Enabling CBRS experimentation and ML-based Incumbent Detection using OpenSAS

Oren Rodney Collaco

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
Computer Engineering

Luiz Pereira da Silva, Chair
Aloizio Pereira da Silva
Carl Dietrich
Amr Haj-Omar

May 5th, 2023
Arlington, Virginia

Keywords: CBRS, Experimentation, Open source, Spectrum Access System (SAS), Spectrum Sharing, Testbed

Copyright 2023, Oren Rodney Collaco
Enabling CBRS experimentation and ML-based Incumbent Detection using OpenSAS

Oren Rodney Collaco

(ABSTRACT)

In 2015, Federal Communications Commission (FCC) enabled shared commercial use of the 3.550-3.700 GHz band. A framework was developed to enable this spectrum-sharing capability which included an automated frequency coordinator called Spectrum Access System (SAS). This work extends the open source SAS based on the aforementioned FCC SAS framework developed by researchers at Virginia Tech Wireless group, with real-time environment sensing capability along with intelligent incumbent detection using Software-defined Radios (SDRs) and a real-time graphical user interface. This extended version is called the OpenSAS. Furthermore, the SAS client and OpenSAS are extended to be compliant with the Wireless Innovation Forum (WINNF) specifications by testing the SAS-CBRS Base Station Device (CBSD) interface with the Google SAS Test Environment. The Environment Sensing Capability (ESC) functionality is evaluated and tested in our xG Testbed to verify its ability to detect the presence of users in the CBRS band. An ML-based feedforward neural network model is employed and trained using simulated radar waveforms as incumbent signals and captured 5G New Radio (NR) signals as a non-incumbent signal to predict whether the detected user is a radar incumbent or an unknown user. If the presence of incumbent radar is detected with an 85% or above certainty, incumbent protection is activated, terminating CBSD grants causing damaging interference to the detected incumbent. A 5G NR signal is used as a non-incumbent user and added to the training dataset to better the ability of the model to reject non-incumbent signals. The model achieves a maximum validation accuracy
of 95.83% for signals in the 40-50 dB Signal-to-Noise Ratio (SNR) range. It achieves an 85.35% accuracy for Over the air (OTA) real-time tests. The non-incumbent 5G NR signal rejection accuracy is 91.30% for a calculated SNR range of 10-20 dB. In conclusion, this work advances state of the art in spectrum sharing systems by presenting an enhanced open source SAS and evaluating the newly added functionalities.
Enabling CBRS experimentation and ML-based Incumbent Detection using OpenSAS

Oren Rodney Collaco

(GENERAL AUDIENCE ABSTRACT)

In 2015, FCC enabled shared commercial use of the 3.550-3.700 GHz band. A framework was developed to enable this spectrum-sharing capability which included an automated frequency coordinator called SAS. The task of the SAS is to make sure no two users use the same spectrum in the same location causing damaging interference to each other. The SAS is also responsible for prioritizing the higher tier users and protecting them from interference from lower tier users. This work extends the open source SAS based on the aforementioned FCC SAS framework developed by researchers at Virginia Tech Wireless group, with real-time environment sensing capability along with intelligent incumbent detection using SDRs and a real-time graphical user interface. This extended version is called the OpenSAS. Furthermore, the SAS client and OpenSAS are extended to be compliant with the WINNF specifications by testing the SAS-CBSD interface with the Google SAS Test Environment. The ESC functionality is evaluated and tested in our xG Testbed to verify its ability to detect the presence of users in the CBRS band. The ESC is used to detect incumbent users (the highest tier) that do not inform the SAS about their use of the spectrum. An ML-based feedforward neural network model is employed and trained using simulated radar waveforms as incumbent signals and captured 5G NR signals as a non-incumbent signal to predict whether the detected user is a radar incumbent or an unknown user. If the presence of incumbent radar is detected with an 85% or above certainty, incumbent protection is activated, terminating CBSD grants causing damaging interference to the detected incumbent.
A 5G NR signal is used as a non-incumbent user and added to the training dataset to better the ability of the model to reject non-incumbent signals. The model achieves a maximum validation accuracy of 95.83% for signals in the 40-50 dB SNR range. It achieves an 85.35% accuracy for OTA real-time tests. The non-incumbent 5G NR signal rejection accuracy is 91.30% for a calculated SNR range of 10-20 dB. In conclusion, this work advances state of the art in spectrum sharing systems by presenting an enhanced open source SAS and evaluating the newly added functionalities.
Dedication

To my family, relatives, and friends, who have provided unwavering support, guidance, and inspiration throughout my journey, fueling my dreams and ambitions with their love and belief in me.
Acknowledgments

