step t_1 is available. The reinforcement learning block relays this information through message passing, and the user acts on it to begin the next time step t_1. The time needed to send this information to the transmitter and coordinate the next action is T_{ACK}.

4.1.1.5 Goodput Calculation

To measure how much data is sent relative to the time between bursts, a goodput calculation is helpful. This measurement is useful in Chapter 8 when determining the effects of latency.

Goodput can be calculated by determining the fraction of the data to the entire transmitted signal. This can be calculated using Equation 4.3 below

$$\text{Goodput (Mbps)} = \frac{(packet\_len\_bytes) \times 8 \text{ bits}}{T_{step}} \tag{4.3}$$

where T_{step} is the time required for one time step as determined by the signal's samping rate F_s. T_{step} is defined in Equation 4.4.

$$T_{step} = \frac{N_{data\_samps} + N_{deadtime}}{F_s} \tag{4.4}$$

The number of data samples is related to the number of subcarriers at the input of the channel and the number of data symbols plus the header. At the input of the channel, there are 10 OFDM symbols, each with 64 subcarriers plus the 16 subcarriers added for the cyclic prefix. Thus, the number of data samples is $10(64 + 16) = 800$. The amount of deadtime depends on what is required. Suppose the deadtime is 2000 samples. Then, for a signal with a sampling rate of 20 Mhz, per IEEE 802.11a protocol, the amount of time required per time step would be

$$T_{step} = \frac{800 + 2000}{20MHz} = 0.14ms \tag{4.5}$$

Thus, this corresponds to a goodput of

$$\text{Goodput (Mbps)} = \frac{80 \times 8 \text{ bits}}{0.14ms} = 4.57 \text{ Mbps} \tag{4.6}$$

This goodput can be validated by the theoretical OFDM data rate. According to [42], the goodput of OFDM signals can be calculated using equation 4.7 below

$$\text{Goodput (Mbps)} = \left[\frac{N_{occupied} \times N_{BPSCS} \times R \times N_{SS}}{T_{sym}}\right] \times \frac{(L_{FFT} + L_{CP}) \times N_P}{N_{data\_samps} + N_{deadtime}} \tag{4.7}$$

where N_{BPSCS} represents the number of coded bits per subcarrier per stream and N_{SS} the number of spatial streams, which would be 1 in this case since this is not a multiple input multiple output (MIMO) scenario. R is the coding rate, as defined in Equation 4.8 below.

$$R = \frac{packet\_len\_bytes}{packet\_len\_bytes + N_{CRC}} \tag{4.8}$$

To find the symbol duration T_{sym}, the time required for the data carriers and the guard must be found, as shown in Equations 4.10 and 4.11.

$$F_{sc} = \frac{F_s}{L_{FFT}} \tag{4.9}$$

$$T_{Data} = \frac{1}{F_{sc}} \tag{4.10}$$

$$T_{GI} = \frac{L_{CP}}{F_s} \tag{4.11}$$

$$T_{sym} = T_{GI} + T_{Data} \tag{4.12}$$

Notice that these equations use the inverse relationship frequency has with time to find the time required for these operation. Because the OFDM symbol is generated using subcarriers, then using the subcarrier spacing of the subcarriers is helpful in finding the time for an OFDM symbol. For example, as per IEEE 802.11a protocol, if the transmitted signal with

64 subcarriers and cyclic prefix length of 16 operates at a sampling rate of $F_s = 20MHz$, then Equation 4.9 becomes

$$F_{sc} = \frac{20MHz}{64} = 312.5kHz \tag{4.13}$$

This translates to

$$T_{Data} = \frac{1}{312.5kHz} = 3.2\mu s \tag{4.14}$$

$$T_{GI} = \frac{16}{20MHz} = 0.8\mu s \tag{4.15}$$

$$T_{sym} = 3.2\mu s + 0.8\mu s = 4\mu s \tag{4.16}$$

Additional, with a 4 byte CRC,

$$R = \frac{80}{80+4} = 0.95 \tag{4.17}$$

These values correspond to a goodput of

$$\text{Goodput (Mbps)} = \left[\frac{48 \times log2(4) \times 0.95 \times 1}{4\mu s}\right] \times \frac{80 \times 7}{800 + 2000} = 4.57 \text{ Mbps} \tag{4.18}$$

which matches the results found in Equation 4.6. These calculations were utilized in measuring the effect latency had in the system in terms of data rate in Chapter 8.

4.1.2 Extension of Framework to 5G

This section loosely extends the framework discussed to 5G NR. The intent is to use 5G numerology 0 settings for the OFDM symbols and characterize the RL problem more to resource blocks rather than channels.

4.1.2.1 5G NR Numerology 0

5G NR is setup similarly to LTE but has flexibility in the configuration of its subcarrier spacing, as shown in Table 4.4, each of which can be used in different cases. Due to USRP constraints on sampling rates, where only a maximum of 10 MHz is possible in the testing setup of this thesis, numerology 0 is used in this work, where each subcarrier has a spacing of 15 kHz [43]. In addition to flexible subcarrier spacing, 5G NR uses the concept of resource

Numerology μ	$\Delta f = 2^{\mu} \times 15[kHz]$
0	15
1	30
2	60
3	120
4	240

Table 4.4: Subcarrier spacing configurations for the different 5G numerologies [43]

blocks, where one resource block consists of 12 OFDM subcarriers and 14 OFDM symbols as shown in Fig. 4.8. A resource block consists of 168 resource elements, where a resource element is 1 OFDM subcarrier in 1 OFDM symbol. For numerology 0, the resource element has a subcarrier spacing of 15 kHz, and the bandwidth of a resource block is $12 \times 15kHz = 180kHz$. Notice that each resource block uses 14 OFDM symbols. In 5G NR, these 14 OFDM symbols make up 1 slot, which is the same configuration this work will follow.

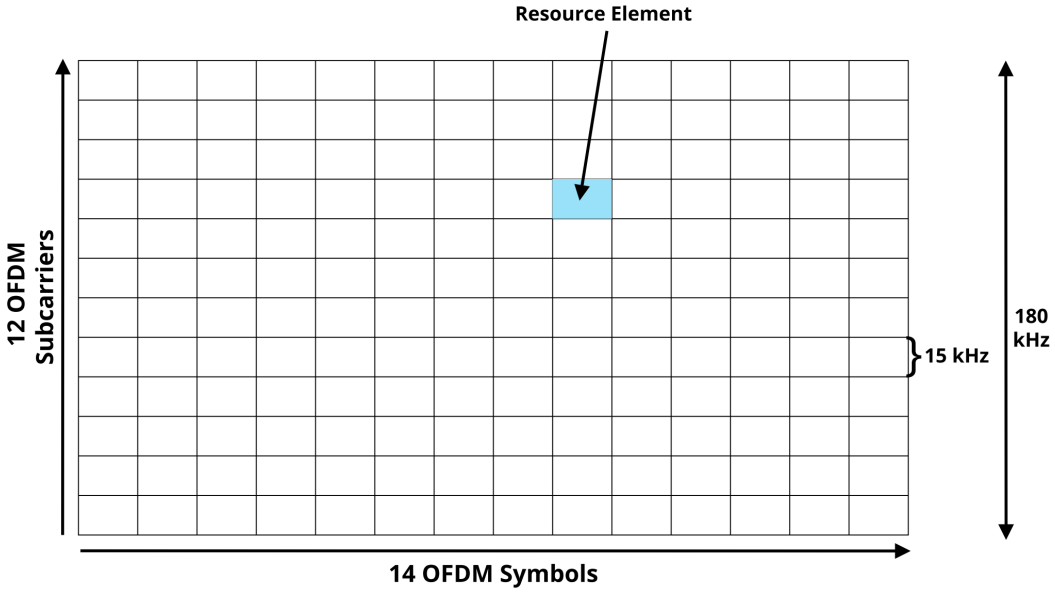


Figure 4.8: Resource block configuration in 5G NR

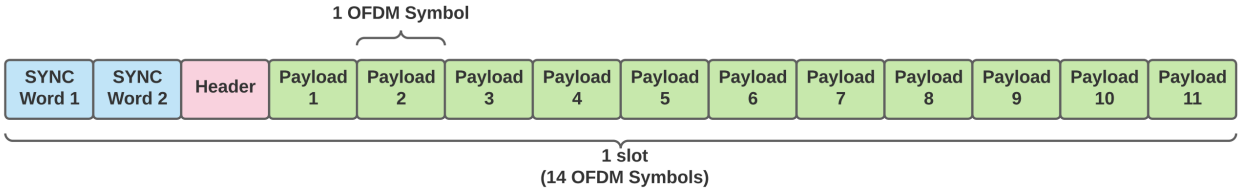


Figure 4.9: 5G slot with 14 OFDM symbols

4.1.2.2 Slot Structure

For the 5G slot configuration, the radio frame structure used is shown in Fig. 4.9, where there are 2 sync words, 1 header, and 11 data payload symbols, totalling 14 OFDM symbols. These values are summarized in Table 4.5. Notice that the frame is similar to IEEE 802.11a but with extra data symbols. This can still be a realistic uplink according to [44], which discusses the different types of Demodulation Reference Signals (DM-RS) used for synchronization.

In particular, mapping B is possible in this work. Mapping B is located in the first symbol of the transmission [44], which is the case in Fig. 4.9. Additionally, this mapping is relative to the start of the slot boundary [44], which is desirable in the case of frequency hopping.

Symbol Type	**Number**
Sync Word Symbols	$N_{sync} = 2$
Header Symbols	$N_H = 1$
Payload Symbols	$N_P = 11$
Total OFDM symbols	$N_{symb} = 14$

Table 4.5: Number of symbols for different symbol types in 1 5G slot

4.1.2.3 Subcarrier Allocation

Table 4.6 summarizes the subcarrier allocation used for the 5G-based application in this work. For the subcarrier allocation, a realistic uplink can be modeled using 64 subcarriers since in a 5G scenario, user equipment (UE) is allocating resource blocks according to 12 subcarriers.

Note that due to GNU Radio constraints, 5G's subcarrier allocation number of 4096 (as discussed in [45]) is not possible. Thus, in this work, the application is 5G-based and not fully 5G. The idea is to stay true to the numerology settings for subcarrier spacing as well as the slot structure.

Subcarrier Allocation Variables	**Value**
FFT Length	$L_{FFT} = 64$
CP Length	$L_{CP} = 4$
Number of Subcarriers	$N_{sc} = 64$
Number of Null Subcarriers	$N_{null} = 16$
Number of Guard Subcarriers	$N_{guard} = 15$
Number of Pilot Subcarriers	$N_{pilots} = 4$
Number of Data Subcarriers	$N_{data} = 44$
Number of Occupied Subcarriers	$N_{occupied} = 48$
Pilot Carrier Indices	(-17, -8, 8, 17)
Data Carrier Indices	(-26 to -18, -16 to -9, -7 to -1, 1 to 7, 9 to 16, 18 to 26)
Pilot Symbols	(1, -1, 1, -1)

Table 4.6: Subcarrier number and allocation indices breakdown for a single 5G-based OFDM symbol

4.1.2.4 Formulation for RL

The main difference for the RL in the 5G-based scenario is the terminology. Q-learning still applies, but instead of thinking in sub-channels as the action space, the action space consists of the selection of resource blocks. This is depicted in Fig. 4.10. The RL agent will transmit a signal that will fill one 5G-slot, which contains 14 OFDM symbols. In this case, the RL agent also has the choice between N allocations, which each contain 4 resource blocks within 1 time slot. Thus, the bandwidth of 1 allocation of 4 resource blocks is $12 \times 4 \times 15kHz = 720kHz$. For $N = 8$, this corresponds to a channel bandwidth of $8 \times 720kHz = 5.76MHz$.

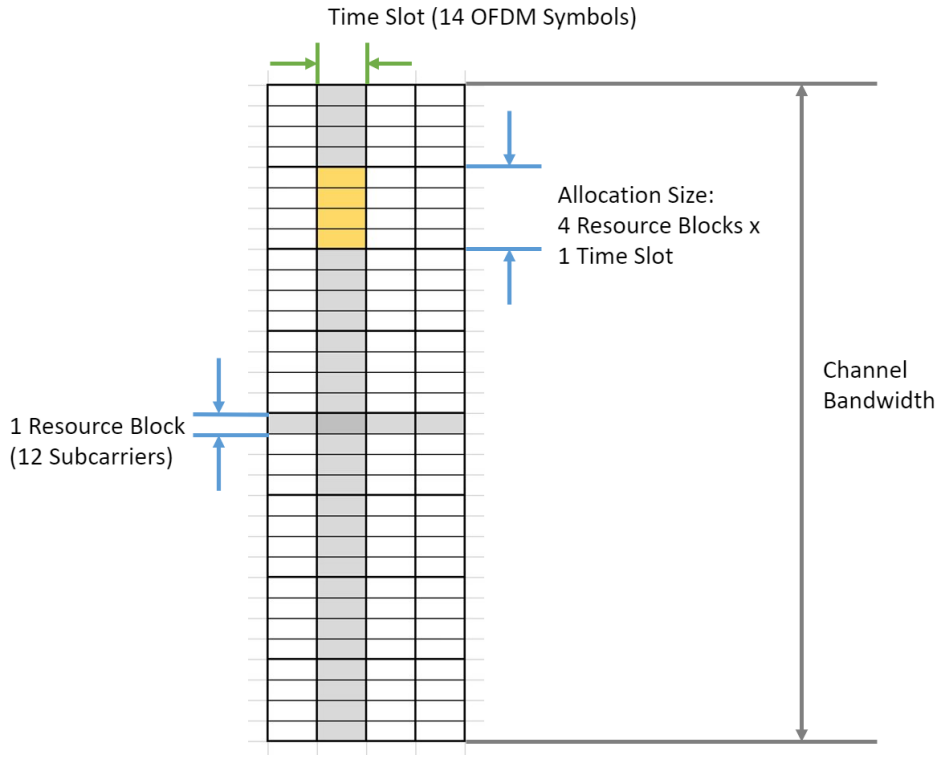


Figure 4.10: 5G channel choices for RL according to slot and resource block terminology

4.1.2.5 Goodput Calculation

To measure how much data is sent relative to the time between bursts, a goodput calculation is helpful. This measurement is useful in Chapter 8 when determining the effects of latency. This section follows the same calculations as in Section 4.1.1.5.

At the input of the channel, there are 1 header symbol and 11 data symbols, each with 64 subcarriers plus the 4 subcarriers added for the cyclic prefix. Thus, the number of data samples is $14(64+4) = 952$. The amount of deadtime depends on what is required. Suppose the deadtime is 2000. Then, for a signal with a sampling rate of 960 kHz, the amount of

time required per time step would be

$$T_{step} = \frac{952 + 2000}{960kHz} = 3.075ms \tag{4.19}$$

Thus, this corresponds to a goodput of

$$\text{Goodput (kbps)} = \frac{117 \times 8 \text{ bits}}{3.075ms} = 304 \text{ kpbs} \tag{4.20}$$

where $packet\_len\_bytes = 117$ was found as follows, where for 5G NR numerology 0 with 11 payload symbols QPSK modulated ($M = 4$),

$$packet\_len\_bytes = 11 \left[\frac{44 \times log_2(4)}{8} \right] - 4 = 117 \text{ bytes.} \tag{4.21}$$

This goodput can be validated by the theoretical OFDM data rate, as shown by the calculations below.

$$F_{sc} = \frac{960kHz}{64} = 15kHz \tag{4.22}$$

This translates to

$$T_{Data} = \frac{1}{15kHz} = 66.67\mu s \tag{4.23}$$

$$T_{GI} = \frac{4}{960kHz} = 4.167\mu s \tag{4.24}$$

$$T_{sym} = 66.67\mu s + 4.167\mu s = 70.84\mu s \tag{4.25}$$

$$R = \frac{117}{117 + 4} = 0.97 \tag{4.26}$$

These values correspond to a goodput of

$$\text{Goodput (Mbps)} = \left[\frac{44 \times log2(4) \times 0.97 \times 1}{70.84\mu s}\right] \times \frac{(64+4) \times 11}{952 + 2000} = 304 \text{ kpbs} \tag{4.27}$$

which matches the results found in Equation 4.20. These calculations were utilized in measuring the effect latency had in the system in terms of data rate in Chapter 8.

4.2 Defining the non-RL Agents: Interferers

The non-RL agents within the environment are the interferers, which are the users that can cause unsuccessful communication to the intelligent user as a result of transmitting over the same sub-channel. Fig. 4.11 discusses the interferer's implementation within GNU Radio.

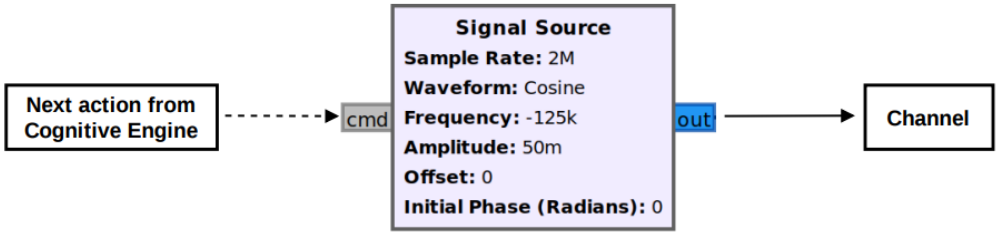


Figure 4.11: Implementation of interferers within GNU Radio. Note that its action comes from a cognitive engine that determines where to go next (the reasoning as to why is discussed in this section). The interferers are not equipped with RL.

One way to add an interferer within the same sub-channel is to add a sinusoid operating at approximately the same frequency, or within the sub-channel bandwidth. Within GNU Radio, the generation of a sinusoid is accomplished through the "Signal Source" block, as shown in Fig. 4.11. This signal operates at the same wideband sampling rate but at a

particular center frequency. This frequency, or next action for the interferer, is sent via message passing (e.g. shown through the dotted arrow). The signal is then generated at this frequency and sent to the channel. Note that the decision making for the interferers is also controlled by the cognitive engine of the custom RL block to better synchronize each time step. GNU Radio does not have synchronous message passing, so controlling all of the users within the environment through the same block helps in decreasing the chance of the users acting in a different time step. Thus, for simplicity, in this system, the decision making of all users are coupled within the same cognitive engine to ensure synchronous transmission across each time step. Even in this format, the OFDM user does not have knowledge of the other users within the environment. It still only relies on its spectrum sensing and receiver to retrieve the state and reward. Thus, the integrity of its RL is maintained regardless of the system centrally controlling other users within the same block. In a real system, however, each user would have their own cognitive engine or control mechanism.

In this system, the movement of the interferer is a fixed random pattern. For example, for a fixed random pattern, if there are 4 sub-channels, the interferer follows a randomly generated hopping sequence that repeats every 4 epochs. This hopping pattern was chosen for two reasons. The first is its randomness poses a challenging hopping pattern to be learned by Q-learning. The second is that encountering this kind of movement by real users in an RF environment would not be uncommon. For instance, Bluetooth, through a pseudo-random hopping sequence, behaves very similarly. Hence, this pattern serves as a realistic scenario of what the system could face in a real environment.

Chapter 5

Software-Defined Radio Implementation of the Transmitter, Receiver, and Channel

This chapter discusses the implementation of the RL agent's OFDM-based transmitter and receiver and the channel over which it operates.

5.1 Defining the RL Agent: OFDM Transmitter

This section details the formation of the RL Agent, which is the OFDM transmitter, as shown in Fig. 3.2.

Fig. 5.1 summarizes the order in which the transmitter chain was implemented. First, the signal was formatted with a standard protocol, such as IEEE 802.11a and 5G New Radio, as discussed in Chapter 4. Then, the data was generated and OFDM modulated. Finally,

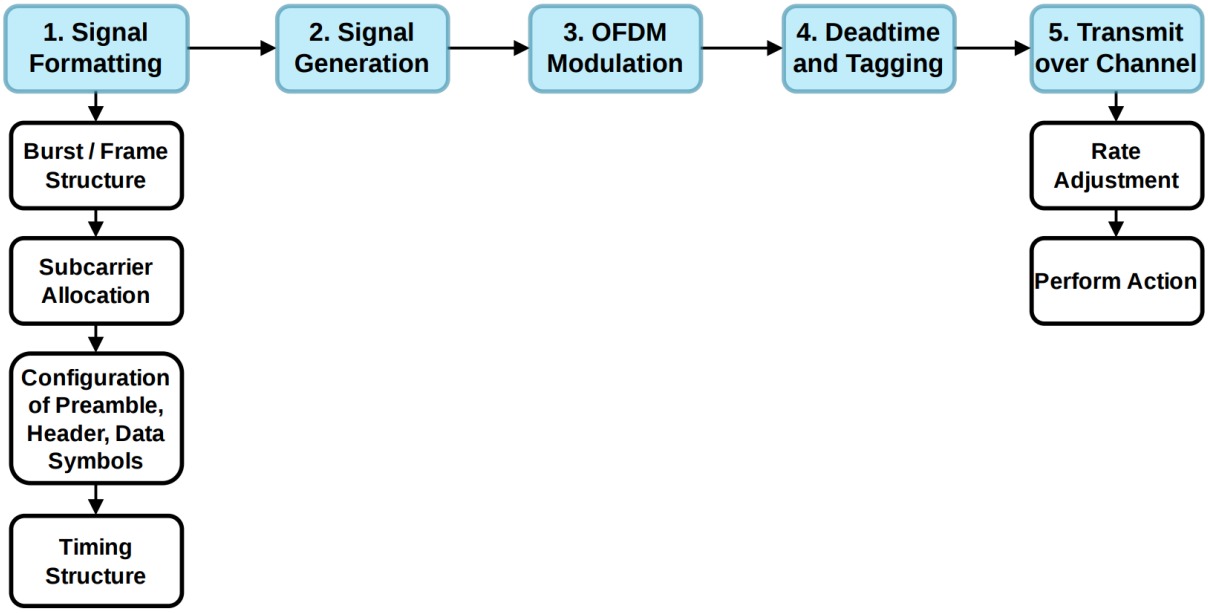


Figure 5.1: Process of implementing OFDM transmitter chain

before transmitting over the channel, the signal's sampling rate was adjusted to the wide-band sampling rate, and the signal's frequency was adjusted to match the action selected by the reinforcement learning algorithm.

5.1.1 Generating the Transmitted Signal in GNU Radio

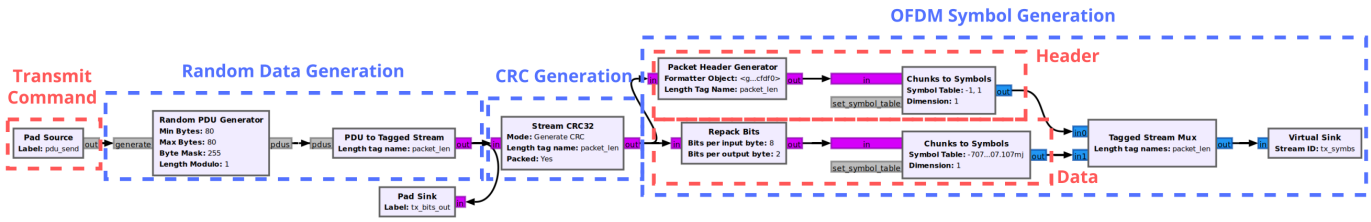


Figure 5.2: Generation of OFDM signal, with header, data payload and CRC

Once the signal is formatted, the signal can be generated. Fig. 5.2 displays the GNU

Radio blocks used in the generation of the un-modulated OFDM symbols. Discussion of the blocks proceeds from left to right, beginning with Transmit Command.

- *Transmit Command.* The command comes from the custom RL block in the main flowgraph. This message tells the transmitter to send another burst and to start the next time step.

- *Random Data Generation.* Generation begins with the "Random PDU Generator" block, which generates a fixed length random bit stream representing values from 0 to 256. This byte stream is then sent through the "PDU to Tagged Stream" block, which attaches a tag called *packet_len* to the stream at the beginning of the transmission. This tag is helpful at the receiver and is needed to properly synchronize and identify the header. It tells the receiver how long the entire transmitted signal should be.

- *CRC Generation.* The "Stream CRC32" block adds an additional 4 bytes (32 bits) to the end of the packet, increasing the value of the tag *packet_len* by 4. The system internally resets this variable. Because the data was packed from the beginning, the "Packed" input for this block is set to "Yes". These added 4 bytes are checked for errors at the receiver. If the CRC fails, the transmission also fails.

- *OFDM Symbol Generation.* This section generates the header and data symbols. The header is generated by formatting it as discussed above through digital.packet_header_ofdm(). This formatter object is passed into the "Packet Header Generator" block, which outputs a tagged stream with the added header. This tagged stream is BPSK modulated

through the "Chunks to Symbols" block, where the symbol table is determined by header_mod.points(). The output is a stream of BPSK modulated complex values. For the data symbols, the bytes are repacked to prepare for modulation through the "Repack Bits" block. Because a byte is represented by 8 bits, the bits per input byte variable is set to 8. Bits per output byte is 2 for QPSK modulation since QPSK has a value of 2 bits per symbol. This repacked byte stream is passed to the "Chunks to Symbols" block, where the symbol table is determined by payload_mod.points(). The output is a stream of QPSK modulated complex values. The header and data streams are merged through the "Tagged Stream Mux" block, which outputs a stream of complex values called *tx_symbs* ready to be OFDM modulated.

5.1.2 Applying OFDM Modulation to the Signal

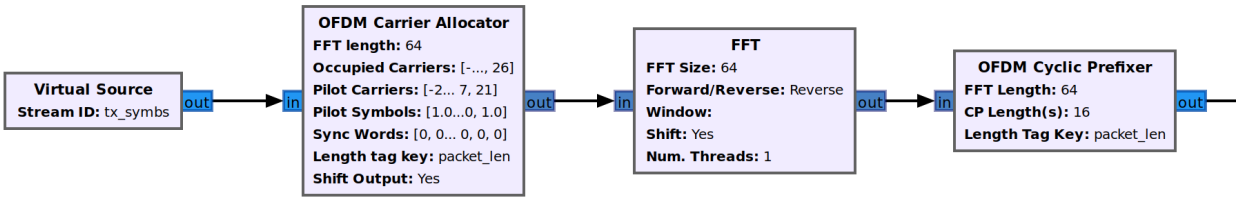


Figure 5.3: OFDM Modulation in GNU Radio

Next, the generated data is OFDM modulated. Fig. 5.3 displays the GNU Radio blocks used in the modulation of the generated symbols. These blocks are discussed below.

- *OFDM Carrier Allocator.* This block allocates the data in the input stream to certain subcarriers, as described in the "Subcarrier Allocation" portion of Section 4.1.1.2. Fig.

5.4 demonstrates what was inserted in GNU Radio for the block to properly perform subcarrier allocation. The carrier sets were inserted using the indices discussed in Table 4.2, and the sync words were generated as discussed in Section 4.1.1.3. Note that this block precedes the IFFT so that the subcarriers are allocated properly.

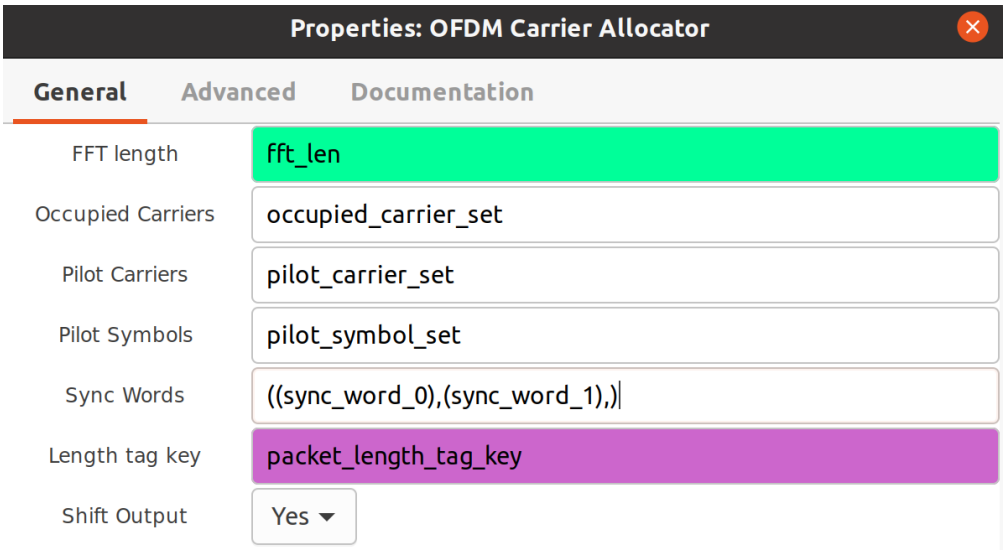


Figure 5.4: Subcarrier allocation in GNU Radio

- *Inverse Fast Fourier Transform (IFFT).* Because, at this point, the signal is in the frequency domain, an IFFT is performed to transfer the complex stream from the frequency to time domain in preparation for transmission over a channel. At the transmitter, in GNU Radio, the "FFT" block is set to 'Reverse', which sets the block to an IFFT operation. This IFFT splits the incoming complex set of symbols into a series of N parallel complex symbols, each of which are modulated by a different subcarrier.

- *OFDM Cyclic Prefixer.* This block adds a cyclic prefix, which is typically 25% of the total number of subcarriers for IEEE 802.11a. Thus, the cyclic prefix length for IEEE 802.11a is 25% of 64, which is $L_{CP} = 16$. For 5G NR, $L_{CP} = 4$. A copy of the last $L_{CP}/64$ (%) of the symbol is appended to the front to alleviate the effects of intersymbol interference (ISI), which occurs when symbols overlap, and fading from a multi-path channel, which occurs when interference along a channel causes multiple versions of the received signal to arrive at the receiver.

5.1.3 Deadtime and Tagging

As discussed in Section 4.1.1.4, tags and deadtime were implemented after OFDM modulating the signal. This sub-section gives further details on why deadtime and tagging were implemented and its implementation in GNU Radio.

Because GNU Radio uses real-time stream processing, transmission is continuous, meaning the transmitter continues to transmit until it runs out of samples. Thus, it is difficult to tell GNU Radio when and for how long to perform certain processes, such as spectrum sensing and RL. This poses a problem to the implementation of reinforcement learning because of inherent latency in real-time systems, which could cause the user to perform its action in the wrong portion of the time step and perhaps within the wrong time step altogether. To measure and prevent this latency, the blocks in Fig. 5.5 were added after the OFDM modulation blocks. It consists of using tagging, message passing, and a period of non-transmission (e.g. deadtime) to understand and overcome this limitation. The sub-sections that follow

outline why deadtime and tagging were needed and how they were implemented in GNU Radio.

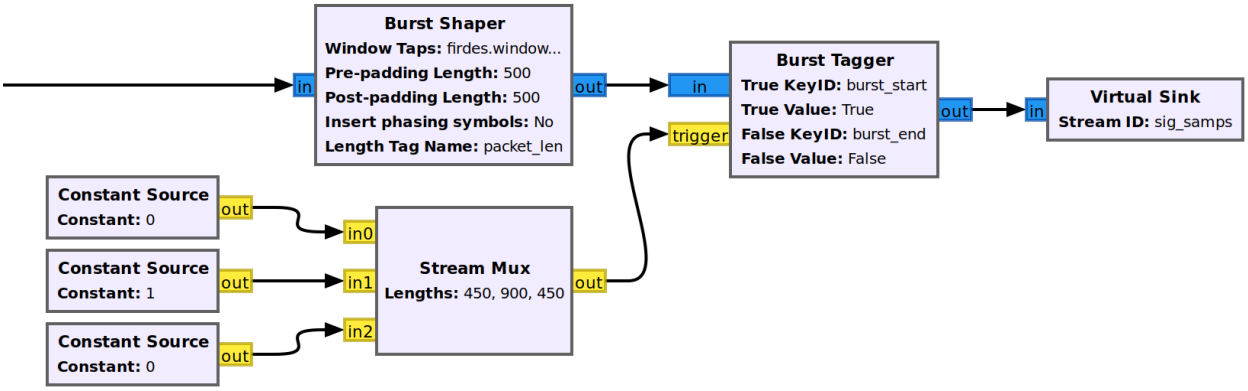


Figure 5.5: Configuration of deadtime and tagging in GNU Radio

5.1.3.1 Duration of Computation - Deadtime

Due to the continuous nature of the timing structure discussed in Fig. 4.6, it can be difficult for the user to determine how long it needs to sense and perform computations (e.g. Q-learning). It is possible to temporarily stop the program to perform these tasks, but when implemented with USRPs for over-the-air (OTA) transmission, the connection could fail. It is also undesirable to continuously spam messages at the USRPs. Therefore, a deadtime can help in giving the USRPs time between transmissions to perform these operations, e.g. sensing and reinforcement learning. The reinforcement learning block, for instance, needs both the sensing results and packet header information for its algorithm. Therefore, the reinforcement learning block needs to wait until both the sensing blocks and receiver are finished processing. Additionally, even after it has received both inputs, the reinforcement

learning block still needs time to predict the transmitter's next action. Thus, the deadtime is also useful for additional computation time needed by the reinforcement learning algorithm, as demonstrated by Fig. 4.6.

5.1.3.2 Time of Computation - Tagging

Contrary to deadtime, which is concerned about "how long" to perform an operation, tagging determines "when" to perform the operation. The asynchronous behavior of blocks within GNU Radio makes it difficult for the user to know when to start sensing. Therefore, tags were added before and after the burst to make a clear distinction between the deadtime and transmission periods such that it knows only to sense the spectrum between the tags and reinforcement learning outside the tags. Additionally, the "packet_len" tag is useful in acquiring the reward. After an action is performed, it is difficult to know when the receiver has the reward ready. However, with an appropriate amount of deadtime, the reward should be available by the beginning of the next epoch. Thus, this tag, always located at the beginning of the transmission, is used to set a definitive time of when the reward should be ready. As a result, the RL block can be confident it is receiving the correct reward. If by chance the RL block does not receive the reward by the time this tag appears, then the receiver has latency and deadtime should be increased.

5.1.3.3 Implementation of Deadtime and Tagging

Once the signal has been OFDM modulated, the "Burst Shaper" and "Burst Tagger" blocks were added. These blocks are used to add tags at a particular time of the signal for the purpose of identifying when the OFDM burst approximately begins and ends. It uses a trigger, based on a constant signal source, to add these tags. Notice that the trigger consists of three separate sources. It starts with a string of 0's, of which construct the first half of the deadtime, followed by a string of 1's. The string of 1's carries the OFDM burst. When the "Burst Tagger" block notices the transition from 0's to 1's, it adds the "burst_start" tag. Another string of 0's was added after the OFDM burst so that the transition from 1's to 0's is noticed by the block. When that trigger occurs, the "burst_end" tag is added. This last string of 0's constructs the last half of the total deadtime. Thus, sensing will only occur during the interval ('burst_start', 'burst_end').

5.1.4 Preparing to Transmit over a Channel

The formation of the signal is now complete. However, it must be properly configured to transmit over a channel. This sub-section details that process.

5.1.4.1 Convert from Baseband to Passband

Because the OFDM burst transmits at its own sampling rate in a system with a wider bandwidth, converting the OFDM burst to the sampling rate that matches the wide-band

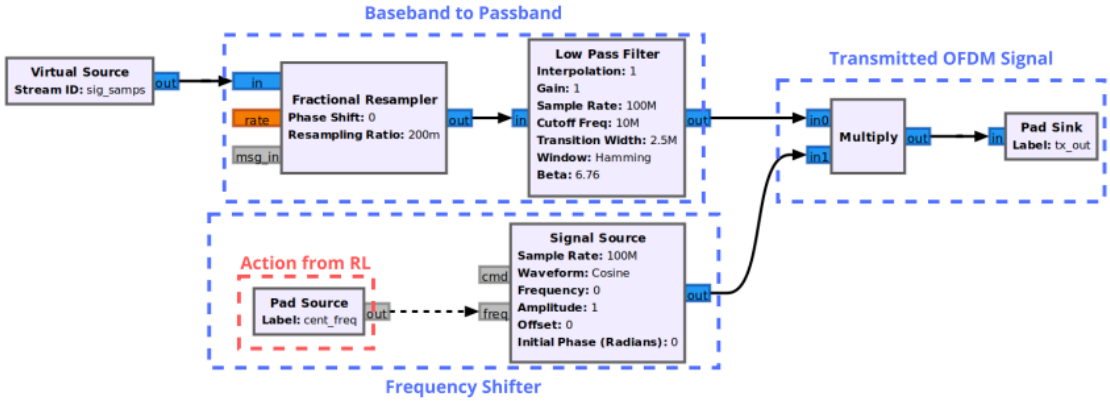


Figure 5.6: Blocks used to prepare the signal for transmission over a channel. The signal is converted from its signal sampling rate to the spectrum's wideband sampling rate. A center frequency among this wideband spectrum is also applied to the signal.

sampling rate of the spectrum is required. This was accomplished in GNU Radio through the "Fractional Resampler" block and by applying a low-pass filter, as displayed in Fig. 5.6. By doing so, the signal can transmit across the channel at its original bandwidth.

5.1.4.2 Performing an Action: Frequency Shifter

Because the transmitter serves as the RL agent, it must perform the action it received from the RL algorithm. In this framework, the action is the center frequency of the chosen sub-channel. In order for the system to have the ability to perform this action, the "Signal Source" block was added. It multiplies the resampled signal by a sinusoid centered at the new frequency, such that the result is the resampled signal at that new frequency. An example output is shown in Fig. 5.7, where the action required the signal to be centered at 0 Hz. This resulting signal is also the OFDM signal transmitted across the channel. It is at this

point that the implementation of the transmitter is complete.

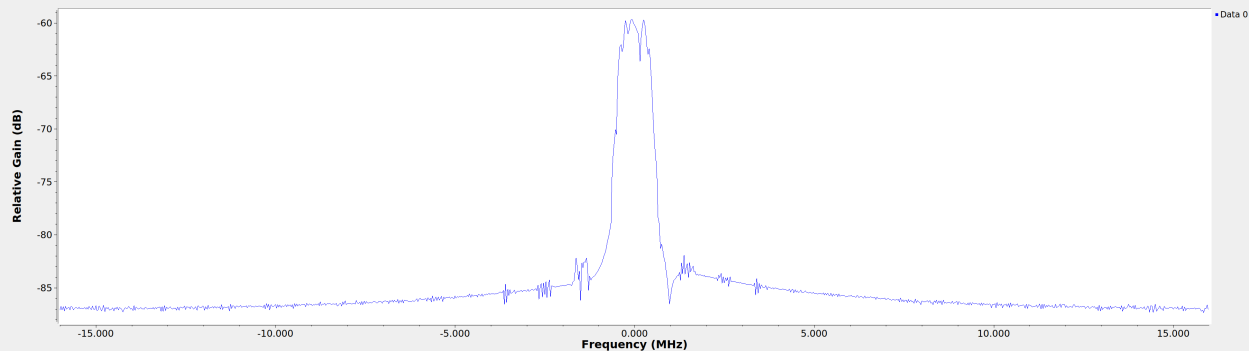


Figure 5.7: Transmitted OFDM signal after resampling and frequency shifting in GNU Radio

Note that the center frequency of the signal is relative to the channel's operating frequency, wideband sampling rate, and sub-channel allocation. For example, suppose a radio was set to transmit across a portion of the spectrum centered at 2.4 GHz, and the number of sub-channels was $N = 8$. Suppose also that the radio can only look at portions of at most 16 MHz, which serves as the wideband sampling rate. With 8 sub-channels, this corresponds to each sub-channel having a bandwidth of $16MHz/8 = 2MHz$, with the possible frequencies spanning $(-8MHz + 2.4GHz, 6MHz + 2.4GHz)$. If the transmitter uses the entire sub-channel, this also means the transmitter has a signal sampling rate of 2 MHz, with possible frequencies spanning $(-8MHz, 6MHz)$. If the action from the RL block is to move 2 MHz to the right, though the radio over-the-air would move to 2.402 GHz, the transmitter would receive the action of 2 MHz, which is relative to 2.402 GHz.

5.1.5 Summary of the OFDM Transmitter

In this section, the implementation of the OFDM transmitter, e.g. the RL agent, in GNU Radio was detailed. The full transmitter chain is shown in Fig. 5.8. To create the transmitted signal, random data was first generated and then formatted for IEEE 802.11a or 5G NR through header and payload streams. These generated streams were merged and then OFDM modulated. Tags were added to the signal to indicate the burst portion of the signal, and deadtime was implemented to handle system latency. The signal was then converted from baseband to passband so that it could transmit across a wideband channel. Before going across the channel, the signal was also shifted to the chosen center frequency from the RL algorithm.

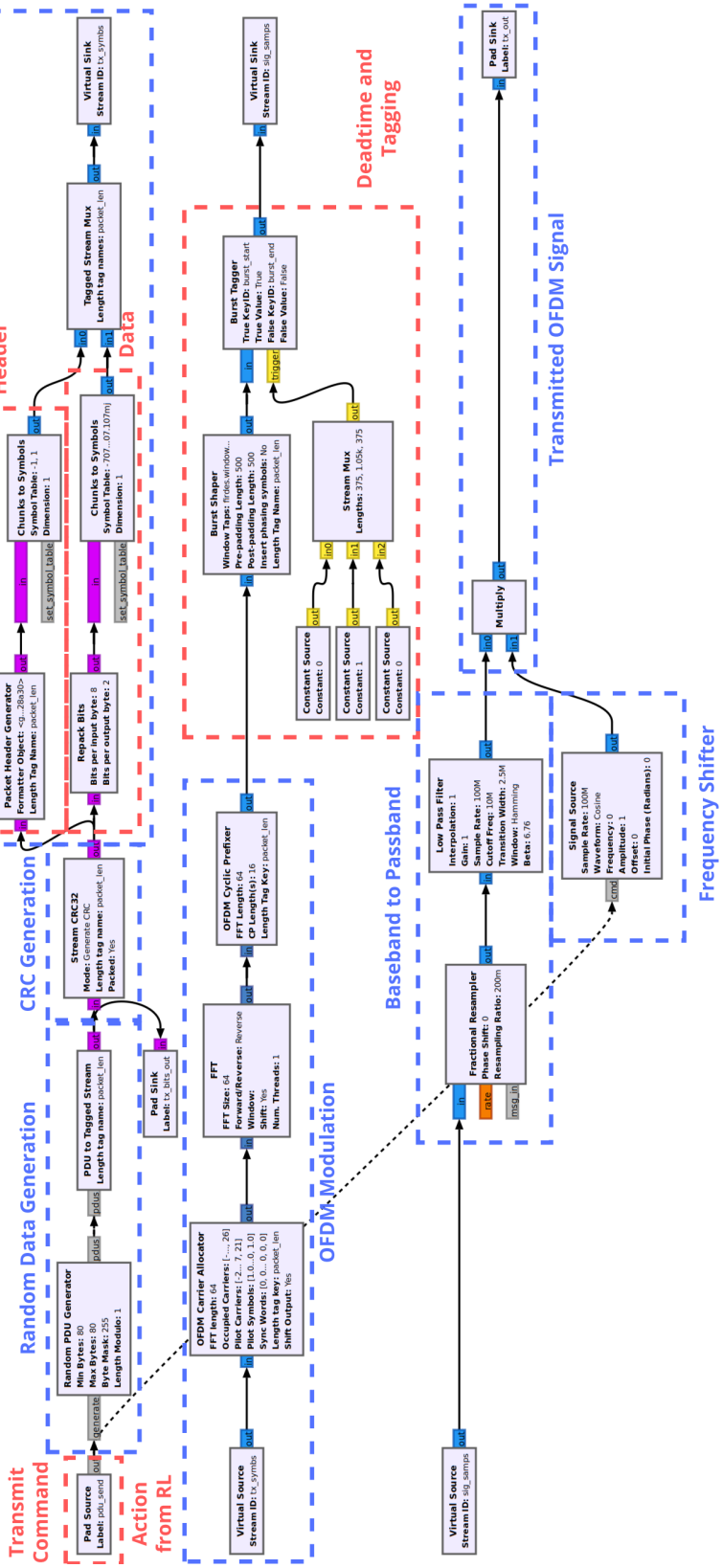
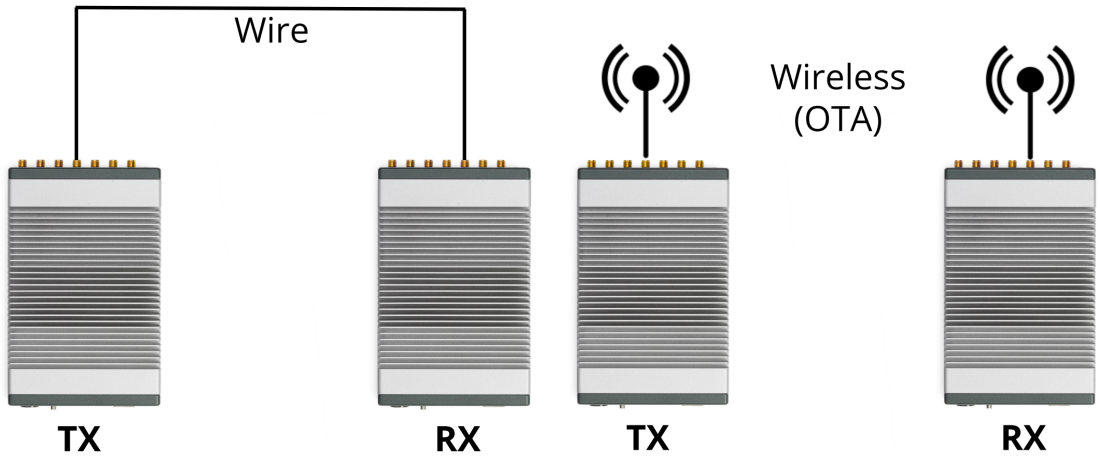


Figure 5.8: OFDM transmitter flowgraph in GNU Radio

5.2 Defining the Environment: Channel

The environment over which users interact is typically through a wireless channel within the radio frequency space. In this work, USRPs are used to transmit and receive. With USRPs, the channel can either be wired or wireless, as shown in Fig. 5.9. For the wired setup in Fig. 5.9a, a wired cable is used to connect the transmitter and receiver. For the wireless setup in Fig. 5.9b, antennas are used. For testing purposes, particularly for when new blocks were added to the framework, the wired configuration was utilized so that testing occurred over more stable environment conditions.



(a) Wired channel setup (b) Wireless channel setup

Figure 5.9: Hardware channel setup with the Ettus USRP E320. The USRP figure was utilized from [46]. Note that both the transmitted OFDM signal and the interferers are transmitting from the same USRP TX. Determined as Fair Use, as explained in Appendix B.

In GNU Radio, the USRPs were accessed through its UHD package. The "USRP Sink" block is used as the transmitter, and the "USRP Source" block is used as the receiver. To configure the transmit power, receive power, and linear gain, the blocks in Fig. 5.10 were used. This script generates a sinusoid at an arbitrary frequency but with the same wideband sampling rate that will be used in the main flowgraph. The reason is because both the transmitter and receiver operate not at the signal sampling rate but at the wideband sampling rate. This script is mainly helpful in determining the operating range in power and gain of the USRPs and was required because automatic gain control (AGC) did not work properly for the short bursts generated by this framework. AGC requires adjusting to the power of the signal, but because the bursts are so short, it is difficult for AGC in the USRPs to perform this function. Lastly, for this work, to conserve energy, it is desirable to operate at the lowest power possible that still results in successful communication.

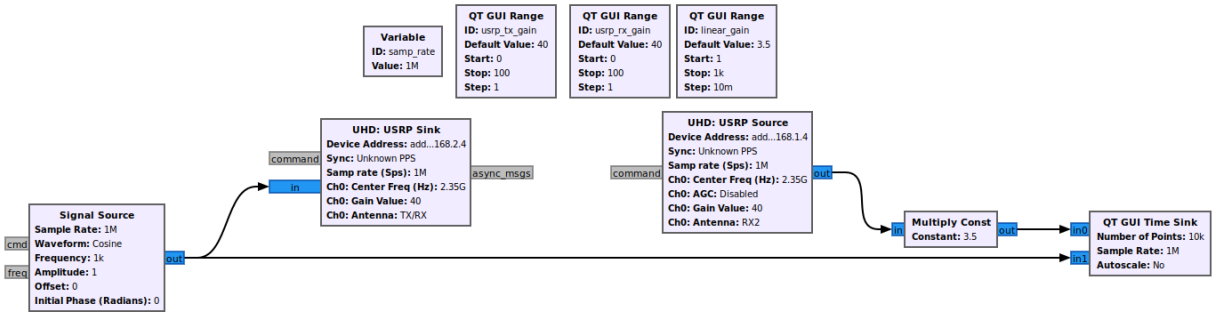


Figure 5.10: GNU Radio script used to configure the power and gain of the USRPs

Examples outputs of the script in Fig. 5.10 are shown in Figures 5.11 and 5.12. A valid output is depicted in Fig. 5.11. To be valid, the amplitude of the OTA signal (in red and blue) must approximately match the amplitude of the simulated sinousoid (in green

and black). More specifically, for a signal with noise, approximately the middle of the amplitude of the OTA signal should match the amplitude of the simulated sinousoid. Fig.

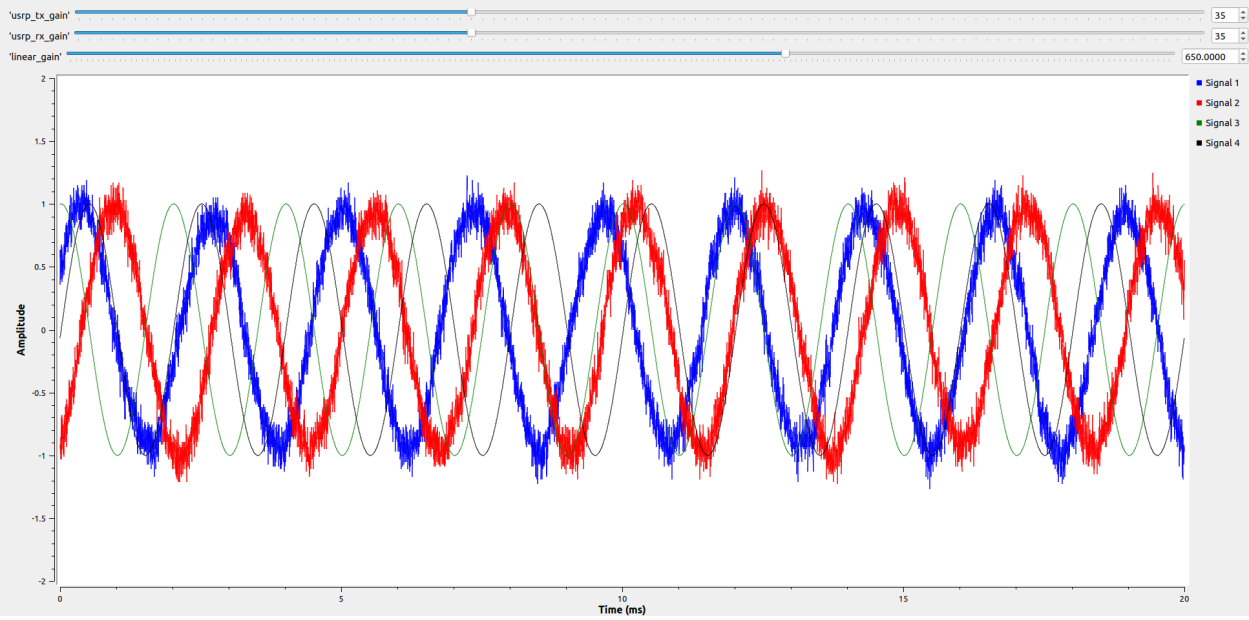
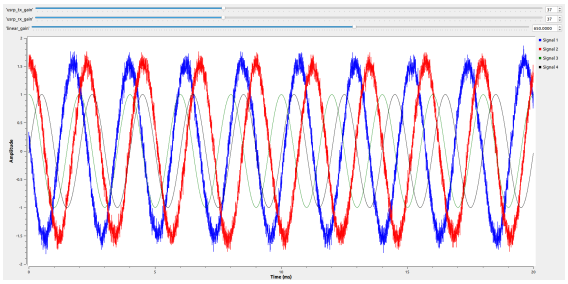


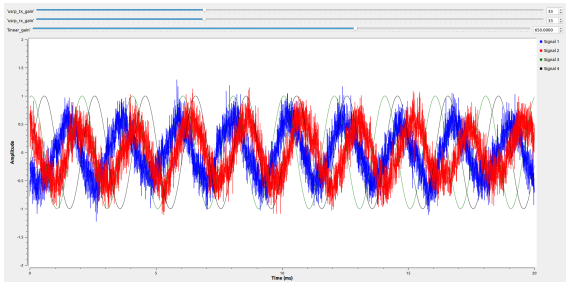
Figure 5.11: Example of valid USRP test result (power matches the test signal and not too close to the noise floor)

5.12 demonstrates different examples of invalid outputs that could result in the receiver not being able to properly demodulate the transmitted signal. The first case, as shown in Fig. 5.12a, occurs when the amplitude of the OTA signal is higher than the amplitude of the simulated sinusoid. This happens when the power of the transmitter and receiver and/or the linear gain are set too high. To resolve the issue, the power can be brought down. The second case, as shown in Fig. 5.12b, occurs when the amplitude of the OTA signal is lower than the amplitude of the simulated sinusoid. This happens when the power of the transmitter and receiver and/or the linear gain are set too low. To resolve the issue, the

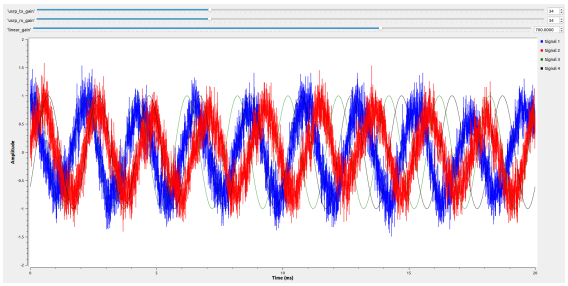
power can be increased. The third case, as shown in Fig. 5.12c, occurs when the OTA signal is too close to the noise floor. Though the amplitude approximately matches the simulated sinusoid's amplitude, this output can occur if the signal is too noisy. This happens when the linear gain is set too high. In this case, the linear gain should be decreased. As a result, an increase in the transmitter and receiver power is necessary to maintain the amplitude.