First and foremost, I express my sincere gratitude to Dr. Aloizio Pereira da Silva for his invaluable guidance, motivation, and support. I extend my thanks to my committee chair, Dr. Luiz DaSilva, and members Dr. Carl Dietrich and Dr. Amr Haj-Omar for their guidance. I dedicate my thesis to my parents, Mr. Rodney Martin John Collaco and Mrs. Ursula Collaco, who have been my pillars of strength. A special mention goes to my father for sparking my interest in this field. I also appreciate my grandmother, Margaret Rebello, and aunts, Erina Reddy and Helen Jadhav, for their unwavering support. I am grateful to my colleague and friend, Asheesh Tripathi, for his invaluable assistance with my thesis. Special acknowledgment goes to my friends and colleagues Aditya Sathish, Jaswanth Sai Reddy, Tarun Rao Keshabhoina, Vikas Krishnan Radhakrishnan, Prateek Sethi, and Tapan Bhatnagar, who have been instrumental in helping me set up experiments and providing valuable feedback. Finally, I extend my heartfelt gratitude to all friends, relatives, and others who have directly or indirectly contributed to the success of this research. Your support has made all the difference, and I am truly grateful. This work was supported by Commonwealth Cyber Initiative (CCI), an investment in the advancement of cyber R&D, innovation, and workforce development. Visit CCI at: www.cyberinitiative.org.
Contents

List of Figures xi

List of Tables xiii

1 Introduction 1

1.1 Purpose of the Research .............................................. 1

1.2 Contributions .......................................................... 1

1.3 Thesis Outline .......................................................... 4

2 Background and Related Work 5

2.1 FCC Rules .................................................................. 5

2.1.1 General CBSD requirements ......................................... 6

2.1.2 User types .............................................................. 6

2.1.3 SAS Requirements ..................................................... 8

2.2 WINNF specifications .................................................. 8

2.3 Citizen Broadband Radio Service (CBRS) Spectrum Survey .............. 10

2.4 Related Work ............................................................. 11

3 Methodology 14
3.1 SDR-based CBSD ............................. 14
3.2 Environmental Sensing ............................................. 15
3.3 Proposed Incumbent Detection Mechanism ................. 16
   3.3.1 Dataset Preparation .................. 17
   3.3.2 Neural Network Model Architecture .......... 17
   3.3.3 Training and Evaluation .................. 17
3.4 Implementation ............................................. 18
   3.4.1 OpenSAS ................................. 18
   3.4.2 SDR-based CBSD .......................... 21
   3.4.3 ESC Node .................................. 26
   3.4.4 Incumbent Detection Mechanism .......... 30
   3.4.5 National Radio Dynamic Zone (NRDZ) Demo .... 35

4 Results .................................................................... 40

4.1 Grant algorithm evaluation ..................................... 41
   4.1.1 Location based interference evaluation .......... 41
4.2 ESC Detection Latencies ..................................... 45
4.3 Model training and OTA results ............................... 47
4.4 Incumbent detection and protection demonstration .... 50
4.5 Dynamic Protected Area (DPA) functionality evaluation .... 53
5 Conclusions & Future work

5.1 Grant Algorithm .................................................. 58

5.2 Incumbent Detection Performance ............................... 58

5.3 Future work .......................................................... 59

5.3.1 ESC sensor framework ......................................... 60

5.3.2 Grant algorithm improvement ................................. 60

5.3.3 Dataset & model improvement ............................... 61

Bibliography ................................................................. 62

Appendices ................................................................. 65

Appendix A First Appendix ............................................. 66

A.1 Grant algorithm and related code ............................... 66
List of Figures

2.1 WINNF SAS architecture. .................................................. 9
2.2 CBSD state diagram. .......................................................... 10
2.3 CBRS spectrum survey (3.55 - 3.56 GHz) in Blacksburg, VA. .... 11
3.1 OpenSAS architecture. ...................................................... 19
3.2 CBSD view - OpenSAS User Interface (UI). ......................... 20
3.3 Spectrum view - OpenSAS UI. ........................................... 20
3.4 SAS client architecture. .................................................... 22
3.5 SAS-CBSD sequence diagram. ............................................ 25
3.6 N310 as ESC sensor deployed in the CCI xG Testbed radio grid network. . 27
3.7 ESC Application flowchart. ................................................. 29
3.8 National Institute of Standards and Technology (NIST) Radar waveform generator. ......................................................... 30
3.9 Adding noise to generated waveforms. .................................. 31
3.10 Neural network model. ..................................................... 33
3.11 Hat Creek Radio Observatory (HCRO) satellite view. ............. 35
3.12 Satellite Orbit Prediction Processor (SOPP)-OpenSAS interface. .... 37
4.1 Grant success – OpenSAS UI. ............................................. 42
4.2 Grant success (close) – OpenSAS UI. .................................................. 43
4.3 Grant conflict – OpenSAS UI. ................................................................. 44
4.4 SNR 20-30 dB dataset training metrics. .................................................. 47
4.5 SNR 30-40 dB dataset training metrics. .................................................. 48
4.6 SNR 40-50 dB dataset training metrics. .................................................. 48
4.7 Timeline of the Incumbent Detection and protection demonstration. ....... 51
4.8 Detected incumbent user – OpenSAS UI. .............................................. 52
4.9 DPA inactive, HCRO1 & HCRO2 transmitting – OpenSAS UI. .............. 55
4.10 DPA active, HCRO1 transmitting after shifting frequency – OpenSAS UI. 56
List of Tables

3.1 Available signal parameters for NIST signal generator [2] .......................... 39

4.1 Average latencies and Standard Deviation (SD) for each operation using 500 sample points. ................................................................. 46

4.2 Training and validation accuracy. ................................................................. 49

4.3 Over the air (OTA) accuracy. ................................................................. 50

4.4 OTA 5G NR rejection accuracy. ................................................................. 50
**Acronyms**

API  Application Programming Interface

ARFCN  Absolute Radio Frequency Channel Number

CA  Certificate Authority

CBRS  Citizen Broadband Radio Service

CBSD  CBRS Base Station Device

CCI  Commonwealth Cyber Initiative

CORNET  Cognitive Radio Network Testbed

CSV  Comma-Separated Values

DPA  Dynamic Protected Area

EIRP  Effective Isotropic Radiated Power

ERP  Effective Radiated Power

ESC  Environment Sensing Capability

FaIR  (F)eder(a)ted (I)ncumbent Detection in CB(R)S

FCC  Federal Communications Commission

FFT  Fast Fourier Transform

FPGA  Field-Programmable Gate Array

FSS  Fixed Satellite Service
GAA General Authorized Access

GPS Global Positioning System

HCRO Hat Creek Radio Observatory

HTTPS Hypertext Transfer Protocol Secure

ID Identifier

IEEE Institute of Electrical and Electronics Engineers

IQ In-phase and Quadrature

JSON JavaScript Object Notation

LTE Long-Term Evolution

ML Machine Learning

MWC Mobile World Congress

NIST National Institute of Standards and Technology

NN Neural Network

NR New Radio

NRDZ National Radio Dynamic Zone

NUC Next Unit of Computing

OTA Over the air

PAL Priority Access License

PPA PAL Protection Area
**PU** Primary User

**RA** Radio Astronomy

**ReLU** Rectified linear unit

**REST** REpresentational State Transfer

**RF** Radio Frequency

**SAS** Spectrum Access System

**SD** Standard Deviation

**SDR** Software-defined Radio

**SFP** Small Form-factor Pluggable

**SNR** Signal-to-Noise Ratio

**SOPP** Satellite Orbit Prediction Processor

**SVM** Support Vector Machine

**TCP** Transmission Control Protocol

**UHD** USRP Hardware Driver

**UI** User Interface

**URL** Uniform Resource Locator

**USRP** Universal Software Radio Peripheral

**WINNF** Wireless Innovation Forum
Chapter 1

Introduction

1.1 Purpose of the Research

The primary purpose of this research is to explore and enhance spectrum management techniques in shared spectrum environments, specifically in the Citizen Broadband Radio Service (CBRS) band. To achieve this goal, the implementation and evaluation of real-time, open-source environmental sensing functionality is critical. In the current state of Spectrum Access System (SAS), as presented by Kikamaze [9] and Makin [11], there are a few limitations such as the lack of real-time web interface and intelligent incumbent detection. Building upon these existing efforts, the open-source SAS is extended to include incumbent detection and a real-time web interface. The research also investigates different ML-based incumbent detection methods, such as Support Vector Machine (SVM) classifiers [1], and opts for a feed-forward neural network approach for signal classification.

1.2 Contributions

Several significant contributions aimed at improving spectrum-sharing systems and their implementation are presented. Building on the existing efforts presented by Kikamaze [9] and Makin [11], an extended version of the open-source SAS [20], called OpenSAS, is provided. The contributions of this work are as follows:
• **Contribution 1:** Enhanced SAS Code: The SAS code has been updated to incorporate Hypertext Transfer Protocol Secure (HTTPS) protocol, aligning it more closely with the ideal Wireless Innovation Forum (WINNF) SAS. Additionally, the grant algorithm is updated to a suburban path loss based model [10] for calculating interference distance for coming grant requests. This updated code is made available on Github at [3]

• **Contribution 2:** Intelligent Incumbent Detection: A standalone application for an Environment Sensing Capability (ESC) node connected to the OpenSAS has been developed. This application integrates with the incumbent signal detection using an Machine Learning (ML)-based feed-forward neural network on the OpenSAS. The current implementation is to detect incumbent (radar) in a 5G NR based experimental CBRS network deployment. The experimental network is initially tested at the CCI xG Testbed [7]. Additionally, this platform could be utilized by the researchers to experiment with their own models for detecting signals of their interest through the ESC node in testbed environments. The C/C++ based ESC application is available at [4]

• **Contribution 3:** 5G NR SDR-based CBSD: A SDR-based 5G NR gNodeB is deployed using srsRAN 5G software implementation as a CBSD. The additional contribution is the development of a SAS client that controls the srsRAN gNodeB based on OpenSAS responses. The conformance of message format and protocol has been verified using the Google SAS test environment.

• **Contribution 4:** Improved OpenSAS Web Interface: The OpenSAS web interface has been enhanced by adding real-time spectrum sensing, as well as views of allocated and detected incumbent spectrum. This improvement allows for better monitoring and management of the available spectrum, making it easier to identify and address potential interference issues. Furthermore, instead of polling for new events, event-based
broadcasting has been introduced using the SocketIO library, allowing the immediate updates of CBSD state and spectrum on the web interface.

**Contribution 5:** Institute of Electrical and Electronics Engineers (IEEE) INFOCOM 2023 Demo Paper: A demo paper has been accepted for presentation at IEEE INFOCOM 2023, which showcases the spectrum sharing for General Authorized Access (GAA) and Priority Access License (PAL) users demonstrated at Mobile World Congress (MWC) Las Vegas 2022. The priority protection for a PAL user over a GAA user was demonstrated using OpenSAS. The PAL and GAA CBSDs were deployed using srsRAN protocol stack and SDRs.