(a) Case 1: Power too high (overshooting)



(b) Case 2: Power too low (undershooting)



(c) Case 3: Good power level but too much noise

Figure 5.12: Examples of invalid USRP test results

5.3 Acquiring the Reward: OFDM Receiver

In this section, how the reward was obtained from the receiver is discussed, along with a detailed summary of how the receive chain was formed. It is important to note that the receiver uses the same protocol formation as the transmitter. Therefore, the burst structure and subcarrier allocation are the same at the receiver.

5.3.1 Preparing for Demodulation

5.3.1.1 Convert from Passband to Baseband

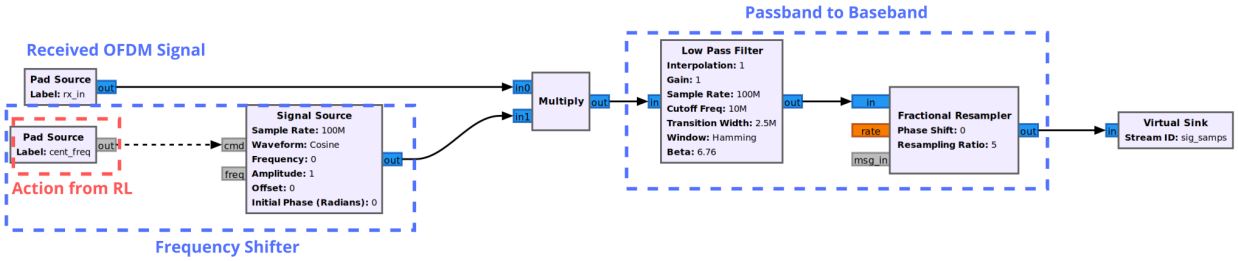


Figure 5.13: Resampling back to the signal sampling rate and frequency shifting to prepare the received signal for demodulation

The first step after transmitting across the channel is to process the signal such that it is prepared for demodulation. There are two adjustments that must be made. When the signal gets to the receive chain, it is still at the transmitted frequency. This frequency must be brought back to 0 Hz so that proper demodulation can occur. To do so, the receiver also requires an action from the cognitive engine. However, while the transmitter gets the true action, e.g. the true frequency, the receiver gets the negative of the true frequency to bring

the signal to 0 Hz. Additionally, the signal is still operating at the wide-band sampling rate. However, for proper OFDM demodulation, the signal must be brought back to its original sampling rate. Therefore, the sampling rate is adjusted once more through the "Fractional Resampler" block and the low pass filter, as shown in Fig. 5.13. These blocks also remove other signals within the wideband spectrum to isolate the signal of interest, which is the received signal. The blocks shown in Fig. 5.13 are reversing the adjustments made at the transmitter in Section 5.1.4 to transmit the signal over a channel.

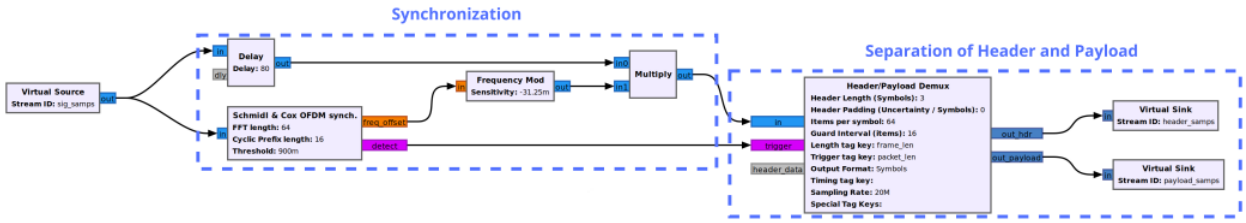


Figure 5.14: Blocks performing synchronization and separating the header and payload symbol streams

5.3.1.2 Synchronization

Before demodulating, the beginning of the OFDM burst must be detected. This is accomplished through synchronization methods, as shown in the synchronization block in Fig. 5.14. One of the most known synchronization algorithms is the Schmidl and Cox method, which in GNU Radio is the "Schmidl and Cox OFDM synch." block. The first output of this block is "freq offset", which estimates the fine frequency offset required to properly demodulate the signal. As described through the GNU Radio documentation [47], the frequency offset

is normalized to

$$f_{norm} = \frac{2(freq_{offset})}{L_{FFT}} \tag{5.1}$$

where division by L_{FFT} is applied to scale the frequency offset by the OFDM symbol duration, which is determined by the length of the FFT, e.g. the number of total subcarriers. The output of Equation 5.1 can be applied through the "Frequency Mod" block, which outputs a frequency modulated signal according to a sensitivity value defined as

$$k = 2\pi \frac{f_{\Delta}}{f_s} \tag{5.2}$$

in the GNU Radio documentation [48]. The sensitivity specifies how much the phase changes, and "Frequency Mod" uses this sensitivity to correct the fine frequency offset [48]. The sensitivity used in this framework is defined in Equation 5.1. This block also internally evaluates the course frequency offset. If a course frequency offset is found, the tag 'ofdm_sync_carr_offset' is internally added to the first item to be corrected later by the "OFDM Frame Equalizer" block [47], [49]. The second output of the "Schmidl and Cox OFDM synch." block detects the start of an OFDM frame. When it does detect the start of the frame, it sends a trigger to the "Header/ Payload Demux" block, which drops samples until it receives a trigger [50].

5.3.1.3 Separation of the Header and Payload

The header and payload signal streams are separated through the "Header/ Payload Demux" block. The parameters used in the configuration of this block are shown in Fig. 5.15. Notice

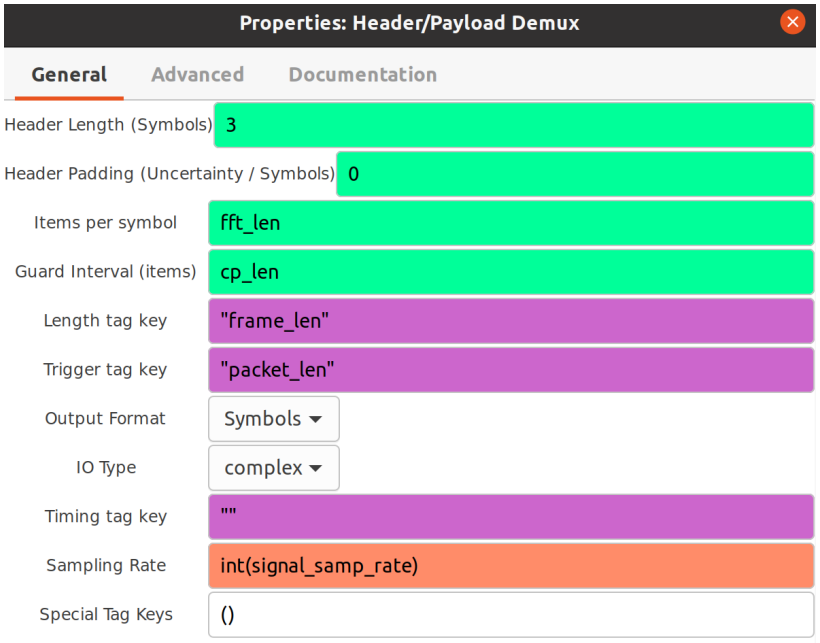


Figure 5.15: Parameters used to configure the "Header / Payload Demux" block in GNU Radio

that the "Header Length" was set to 3 symbols. This is to account for the two synchronization words, which are not removed until the header stream is demodulated. Also, this block drops the cyclic prefix added to each symbol, since it is no longer needed.

5.3.2 OFDM Demodulation

Once the signal is ready for demodulation, it goes through a series of blocks to retain the original bit stream. The process is described through Fig. 5.16, which displays the blocks required for both the header and payload streams.

The basic OFDM demodulation process is detailed below. The process is mostly the same

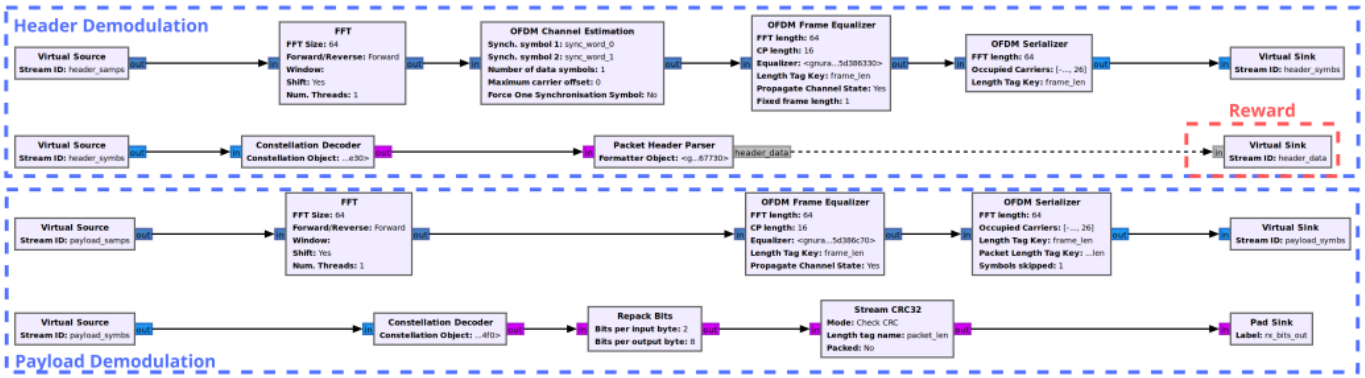


Figure 5.16: Blocks used in GNU Radio for the OFDM demodulation of the header and payload streams

for the header and payload, except that the payload does not use channel estimation because the "OFDM Channel Estimation" block in GNU Radio expects synchronization symbols as input, which is only included in the header stream [41]. The payload stream only contains the data of the 7 payload symbols, whereas the header streams contains the two synch words and the one header symbol.

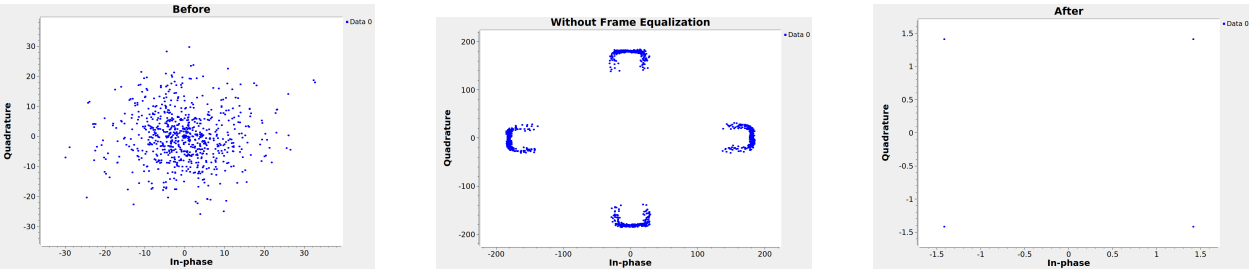
- *FFT*: The bit stream is decoded in the frequency domain. Therefore, both the header and payload streams must be converted from the time to frequency domain, which is accomplished through the "FFT" block in GNU Radio, which is set to "Forward" to perform the FFT operation.

- *Channel Estimation*: When transmitting over a channel, the received signal can become distorted due to poor channel conditions. To recover the original signal, channel estimation is performed. Channel estimation estimates the effect of the channel and

compensates for that effect at the receiver. For OFDM, channel estimation typically estimates the channel response of each subcarrier through the known preamble and pilot symbols. This is accomplished through the "OFDM Channel Estimation" block, of which also estimates the course frequency offset. This block additionally removes the two synch words and outputs the header data only [41]. This block is only applied for the header.

- *Channel Equalization*: Equalization uses known information of the data and the channel estimates to equalize the OFDM data, meaning it attempts to reverse the distortion caused by the channel to recover the transmitted signal. In GNU Radio, equalization is accomplished through the "OFDM Frame Equalizer", which corrects the course frequency offset and performs equalization to recover the original signal [49]. For example, Fig. 5.17 displays the decoded constellation after going through the receiver chain. Fig. 5.17a shows the results of no frequency correction, which would occur before the demodulation process began. At this point, the constellation is spread, which is an indication of distortion across the channel, and hence synchronization and equalization methods are needed. After applying synchronization and separating the streams, Fig. 5.17b shows the decoded constellation without the course frequency offset correction and equalization, e.g. without the "OFDM Frame Equalizer" block. With this block, a result as shown in Fig. 5.17c occurs, which is the desired output for a QPSK modulated signal.

- *Serialization*: At this point, processing has occurred through N parallel streams of



(a) Before any correction

(b) With only fine frequency offset correction

(c) After all corrections, with fine and course frequency offset correction

Figure 5.17: Effect of frequency offset correction

data, where N represents the number of subcarriers. The "OFDM Serializer" block is used to combine the N parallel streams into one stream [51].

- *Decoding to Bit Stream*: After the stream has been serialized, it is ready to be converted back to its original bit stream. It does this through the "Constellation Decoder" block, which requires the modulation object of the stream, such as digital.constellation_qpsk() for the payload and digital.constellation_bpsk() for the header. This block demodulates the data according to the stream's respective modulation scheme, where the output is a stream of packed bytes.

For the payload stream, the demodulated data is converted back to the original bit stream. That process is described below. Note that the payload is only demodulated if a non-corrupted CRC is found in the header.

- *Convert Bytes to Bits*: To get bits, the "Repack Bits" block is used to convert the

packed bytes to bits, unpacking the bytes to get a length of 8 bits each.

- *Check CRC*: Bit errors are checked via the "Stream CRC32" block in GNU Radio. It uses a 32-bit cyclic redundancy check. If the bits pass through this block, then demodulation was successful. Otherwise, this block fails and will return nothing.

Note that the header stream passes through the "Packet Header Parser" block. This block decodes the header information in an attempt to retrieve the stored header information and outputs it as a message. If this block fails, then it will return *False*, as a result of the header being corrupted. It can also return *False* if the threshold of the "Schmidl and Cox OFDM synch." block is too low. If it is too low, many false detections from interference and/or noisy channel conditions can occur, of which backlogs the "Packet Header Parser" block with multiple *False* readings. This can cause latency to increase in the receiver because these false detections can delay when it detects the start of the OFDM transmission. This behavior is due to the message queue of GNU Radio. Thus, it is desirable to choose an appropriate threshold so that the message queue is not overloaded, resulting in lower latency in the receiver chain. Conversely, for a true detection, the "Packet Header Parser" block provides header information, such as the frame number and packet length. As displayed in Fig. 5.16, the output of this block serves as the reward. If the header passed, then communication was successful, and a positive reward is returned to the cognitive engine. If the header did not pass, then communication was unsuccessful as a result of being interfered with, and a negative reward is returned to the cognitive engine. For the reward, the assumption is that if the header does not pass, then the payload stream is most likely corrupted as well. Likewise,

if the header did pass, the assumption is that the payload stream was also not interfered with.

5.3.3 Summary of the OFDM Receiver

In this section, the implementation of the OFDM receiver in GNU Radio was detailed. The full receiver chain is shown in Fig. 5.18. The received signal prepared for demodulation by converting the signal from passband to baseband, used synchronization methods to correct frequency offsets, and then separated the header and payload stream to be demodulated separately. The OFDM demodulation process was then described, where the payload stream was converted to a bit stream and the header stream was used to find the reward.

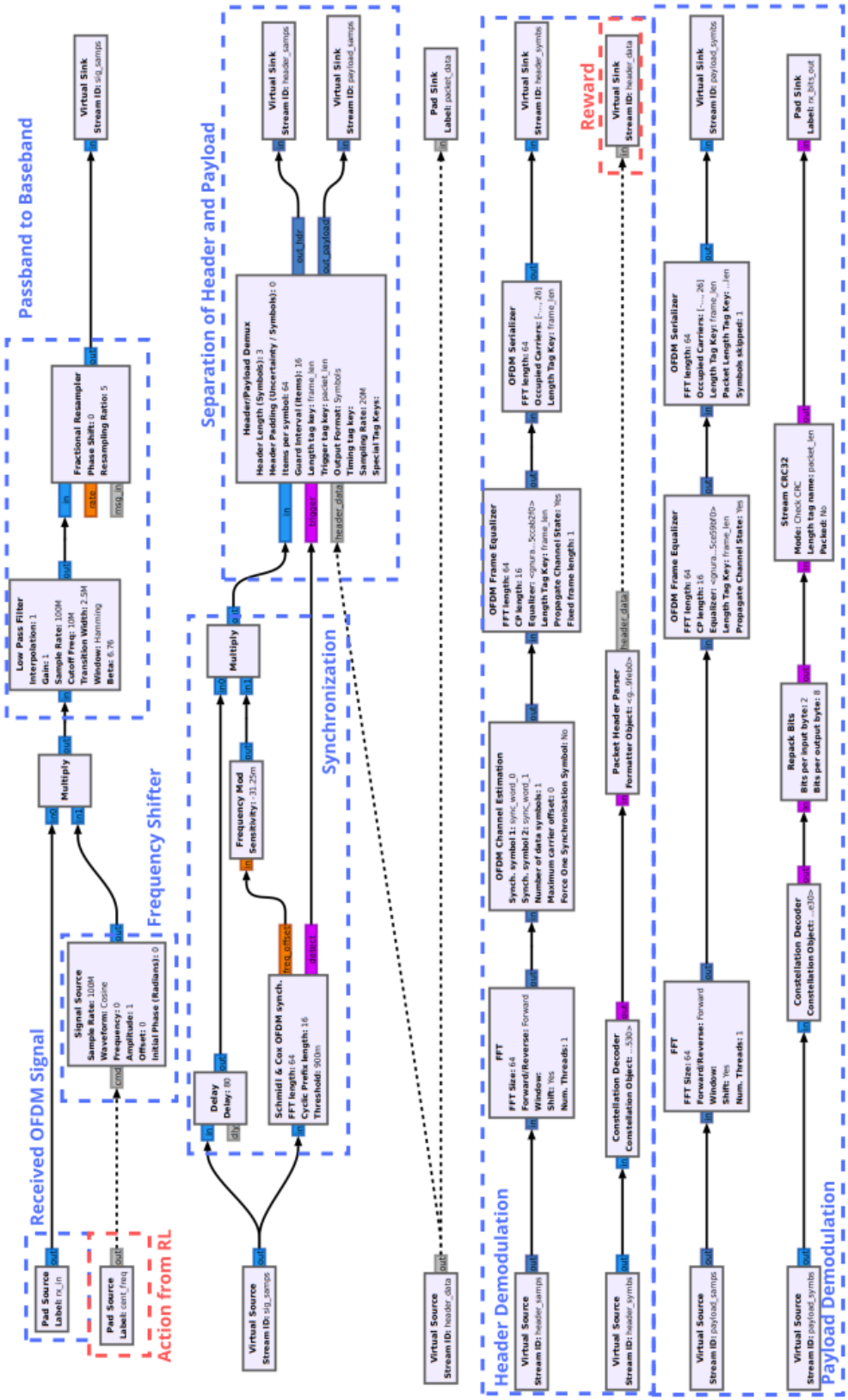


Figure 5.18: OFDM receiver flowgraph in GNU Radio

Chapter 6

Software-Defined Radio Implementation of Spectrum Sensing

This chapter discusses the implementation of the spectrum sensing detectors used to acquire the state of the RF environment, as shown in Fig. 3.2. Detectors implemented include the polyphase channelizer detector, a time-based energy detection method, and a PSD-based detector, a frequency-based energy detection method. Two were implemented so that the computational time difference between sensing algorithms could be measured and analyzed to infer whether RL is limited by different detectors.

6.1 Acquiring the State: Wideband Spectrum Sensing (WBSS)

This section details the implementation of the two detectors used to acquire the state observation for the RL algorithm. First is the polyphase channelizer detector, followed by the PSD-based detector. The performance of these two detectors are compared in Chapter 8. More specifically, any differences between the two detectors, particularly in processing latency, is investigated.

6.1.1 Polyphase Channelizer Detector

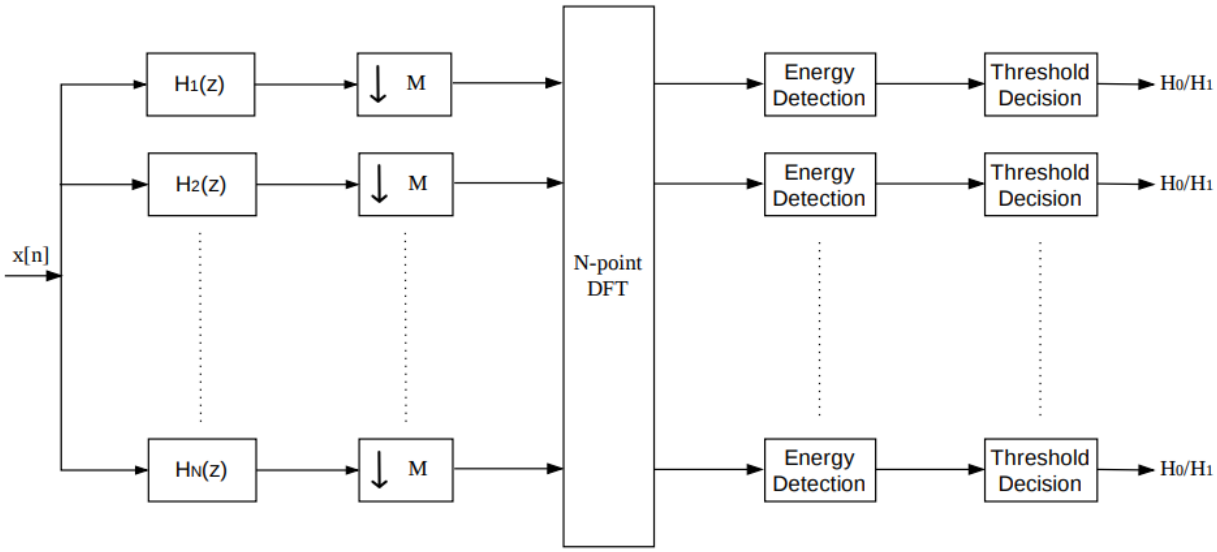


Figure 6.1: Filter bank-based sensing with a polyphase channelizer and energy detection [52]

The environment state is acquired through a polyphase channelizer detector, which con-

sists of the polyphase filter bank shown in Fig. 6.1. A wideband signal $x[n]$ is split into a subset of N sub-signals by applying a low pass filter $H(z)$ and downsampling by M to convert from the wide-band sampling rate to the signal's sample rate. Narrowband sensing, e.g. energy detection, is applied to each individual sub-signal to get a representation of signal presence in the environment. The energy of one sub-channel, say x_1 with K number of samples would be

$$E[x_1] = \frac{1}{K} \sum_{i=1}^{K} |x_1[i]|^2 \tag{6.1}$$

as defined in [53]. Equation 6.1 is applied to every sub-signal. Once the energy has been found, a threshold can be applied to determine the presence of a signal. Determining the presence of a signal is based on a Hypothesis test, with two possible outputs H_0 and H_1. H_0 is the case where no signal is present, when only the noise $w[n]$ is captured. H_1 captures the signal $x[n]$ plus the noise $w[n]$. This is summarized below, as defined in [53].

$$H_0 : y[n] = w[n] \tag{6.2}$$

$$H_1 : y[n] = x[n] + w[n] \tag{6.3}$$

A signal is present if the energy is above some fixed noise threshold λ. Based on that value, if $E[x_1] > \lambda$, then H_1 is chosen, and the output would be a 1. This result represents an occupied sub-channel. If $E[x_1] < \lambda$, then H_0 is chosen, and the output would be a 0. This result represents an unoccupied sub-channel.

Within GNU Radio, the detector was implemented as shown in Fig. 6.2. It takes the transmitted stream after it travels over the channel and passes it through the "Polyphase

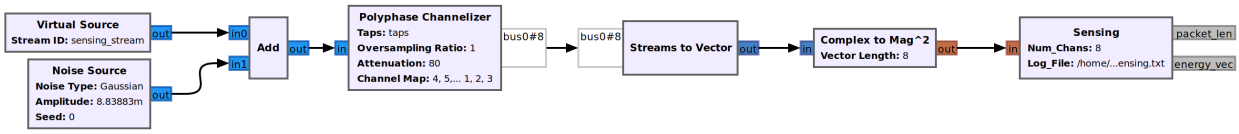


Figure 6.2: Blocks used to implement the detector in GNU Radio

Channelizer" block in GNU Radio. Note that the detector uses an Additive White Gaussian Noise (AWGN) channel rather than the received stream from the USRP. This is because the USRPs remove the tags. Therefore, the sensor occurs at the transmitter, rather than the receiver, to ensure that sensing occurs during the burst region through the use of these tags. Otherwise, it is difficult to tell the sensor when to start and stop sensing. The same applies for the PSD sensor. Using AWGN as the channel for the detector is analogous to a transmitter listening to the real RF spectrum while transmitting or during deadtime in a real world scenario. As such, having the detector at the receiver is not needed if the transmitter can observe the same spectrum.

In reference to Fig. 6.1, the "Polyphase Channelizer" block performs the wideband to narrowband operation, where the signal $x[n]$ is split into N sub-signals. The channelized output was then converted to a vector through the "Streams to Vector" block. For the vector output, which contains N parallel vectors, the "Complex to Mag Squared" block is used to perform the $|x_i|^2$ operation of Equation 6.1. The custom "Sensing" block averages the energy between the tags "burst_start" and "burst_end". The result, e.g. the state observation, is sent via message passing to the cognitive engine. Notice that the "Sensing" block has a second output, packet_len. This message occurs when the block finds the beginning of a

transmission, signaling the start of a time step. This message is sent to the RL block to signal that the receiver from the previous time step should be completed by this point. This is also applied to the output of the PSD sensor.

6.1.2 PSD-Based Detector

The PSD-based detector is also based on energy detection but without the channelizer. In fact, the PSD-based detector calculates the energy directly in the frequency domain, whereas the channelizer performs energy detection from the time domain signal. Thus, rather than using the "Polyphase Channelizer" block, the "FFT" block is used. Then, the average power is found through the "Complex to Mag Squared" block. The same threshold process applies as well. However, because the PSD detector does not contain a channelizer, the custom "PSD Sensing" block is used to break the FFT stream from the FFT size N_{FFT} to N individual outputs, one for each sub-channel. Additionally, to ensure the results are from the current time step, the "Tagged Stream Align" block is used to align the FFT to the beginning of the sensing region of the signal, 'burst_start'.