**Contribution 6:** NRDZ Spectrum Sharing Demo: The functionality of OpenSAS has been extended and integrated with Radio Astronomy (RA) observatory systems to facilitate the spectrum sharing for the NRDZ demo at Hat Creek Radio Observatory (HCRO). This contribution was a collaborative effort with Wireless@VT research group and University of Colorado, Boulder. The DPA based protection zone and reservation mechanism was integrated with OpenSAS and the DPA triggered RA protection from damaging interference was demonstrated. More details are provided in Chapter 3.4.

Through a series of demonstrations and scenarios, the new capabilities are evaluated, and the results are presented. In the first scenario, a radar-based incumbent becomes active in one of the channels used by the PAL CBSD. In this case, the OpenSAS detects the incumbent using the ESC in 86.7 percent of the instances during a test run consisting of 1,000 trials. Upon detecting the incumbent, the OpenSAS terminates the grants of the CBSDs transmitting in the same channel.

In the second scenario, an unauthorized 5G base station becomes active in one of the channels. The ESC node detects this signal and forwards it to the OpenSAS for incumbent
detection. Utilizing the Neural Network (NN) model, the OpenSAS determines that the signal is not a radar-based incumbent and displays this information on the Web UI as an unauthorized transmission.

1.3 Thesis Outline

Chapter 2 offers an overview of the background and related work in the SAS and CBRS ecosystem. It elaborates on the rules and protocols established by the FCC and WINNF for the SAS and examines their relevance to the OpenSAS. Furthermore, this chapter discusses related research in this field.

Chapter 3 delves into the methodology and implementation aspects of the work presented in this thesis. It covers the details of the software developed and the hardware employed in the research. Chapter 4 presents the results, analysis, and observations derived from this work, along with explanations on how the results were obtained.

Chapter 5 encompasses the discussion and conclusions drawn from the research. It concludes the thesis with reflections on potential future work in this area of study.
Chapter 2

Background and Related Work

The background chapter introduces the key concepts about the CBRS and SAS ecosystem. It lays out the relevant rules from the FCC about spectrum management and incumbent protection. These rules are referenced throughout the later chapters. The OpenSAS, introduced in the earlier chapter, strives to be complaint to relevant FCC rules and WINNF specifications highlighted in this chapter.

2.1 FCC Rules

FCC provides regulations governing the use of the CBRS band in the FCC Title 47 Part 96 document [6]. That document describes the rules to be followed by each CBSD transmitting in the CBRS band. Since the OpenSAS, introduced in the previous chapter, is one of the main components of this thesis, all of these FCC rules are considered and used as a template for the functionalities added to the OpenSAS. The OpenSAS follows the FCC rules in this Section to provide incumbent or Primary Users (PUs) interference protection from lower priority users such as GAA users and PAL users. This work further uses these to demonstrate primary user protection in a testbed environment using the sensing capability. The FCC document provides the following rules for various parts of the CBRS ecosystem [6].
CHAPTER 2. BACKGROUND AND RELATED WORK

2.1.1 General CBSD requirements

General CBSD requirements are applicable to all the CBSDs operating in the CBRS band. The SDR-based CBSD designed and prototyped in this work aims to comply with these requirements. The requirements include reporting the details of the CBSD to the SAS, registering, and getting authorization to transmit in a particular channel in the band. During the registration process, the antenna details, category, FCC identification number and other deployment details are communicated to the OpenSAS. If the CBSD wants to change its operating parameters, it updates these changes to the OpenSAS. Although FCC rules require certain software security features to protect the communication between the CBSD and the SAS, since the SDR based CBSD does communicate with a commercial SAS, this requirement is not completely followed. Similarly, the firmware and software running on the CBSD ideally should be well certified and protected against modification. However, for experimentation and research purposes, this rule is not followed either. These are not limitations but the nature of a SDR-based CBSD used for experimentation and research purposes. This work also forms part of the basis for a project where an outdoor CBRS experimentation network is planned to be deployed. These software security rules will be considered when deploying the SDR-based CBSD in the project.

2.1.2 User types

According to the aforementioned FCC document, there are three user types in the CBRS band. The following are the categories.
2.1. FCC Rules

**Incumbent/Primary**

FCC considers Fixed Satellite Service (FSS) earth stations and other federal users as incumbent or primary users. They are always protected from damaging interference by the SAS. Some incumbents can inform the SAS about their transmission timings and details while others transmit without informing the SAS. The non-informing incumbents are detected using the SAS’s ESC. In context of OpenSAS, this work considers the signature of a transmitter in a non-authorised channel to determine if it is an incumbent.

**PAL**

Users that hold a Priority Access License (PAL) for a given geographic location can make use of the higher priority in that location. PAL users are limited to the 3550 MHz to 3650 MHz part of the spectrum and each PAL consists of a 10 MHz channel. Virginia Tech holds eight 10 MHz blocks PAL in Craig and Montgomery counties. This will form the basis of previously mentioned project for deploying the experimental CBRS network employing SDR-based CBSDs.