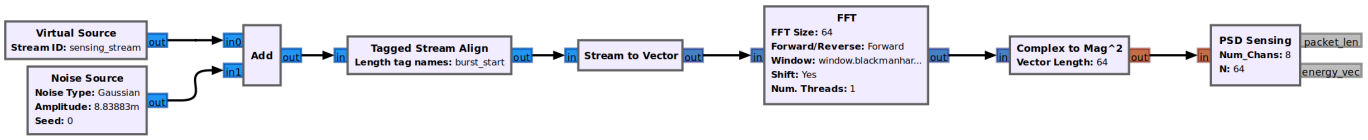


Figure 6.3: Blocks used to create the PSD-based detector in GNU Radio

Chapter 7

Software-Defined Radio Implementation of Reinforcement Learning

This chapter discusses the implementation of the custom RL block, also called the cognitive engine, within GNU Radio. Implementation discussion is less of how Q-learning was implemented, which was explained in Chapter 2, and more of how state and reward feedback was made accessible to the cognitive engine and how the cognitive engine was able to process that feedback through GNU Radio message handlers.

7.1 Deciding the Action: Custom Reinforcement Learning Block

The last part of the reinforcement learning implementation in GNU Radio is the incorporation of the actual algorithm, e.g. Q-learning, through a custom block. The overview of this block is shown in Fig. 7.1. This figure shows the necessary inputs required for the algorithm to work properly as well as the outputs (e.g. actions) determined by Q-learning for the transmitter and receiver and a fixed random policy and/or sensing-based policy for the interferers. Refer back to Section 3.1.2 for the formulation of the RL used in this work and Chapter 2 for the explanation of Q-learning.

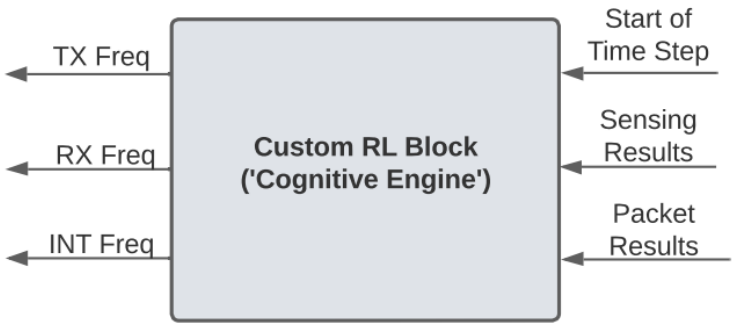


Figure 7.1: I/O Diagram of the Custom RL Block in GNU Radio

The required inputs are the sensing and packet results and a notification of the start of the next time step. These inputs are passed in the form of a message type in GNU Radio. Messages in GNU Radio are controlled through message handlers, which are functions set to receive the message input through GNU Radio's *pmt* library [54]. This is also how each of

the three input messages are controlled within the custom RL block, with each performing different functions, as described in Fig. 7.2.

Starting with the message handler for the sensing result, the sensing vector from the detector is stored in a variable, where a threshold based on the noise floor is applied to assign a 0 or 1 to unoccupied and occupied sub-channels, respectively. Additionally, the sensing message handler also controls the transmission of the next burst for the next time step. It does so by sending the next action determined by each user's policy to the transmitter's "Random PDU Generator" block, as described in Section 5.1. This next action tells this block that the cognitive engine is ready to process the next burst. However, it only does this within a set simulation time t. During this time, the policy of each user is enacted to determine their next action for the next time step.

For the transmitter and receiver, Q-learning is used. For the interferers, a fixed random or sensing-based policy is used to determine the next action. Note that the decision making for the interferers remains independent of Q-learning. The decision making for the interferers are performed in the same block as Q-learning so that internally, sensing and packet errors can be checked throughout the process for testing purposes. Sensing errors can occur when the detector has an incorrect threshold and/or when sensing results are delayed from latency. Packet errors can occur when the environment is too noisy, the Schmidle and Cox detection threshold is set too low, and/or from receiver latency. Threshold errors can easily be mitigated through trial and error, and in a noisy environment, the transmitter and receiver power can be increased to raise the signal from the noise floor. In the case of

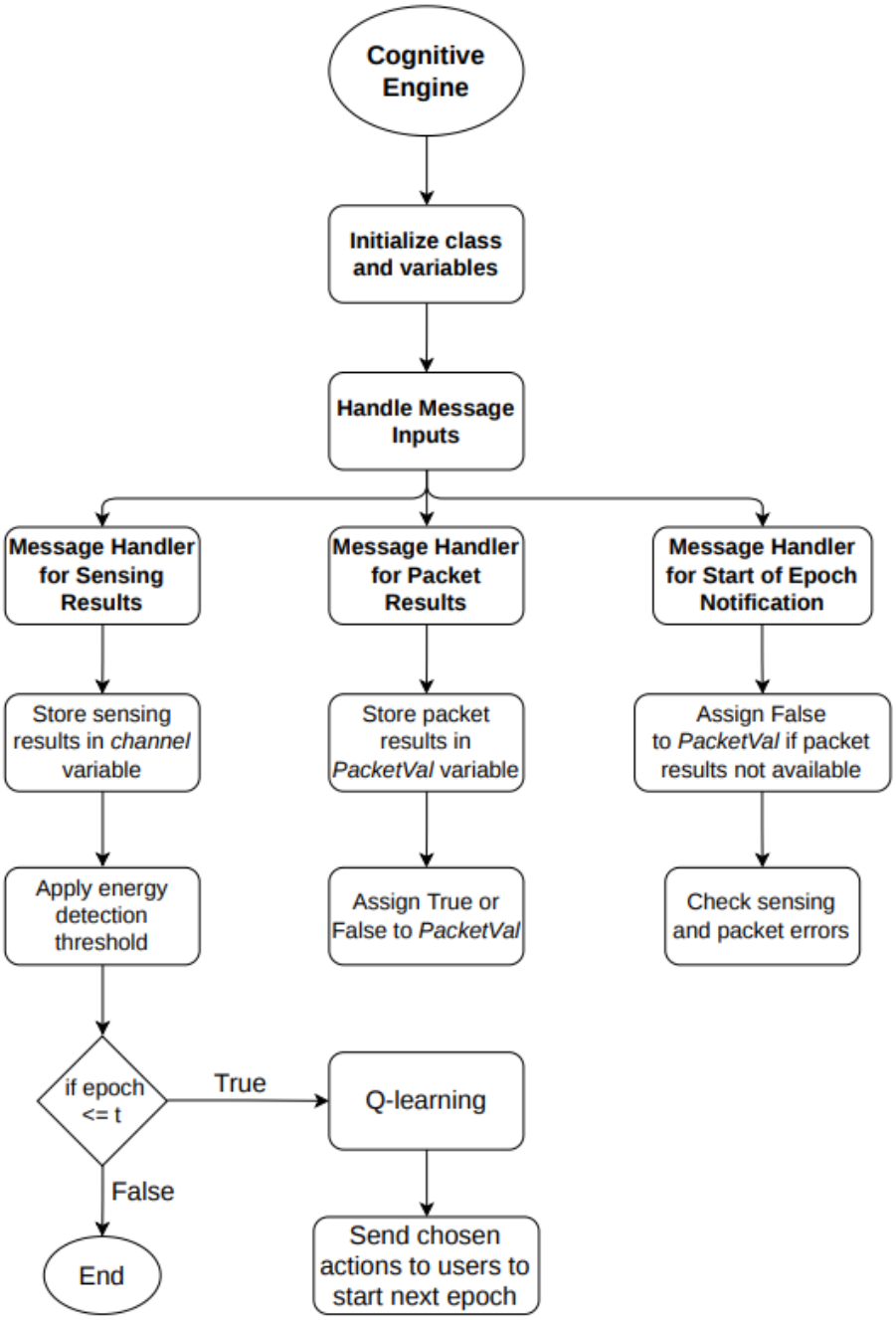


Figure 7.2: Function of the custom RL block, known as the "Cognitive Engine". Its main purposes are to handle the message inputs from the receiver and detector and to determine the actions for the next time step.

latency, deadtime should be increased to lower the number of errors. The minimum amount of deadtime required for no sensing and packet errors is investigated in Chapter 8.

For the message handler for the packet result, the packet header parser from the receiver, as described in Section 5.3, sends a *True* or *False*, which is stored in a variable. There is also a message handler to identify the beginning of the next time step to put a deadline on the packet results of each time step. This message is used to tell the RL block that the receiver should have its results available by the first item of the next time step. If not, which could occur if latency exists in the receiver and the deadtime is not large enough, a *False* is assigned to the packet variable. A *False* reading in this case is not due to a corrupted header, but instead it is due to a missed packet caused from system latency. How much deadtime is needed due to receiver latency is investigated in Chapter 8.

It is important to note that both the packet and sensing results must arrive to the reinforcement learning algorithm function within the same time step for accurate results. However, the asynchronous nature of messages in GNU Radio posed a challenge. When testing the algorithm, it was found that the sensing and packet results could belong to different time steps, which could ultimately affect the accuracy of the reinforcement learning algorithm since the state and reward inputs would not be from the current time step. This is in direct violation of the Markov property of a Markov Decision Process, which requires results from the current observation. Though the algorithm would still train, having this occur would be detrimental to the performance of the system over time because the policy would be outdated and behind the current state of the environment. The consequence would

be a higher probability of unsuccessful communication and an agent with unstable behavior. To mitigate this problem, deadtime and tagging was implemented, as described in Section 5.1.3. Additionally, by adding deadtime, latency caused by feedback can be measured and analyzed. This is further investigated in Chapter 8.

7.2 Verification of Q-learning

Parameters used for simulation are summarized in Table 7.1. Users involved are one OFDM user using IEEE 802.11a protocol and two interferers. The movement of the interferers is a repeated random pattern that is initialized at the beginning.

In this section, the performance of Q-learning is verified to validate the framework implemented in this chapter. Fig. 7.3 displays a spectrum waterfall of the radio frequency environment over a range of epochs for 8 sub-channels to visually demonstrate the performance. The OFDM user is the vertical signal (has bandwidth) spanning the entire channel. The interferers are the short horizontal tones, as shown in white on the left. Collisions occur when the interference overlaps with the RL agent, as highlighted by the red boxes on the left, in the pre-learned region. The left shows such a collision within the pre-learned region where the policy was not yet optimal, and the right displays the learned region, where the user has accurately predicted where the interferer will go next, as indicated by its hopping from 1 to 2.

For this thesis, Q-learning converges when the packet errors tend to 0 as learning con-

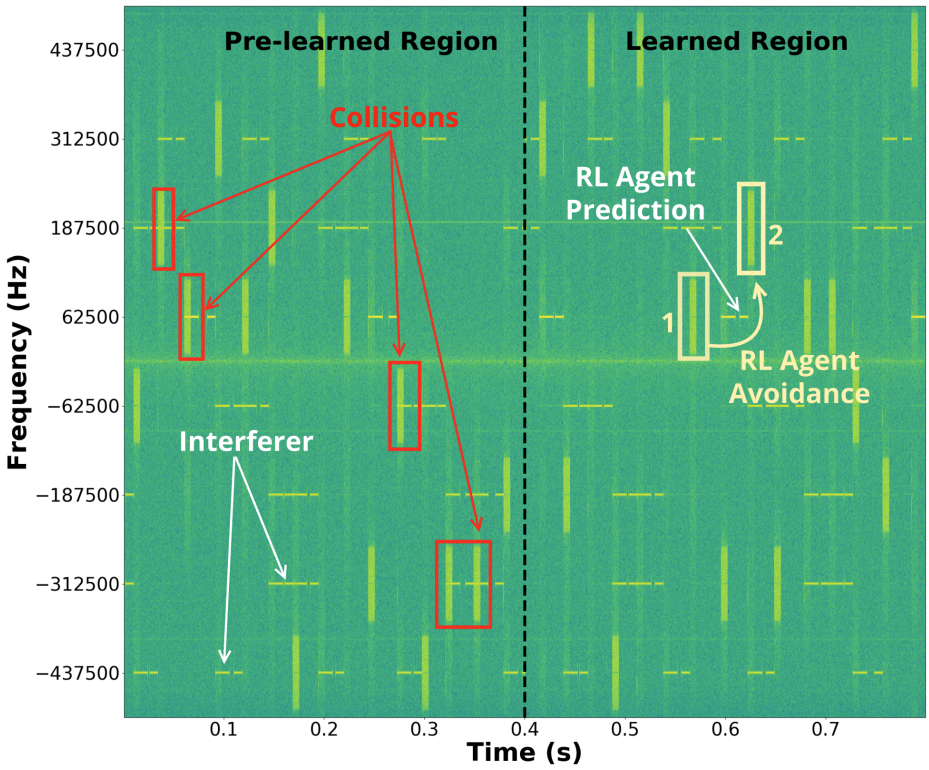


Figure 7.3: Spectrum waterfall showing activity of movement for the RL agent and interferers over 8 sub-channels within the RF environment.

tinues. This behavior is demonstrated in Fig. 7.4, where the RL agent learned to avoid the interferers as the number of epochs increased. By epoch 50, the number of packet errors had converged to 0, indicating that the RL agent had learned the pattern of movement of the interference within the RF environment.

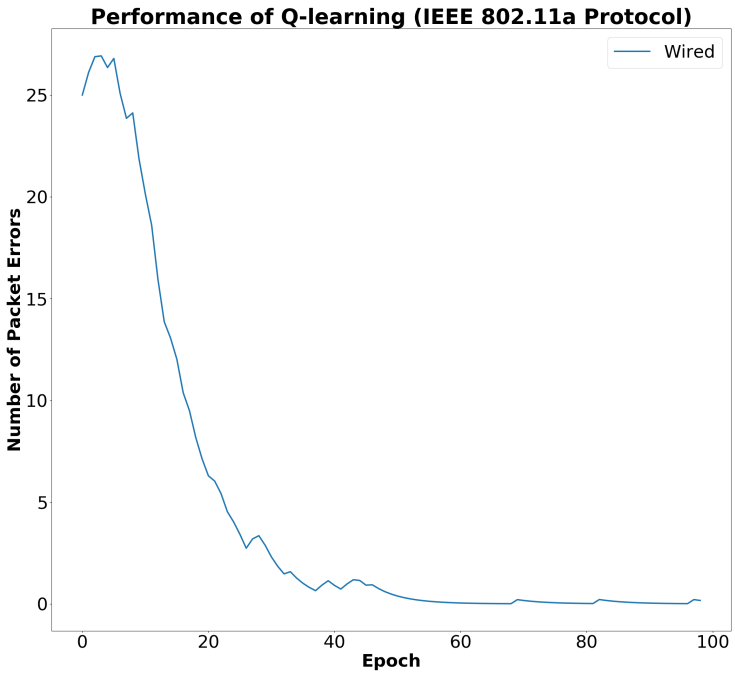


Figure 7.4: Simulation for 8 sub-channels of a transmitted signal with the IEEE 802.11a protocol standard demonstrating that Q-learning performs as expected in the framework outlined in this chapter.

Learning Rate	$\alpha = 0.9$
Discount Factor	$\gamma = 0.1$
Epsilon Decay Rate	0.1
Minimum Epsilon	$\epsilon_{min} = 0.001$
Maximum Epsilon	$\epsilon_{max} = 1$
Number of Channels	N = 8
Number of Symbols	10
Number of Header Symbols	1
Number of Data Symbols	7
Number of Data Samples	800
Duty Cycle	29%
Wideband Sample Rate	1 MHz
Signal Sample Rate	125 kHz
Number of Epochs	100
Time per Epoch	22.4 ms
Data Transmission Time	$\mathrm{T}_D = 6.4ms$
Sensing Time	$\mathrm{T}_{WBSS} = 6.4ms$
Deadtime	$\mathrm{T}_S + T_{RL} + T_{ACK} = 16ms$
Goodput	29 kbps

Table 7.1: Notable parameters that determine the generation of the transmitted signal and structure of reinforcement learning

Chapter 8

Analysis of the Performance of Reinforcement Learning in Real-World Implementations

This chapter analyzes the framework designed in Chapters 3 to 7 through simulation and over-the-air / over-the-wire testing. The purpose is to find any limitations that impair the RL learning process, and from these limitations, emphasize the considerations that should be included in the design process of an RF-based RL system. Any limitations are examined by their effects on the convergence of the RL algorithm, which occurs when the number of packet errors goes to 0 over time. This convergence is mostly affected by issues in the state provided by spectrum sensing and the reward provided by the receiver. Thus, different practical situations are tested to investigate their effects on the results from the spectrum sensing detector and the receiver. Situations of interest include system latency, the use of different spectrum sensing detectors, hardware memory limitations, and impairments of a

wireless channel.

8.1 Issues in the RL Process

The integrity of the RL process and its ability to converge to an optimal behavior depends on its state and reward. Here integrity refers to how much the state and reward appropriately represent the environment and agent performance. If the state and the reward properly capture the condition of the environment and the agent's performance, then RL has a higher probability of performing as expected. However, if the state and reward do not give valuable feedback, then RL often produces a sub-optimal behavior or does not work at all. This can happen in a real system with RL because of the imperfections that naturally occur. These imperfections cause limitations in the system that must be mitigated in order for RL to function properly. Though it is often difficult to understand where, when, and how these limitations occur, what remains constant is how RL can fail - through invalid state and reward feedback. Therefore, by examining what causes errors in the state and reward, limitations that affect the convergence of RL can be found. This analysis is important to the design of RFRL systems because these limitations are not considered in simulation-only investigations of RL for communications. Thus, by examining them, they can be mitigated before real-world deployment.

8.1.1 State Errors

The state is found through spectrum sensing. Throughout implementation and testing, the following possible sensing errors were observed. Note that the detector captures signal presence within the RF environment. Its sensing result contains an array of 0s and 1s, where 0 indicates an empty sub-channel and 1 an occupied sub-channel.

- **Detection threshold is too high.** When the threshold is too high, the detector fails to detect the presence of users. For example, if $N = 4$ and the users were assigned to f_0, f_2, and f_3, the true state is $[1, 0, 1, 1]$, but the detector sent $[0, 0, 0, 0]$. The threshold value can be adjusted and should be checked before RL training starts. However, note that finding an appropriate threshold is often difficult; thus, this kind of error can occur easily if not set carefully.

- **Detection threshold is too low.** When the threshold is too low, the detector captures false detections caused by spikes in noise, resulting in it reporting the presence of "signals" that are actually noise. For example, if $N = 4$ and the users were assigned to f_0, f_2, and f_3, the true state is $[1, 0, 1, 1]$, but the detector sent $[1, 1, 1, 1]$. Again, the threshold value can be adjusted and should be checked before RL training starts.

- **Computation time of the sensor.** Latency in the system could cause an asynchronous nature to sensing. For instance, if the transmitted signal is transmitting too fast for the detector, because the detector operates at a fixed rate, there is system latency caused by the detector. Thus, a delayed state observation can occur from

processing the signal too slowly, where the detector gives results for time t at time $t+1, t+2, t+3, ...$, depending on how large the latency is. If the detector is too fast and system latency is caused by another component, then a similar result could happen, where the sensing result fails to detect the presence of signals due to processing the signal too quickly. Determining this latency is part of the contributions of this thesis and is mitigated through a duty cycle to ensure state and reward results are current to the current time step. Thus, a lower duty cycle (more deadtime) can be implemented to mitigate these issues.

8.1.2 Reward Errors

Another source of error in the RL process is an incorrect reward. A reward in a given epoch can be incorrect for the following reasons.

- **Improper thresholding of the synch word detector.** If the probability of detection P_d threshold is too low in the "Schmidl and Cox OFDM Synch" block, the number of false detections due to a higher noise floor and/or the presence of interferers increases. Thus, this threshold value must be adjusted depending on the noise environment. Typically, a value greater than 0.9 is sufficient. Additionally, these false detections could be causing latency at the receiver. If the threshold is too low, the message queue for the packet header parser in GNU Radio gets overloaded with many false detection messages before it sees a $True$. This could also cause the RL block to never receive feedback from the receiver altogether, particularly in high noise scenarios.

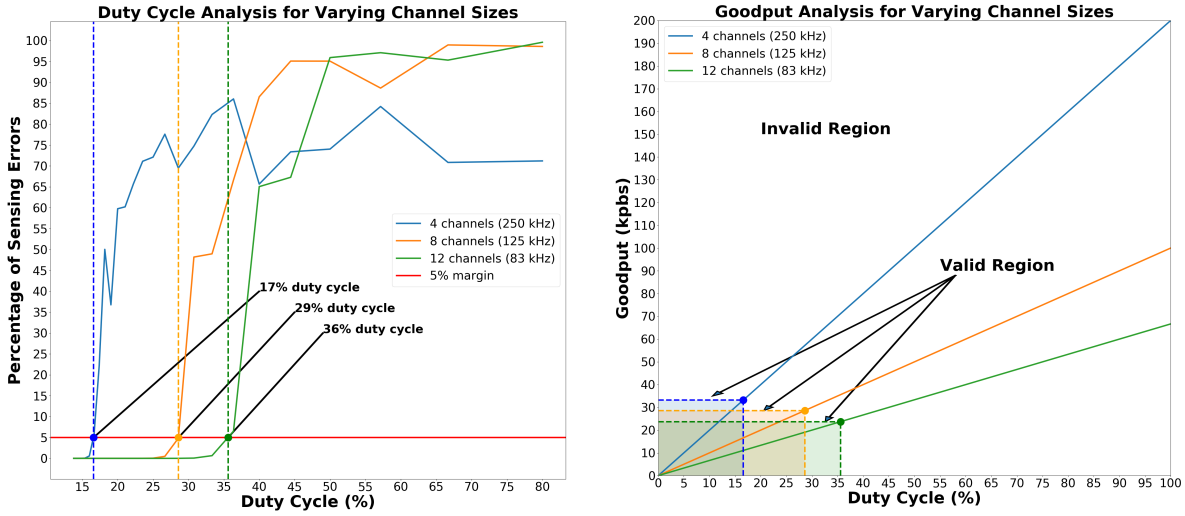
If the threshold is raised, then the message queue has less messages to parse, and the *True* value can be sent to the RL block quicker.

- **Slow processing time of the receive chain.** If the receiver is still processing while the rest of the system is done, then it is possible for the reward for time t to be given to the custom RL block at time $t+1, t+2, t+3$, etc. In this case, deadtime can be added.

8.2 Computational and Latency Effects

In this section, the effects of latency are analyzed through the implementation of a duty cycle around the OFDM burst, as shown in Fig. 5.7. Through this duty cycle, a lower bound on the time required for valid state and reward feedback to reach the custom RL block can be found. More specifically, this section measures the amount of time necessary for proper spectrum sensing, receiver processing, and any computation time, such as for the RL algorithm. To measure this latency, the duty cycle was varied until the percentage of sensing errors became less than a fixed allowable error margin, in this case 5%. This 5% error margin was observed to still produce convergence of the RL algorithm. To understand if a sensing error occurred in a given epoch, the sensing results are checked for correctness based on the actions the OFDM user and the interferers were supposed to take as decided in the previous epoch. Note that all testing for this section occurred over USRPs, and the spectrum sensing detector utilized was the PSD-based detector.

8.2.1 Measuring and Mitigating Latency



(a) Trade-space predicting regions of RL accuracy for varying duty cycles and sub-channel sizes

(b) Goodput trade-space showing the range of effective data rates given a maximum allowable duty cycle for varying channel sizes

Figure 8.1: Trade-space formulation predicting regions of RL accuracy due to latency mitigation from a duty cycle implementation for varying channel sizes N. Note that each N corresponds to a different signal sampling rate. For example, a signal operating in a $N = 4$ sub-channel configuration with 1 MHz available bandwidth operates at 250 kHz (IEEE 802.11a protocol).

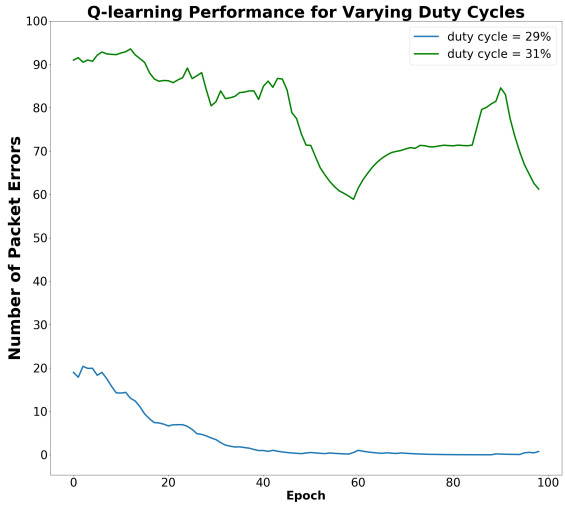
Fixed at a wideband sampling rate of 1 MHz formatted with the IEEE 802.11a-like protocol, sub-channel numbers of $N = 4, N = 8$, and $N = 12$ were tested to examine how the framework was effected by varying signal rates, as determined by N, and how much deadtime (e.g. duty cycle) was required for the detector to produce non-delayed state results based on the signal's rate. These results are shown in Fig. 8.1. Fig. 8.1a demonstrates the trade-space

Number of Channels N	Signal Sample Rate	Theoretical Goodput	Maximum Goodput	Goodput Ratio
4	250 kHz	200 kbps	33 kpbs	0.17
8	125 kHz	100 kbps	29 kbps	0.29
12	83 kHz	67 kbps	24 kbps	0.36

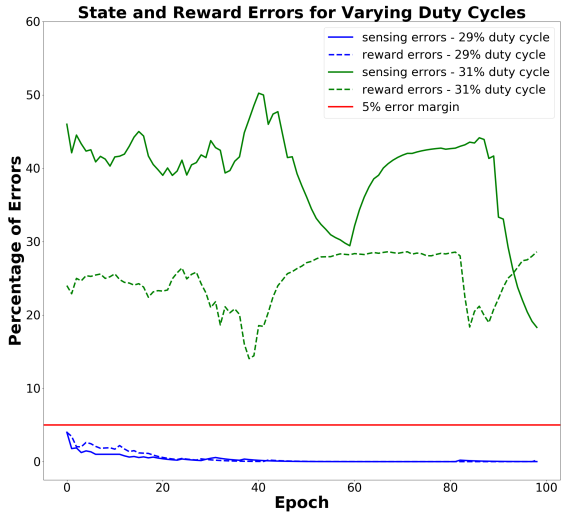
Table 8.1: Goodput analysis for varying sampling rates, comparing theoretical with experimental (IEEE 802.11a protocol)

that was formulated as a result of testing each N, with its respective signal sampling rate, across a range of duty cycles. The maximum allowed duty cycle to ensure accurate RL was found by determining what duty cycle resulted in a sensing error percentage occurring below a 5% error margin. Fig. 8.1b shows the effective range of data rates (goodput), as calculated in Equation 4.3, that are possible for the system given the maximum duty cycle found for each N. For $N = 4$, with a signal sampling rate of 250 kHz, the upper limit on the duty cycle was found to be 17%. This corresponds to a maximum data rate of 33 kpbs, given a theoretical maximum of 200 kpbs. For $N = 8$, with a signal sampling rate of 125 kHz, the upper limit on the duty cycle was found to be 29%. This corresponds to a maximum data rate of 29 kpbs, given a theoretical maximum of 100 kpbs. For $N = 12$, with a signal sampling rate of approximately 83 kHz, the upper limit on the duty cycle was found to be 36%. This corresponds to a maximum data rate of 24 kpbs, given a theoretical maximum of 67 kpbs. These results are summarized in Table 8.1.