**GAA**

GAA users are the lowest priority and must always protect higher priority user from damaging interference. Every CBSD that registers with the OpenSAS that is not on the PAL list is by default considered a GAA CBSD. The SAS does not protect GAA users from interference protection from other GAA users. When PAL frequencies are not in use, the SAS may grant GAA users permission to transmit on them.
2.1.3 SAS Requirements

The SAS requirements help guarantee that the SAS performs its job of spectrum sharing between the three user tiers and protecting incumbents/PUs from damaging interference from secondary users. We try to make the OpenSAS as close to these requirements as possible while taking in consideration our own use case. In summary, the following are the main requirements of the SAS provided in subpart F of the document [6] and achieved in the OpenSAS:

- Determine and provide permissible channels or frequencies and permissible transmission power to CBSDs at their location. In its current implementation, the OpenSAS does this using the location of the registering CBSD and the channel being asked for.
- Register and authenticate the identification information and locations of the CBSDs.
- Protect incumbents/PUs from harmful interference by controlling the CBSDs appropriately.
- Use ESC sensors to acquire incumbent (in this case, an incumbent signal is generated using an SDR) activity and instruct CBSDs to move channels or cease transmissions.
- Ensure secure and reliable communication between SAS and CBSDs.

2.2 WINNF specifications

The WINNF brings to reality the requirements laid out by the FCC via protocol specifications. WINNF specifies the protocol for communication between the CBSD and the SAS [19]. It lays out the type of request the CBRS can make to the SAS and the protocol to be used for the communication. Figure 2.1 shows the general architecture of the SAS and CBRS ecosystem according to WINNF. The most relevant part of the WINNF specification to this thesis
is the SAS-CBSD protocol. OpenSAS follows this messaging protocol for communication with the SDR based CBSD.

\begin{figure}[h]
\centering
\includegraphics[width=0.8\textwidth]{sas_architecture}
\caption{WINNF SAS architecture.}
\end{figure}

**SAS-CBSD interface**

This interface consists of five requests made to publicly available Uniform Resource Locator (URL)s via HTTPS. These requests are used by the CBSD to inquire for available spectrum at its geolocation and to acquire a grant to transmit in available spectrum. Figure 2.2 shows the state the CBSD has after each request. Once the SAS grants the spectrum to the CBSD, it has to send a heartbeat request before starting transmissions. The \textit{TransmitExpireTime} received in the heartbeat response determines the time until the CBSD can continue transmitting. To renew this time, the CBSD periodically sends heartbeat request to the SAS,
letting it know that it is still transmitting.

![Diagram of CBSD state transitions](image)

Figure 2.2: CBSD state diagram.

## 2.3 CBRS Spectrum Survey

To verify that the CBRS band is not being used in the area where the CBRS experimentation network is projected to be deployed, this work includes a spectrum survey of the planned area of deployment in Blacksburg, VA. To carry out this spectrum survey, two students borrowed an off-road gator vehicle to mount a CBRS antenna and carry a portable spectrum analyser to take measurements along the sensor placement path. The spectrum analyser was configured to channel scanner mode and connected to a Global Positioning System (GPS) antenna to capture latitude and longitude co-ordinates. The 150 MHz CBRS band was divided into 10 MHz chunks in the channel scanner mode. The result was power levels in each 10MHz channel for the 15 channels. This data was exported to a Comma-Separated Values (CSV) file and 15 heat maps were created corresponding to each channel. Figure 2.3 shows the
channel 1 (3.55 - 3.56 GHz) heat map produced in this survey.

![Channel 1 - 3.55 - 3.56GHz](image)

Figure 2.3: CBRS spectrum survey (3.55 - 3.56 GHz) in Blacksburg, VA.

### 2.4 Related Work

The SAS used in this thesis is an extended version of the open-source SAS described in Kikamaze’s thesis [9]. In this work, the author demonstrates and evaluates a SAS based on the WINNF. It provides some basic functionalities such as detection of incumbent using power based sensing. The basic SAS-CBSD interface is implemented; however, compliance is not tested with a commercial SAS. The author uses GNURadio application and SDR for sensing.
the environment and also determining the location of the PU using the power data from each sensor using trilateration. This SAS is further modified and used in Makin’s thesis [11] to implement a PU obfuscation scheme. The author proposes four obfuscation schemes and evaluates them using simulations and experimentation. Although sensing capability is demonstrated in the aforementioned work [9], it does not include any type of incumbent detection. Furthermore, there is lack of a real-time web interface, which is addressed in this thesis.

Caromi and Souryal [1] evaluate the use of SVM classifiers for incumbent sensors. They use field measured data of incumbent signals to train and test two classifier models. They use samples captured from in-band radar and out-of-band emissions from adjacent users at US coastal locations with added computer-generated Long-Term Evolution (LTE) signals and Gaussian noise. They find that both types of classifiers perform similarly with Gaussian noise or LTE emissions; however, the peak analysis classifier performs better than the other in adjacent band emissions. However, this thesis uses computer-generated radar waveforms generated using NIST’s incumbent simulated radar signal generator [2] since captured data for incumbent data is not publicly available to the best of our knowledge. Furthermore, instead of SVM classifiers, this work uses feed forward neural networks to classify the signal detected by the ESC sensor to eliminate the need for feature extraction.