To validate these results, refer to Fig. 8.2, which shows the performance for two different duty cycles of an 8 sub-channel configuration. Of the two duty cycles, one causes inaccurate RL ($> 29\%$) and the other accurate RL ($\leq 29\%$). Fig. 8.2a displays the Q-learning



(a) Results showing that operating at a duty cycle higher than the allowed maximum (29%) results in undesirable Q-learning performance



(b) Percentage of state and reward errors over 100 epochs, showing higher percentages for duty cycle higher than the allowed maximum

Figure 8.2: Demonstrating effect of duty cycle to RL performance due to system latency

performance, where theoretically, the number of packet errors should go to 0 at the end of learning. For a duty cycle of 29%, this convergence behavior was observed. However, for a duty cycle of 31%, which is outside the 5% error margin region, the Q-learning performance proved to be very unstable and fails to converge. In Fig. 8.2b, it was verified that this unstable behavior for a duty cycle of 31% was a consequence of sensing errors that remained in the 20% to 55% sensing error range and reward errors within the 25% to 30% reward error range, likely due to latency within the system. However, for a duty cycle of 29%, the percentage of sensing and reward errors remained below the 5% margin, verifying the duty

cycle limitation due to latency found in Fig. 8.1a.

From these results, for accurate RL, it was found that the system requires lower duty cycles, and thus lower goodput, for higher-speed processing, as indicated in Fig. 8.1a by the 20% decrease in duty cycle required for 250 kHz versus 83 kHz. Intuitively, this makes sense because the samples are being processed faster. Therefore, in the design and implementation of RFRL systems, the sampling rates of the signal and processing speed of individual components should be considered. If an engineer is more concerned about goodput efficiency, perhaps a smaller sampling rate should be used such that the accuracy of RL is maintained. Overall, the trade-space shown in this section should give engineers an example of how to formulate a trade-off analysis between a parameter of their choice and RL accuracy so that the operating regions of the chosen parameters can be set appropriately to achieve and maintain accurate RL. The rest of this section demonstrates how using the maximum operating duty cycle results in convergent RL performance, effectively successfully mitigating system latency for each of the sub-channel configurations.

To demonstrate what results from adhering to the maximum allowed duty cycle, refer to Fig. 8.3. Fig. 8.3 shows, for a different number of available channels, the number of packet errors that occurred throughout the learning process. As shown, all sub-channels, each with their respective duty cycles of 17%, 29%, and 36%, were able to converge to approximately 0 packet errors, verifying that the implemented duty cycle does aid in mitigating existing system latency and produces RL convergence behavior that matches the theoretical results discussed in simulation-based prior art, where hardware effects are not considered. Here,

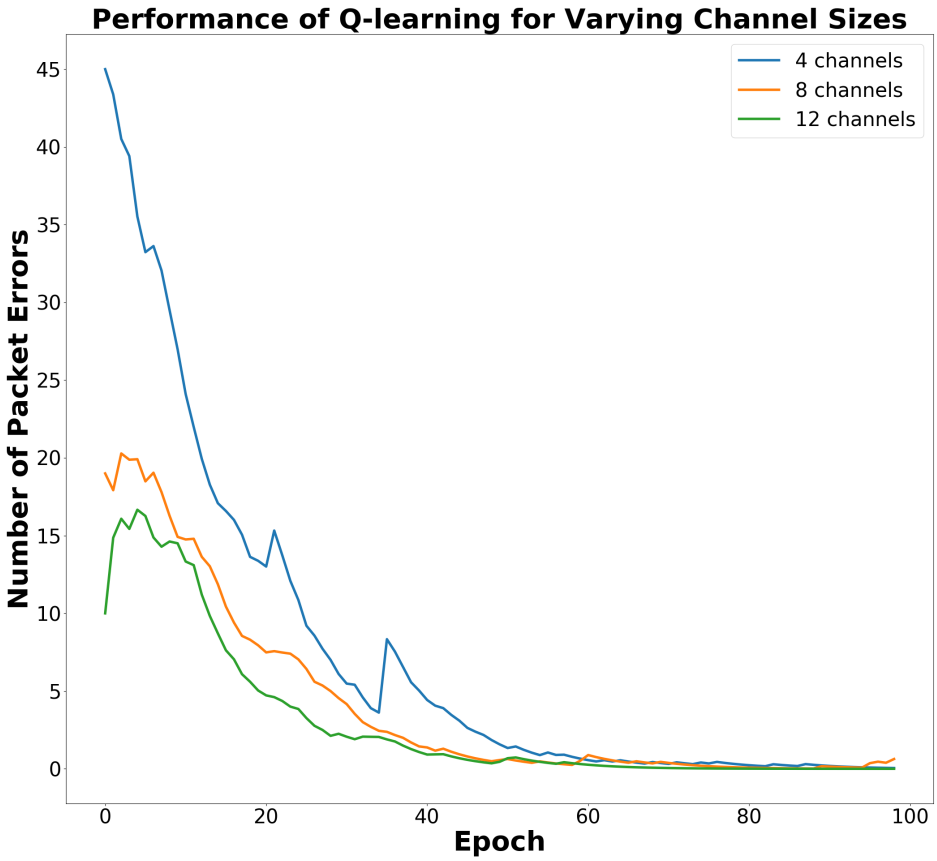


Figure 8.3: Q-learning performance for varying channel numbers, displaying the number of packet errors decreasing over epoch time

because latency is mitigated by the duty cycle, the packet errors at the beginning are due to the interference pattern and the random policy followed at the very beginning of the RL learning process.

In Fig. 8.3, notice that each sub-channel N converged to 0 at different rates. For 4 sub-channels, it starts at a much higher number of packet errors. This makes sense since there are three total users within the environment, and thus, more opportunity to collide with another. As the number of sub-channels increase with a fixed number of users, the

number of packet errors begin to decrease at the beginning. This also makes sense because the probability of colliding with another user randomly is much lower. For 8 sub-channels, this results in a 56% decrease in collision probability compared to 4 sub-channels, and for 12 sub-channels, that percentage increases to 67%. Therefore, from this logic, because it started with a higher number of packet errors, 4 sub-channels should have a faster learning curve. This trend is shown in Fig. 8.3. Notice before epoch 60, the slope for 4 sub-channels is much steeper than the other two curves. Furthermore, by comparing to the slope for 4 sub-channels, the slope for 8 sub-channels decreases by 44% and 56% for 12, which verifies the above assumption. Therefore, trends and convergent behavior shown in Fig. 8.3 verify the theoretical operation of Q-learning, implemented with the maximum allowed duty cycle, within the designed system.

8.2.2 Comparison of Different Physical Waveforms based on Existing Protocols

In this section, the performance between the IEEE 802.11a-like and 5G NR-like protocols are compared. Note that in the case of 5G NR, the signal sampling rate remains at a constant 960 kHz but the wideband sampling rate varies based on N. This is opposite the case of IEEE 802.11a, where the wideband sampling rate was a constant 1 MHz but the signal sampling rate varied based on N. The reasoning being that the signal sampling rate must be kept at a constant 960 kHz to maintain numerology 0's subcarrier spacing of 15 kHz, as discussed in Section 4.1.2.

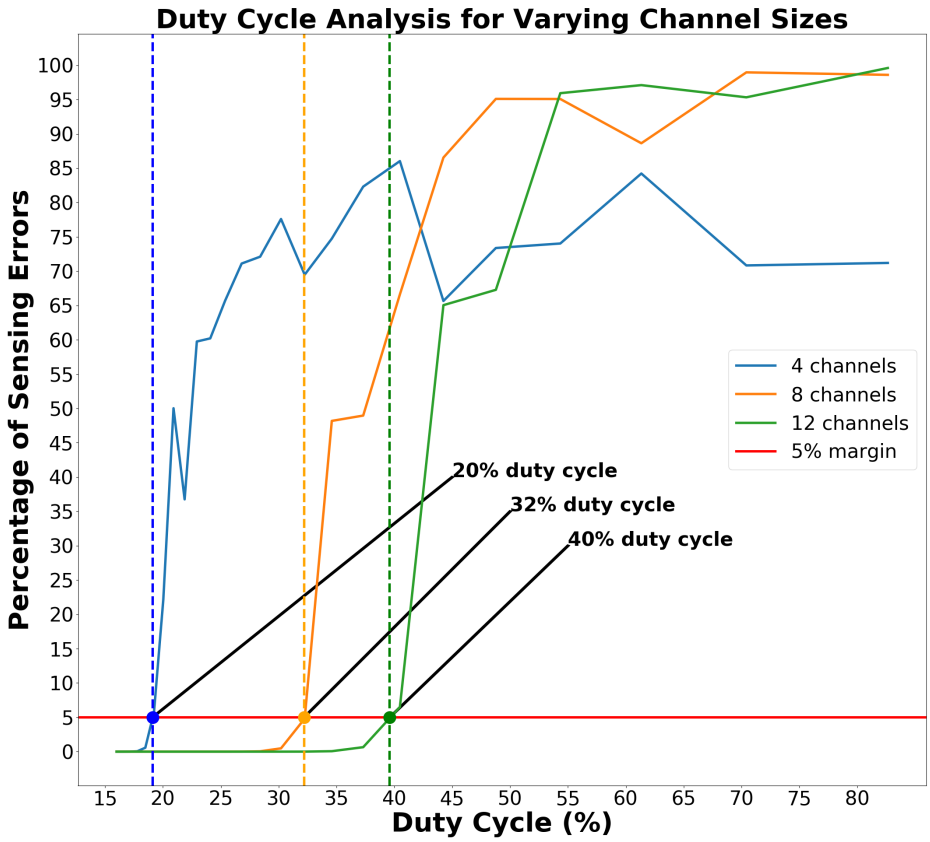


Figure 8.4: Trade-space formulation predicting regions of RL accuracy due to latency mitigation from a duty cycle implementation for varying channel sizes N for the 5G NR protocol. Note that each N corresponds to a different wideband sampling rate. For example, a signal with a 960 kHz sampling rate operating in a $N = 4$ sub-channel has a wideband sampling rate of 3.84 MHz

The trade-space for 5G NR between duty cycle, goodput, and latency is shown in Fig. 8.4, where a maximum duty cycle of 20%, 32%, and 40% were found for $N = 4, 8$, and 12 sub-channels, respectively. This is a 17% increase in duty cycle for all sub-channels. However, this increase in duty cycle is due to the increase in the number of payload symbols, which is 11 for 5G NR compared to 7 for IEEE 802.11a. Therefore, it was observed that

the amount of deadtime used remained constant, indicating that the latency is constant across sub-channel configurations for both protocols. Meaning, latency is constant across protocol configurations when the wideband sampling rate and signal sampling rate differ by a factor of N. In other words, for each sub-channel configuration N, the same deadtime settings can be used across protocols to mitigate latency. In the design and implementation of RFRL systems, this is helpful because the same analysis does not have to be repeated across protocols. Instead, an engineer can test the latency for one implementation and apply the results to other protocols. Based on the results, this is because the system latency is seemingly dependent on the signal processing speed and/or processing speed of other components, rather than the frame protocol structure.

Here, the Q-learning curves for $N = 4$ and $N = 8$ sub-channels is compared for the IEEE 802.11a and 5G NR protocols. Note that $N = 12$ was not performed for 5G NR because of USRP limitations in sampling rate. The $N = 12$ configuration would have required a 11.52 MHz wideband sampling rate. However, the networking of the USRP to the computer cannot operate above 10 MHz. The parameters used for both protocols are summarized in Table 8.2. Notice that the time for deadtime for the IEEE 802.11a-like protocol remains relatively constant across sub-channel configurations. However, this is not the case for 5G NR, where the time for deadtime is 2.1 ms and 4.2 ms for $N = 4, 8$, respectively. This is because, for 5G NR for each N, the wideband sampling rate was changed rather than the signal sampling rate. However, as explained above, the number of samples used for deadtime, and thus the processing time, for N across protocols remained the same. For instance, the number of

Parameters	**IEEE 802.11a**	**5G NR**
Learning Rate	$\alpha = 0.9$	$\alpha = 0.9$
Discount Factor	$\gamma = 0.1$	$\gamma = 0.1$
Epsilon Decay Rate	0.1	0.1
Minimum Epsilon	$\epsilon_{min} = 0.001$	$\epsilon_{min} = 0.001$
Maximum Epsilon	$\epsilon_{max} = 1$	$\epsilon_{max} = 1$
Number of Sub-channels	N = 4, 8	N = 4, 8
Number of Symbols	10	14
Number of Header Symbols	1	1
Number of Data Symbols	7	11
Number of Data Samples	800	952
Duty Cycle	17%, 29%	20%, 32%
Wideband Sample Rate	1 MHz	3.84 MHz, 7.68 MHz
Signal Sample Rate	250 kHz, 125 kHz	960 kHz
Number of Epochs	100	100
Time per Epoch	19.2 ms, 22.4 ms	3.1 ms, 5.2 ms
Data Transmission Time	$\mathrm{T}_D = 6.4ms$	$\mathrm{T}_D = 1ms$
Sensing Time	$\mathrm{T}_{WBSS} = 6.4ms$	$\mathrm{T}_{WBSS} = 1ms$
Deadtime	$\mathrm{T}_S + T_{RL} + T_{ACK} = 16ms, 16ms$	$\mathrm{T}_S + T_{RL} + T_{ACK} = 2.1ms, 4.2ms$
Goodput	33 kbps, 29 kbps	156 kbps, 250 kbps

Table 8.2: Notable parameters that determine the generation of the transmitted signal and structure of reinforcement learning for the IEEE 802.11a and 5G NR protocols

deadtime samples for $N = 4$ was the same for both protocols, and likewise, the number of deadtime samples used for $N = 12$ was the same as well. Therefore, it can be concluded that the number of samples used to mitigate latency is constant across protocol configurations, as long as the wideband sampling rate and signal sampling rate differ by the same factor. For example, for IEEE 802.11a for $N = 8$, the wideband sampling rate divided by the signal sampling rate was 1MHz/125kHz = 8, which matches $N = 8$. The same is true for $N = 8$

for 5G NR, where 7.68MHz/960kHz = 8. However, for a constant signal sampling rate, the actual time in ms of latency increases as the wideband sampling rate increases. This increase has no effect on the system because measured latency (in ms) is dependent on the number of deadtime samples. Though it increases with the wideband sampling rate, the number of deadtime samples is constant. In other words, the latency in ms is scaled by the same increase in the wideband sampling rate, resulting in a higher latency time (in ms) but a constant number of samples for deadtime. For a constant wideband sampling rate, as is the case for the IEEE 802.11a-like case, the latency in ms remains relatively constant. However, when the signal sampling rate is increased by its increasing sub-channel configuration N, as shown in the trade-space analysis, more samples of deadtime are required to account for latency. Thus, in a system with a fixed rate, when the signal sampling rate increases, latency (in ms) decreases if the deadtime is not adjusted. Decreasing latency would normally be desired, but if the deadtime is not adjusted to account for latency according to the new rate, there could be issues where there is not enough deadtime to properly perform RL. Therefore, it can be concluded that latency is caused by the speed of the individual components. The latency of each component should be considered such that it does not operate too fast for what the system can handle. In other words, when designing and implementing an RFRL system, the latency depends on the wideband-to-signal sampling rate ratio, which is N. Thus, the number of samples used for deadtime will change if the wideband-to-signal sampling rate ratio is changed. The smaller the ratio, which operates under a faster signal sampling rate, the greater the latency (in number of samples). Similarly, the larger the ratio, which operates

under a slower signal sampling rate, the smaller the latency (in number of samples).

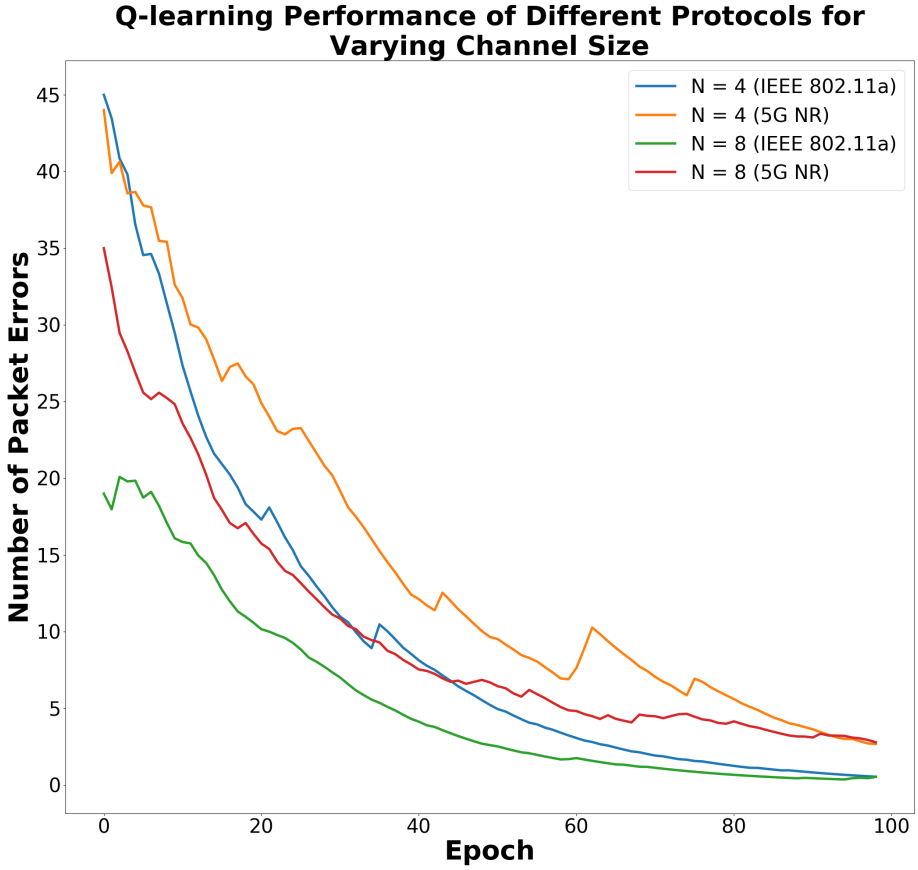


Figure 8.5: Comparing the Q-learning performance for varying channel numbers for different protocols, displaying the number of packet errors decreasing over epoch time

Fig. 8.5 compares the Q-learning convergence curves for the IEEE 802.11a-like and 5G NR-like protocols. The results show that the 5G NR case performs worse than the IEEE 802.11a-like protocol. In fact, unlike the IEEE 802.11a case, one to two sub-channels of the 5G NR-case spanned the WiFi band of 2.4 GHz, which has very active activity,. Also, because the spectrum sensing detector is at the transmitter, the signal activity over the WiFi

band was not represented in the environment. meaning the receiver sees the WiFi band, as indicated by more false packets, but not at the transmitter. This could lead to state and reward errors in the RL learning process. Future work would be to test both protocols over a quiet industrial, scientific, and medical-purposed (ISM) band to fully investigate whether the worse 5G NR performance was due to WiFi-band activity.

Despite a slightly worse performance, the trends between the two protocols were the same. The 4 sub-channel configuration still started with more packet errors at the beginning than the 8 sub-channel case, a direct consequence of the RL agent having less options in the $N = 4$ case. Additionally, the 4 sub-channel configuration still learned and converged at a similar, faster rate, as indicated by the steep slopes, than the 8 sub-channel configuration. Therefore, the similarity in Q-learning behavior indicates that the differences in protocol does not affect its learning behavior.

8.2.3 Comparison of Different Spectrum Sensing Detectors

In this section, the effect different spectrum sensing detectors has on RL is investigated. Here, the setup described in Section 6.1.1 is used for the polyphase channelizer detector. The main focus was to investigate the impact of sensing algorithm complexity, which can be measured by how long it takes to execute it, e.g. processing time. In order to measure the difference in processing time between the two detectors, both detectors operated during training, and both were processed by the custom RL block. However, the timing of the next epoch was determined by the slower of the two detectors. Because the RL block gets sensing

messages from both detectors, it knows which one is slower in each epoch.

For sub-channel configurations of N = 4, 8, and 12 for each epoch, the difference in processing time was found by subtracting the time recorded for each detector, as determined by the cognitive engine. The result is displayed in Fig. 8.6, which shows the difference in processing time over the duration of 100 epochs. It was found that for each sub-channel configuration, the PSD detector is slower than the channelizer detector, by on average approximately 11 ms, 14 ms, and 17 ms for N = 4, 8, and 12 sub-channels, respectively.

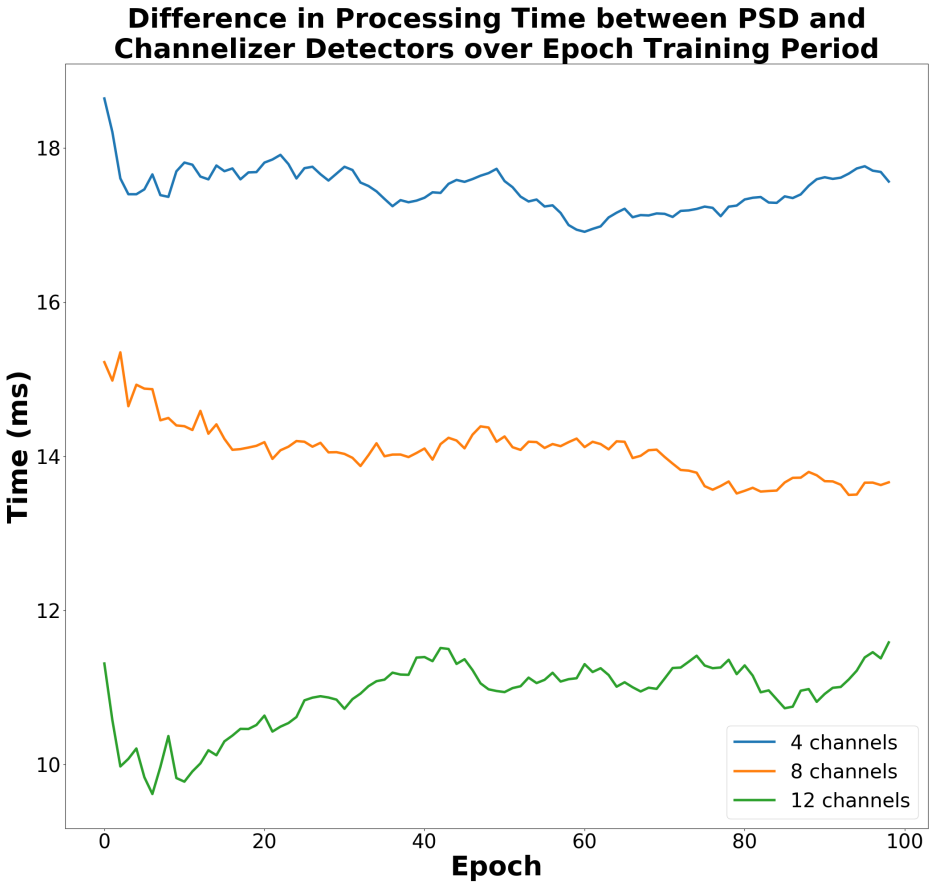


Figure 8.6: Comparing the processing time of the polyphase channelizer detector versus the PSD-based detector over epoch time

The effect of the signal sampling rate discovered in Section 8.2 was found to have an effect on the detectors as well. For $N = 4$, where the signal operates at 250 kHz, the average amount of processing time between the two detectors is 17 ms. However, for $N = 12$, which operates at 83 kHz, this processing time difference decreases approximately 35% to 11 ms. Thus, the PSD-based detector processes less efficiently at higher sampling rates, indicating that from a processing perspective, the channelizer detector is a better choice to decrease latency in the system. However, as discussed in Section 8.2, this effect is actually the opposite. In Section 8.2, it was found that a lower duty cycle was required to properly mitigate latency for higher sampling rates, leading to the conclusion that processing rates of the framework are set by the wideband sampling rate and not the signal sampling rate. Thus, although higher signal sampling rates have faster processing times (e.g. transmits faster), other processes within the system that are fixed to the entire framework's rate cannot process any faster, causing latency. This can lead to a situation where multiple signals can be transmitted before the RL block has time to process it.

8.2.4 Effects of Memory Limitations

Another limitation that can pose a problem to real-time communication systems with RL is the memory required to represent a state and action space. For example, in the simulations presented in this work, for N sub-channels, the memory requirements are

$$M = N2^N. \tag{8.1}$$

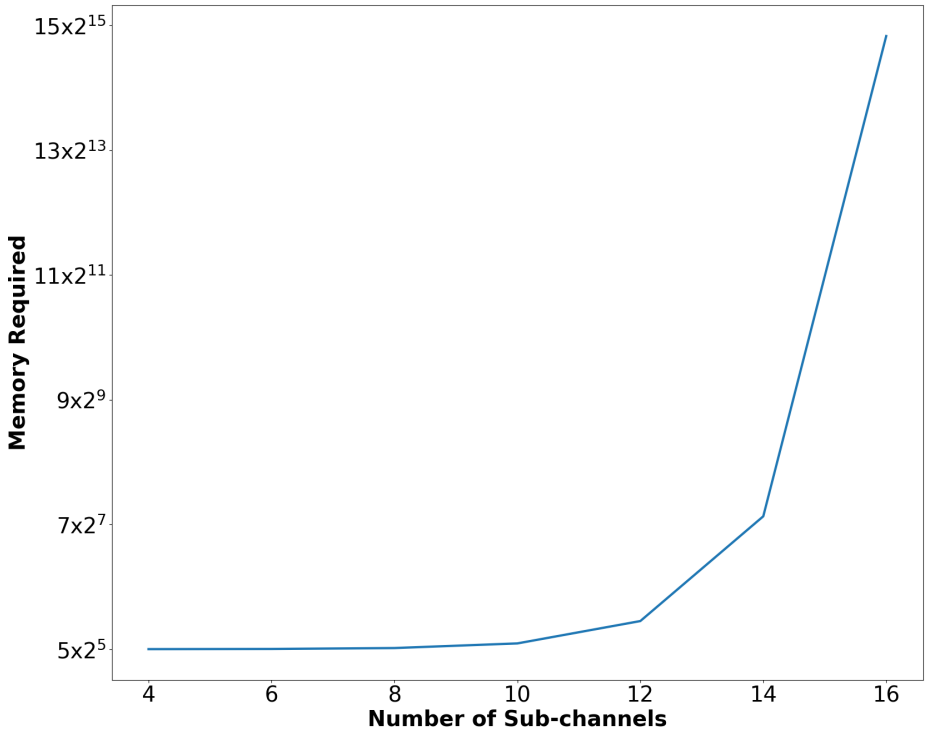


Figure 8.7: Memory analysis for state space

From this expression, it is clear that the state space can explode very quickly as N increases, as shown in Fig. 8.7 (e.g. from 12 to 16 sub-channels). This issue is particularly present in discrete, tabular RL methods, such as Q-learning. For example, when testing the RL framework in this thesis, GNU Radio refused to accept more than 16 sub-channels without giving a memory error, limiting the system to $N \leq 16$. Additionally, as the state and action spaces increase, the amount of time required to optimize the policy explodes as well. In the case of Q-learning, though among the simplest RL algorithms to implement, the values in the Q-table table for each state-action pair cannot be updated well enough without appropriate training time because each state-action pair are not visited enough. When state-action pairs

are neglected during the training process as a consequence of small training times, after an "optimal" policy has been reached, the system will not know what to do with that state-action pair once it sees it because it has not seen it enough during training to learn how to handle it. Therefore, as memory increases, training time must also increase. However, real world RF systems must be able to train and react quickly, which cannot be achieved with high memory requirements, and thus large state and action spaces. Therefore, an RFRL system, if demanding large state and action spaces, is better off with a function approximation-based RL algorithm, where function approximators are used to estimate Q through a set of features. One such way is to apply deep reinforcement learning, which uses a neural network to estimate Q. However, because deep RL is more computational complex than Q-learning and requires high accuracy, it is important to understand how long a real system would need to process that algorithm and prevent feedback latency. Thus, a trade-space analysis for deep RL in a similar context would be an interesting application for future work.

8.3 Real-World Propagation Effects

In this section, different effects were tested over the wireless channel. First, tested OTA using USRPs, the power and distance between the transmitter (TX) and receiver (RX) were varied to see how power and distance affect RL performance. Second, the effect of Doppler shifts was simulated in GNU Radio (not OTA) by varying different velocities to see what happens to the performance of RL under mobile conditions. Note that the same settings listed in Table 7.1 were used to test these configurations, and the PSD-based detector was

used for spectrum sensing.

8.3.1 Effect of Power and TX/RX Separation

In this section, the power was varied to understand how power of the signal, as the TX/RX separation increases, affects the performance of RL. The distances used are shown in Fig. A.1 in Appendix A.1. A smaller separation is characterized by a 6.5 inch TX/RX separation. A larger separation is characterized by a 65 inch TX/RX separation, which is 10 times bigger than the small configuration. The power settings include a lower and higher power level. How these were chosen is explained and displayed in Appendix A.2. Note that the power, for both lower and higher settings, were set to be approximately the same SNR for all distance configurations.

First, the signal-to-noise ratio (SNR) is obtained by comparing the power level of the noise floor P_n to the power level of the signal plus noise P_{s+n}. This was accomplished by plotting, in MATLAB, the PSD of the received time-version signal captured in GNU Radio by a file sink. Because the entire signal is padded with deadtime, power of the noise floor was captured by removing the OFDM-part of the signal, such that only periods of deadtime remained. For the power of the OFDM signal, the deadtime was removed, such that only the OFDM signal remained. Then, after plotting, the SNR was calculated using Equation 8.2 below.

$$SNR = \frac{P_{s+n} - P_n}{P_n} = \frac{P_s}{P_n} \tag{8.2}$$

This SNR gives insight on how much the received signal is affected by noise. It is of interest to see how this noise affects the performance of RL. Thus, the effect of power (and thus noise) on the system is analyzed by comparing the Q-learning performance resulting from a lower power setting versus a higher power setting for each TX/RX separation distance.

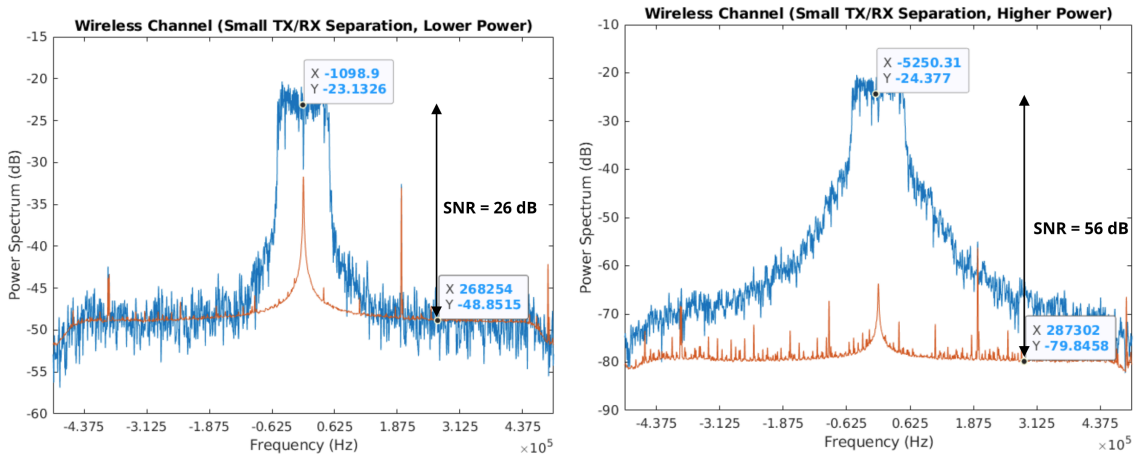
8.3.1.1 Smaller TX/RX separation distance

For the smaller TX/RX separation configuration, a distance of 6.5 inches was used. The PSDs of the received signal for lower and higher power configurations are shown in Fig. 8.8. The orange plot represents the PSD of the noise floor, and the blue plot represents the PSD of the received OFDM signal. Notice that at 0 Hz, the noise floor spikes closer to the OFDM signal. This is due to leakage from the hardware's local oscillator (LO) and is normal to see in radio hardware. Additionally, other spikes in the noise floor are characterized by signal activity in the band, which is normal to see as operation occurred around the WiFi band.

Fig. 8.8a shows the PSD for a lower power configuration. The power of the noise floor is approximately -49 dB, and the power of the OFDM signal is approximately -23 dB. Thus, the SNR of the lower power small separation configuration is

$$SNR = -23 \text{ dB} - (-49 \text{ dB}) = 26 \text{ dB} \tag{8.3}$$

Fig. 8.8b shows the PSD for a higher power configuration. The power of the noise floor is approximately -80 dB, and the power of the OFDM signal is approximately -24 dB. Thus,



(a) PSD of OFDM signal and noise floor for lower power

(b) PSD of OFDM signal and noise floor for higher power

Figure 8.8: PSD of OFDM signal and noise floor for smaller TX/RX separation, with the SNR annotated

the SNR of the higher power small separation configuration is

$$SNR = -24 \text{ dB} - (-80 \text{ dB}) = 56 \text{ dB} \tag{8.4}$$

which is 30 dB higher than the lower power configuration. Therefore, the RL performance was evaluated for each SNR with an $N = 8$ sub-channel configuration to see how noise affects training and convergence behavior for a small TX/RX separation setting.

The Q-learning performance for both power configurations is shown in Fig. 8.9. For convergence, the number of packet errors should go to 0 as the number of epochs increase. This behavior for both lower and higher power configurations is observed in Fig. 8.9. Fig. 8.9 demonstrates that the convergence behavior for the lower and higher power curves are nearly the same. This indicates that both the lower and higher power cases are not as

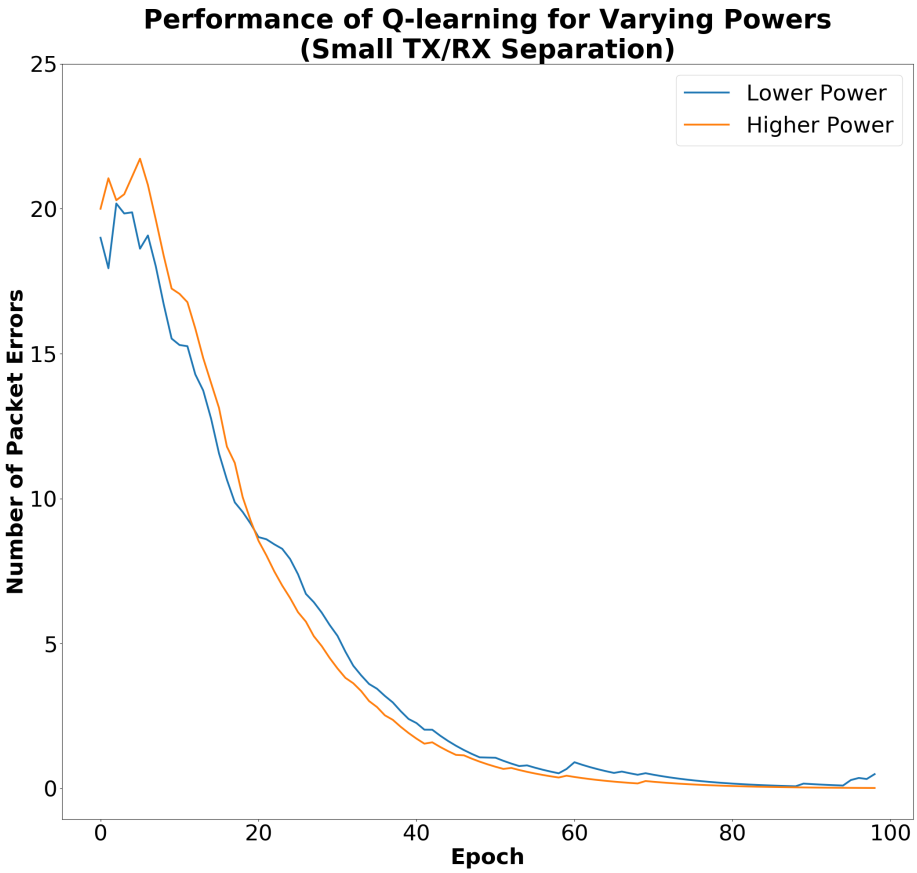
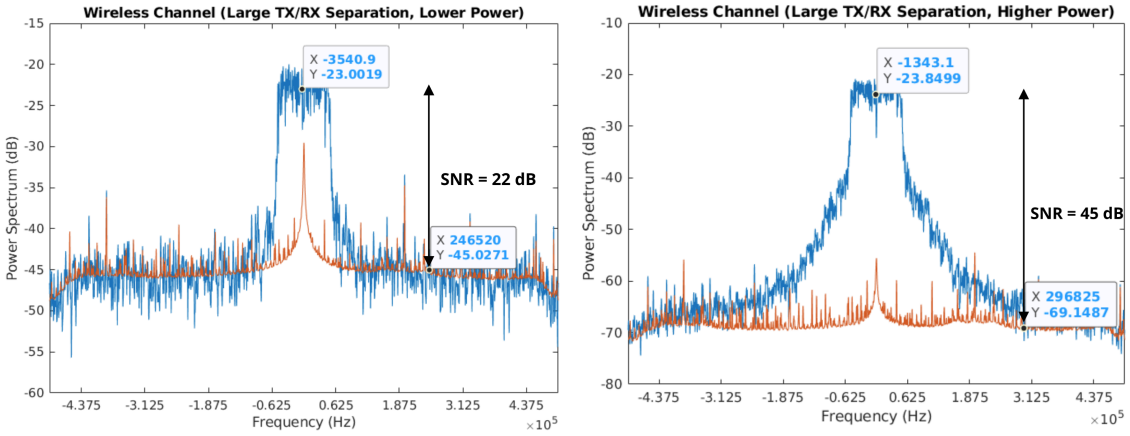


Figure 8.9: Q-learning performance for smaller TX/RX separation with varying power configurations, showing convergence behavior as training continues

impacted by noise. However, note that the higher power case has more packet errors until after epoch 20. Every test is different, and actions at the beginning are chosen randomly. Therefore, these random actions for the higher power test run happened to be more actions that were the same as the interferers'. However, at the end of training, the higher power case, though just slightly, performed better than the lower power case, as expected.

8.3.1.2 Larger TX/RX separation distance

For the larger TX/RX separation configuration, a distance of 65 inches was used, which is 10 times larger than the small separation configuration of 6.5 inches. The PSDs of the received signal for lower and higher power configurations are shown in Fig. 8.10b.



(a) PSD of OFDM signal and noise floor for lower power

(b) PSD of OFDM signal and noise floor for higher power

Figure 8.10: PSD of OFDM signal and noise floor for larger TX/RX separation, with the SNR annotated

Fig. 8.10a shows the PSD for a lower power configuration. The power of the noise floor is approximately -45 dB, and the power of the OFDM signal is approximately -23 dB. Thus, the SNR of the lower power larger separation configuration is

$$SNR = -23 \text{ dB} - (-45 \text{ dB}) = 22 \text{ dB} \tag{8.5}$$

Fig. 8.10b shows the PSD for a higher power configuration. The power of the noise floor is

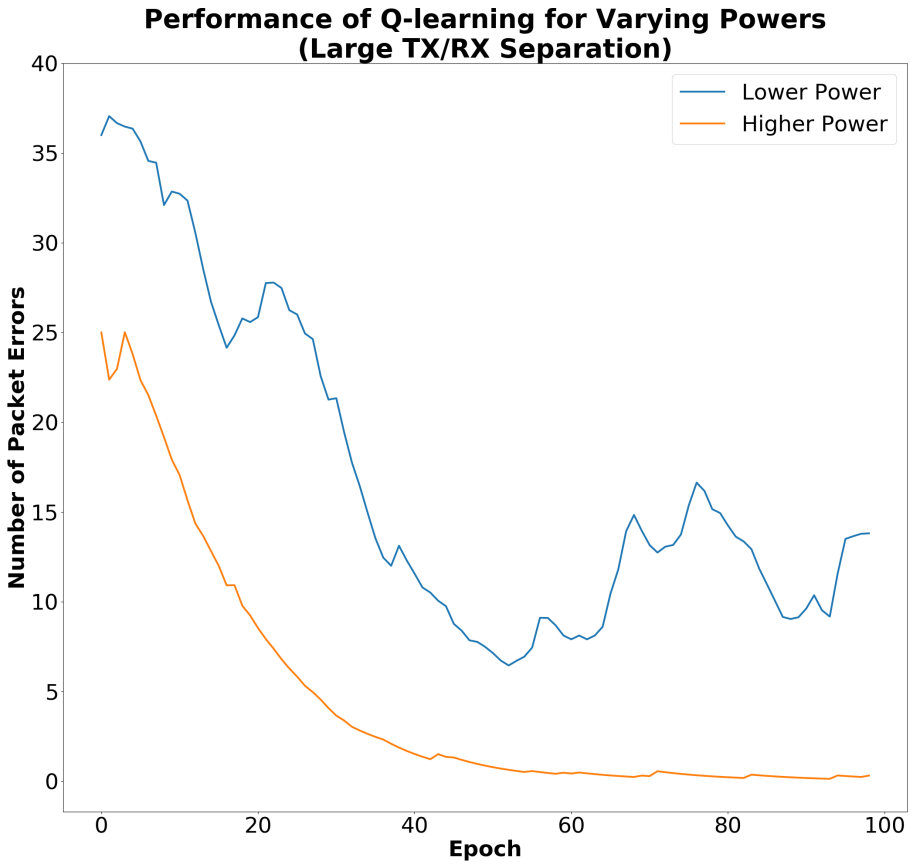


Figure 8.11: Q-learning performance for larger TX/RX separation with varying power configurations, showing convergence behavior as training continues

approximately -69 dB, and the power of the OFDM signal is approximately -24 dB. Thus, the SNR of the higher power larger separation configuration is

$$SNR = -24 \text{ dB} - (-69 \text{ dB}) = 45 \text{ dB} \tag{8.6}$$

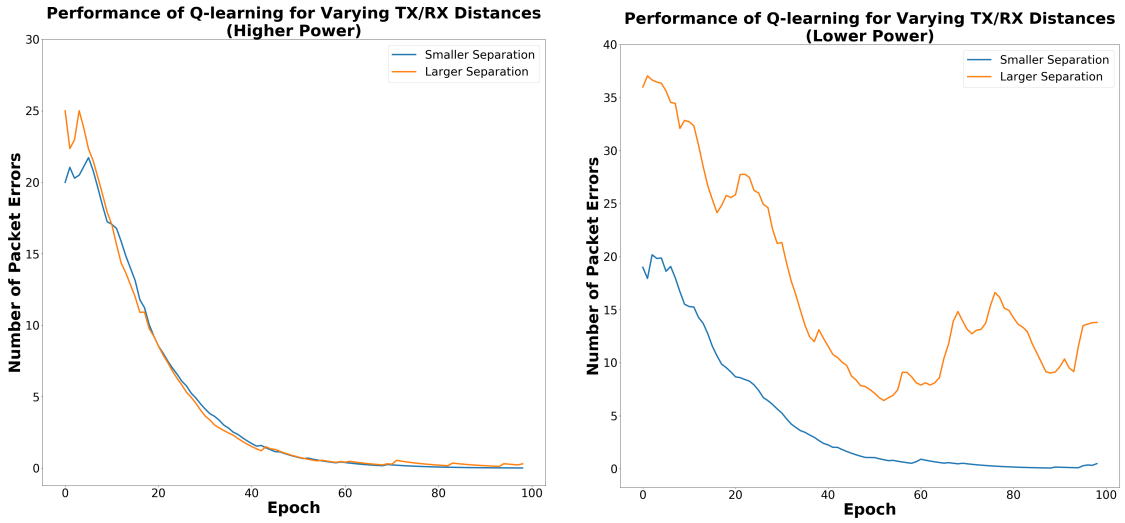
which is 23 dB higher than the lower power configuration. Therefore, the RL performance was evaluated for each SNR with an $N = 8$ sub-channel configuration to see how noise affects training and convergence behavior for a larger TX/RX separation setting.

The Q-learning performance for both power configurations is shown in Fig. 8.11. Again, for convergence, the number of packet errors should go to 0 as the number of epochs increase. This behavior is again observed for the higher power setting as shown in Fig. 8.11. However, the lower power setting fails to converge, unlike the lower power case for the smaller distance configuration. Therefore, it can be concluded that both power and distance affects the performance of RL.

8.3.1.3 Summary of Results

To summarize the effects power level and TX/RX separation distances have on RL performance, refer to Fig. 8.12. In Fig. 8.12a, Q-learning convergence results for the higher power setting were plotted for each TX/RX separation to demonstrate how the performance of RL is affected by noise when the distance between the transmitter and receiver increases when operating at higher power. At a higher power setting, the expectation is that the effect of noise should be reduced. This was observed in Fig. 8.12a by the convergence behavior demonstrated by both separation configurations, indicating that as long as the power is high enough, RL performance will not be affected because the signal power is high enough to mitigate the effects of noise.

However, the same conclusions were not reached for a lower power setting. In Fig. 8.12b, Q-learning convergence results for the lower power setting were plotted for each TX/RX separation to demonstrate how the performance of RL is affected by noise when the distance between the transmitter and receiver increases when operating at lower power. The results



(a) Q-learning performance for higher power configuration

(b) Q-learning performance for lower power configuration

Figure 8.12: Convergence behavior of different TX/RX separations, comparing differences in convergence for both power setting

show that as distance increases, it becomes more difficult to achieve convergent behavior for a lower power setting. Thus, in this testing environment, it can be concluded that as the TX/RX separation distance increases, the performance of RL does not reach optimal behavior with a lower power setting.

Note that the power was constant across configurations. Therefore, path loss should not affect the performance of the system. However, because these powers and distances were tested OTA through USRPs, multipath can occur, especially at higher frequencies. In this thesis, the operating frequency was close to the WiFi band, 2.35 GHz. Therefore, it is possible for the transmitted signal to reflect and bounce off objects, even a human, when

the TX and RX are separated. This effect is seen by the poor low power performance of the larger TX/RX separation. Thus, it can be concluded that RL is affected by multipath in this testing environment when operating at a low power, as expected since the designed framework was not designed to mitigate multipath effects. Note that multipath, in this case, can be mitigated by applying a higher power to overcome multipath effects, such as changes in the signal power of the received signal.

If a received signal is affected by multipath, then the received signal can be distorted, causing detection and demodulation issues at the receiver. This, of course, also causes errors in the RL process because the reward would be wrong since the receiver cannot handle multipath. With a low power setting, the system was already at its minimum power setting for 0 packet errors before RL was applied. Thus, in the presence of multipath, the results shown in this section showed that this minimum is too low for RL to recover if the signal power dips too close to the noise floor. If the signal power is too low, it is possible that the state and reward could be wrong. For instance, if the threshold for the sensor is set for the original power level, then the signal power could result in an energy value less than that threshold due to variations in signal power. Then, the detector cannot accurately detect the presence of signals within the environment. Additionally, if the detection threshold at the receiver for the "Scmidl and Cox OFDM synch" block is too high, then it could fail to detect an OFDM signal that has a reduced power level, resulting in an incorrect reward. Both of these cases could harm the RL training process and result in the non-convergent behavior shown in Fig. 8.12b. Because the higher power setting resulted in optimal RL performance,

as shown in Fig. 8.12a, an easy, quick fix for possible path loss would be to increase the power until convergence behavior is achieved and that would serve as the new minimum lower power setting.

These results can be useful for engineers in the design process because they give insight to the power limitations on RL performance due to wireless channel impairments, such as noise and multipath existing within the natural environment. This section also demonstrated the effect distance has on the performance of RL, with undesired behavior perhaps caused by naturally occurring multipath. It was concluded that as distance increases, the policy resulting from RL struggles to meet convergent behavior when operating with minimum power. However, this issue can be mitigated by increasing the power.

8.3.2 Effect of Doppler Shift

In this section, a Doppler shift is simulated by varying the velocity to see how mobile devices with RL are affected by this phenomenon. Doppler shift occurs when the transmitter and receiver move at a velocity relative to each other, and as a result, the frequency of the signal is shifted, according to that relative velocity. This shift in frequency is the Doppler frequency. Doppler shift also depends on the angle of arrival θ, e.g. the direction of signal travel. However, this is not simulated in this work. Note that in this section, the Doppler shift effect is simulated in GNU Radio through a simulated channel model, rather than OTA through USRPs. The simulation in this section also uses the PSD-based detector, IEEE 802.11a-like protocol formatting, and an 8 sub-channel configuration following the

parameters as summarized in Table 7.1.

8.3.2.1 Change from USRP to Channel Model

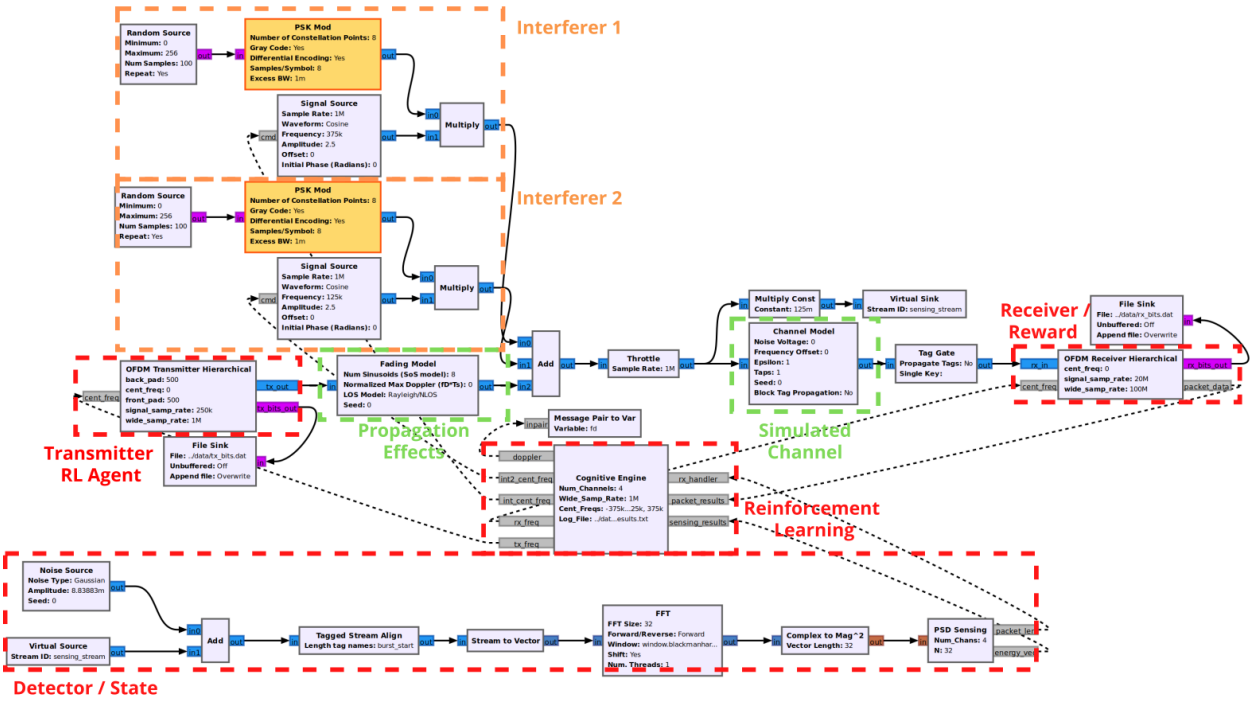


Figure 8.13: Framework in GNU Radio modified to simulate fading effects, as indicated by the added simulated channel blocks highlighted in green

In this section, the effect of mobility through the application of a Doppler shift on the performance of Q-learning is investigated. Because it is difficult to simulate over USRPs, the channel was replaced with the "Channel Model" block as shown in Fig. 8.13, where the "Fading Model" block was used to add a Doppler shift through the "Normalized Max Doppler" entry. The same transmitter, receiver, and spectrum sensing detector was used, and PSK-modulated interferers were utilized.

8.3.2.2 Important Calculations

The Doppler shift f_d can be found using Equation 8.7 below

$$f_d = \frac{f_c[Hz]v[m/s]}{c[m/s]} \tag{8.7}$$

where f_c is the carrier frequency (in this work 2.35 GHz), v the velocity of the moving device, and c the speed of light, which is always $c = 3 \times 10^8$ m/s. For the "Fading Model" block, this needs to be normalized by the wideband sampling rate F_W as follows in Equation 8.8

$$f_{d\_norm} = \frac{f_d[Hz]}{F_W[Hz]}. \tag{8.8}$$

How this Doppler shift affects the signal can be seen through the coherence time T_c, which defines the amount of time in which the channel is said to not be varying. According to [55], a practical calculation for coherence time is defined as

$$T_c = \frac{0.423}{f_d[Hz]}. \tag{8.9}$$

8.3.2.3 Simulation Parameters

Table 8.3 summarizes the settings used for the OFDM signal, Q-learning, and fading in this section's simulation. Further explanation of the chosen velocities are described in Table 8.4, where for each velocity, the corresponding Doppler shift, coherence time are given, and the number of times N_f that fading can cause a change based on the coherence time. Additionally, an example use case of each velocity is described. For example, a person walking with a cellular device might walk at a speed of $v = 1$ m/s. The RL response to

Learning Rate	$\alpha = 0.9$
Discount Factor	$\gamma = 0.1$
Epsilon Decay Rate	0.1
Minimum Epsilon	$\epsilon_{min} = 0.001$
Maximum Epsilon	$\epsilon_{max} = 1$
Number of Channels	N = 8
Number of Symbols	10
Number of Header Symbols	1
Number of Data Symbols	7
Number of Data Samples	800
Duty Cycle	14%
Wideband Sample Rate	1 MHz
Signal Sample Rate	125 kHz
Number of Epochs	100
Time per Epoch	46.4 ms
Data Transmission Time	$\mathrm{T}_D = 6.4ms$
Sensing Time	$\mathrm{T}_{WBSS} = 6.4ms$
Deadtime	$\mathrm{T}_S + T_{RL} + T_{ACK} = 40ms$
Velocity (m/s)	0, 1, 2, 3, 4, 5, 10, 30

Table 8.3: Notable parameters that determine the generation of the transmitted signal, structure of reinforcement learning, and simulation propagation effects

each of these velocities was investigated to determine the effect of small-scale fading on the performance of RL.