Troglia et al. [17] propose an ESC network with multiple sensor nodes called (F)eder(a)ted (I)ncumbent Detection in CB(R)S (FaIR) based on distributed learning framework. The authors propose that each ESC node performs the incumbent detection locally by pulling a globally maintained current detection model from a central server. This model is evaluated with local spectrum data and the model is updated intermittently. The central server updates the global model by averaging models across nodes. The authors highlight that no In-phase and Quadrature (IQ) streaming is required to detect the incumbent in this scenario.
and better average detection accuracy is achieved. However, our proposed method differs from this technique in that it does not include a distributed learning framework but a central detection model inside the OpenSAS. The implementation and evaluation of this framework [17] using distribution edge based incumbent detection can be treated as future work for extending OpenSAS capabilities.
Chapter 3

Methodology

This chapter describes the methodology and implementation of the main contributions listed in the Chapter 1. The first three sections describe the proposed methods for 5G NR SDR-based CBSD, environmental sensing and finally, the incumbent detection mechanism. The last section describes the implementation of the event-based OpenSAS UI, 5G NR SDR-based CBSD prototype, the ESC nodes, and the detection of the incumbent using a feed forward neural network.

3.1 SDR-based CBSD

To create an SDR-based CBSD that deploys a 5G NR network in an experimental CBRS environment, this thesis uses the srsRAN 5G software implementation along with a supported USRP controlled using a SAS client. We further develop a HTTPS based SAS client to communicate with the OpenSAS as well as the Google SAS test environment. The Google SAS test environment is used as a way to perform conformance testing against the WINNF specifications. The gNode configuration file contains the operating parameters of interest, such as the Absolute Radio Frequency Channel Number (ARFCN) value (determines the frequency of operation) and Radio Frequency (RF) frontend gains. The proposed method involves starting the srsgnb program once all the correct parameters are set in the configuration file based on the response received from the SAS. It further involves that the SAS client
3.2 Environmental Sensing

In the WINNF-based SAS, ESC is used to detect incumbents transmitting in a given geolocation. The WINNF specification includes incumbent informing, where the incumbent directly inform the SAS of its intention to transmit. The SAS, based on this information, can order any CBSD that might cause damaging interference to cease transmissions or to move to another channel. For incumbents that do not inform the SAS, it uses ESC sensors to detect signals beyond the baseline power expected in a given channel. This work adds a similar functionality to OpenSAS.

The method used in this work proposes SDRs with limited compute functionality to behave as a self contained ESC sensor node. In order to sense power levels across the entire 150 MHz spectrum two possible methods were considered:

- Setting the sample rate to the channel bandwidth and the center frequency to the center of the channel. After capturing enough samples for the Fast Fourier Transform (FFT), move the center frequency to the next channel and capture IQ samples for that channel. Do this for all channels and repeat. However, changing the center frequency takes time and increases time to detect power at a given channel.

- Set the sample frequency to capture the entire CBRS band (or any band of interest). This is the ideal method since it requires the lowest time between two consecutive power checks across the entire spectrum. However, the SDR might be limited in the sampling rate or analog bandwidth for a given channel.
The second method was chosen; however, due to analog bandwidth limit of 100MHz, only 10 channels are considered. The SDR captures IQ samples at its maximum quadrature sampling rate and computes the power (dBm) per 10MHz for each channel.

Capturing power at each band provides two possibilities. The first one is computing the FFT on the embedded processor, and the second is using the Field-Programmable Gate Array (FPGA) on the SDR to compute the FFT. Based on the number of FFT bins, each channel power is calculated by averaging the bins corresponding to a given channel. If the channel power exceeds a certain threshold, the SDR changes the center frequency and sampling rate to capture the signal above threshold. It captures the signal for a set duration and sends the raw IQ data to the OpenSAS via an HTTPS request.

### 3.3 Proposed Incumbent Detection Mechanism

This section presents a proposed incumbent detection mechanism for identifying radar signals in the presence of other communication signals, such as 5G NR. The goal is to build a neural network model capable of accurately predicting the presence of incumbent radar signals in the spectrum while rejecting other types of signals. To achieve this, the model is trained on a mixed dataset containing both simulated radar signals and real-world 5G NR signals. The mixed dataset can prevent the model from overfitting to recognize the only the presence of a signal as incumbent. This may cause any signal to be detected as incumbent radar signal which not desired.
3.3. PROPOSED INCUMBENT DETECTION MECHANISM

3.3.1 Dataset Preparation

The NIST simulated waveform generator is used to create a dataset of simulated radar signals. These signals are generated with various waveform types and SNRs. In addition, 5G NR signals are captured using an ESC sensor node and OpenSAS, and are labeled as non-incumbent signals. The captured 5G NR signals are then combined with the simulated radar signals to create a mixed dataset for training and evaluation purposes.

3.3.2 Neural Network Model Architecture

The proposed neural network model consists of multiple layers, including dense layers, dropout layers, and an output layer with a sigmoid activation function. The model uses Rectified linear unit (ReLU) activation functions, HeNormal kernel initializers, and L2 regularization for the dense layers. Dropout layers are added after the dense layers to reduce overfitting. The output layer produces a value between 0 and 1, representing the probability of the input belonging to a radar incumbent class.