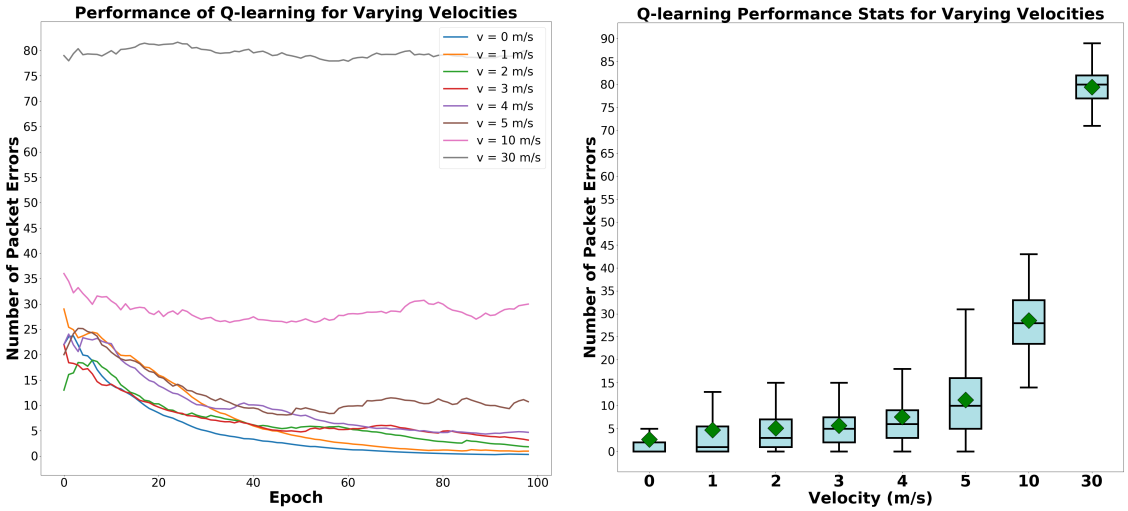
$v[m/s]$	$f_d[Hz]$	$T_c[ms]$	N_f	**Description**
0	0	N/A	N/A	Equivalent to not moving
1	7.8	54	0.12	Equivalent to a walking pedestrian
2	15.7	27	0.24	Equivalent to a jogging pedestrian
3	23.5	18	0.36	Equivalent to the average running speed
4	31.3	13.5	0.47	Equivalent to a better-than-average running speed
5	39.2	10.8	0.6	Equivalent to the average elite running speed
10	78.3	5.4	1.2	Equivalent to a car in a residential area
30	235	1.8	3.6	Equivalent to a car on the interstate

Table 8.4: Different velocities to test in this section, their respective Doppler shifts, and a description of an example use case

8.3.2.4 Q-learning Performance when Varying the Doppler Shift

Fig. 8.14a shows the results of applying different velocities (e.g. Doppler shifts) to an 8 sub-channel configuration. From this plot, it is clear that as the velocity increases, particularly from 5 m/s to 30 m/s, the performance of RL is negatively affected, as indicated by its non-convergent behavior to 0. This indicates that RL is negatively affected by small-scale fading. This is better observed in Fig. 8.14b. The baseline measurement is for $v = 0$ m/s, e.g. a device not moving. The further the statistics vary from it, the worse the RL performance. As v increases, the spread and box width slowly begins to vary from $v = 0$ m/s. However, for $v \leq 4$ m/s, it does not vary so much that it leads to non-convergent behavior. This makes sense because the fading occurs slowly, as indicated by N_f in Table 8.4, where the fading can change the channel less than 0.5 times throughout the signal. However, for $v \leq 4$ m/s, though gradual, the RL performance does get increasingly worse, with each increasing

v converging to a higher number of packet errors. This indicates that as v increases, the likelihood of the RL algorithm producing a sub-optimal policy increases, until it ceases to converge altogether, as is the case starting from $v \geq 5$ m/s.

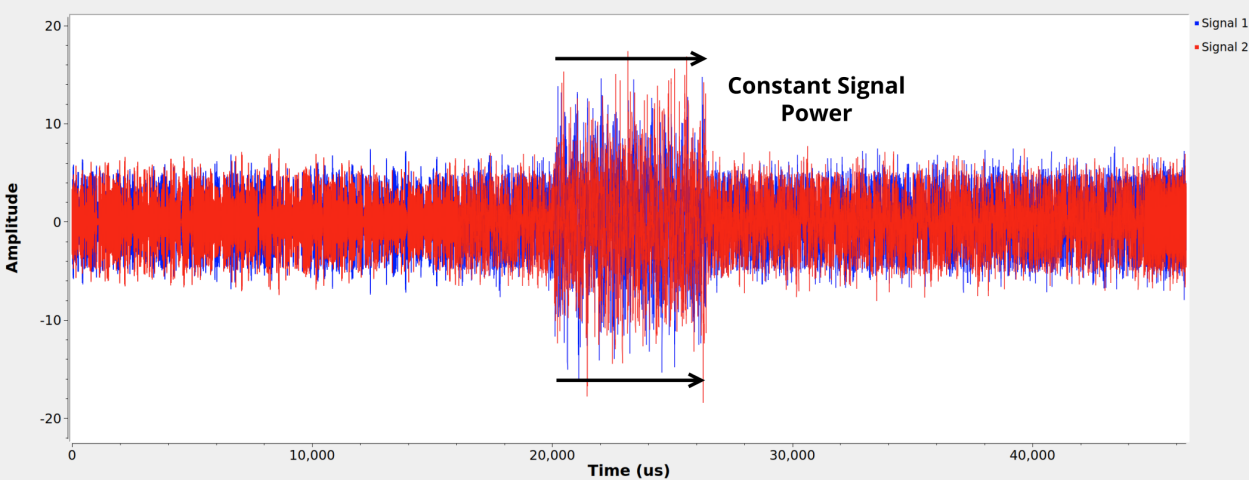


(a) Q-learning performance curves for varying velocities

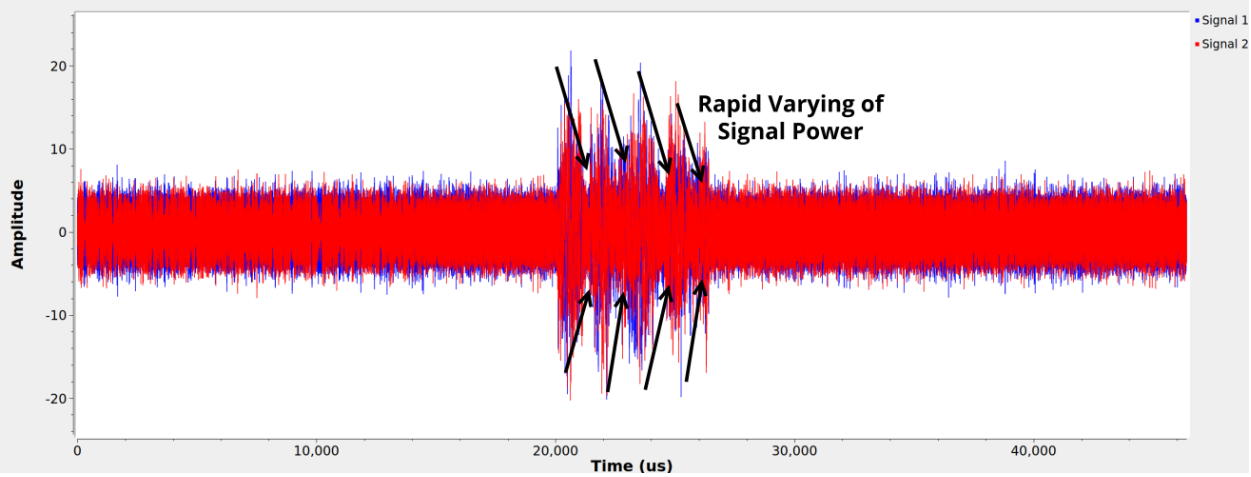
(b) Q-learning performance statistics for varying velocities

Figure 8.14: Q-learning performance when simulating different Doppler shifts for an 8 sub-channel configuration

Notice that for $v = 10$ m/s and $v = 30m/s$, there are a high number of packet errors. At this point, the signal can be highly affected by fading because it changes faster, as indicated by N_f in Table 8.4, where for $v = 10$ m/s and $v = 30$ m/s, fading can cause changes 1.2 to 3.6 times throughout the signal's duration. According to [56], this can cause distortion of the signal. Also, the faster fading can cause changes in the amplitude and phase of the signal, as shown in Fig. 8.15. Fig. 8.15a shows the case when no Doppler is applied, demonstrating a



(a) Received signal *without* Doppler shift (v = 0 m/s)



(b) Received signal *with* high Doppler shift (v = 30 m/s)

Figure 8.15: Visual demonstration of propagation effects on the received signal (as generated in GNU Radio by the implementation of a Doppler shift through the "Fading Model" block)

constant signal power at the receiver. However, this is not the case for increasing velocities, as shown in Fig. 8.15b, where it is clear that the signal is being affected by changes in signal

power. This can cause issues at the receiver when the signal power looks like noise in the locations where the sync words and header are supposed to be. As a result, it is possible that the receiver will not be able to detect the start of the signal. Additionally, at the receiver, the negative of the chosen center frequency for the next time step is fed to the receiver to bring the signal back to 0 Hz for demodulation. However, the higher the Doppler shift, the higher the probability that the center frequency of the received signal differs from the transmitted signal. Therefore, this essentially creates a frequency offset that the receiver cannot correct if the Doppler shift is high enough.

To mitigate Doppler shift, forward-error-correction (FEC) coding and/or interleaving is implemented [56]. The addition of FEC coding can add redundancy, such that the throughput is reduced, forcing the fading into a slow fade that does not affect the signal as much. Interleaving can also be implemented. With interleaving, the signal is dispersed in time, with each piece having less of a fading effect applied [56].

Note in [36], the authors claim that applying a real time-varying channel theoretically causes from approximately 4% in delayed RL observations for $v = 1$ m/s to up to 18% for $v = 5$ m/s as a consequence of latency introduced by a real channel, thus requiring a need for mitigating latency. However, their claims were not verified in simulation. This thesis, however, through the simulations performed in this section, does investigate their assumptions. In this section, as shown in Fig. 8.14a, the RL algorithm produced convergent results for $v \leq 4$ m/s. No extra latency was observed, as the percentage of sensing errors occurred below 10% for all velocities. Therefore, the implemented duty cycle around the

burst seemingly has the same effect as the redundancy from the FEC coding, where because the throughput is reduced, the signal has a slower fade applied. Thus, though a duty cycle reduces throughput, it helps reduce the effects of Doppler shift occurring at lower velocities. However, the duty cycle could not help for higher Doppler shifts caused by higher velocities, such as is the case for $v \geq 5$ m/s. In this case, higher percentages reward errors were observed, suggesting issues in the receiver due to frequency shifts in the received signal. However, sensing errors remained low. Therefore, it can be concluded that the duty cycle implemented in this thesis helped with any latency existing in the system, but cannot handle high Doppler shifts, such as ones caused by a moving car on the highway. This was expected since the system was not designed with Doppler shift mitigating techniques. Therefore, to fully mitigate Doppler shift, the RFRL system would need to be implemented with mitigating techniques, such as the FEC coding and interleaving methods dicussed above. However, as the system is now, a duty cycle can be used to mitigate low Doppler shifts, such as ones that occur from walking or running.

Chapter 9

Conclusion

As discussed in this thesis, existing literature in RL for spectrum avoidance do not consider the practical limitations of implementing RL in RF-based systems, neglecting real world impairments, such as latency, that could negate the usefulness of RL altogether. The focus of research in this area today is more concerned about RL's advancement in RF, not its practicality, where the weaknesses to RL caused by effects of real systems should be analyzed as well. Therefore, motivated to find these weaknesses such that RF-based RL systems can be deployed more smoothly as a result, the purpose of this thesis was to investigate the limitations that could diminish the effectiveness of RL in real-time RF systems, and based on these limitations, stress additional considerations in the design and operation of real radio frequency reinforcement learning (RFRL) systems. To do so, an over-the-air OFDM-based reinforcement learning framework was implemented in GNU Radio, where implementation details were discussed in Chapters 3 to 7. This framework was designed to be flexible to different physical waveform structures based on existing protocols, such as IEEE 802.11a and

5G NR, as formatted in Chapter 4, and can operate over a wireless channel using USRPs. Different effects, such as latency, wireless channel impairments, and differing protocols and spectrum sensing components, were simulated and analyzed to determine the limitations facing the implementation of RL in RF systems. A summary of the resulting analysis is given below.

- **Latency can be measured and mitigated through a duty cycle, which changes depending on the rate of individual components**. It was found in Section 8.2 that adding a duty cycle can both measure and mitigate existing latency within the system. Latency can be measured by determining what maximum allowable duty cycle ensures accurate RL, such that there exists less than 5% in state and reward errors, where the 5% error margin was the set tolerable error percentage that was observed to lead to RL convergence. This error margin can be changed to meet the needs of certain design requirements, such as allowing more error for a higher duty cycle, but the process of determining any latency remains the same. Regardless of the error margin, it was also found that the maximum allowable duty cycle was lowered to meet the requirements of faster individual components in the system, such as a higher signal sampling rate and/or faster computation time of the sensing algorithm, as was the case for the polyphase channelizer detector. Thus, it was concluded that changes in allowable duty cycle must be made when individual components, such as the signal rate, spectrum sensing algorithm, RL algorithm, etc., are changed.

- **Duty cycle is dependent on the wideband-to-signal sampling rate ratio.**

As concluded in Section 8.2.2, the deadtime remains constant across different frame structures as long as both the wideband sampling rate and signal sampling rate differ by a factor of N. This is because the latency (in ms) is scaled as well, as demonstrated by the increase in latency (in ms) when the wideband sampling rate was scaled by N times the sampling rate for the 5G NR-like protocol implementation. However, once N changed, deadtime changed, due to the new rate controlling individual components, as demonstrated in Section 8.2 for the IEEE 802.11a-like protocol implementation. For example, deadtime must be added to handle components with faster rates, as found in the case of the polyphase channelizer detector discussed in Section 8.2.3.

- **Duty cycle was lowered to synchronize the state and reward feedback to the same time step**. To ensure the state and reward arrive to the cognitive engine within the current time step, duty cycle was lowered to allow the time needed for that feedback to arrive. This is based on the latency of the individual components, especially the detector and receiver. Therefore, when implementing RL for RF, the latency caused by the detector and receiver must be determined to properly synchronize the state and reward feedback to the right time step. Such a way to do so is through the duty cycle trade-space presented in this thesis, which determined the duty cycles that resulted in synchronized state and reward feedback.

- **A trade-space analysis can determine the effective ranges of parameter values required for accurate RL.** As shown in Section 8.2, a trade-space between latency, duty cycle, and goodput can be helpful in predicting how latency causes in-

accurate RL. More specifically, the trade-space can be used to determine the effective ranges of duty cycle and goodput that result in accurate RL. This trade-space formulation is not limited to these parameters, where other problems besides latency, such as path loss and fading, can be investigated. Once a limitation is found, the formulation of this trade-space can be used to predict the effective range of the chosen parameters, selected according to the problem of interest, that ensures proper RL performance. To do so, vary the parameter values, record its effect on RL, and plot the resulting trade-space to locate the regions of accurate RL.

- **RFRL systems cannot handle large state and action spaces with discrete RL algorithms**. As discussed in Section 8.2.4, the internal memory of the computer could support sub-channel configurations of up to $N \leq 16$. Otherwise, memory errors from the GNU Radio error system occurred, which is indicative of a memory limitation for discrete RL methods, such as Q-learning. Problems requiring large state and action spaces should utilize function approximation RL methods, such as DRL, to lower the memory and computation requirements.

- **The designed RFRL framework is vulnerable to multipath and Doppler shifts**. Section 8.3.1 concluded that naturally occurring multipath within the test environment can cause RL to result in a non-convergent and/or sub-optimal policy when operating over minimum power as the TX/RX separation distance increases. This makes sense because as distance increases, the signal has more opportunity reflect and bounce off objects within the room, causing multiple versions of the signal to arrive

at the receiver, hence multipath. Thus, it was concluded that the RL process is affected by both power and distance of transmission. Because the maximum power case showed convergent behavior, an easy suggestion would be to increase the transmit power to decrease the effect of multipath. However, high transmit power is not energy efficient. Therefore, to mitigate multipath efficiently, multipath mitigating techniques should be implemented. Additionally, it was found in Section 8.3.2 that low Doppler shifts can be mitigated through a duty cycle, but high Doppler shifts require additional protection of the signal, such as FEC coding and interleaving. As velocity increases, Doppler shift increases, causing detrimental effects on the performance of RL. FEC coding and interleave can be used to mitigate this effect. However, the effects of Doppler shifts akin to walking or running could be minimized through a duty cycle. This thesis showed that the latency concerns caused by a Doppler shift discussed in [36] could be mitigated through a duty cycle, but unlike that study, this thesis asserts that additional protection from Doppler shift is required for RL to work properly for all velocities.

From this analysis, it is clear that other than the formulation of the RL scenario and the components needed to get a state and reward, the following considerations can be helpful in the RL design process. First, the trade-space presented in this thesis can be helpful in predicting the effective range of the operating parameters required for optimal RL performance. Engineers can formulate their own trade-space, but the idea remains the same - the result is a better understanding of the system's operating boundaries. Consequently, one can make more informed design decisions for a real-time RL system. Second, for proper state and

reward feedback, the spectrum sensing detector and the receiver should be able to operate in dynamic RF channels, such as channels with high noise or fading. Engineers should understand the kinds of channels the RL system would be operating over and design the detector and receiver based on the channel environments it could face. Otherwise, channel impairments can cripple the RL learning process. Third, there is an inverse relationship between an individual component's processing efficiency and system latency. To avoid individual components from operating significantly faster than the rate of the system, such that the reward and state are de-synchronized to the current time step, the mitigation and/or reduction of latency should be determined by the slowest processing component. Lastly, the algorithm complexity of components, such as the algorithm used for RL, should be considered when also trying to minimize the system's latency.

9.1 Challenges of Implementing Reinforcement Learning in Real-time Systems

When implementing the framework presented in this thesis, there were a few obstacles that posed a challenge to its success but were observed to be among the most important in the design of the system. By summarizing them here, the hope is that these observations can help make the design and implementation of future RFRL systems a smoother process. These obstacles were not obvious in the beginning design stages, but after completion of this thesis, it was concluded that it is of upmost importance that they should be major

considerations in the implementation of an RFRL system so that RL operates properly in the presence of its limitations. Note that these challenges were met because they were not considered in existing literature, making it difficult to predict these issues. However, this thesis summarizes them here for engineers to reference in the design of their own RFRL systems. They are summarized and explained below.

- **How to know when to start and stop sensing.** In a streaming system, it is difficult to know when to start and stop an operation, hence why the OFDM receiver contains two synchronization words to indicate the beginning of a signal. The same was found to be true for the spectrum sensing detector, which does not obtain a synchronization algorithm used to detect the start of a frame nor any way to detect the end of a frame. Fortunately, GNU Radio enables the use of tags, which can be placed at any sample of the signal. Therefore, for spectrum sensing, tags were used to indicate the start and end of the OFDM frame. However, when passing through the channel, the tags are removed. Therefore, in this thesis, spectrum sensing occurred at the transmitter to bypass this issue. Overall, when implementing RL for RF, *how to tell the system <u>when</u> to sense the RF environment should be a consideration in the design process.*

- **How to know when to stop waiting for the reward from the receiver.** If the signal is corrupted, it is possible that the signal is not detected. If this happens for a given time step, it is difficult for the cognitive engine to know when to stop waiting for the reward and if it gets a result, whether that reward belongs to the current time step. Therefore, when implementing RL for RF, *the waiting duration for the reward*

should be considered to ensure the reward is current to the current time step. In this work, the waiting duration was determined by the beginning of the next time step, as indicated by the tag marking the beginning of the transmitted signal.

9.2 Limitations of the Current Work and Future Work

The limitations of this work are summarized as follows and are mainly due to assumptions that were made to simplify the framework's implementation. Future work of this thesis, indicated by italicized text, is based on these limitations as well as areas of interest that would be interesting to investigate.

- **Spectrum sensing occurs at the transmitter**. As explained above, tags are removed before the signal propagates over the channel. Therefore, in this thesis, spectrum sensing occurred at the transmitter, rather than at the receiver. As a result, the detector does not capture the real RF environment when operating over-the-air. Thus, it is impossible for the framework, as it is now, to be able to capture natural interference occurring over-the-air. However, it was observed that the structure of the OFDM signal was able to successfully handle the presence of naturally-occurring interference. This was likely because the power of the OFDM signal overpowered them, especially since the natural interference was much closer to the noise floor. However, to make the framework more practical in a realistic scenario, spectrum sensing should occur at the receiver so that the detector can catch real interference naturally occurring within the

RF environment. Therefore, future work includes the *implementation of the spectrum sensing detector at the receiver*.

- **Reward relies on the header, not the payload due to the assumption that interference affects both the header and payload**. Because forward error correction (FEC) coding was not integrated, the reward was based on whether the packet header was successfully received or not, which was determined through the packet header parser in GNU Radio. The packet header chain in the receiver sends a $True$ or $False$ as a message to the cognitive engine to indicate the success or failure of communication. The packet header can indicate that the header was not corrupted but does not necessarily mean that the payload was not corrupted. In this work, the following assumption was made - if the packet header passed, then the payload passed. This assumption is valid for this thesis because the implemented interferers were set to interfere with the entire sub-channel, rather than a portion of the signal. Thus, the assumption holds because if interference occurred, then both the header and payload was corrupted. However, interference can be target to only the payload as well. Additionally, in fading environments, the payload can be corrupted due to the variations in signal power and phase. Therefore, to fully characterize the quality of the communication link, the entire signal should be represented by the reward. However, this requires the implementation of FEC coding. To protect the payload bits, implementing FEC coding could improve the communication performance of the system. FEC could also help mitigate losses due to variations in signal power caused by fading.

FEC with redundancy could also help slow the impact of fading. Therefore, future work includes *extending the reward to include a bit-error-rate (BER) of the payload and the implementation of FEC coding to protect the payload.*

- **Spectrum sensing cannot handle fading environments.** When simulating the effect of fading, an observation was that the spectrum sensing detector should be a detector designed for fading channels, as changes in received signal power can force the signal to drop below the set threshold. Consequently, the detector could fail to detect the signal's presence. Additionally, the implementation of a dynamic threshold through ML methods would be an interesting research problem. Therefore, future work includes the *implementation and analysis of a detector that can handle fading environments.*

- **Q-learning is limited to small state and action spaces due to memory**. In Section 8.2.4, it was discussed that Q-learning in the framework designed in this thesis could not handle memory requirements for $N \geq 16$ sub-channels. Therefore, the explosion of the state space serves as a major limitation to discrete RL methods, such as Q-learning. If the problem required a finer representation of the RF spectrum, such that the state and action spaces are large, then a function approximation would have to be utilized to estimate Q. One such way is through DRL. However, it would be interesting to see how much the implementation of DRL effects the latency of the system, since DRL is widely known to be more computationally complex. As a result, more computational resources from the system and hardware could be required and latency

could occur as a consequence. Therefore, future work includes the *implementation and analysis of DRL.*

These limitations serve as possible areas of improvements of the framework presented in this thesis and are good directions for future work.

Appendix A

USRP Configurations

Here, the power and distance configurations used in Section 8.3.1 are described. Figures used to demonstrate separation distances show a representation of the physical distance used.

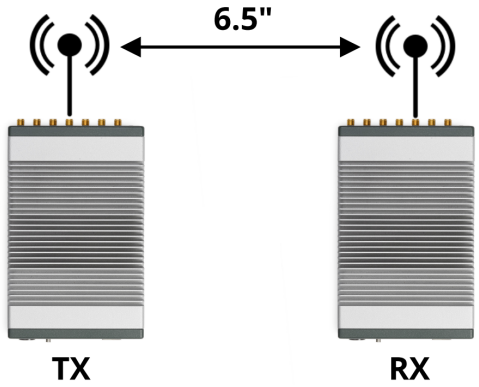
A.1 TX/RX Separation Configurations

When separating the USRPs, the distance between the antennas were measured with a ruler and matched to the distances provided in Fig. A.1.

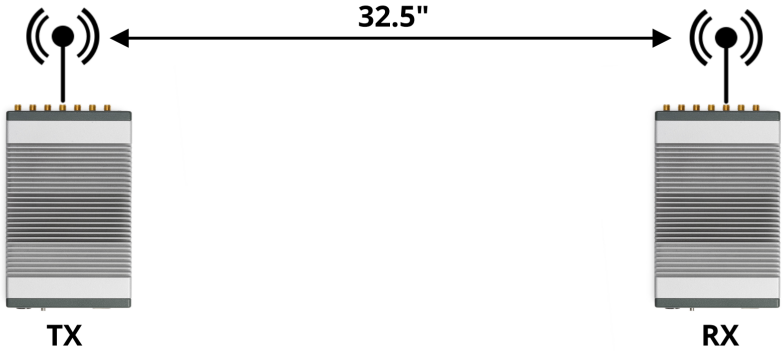
A.2 Power Configurations

As shown in Fig. A.2a, a minimum power level was determined by maximizing the amount of noise that could be present within the system until packet errors occurred. The maximum power level, as demonstrated in Fig. A.2b, was determined from the signal power that

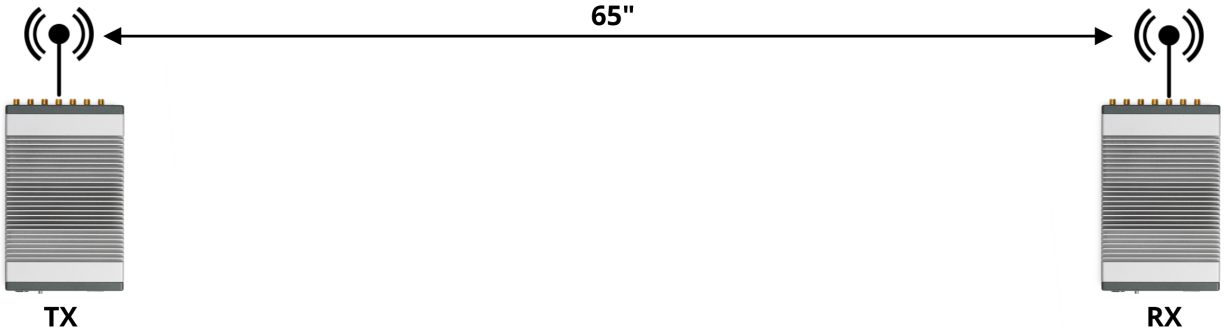
resulted when minimizing the amount of noise. Here, the USRP signal should approximately match the test signal.



(a) Small separation between TX and RX

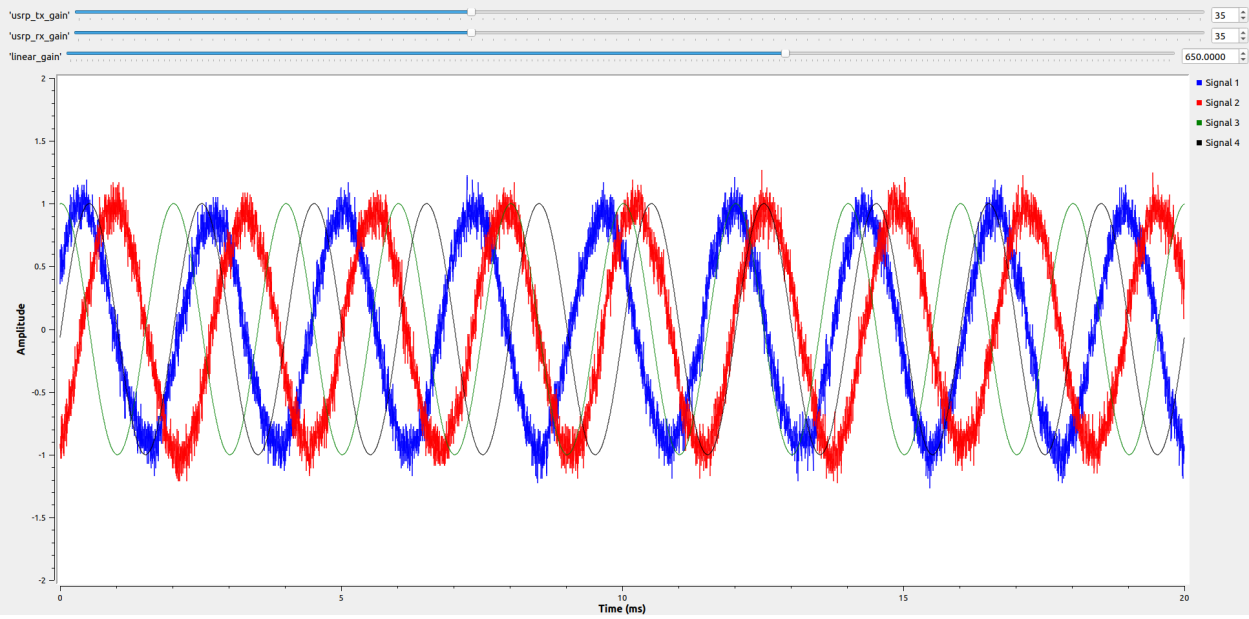


(b) Moderate separation between TX and RX

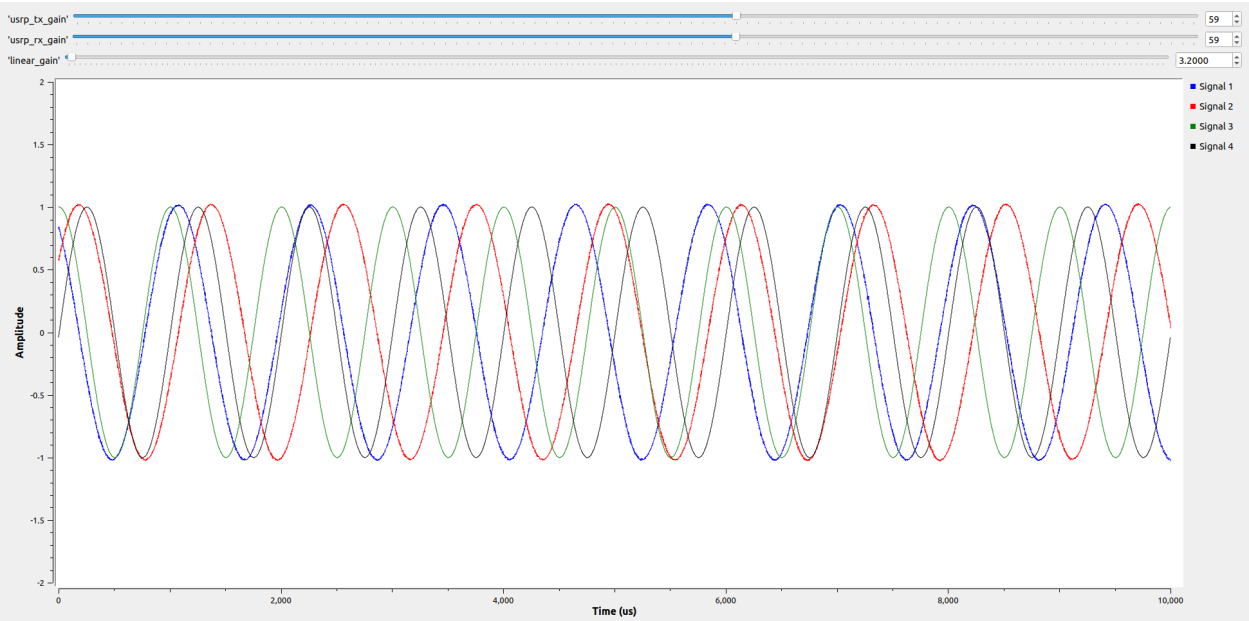


(c) Large separation between TX and RX

Figure A.1: USRP configurations for varying TX/RX separation distances



(a) Lower power setting for small separation (higher noise, more energy efficient)



(b) Higher power setting for small separation (lower noise, more energy inefficient)

Figure A.2: USRP power level configurations as determined by the USRP test script described in Chapter 5. Blue and green represent the complex signal of the test signal. Blue and red represent the complex signal of the USRP signal.

Appendix B

Fair Use Documentation

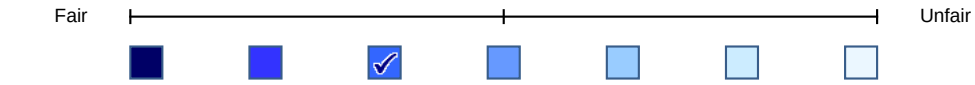
Fair Use Evaluation Documentation

Compiled using the **Fair Use Evaluator** [cc] 2008 Michael Brewer & the Office for Information Technology Policy, http://librarycopyright.net/fairuse/

Name:	Alyse Jones
Job Title:	Graduate Research Assistant
Institution:	Virginia Tech
Title of Work Used:	Result length (802.11a/g/j/p ofdm)
Copyright Holder:	Keysight Technologies, Inc.
Publication Status:	Unknown
Publisher:	
Place of Publication:	
Publication Year:	
Description of Work:	Informational website on OFDM
Date of Evaluation:	July 7, 2022
Date of Intended Use:	July 7, 2022

Describe the Purpose and Character of Your Intended Use:

[+] Use is for "criticism, comment, news reporting, teaching, (including multiple print copies for classroom use), scholarship or research"
[-] Original work is simply duplicated, or reused toward its original intention, rather than being used to create a new work with a new purpose
[+] Use is one-time, or is only occasional or spontaneous
[+] Use is clearly defined and is restricted in scope (limited duration, not iterative, restricted access, etc.)
[+] Use is not-for-profit



Describe the Nature of Your Intended Use of the Copyrighted Work:

[+] Work to be used is primarily of a factual nature (non-fiction, collection of facts, etc.)
[+] Work to be used has been previously PUBLISHED

Fair — Unfair

Describe the Amount of Your Intended Use in Relation to the Copyrighted Work as a Whole:

[+] Only limited and reasonable portions will be used
[+] Only the amount required to achieve the stated, socially-beneficial purpose or objective will be used (be that educational, artistic, scholarly, journalistic, etc.)
[+] The portion used is not the "heart" of the work (the portion considered most central to the work as a whole)

Fair — Unfair

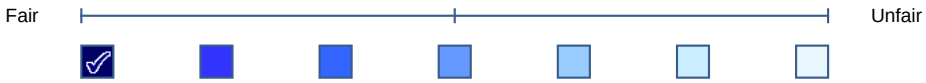
Describe the Effect of Your Intended Use on the Potential Market or Value of the Copyrighted Work:

[+] The copy of the work to be used is a legal copy
[+] Proper attribution will be given with the intended use

Fair — Unfair

The Average "Fairness Level," Based on Your Rating of Each of the 4 Factors, Is:

[see tool disclaimer for important clarifying information]:



Based on the information and justification I have provided above, I, Alyse Jones, am asserting this use is **FAIR** under Section 107 of the U.S. Copyright Code.

Signature: _____

Date of Signature: _____

***Disclaimer:** This document is intended to help you* collect, organize & archive the information you might need to support your fair use evaluation. *It is not a source of legal advice or assistance. The results are only as good as the input you have provided by are intended to suggest next steps, and not to provide a final judgment. It is recommended that* you share this evaluation with a copyright specialist before proceeding with your intended use.

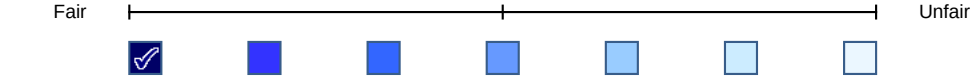
Fair Use Evaluation Documentation

Compiled using the **Fair Use Evaluator** [cc] 2008 Michael Brewer & the Office for Information Technology Policy, http://librarycopyright.net/fairuse/

Name:	Alyse Jones
Job Title:	Graduate Research Assistant
Institution:	Virginia Tech
Title of Work Used:	Innovative methods for fair coexistence between lte and wi-fi in unlicensed spectrum
Copyright Holder:	Vasilis Maglogiannis
Publication Status:	Published
Publisher:	
Place of Publication:	
Publication Year:	2018
Description of Work:	PhD Dissertation
Date of Evaluation:	July 7, 2022
Date of Intended Use:	July 7, 2022

Describe the Purpose and Character of Your Intended Use:

[+] Use is for "criticism, comment, news reporting, teaching, (including multiple print copies for classroom use), scholarship or research"
[+] Use is transformative, i.e. it uses the existing work in a new way (creates an index to the work) or for a new purpose (parody, pastiche, instructional materials, etc.) Transformative works are favored because the purpose of U.S. Copyright Law is to encourage the development and dissemination of new knowledge to benefit the public and thereby advance learning.
[+] Use is not-for-profit
[+] Use is clearly defined and is restricted in scope (limited duration, not iterative, restricted access, etc.)
[+] Use is one-time, or is only occasional or spontaneous



Describe the Nature of Your Intended Use of the Copyrighted Work:

[+] Work to be used has been previously PUBLISHED

Fair — Unfair

Describe the Amount of Your Intended Use in Relation to the Copyrighted Work as a Whole:

[+] Only limited and reasonable portions will be used
[+] The portion used is not the "heart" of the work (the portion considered most central to the work as a whole)
[+] Only the amount required to achieve the stated, socially-beneficial purpose or objective will be used (be that educational, artistic, scholarly, journalistic, etc.)

Fair — Unfair

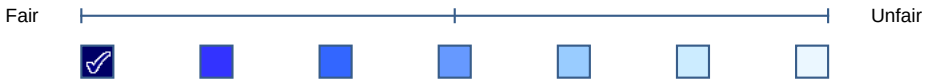
Describe the Effect of Your Intended Use on the Potential Market or Value of the Copyrighted Work:

[+] The copy of the work to be used is a legal copy
[+] Proper attribution will be given with the intended use

Fair — Unfair

The Average "Fairness Level," Based on Your Rating of Each of the 4 Factors, Is:

[see tool disclaimer for important clarifying information]:



Based on the information and justification I have provided above, I, Alyse Jones, am asserting this use is **FAIR** under Section 107 of the U.S. Copyright Code.

Signature: _____

Date of Signature: _____

***Disclaimer:** This document is intended to help you* collect, organize & archive the information you might need to support your fair use evaluation. *It is not a source of legal advice or assistance. The results are only as good as the input you have provided by are intended to suggest next steps, and not to provide a final judgment. It is recommended that* you share this evaluation with a copyright specialist before proceeding with your intended use.

Fair Use Evaluation Documentation

Compiled using the **Fair Use Evaluator** [cc] 2008 Michael Brewer & the Office for Information Technology Policy, http://librarycopyright.net/fairuse/

Name:	Alyse Jones
Job Title:	Graduate Research Assistant
Institution:	Virginia Tech
Title of Work Used:	Usrp e320 software defined radio
Copyright Holder:	Ettus Research
Publication Status:	Published
Publisher:	National Instruments
Place of Publication:	
Publication Year:	
Description of Work:	Figure of product
Date of Evaluation:	July 7, 2022
Date of Intended Use:	July 7, 2022

Describe the Purpose and Character of Your Intended Use:

[+] Use is for "criticism, comment, news reporting, teaching, (including multiple print copies for classroom use), scholarship or research"
[+] Use is transformative, i.e. it uses the existing work in a new way (creates an index to the work) or for a new purpose (parody, pastiche, instructional materials, etc.) Transformative works are favored because the purpose of U.S. Copyright Law is to encourage the development and dissemination of new knowledge to benefit the public and thereby advance learning.
[+] Use is not-for-profit
[+] Use is clearly defined and is restricted in scope (limited duration, not iterative, restricted access, etc.)
[+] Use is one-time, or is only occasional or spontaneous

Fair — Unfair

Describe the Nature of Your Intended Use of the Copyrighted Work:

[+] Work to be used has been previously PUBLISHED
[+] Work to be used is primarily of a factual nature (non-fiction, collection of facts, etc.)
[-] Work is a "consumable" (workbooks or other such educational or other materials that are typically used only once)

Fair — Unfair

Describe the Amount of Your Intended Use in Relation to the Copyrighted Work as a Whole:

[+] Only limited and reasonable portions will be used
[+] Only the amount required to achieve the stated, socially-beneficial purpose or objective will be used (be that educational, artistic, scholarly, journalistic, etc.)

Fair — Unfair

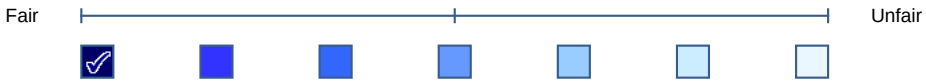
Describe the Effect of Your Intended Use on the Potential Market or Value of the Copyrighted Work:

[+] The work is NOT currently under commercial exploitation (out of print, no licensing available, etc.)
[+] The copy of the work to be used is a legal copy
[+] Proper attribution will be given with the intended use

Fair — Unfair

The Average "Fairness Level," Based on Your Rating of Each of the 4 Factors, Is:

[see tool disclaimer for important clarifying information]:



Based on the information and justification I have provided above, I, Alyse Jones, am asserting this use is **FAIR** under Section 107 of the U.S. Copyright Code.

Signature: _____

Date of Signature: _____

***Disclaimer:** This document is intended to help you* collect, organize & archive the information you might need to support your fair use evaluation. *It is not a source of legal advice or assistance. The results are only as good as the input you have provided by are intended to suggest next steps, and not to provide a final judgment. It is recommended that* you share this evaluation with a copyright specialist before proceeding with your intended use.

Bibliography

[1] *State of the iot 2020: 12 billion iot connections, surpassing non-iot for the first time*, Nov. 2021. [Online]. Available: https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/.

[2] O. R. Centre and H. Tang, *Cognitive radio networks for tactical wireless communications*, 2014.

[3] J. Mitola and G. Maguire, "Cognitive radio: Making software radios more personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, 1999. DOI: 10.1109/98.788210.

[4] *Cognitive radio for public safety*, Oct. 2016. [Online]. Available: https://www.fcc.gov/general/cognitive-radio-public-safety#:~:text=The%5C%20FCC%5C%20previously%5C%20defined%5C%20CR,environment%5C%20in%5C%20which%5C%20it%5C%20operates.%5C%E2%5C%80%5C%9D.

[5] T. J. Willink, ser. Emerging Wireless Technologies, RTO, 2007, pp. 8–1-8–20. [Online]. Available: http://www.rto.nato.int.

[6] Q. Zhou, Y. Li, Y. Niu, Z. Qin, L. Zhao, and J. Wang, ""one plus one is greater than two": Defeating intelligent dynamic jamming with collaborative multi-agent reinforcement learning," in *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, 2020, pp. 1522–1526. DOI: 10.1109/ICCC51575.2020.9345127.

[7] B. P. Lathi and Z. Ding, "Frequency hopping spread spectrum (fhss) systems," in *Modern digital and Analog Communication Systems*. Oxford University Press, 2019, pp. 691–695.

[8] ——, "Direct sequence spread spectrum," in *Modern digital and Analog Communication Systems*. Oxford University Press, 2019, pp. 702–707.

[9] B. Wang, Y. Wu, K. R. Liu, and T. C. Clancy, "An anti-jamming stochastic game for cognitive radio networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 4, pp. 877–889, 2011. DOI: 10.1109/JSAC.2011.110418.

[10] S. Singh and A. Trivedi, "Anti-jamming in cognitive radio networks using reinforcement learning algorithms," in *2012 Ninth International Conference on Wireless and Optical Communications Networks (WOCN)*, 2012, pp. 1–5. DOI: 10.1109/WOCN.2012.6331885.

[11] F. Slimeni, B. Scheers, Z. Chtourou, and V. Le Nir, "Jamming mitigation in cognitive radio networks using a modified q-learning algorithm," in *2015 International Conference on Military Communications and Information Systems (ICMCIS)*, 2015, pp. 1–7. DOI: 10.1109/ICMCIS.2015.7158697.

[12] B. Fette, "Fourteen years of cognitive radio development," in *MILCOM 2013 - 2013 IEEE Military Communications Conference*, 2013, pp. 1166–1175. DOI: 10.1109/MILCOM.2013.200.

[13] S. Haykin, "Cognitive radio: Brain-empowered wireless communications," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, pp. 201–220, 2005. DOI: 10.1109/JSAC.2004.839380.

[14] A. G. Fragkiadakis, E. Z. Tragos, and I. G. Askoxylakis, "A survey on security threats and detection techniques in cognitive radio networks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 428–445, 2013. DOI: 10.1109/surv.2011.122211.00162.

[15] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*. The MIT Press, 2018.

[16] K.-L. A. Yau, P. Komisarczuk, and P. D. Teal, "Applications of reinforcement learning to cognitive radio networks," in *2010 IEEE International Conference on Communications Workshops*, 2010, pp. 1–6. DOI: 10.1109/ICCW.2010.5503970.

[17] Y. Li, X. Wang, D. Liu, Q. Guo, X. Liu, J. Zhang, and Y. Xu, "On the performance of deep reinforcement learning-based anti-jamming method confronting intelligent jammer," *Applied Sciences*, vol. 9, no. 7, p. 1361, 2019. DOI: 10.3390/app9071361.

[18] M. A. Aref, S. K. Jayaweera, and S. Machuzak, "Multi-agent reinforcement learning based cognitive anti-jamming," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, 2017, pp. 1–6. DOI: 10.1109/WCNC.2017.7925694.

[19] *Markov decision process (mdp) toolbox: Mdp module¶*. [Online]. Available: https://pymdptoolbox.readthedocs.io/en/latest/api/mdp.html#mdptoolbox.mdp.ValueIteration.

[20] Baeldung, *Epsilon-greedy q-learning*, Jan. 2021. [Online]. Available: https://www.baeldung.com/cs/epsilon-greedy-q-learning.

[21] A. Violante, *Simple reinforcement learning: Q-learning*, Jul. 2019. [Online]. Available: https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56.

[22] F. Slimeni, Z. Chtourou, and A. B. Amor, "Reinforcement learning based anti-jamming cognitive radio channel selection," in *2020 4th International Conference on Advanced Systems and Emergent Technologies (IC_ASET)*, 2020, pp. 431–435. DOI: 10.1109/IC_ASET49463.2020.9318287.

[23] F. Yao and L. Jia, "A collaborative multi-agent reinforcement learning anti-jamming algorithm in wireless networks," *IEEE Wireless Communications Letters*, vol. 8, no. 4, pp. 1024–1027, 2019. DOI: 10.1109/LWC.2019.2904486.

[24] Q. Zhou, Y. Li, and Y. Niu, "Intelligent anti-jamming communication for wireless sensor networks: A multi-agent reinforcement learning approach," *IEEE Open Journal*

of the Communications Society, vol. 2, pp. 775–784, 2021. DOI: 10.1109/OJCOMS.2021.3056113.

[25] X. Liu, Y. Xu, L. Jia, Q. Wu, and A. Anpalagan, "Anti-jamming communications using spectrum waterfall: A deep reinforcement learning approach," *IEEE Communications Letters*, vol. 22, no. 5, pp. 998–1001, 2018. DOI: 10.1109/LCOMM.2018.2815018.

[26] S. Liu, Y. Xu, X. Chen, X. Wang, M. Wang, W. Li, Y. Li, and Y. Xu, "Pattern-aware intelligent anti-jamming communication: A sequential deep reinforcement learning approach," *IEEE Access*, vol. 7, pp. 169 204–169 216, 2019. DOI: 10.1109/ACCESS.2019.2954531.

[27] Y. Li, Y. Xu, Y. Xu, X. Liu, X. Wang, W. Li, and A. Anpalagan, "Dynamic spectrum anti-jamming in broadband communications: A hierarchical deep reinforcement learning approach," *IEEE Wireless Communications Letters*, vol. 9, no. 10, pp. 1616–1619, 2020. DOI: 10.1109/LWC.2020.2999333.

[28] J. Xu, H. Lou, W. Zhang, and G. Sang, "An intelligent anti-jamming scheme for cognitive radio based on deep reinforcement learning," *IEEE Access*, vol. 8, pp. 202 563–202 572, 2020. DOI: 10.1109/ACCESS.2020.3036027.

[29] Y. Ding, F. Yang, and J. Feng, "Intelligent cognitive anti-jamming algorithm based on long short-term memory network," in *2020 IEEE 3rd International Conference on Electronics and Communication Engineering (ICECE)*, 2020, pp. 76–82. DOI: 10.1109/ICECE51594.2020.9353032.

[30] M. A. Aref and S. K. Jayaweera, "Robust deep reinforcement learning for interference avoidance in wideband spectrum," in *2019 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW)*, 2019, pp. 1–5. DOI: 10.1109/CCAAW.2019.8904887.

[31] X. Wang, J. Wang, Y. Xu, J. Chen, L. Jia, X. Liu, and Y. Yang, "Dynamic spectrum anti-jamming communications: Challenges and opportunities," *IEEE Communications Magazine*, vol. 58, no. 2, pp. 79–85, 2020. DOI: 10.1109/mcom.001.1900530.

[32] S. Machuzak and S. K. Jayaweera, "Reinforcement learning based anti-jamming with wideband autonomous cognitive radios," in *2016 IEEE/CIC International Conference on Communications in China (ICCC)*, 2016, pp. 1–5. DOI: 10.1109/ICCChina.2016.7636793.

[33] L. Kong, Y. Xu, Y. Zhang, X. Pei, M. Ke, X. Wang, W. Bai, and Z. Feng, "A reinforcement learning approach for dynamic spectrum anti-jamming in fading environment," in *2018 IEEE 18th International Conference on Communication Technology (ICCT)*, 2018, pp. 51–58. DOI: 10.1109/ICCT.2018.8600218.

[34] X. Pei, X. Wang, J. Yao, C. Yao, J. Ge, L. Huang, and D. Liu, "Joint time-frequency anti-jamming communications: A reinforcement learning approach," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, 2019, pp. 1–6. DOI: 10.1109/WCSP.2019.8928061.

[35] F. Slimeni, Z. Chtourou, B. Scheers, V. L. Nir, and R. Attia, "Cooperative q-learning based channel selection for cognitive radio networks," *Wireless Networks*, vol. 25, no. 7, pp. 4161–4171, 2018. DOI: 10.1007/s11276-018-1737-9.

[36] A. Zubow, S. Rösler, P. Gawłowicz, and F. Dressler, "Grgym," *Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications*, 2021. DOI: 10.1145/3446382.3448358.

[37] Y. Ren, P. Dmochowski, and P. Komisarczuk, "Analysis and implementation of reinforcement learning on a gnu radio cognitive radio platform," in *2010 Proceedings of the Fifth International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, 2010, pp. 1–6.

[38] *Result length (802.11a/g/j/p ofdm)*. [Online]. Available: https://rfmw.em.keysight.com/wireless/helpfiles/89600b/webhelp/subsystems/wlan-ofdm/Content/dlg_ofdm_time_resultlenpara.htm.

[39] *802.11 ofdm overview*. [Online]. Available: https://rfmw.em.keysight.com/wireless/helpfiles/89600b/webhelp/subsystems/wlan-ofdm/content/ofdm_80211-overview.htm.

[40] V. Maglogiannis, "Innovative methods for fair coexistence between lte and wi-fi in unlicensed spectrum," Ph.D. dissertation, Jan. 2018.

[41] *Ofdm channel estimation*. [Online]. Available: https://wiki.gnuradio.org/index.php/OFDM_Channel_Estimation.

[42] F. Vergès, *Mcs table (updated with 802.11ax data rates)*, Apr. 2019. [Online]. Available: https://semfionetworks.com/blog/mcs-table-updated-with-80211ax-data-rates/.

[43] *5g/nr - frame structure*. [Online]. Available: https://www.sharetechnote.com/html/5G/5G_FrameStructure.html#SS_PBCH_FrequencyDomainResourceAllocation.

[44] E. Dahlman, S. Parkvall, and J. Skold, "Reference signals," in *5G nr: The next generation wireless access technology*. Elsevier Science Publishing Co Inc, 2018, pp. 172–182.

[45] *5g - frame structure*. [Online]. Available: https://www.sharetechnote.com/html/5G/5G_FrameStructure_Candidate.html#Factors_determining_Subframe_Structure.

[46] E. Research, *Usrp e320 software defined radio*. [Online]. Available: https://www.ettus.com/all-products/usrp-e320/.

[47] *Schmidl amp; cox ofdm synch.* [Online]. Available: https://wiki.gnuradio.org/index.php/Schmidl_%5C%26_Cox_OFDM_synch..

[48] *Frequency mod*. [Online]. Available: https://wiki.gnuradio.org/index.php/Frequency_Mod#:~:text=This%5C%20block%5C%20is%5C%20an%5C%20input,to%5C%20sensitivity%5C%20and%5C%20input%5C%20amplitude..

[49] *Ofdm frame equalizer*. [Online]. Available: https://wiki.gnuradio.org/index.php/OFDM_Frame_Equalizer.

[50] *Header/payload demux*. [Online]. Available: https://wiki.gnuradio.org/index.php/Header/Payload_Demux.

[51] *Ofdm serializer*. [Online]. Available: https://wiki.gnuradio.org/index.php/OFDM_Serializer.

[52] J. H. Reed, "Digital filter banks," in *Software radio: A modern approach to radio engineering*. Prentice Hall, 2002, pp. 101–114.

[53] P. Pandya, A. Durvesh, and N. Parekh, "Energy detection based spectrum sensing for cognitive radio network," in *2015 Fifth International Conference on Communication Systems and Network Technologies*, 2015, pp. 201–206. DOI: 10.1109/CSNT.2015.264.

[54] *Polymorphic types (pmts)*. [Online]. Available: https://wiki.gnuradio.org/index.php/Polymorphic_Types_(PMTs).

[55] T. Rappaport, *Wireless Communications*. Jan. 2002.

[56] B. Sklar, *Mitigating the degradation effects of fading channels*. [Online]. Available: https://faculty.uml.edu/jweitzen/16.582/16582ClassNotes/art_sklar6_mitigating.pdf.