3.3.3 Training and Evaluation

The model is trained using the Adam optimizer and binary crossentropy as the loss function. The learning rate is experimented with to achieve a balance between model convergence speed and model stability. Early stopping is employed during training to prevent overfitting. The training process is monitored using TensorBoard, which provides useful visualizations and insights during training.

The performance of the model is evaluated using accuracy as a performance metric. The accuracy value determines the probability of it being a radar incumbent. A learning rate is
chosen such that it strikes a balance between convergence speed and model stability. Loss and accuracy metrics for training and validation are recorded and plotted to visualize the model’s performance.

3.4 Implementation

This section describes the OpenSAS event-based UI enhancement and details about the grant algorithm (Section 3.4.1). It provides details for the SAS client and hardware used in the SDR-based CBSD (Section 3.4.2). It also walks through the ESC sensor node implementation (Section 3.4.3). In addition it provides details on the ML model, dataset and training process for incumbent detection (Section 3.4.4). Finally, it goes over the capabilities added to the OpenSAS for the collaborative HCRO-NRDZ demo (Section 3.4.5).

3.4.1 OpenSAS

The OpenSAS manages and enforces spectrum via the SAS-CBSD interface and the ESC sensor nodes. The important aspects inside the OpenSAS to achieve this include the grant algorithm and the incumbent detection model. The architecture diagram for the OpenSAS is provided in Figure 3.1.

Web Interface

This work improves the web interface (or web client) by using SocketIO [14] for event-based communication between the OpenSAS and the web interface. The existing implementation used REpresentational State Transfer (REST) Application Programming Interfaces (APIs) to update the CBSD states on the interface. In order to display the most recent information
from OpenSAS or to make changes in the CBSD states (such as manually drop grants or deregister), the web client has to make a request to get the most recent information. However, since SocketIO uses web sockets as the underlying protocol to establish a Transmission Control Protocol (TCP) tunnel between the OpenSAS and the web client, sending events from OpenSAS everytime a new event occurs (new CBSD registration, state change, power detection on ESC sensors, etc.), becomes possible. This enables the web interface to always depict the current state of CBSDs registered and granted by the OpenSAS. Figure 3.2 shows the CBSDs page of the OpenSAS UI. It shows the current status of already registered CBSDs and new CBSDs that register automatically show up without refreshing.
CHAPTER 3. METHODOLOGY

Figure 3.2: CBSD view - OpenSAS UI.

Figure 3.3: Spectrum view - OpenSAS UI.
3.4. Implementation

Grant Algorithm

The grant algorithm determines how the OpenSAS responds to incoming grant requests. The grant algorithm in [20] considers overlapping frequency ranges with existing grants to determine if a new grant was given out. This grant algorithm is improved to consider the CBSD location (latitude and longitude) and line of sight distance from other granted CBSDs when grants are processed (Contribution 5). The grant algorithm is further improved to compute free space path loss for every incoming CBSD grant request. A power threshold value is used to determine the transmission radius of incoming grant requests, and the grant request is granted if existing grants do not coincide. Furthermore, for the NRDZ demo (Contribution 6), the grant algorithm is further enhanced to compute aggregate interference in a DPA area when determining if a CBSD is granted spectrum. The code for the grant algorithm is provided in appendix A.1

3.4.2 SDR-based CBSD

A SDR-based CBSD is a prototype base station transmitting in the CBRS band. It consists of three parts: the SAS client, the srsRAN gNodeB software stack, and the RF frontend. The hardware used to run the gNodeB application and RF frontend is described in Section 3.4.2.

SAS Client

The SAS client communicates with the SAS for spectrum sharing. The SAS client used in [11] was based on the SocketIO library, which itself is based on Web Sockets. According to WINNF, the SAS-CBSD interface uses HTTPS for communication. Furthermore, this client
was designed with the Cognitive Radio Network Testbed (CORNET) testbed [11] in mind and included message format specific to the CORNET testbed. Since our work tests the SDR-based CBSD with the Google SAS test environment, the client is re-written for this purpose. Figure 3.4 shows the block diagram of the SAS Client.

![Figure 3.4: SAS client architecture.](image)

The requests library is used to implement the HTTPS functionality. The following code snippet shows how the HTTPS request is made.

```python
Res = requests.post(Url, json=Json, verify=s.SAS_CA_PATH,
   cert=(s.SAS_CLIENT_CERT_PATH, s.SAS_CLIENT_KEY_PATH))
```

Each request is made secure with the client keys signed by the Certificate Authority (CA) certificate. The client key and certificate are provided by Google for its test environment for HTTPS communication. For the OpenSAS, it is generated on the OpenSAS server using
3.4. Implementation

OpenSSL [12] The SAS client makes the following request to the Google Test SAS or the OpenSAS to achieve spectrum sharing:

- Registration
- Spectrum Inquiry
- Grant request
- Heartbeat request: This request is made continuously by the SAS client as long as the CBSD is transmitting. It achieves two goals: informing the SAS about active transmission and enabling the SAS’s indirect control of the transmission. The SAS can return a grant terminated response to stop the CBSD from transmitting or suggest operational parameters to change the CBSD’s transmission power or frequency.
- Grant relinquishment
- Deregistration

Figure 3.5 shows the flowchart of the aforementioned requests of the SDR-based CBSD. Each request and response is numbered from 1 through 14. The first request made by the CBSD is the registration request (1). The registration request is used by the CBSD to provide details to the SAS such as FCC Identifier (ID), location and antenna details (azimuth, height and type). The SAS responds with a registration response containing the CBSD ID (2). The spectrum inquiry request (3) is an optional request made by the CBSD to check if a given channel is available. The SAS responds (4) with the channel and maximum Effective Isotropic Radiated Power (EIRP) if available. Next, CBSD tries to make grant request for a certain channel (5). If the grant request is successful, it receives a grant response (6) with the grantId and the grantExpireTime. The spectrum selection logic and heartbeat block sets the ARFCN and gain values inside the config file based on the grant response (6) it receives from the SAS. Once the values are set, the SAS client sends out the first heartbeat (7) and starts transmitting if it receives a heartbeatbeat response (8) code 0
(SUCCESS). The heartbeat response (8) includes the important TransmitExpiryTime and heartbeatInterval parameters that determine the time until the grant expires organically. However, the SAS client will send a heartbeat request (9) every heartbeatInterval to get an updated TransmitExpiryTime. This allows continuous transmission until the grant is terminated by the SAS due to external factors or the CBSD is turned off. If while heartbeating (7,8,9,10), the SAS client receives a non-zero (grant terminated) response (9) code, it will terminate the srsgnb process and keep trying to get the same grant. If it gets the grant, the srsgnb process is initiated again and the network becomes active. Finally, if the CBSD does not require the channel anymore, it sends a grant relinquish request (11) and receives a grant relinquish response (12). The CBSD deregisters itself by sending a deregistration request (13) using the CBSD ID it received in the registration response (1). It receives a deregistration response (14) if successfully deregistered.

**Hardware & srsRAN**

The srsRAN documentation provides minimum hardware requirements for running the srsRAN software application [15]. These requirements are considered and the computational resource is determined to run the software. In our implementation, the Intel Next Unit of Computing (NUC) is used as the compute resource and x310 is used as the RF frontend. The Intel NUC consists of an Intel i9-12900K processor with 3.6GHz clock speed and 5.1GHz Turbo boost speed. The NUC is equipped with a 10G interface card to link to the Small Form-factor Pluggable (SFP)+ port on the USRP. The aforementioned document provided guidelines to disable certain power saving features of the processor to achieve maximum stability when running srsRAN. These include changing the processor governor for power saving to performance, disabling processor sleep states in BIOS and turning off hyperthreading. This achieves stable performance when running srsRAN gNodeB.
Figure 3.5: SAS-CBSD sequence diagram.
3.4.3 ESC Node

The ESC sensor node is used to sense RF power levels in the desired spectrum. It consists of the RF frontend and the software application that connects to the OpenSAS.

RF Frontend

To implement the sensor, USRP N310 is used in its embedded mode. N310 has a built-in dual-core ARM 32-bit processor and Zynq 7100 series FPGA. The USRP Hardware Driver (UHD) driver includes examples on how to use the UHD driver to interface with the N310. The N310 is connected the backhaul switch, which finally interfaces with the Openstack deployment in the xG Testbed. Figure 3.6 shows the N310 deployed in the ceiling of the CCI xG testbed. N310 supports three master clocks which determine the possible range of sampling rates. For the ESC sensor node, the ideal N310 supported sample rate would be 153.6 MHz to capture the power for the entire 150 MHz spectrum. However, the N310 datasheet [13] shows that each channel has a maximum front-end analog bandwidth of 100 MHz. Thus, it can only capture a maximum bandwidth of 100MHz without sweeping or changing the center frequency.

ESC Application

The ESC application is responsible for continuously monitoring the spectrum for activity. It is implemented using C/C++ programming language and based on the examples provided in the UHD repository by Ettus Research [5] It can either run with the USRP in host mode or in embedded mode. Only USRPs with an internal processor and embedded Linux operating system support running the application in embedded mode. The application is
tested to work with X310 when running on a dedicated compute resource such as personal computer or virtual machine as well. However, using X310 requires minor changes in the application code to support the different sampling rates available on the X310. For this work, the ESC application is used in embedded mode on the N310. Figure 3.7 provides the flowchart of the ESC node operation. As mentioned in Section 3.4.3, the N310 supports a single channel analog bandwidth of 100MHz, limiting the amount of spectrum that can be observed without changing the center frequency. Hence, only 100 MHz of spectrum is observed by the current implementation running on the N310. The ESC application works as follows: The center frequency is set to 3650 MHz to capture 100 MHz supported by the analog frontend. The sampling rate is set to 122.88 MHz, as it is enough to capture the 100 MHz. It also allows changing the sampling rate to 10.24 MHz (divide by 12) without changing the master clock. The 10.24 MHz sampling rate is important because the ML model is trained using this sampling rate dataset. At this sampling rate, the ESC program
commands the USRP to send 512 samples using the UHD API function `issue_stream_cmd`. The `stream_cmd_t` is set to `STREAM_MODE_NUM_SAMPS_AND_MORE` and `num_samps` set to 512, making the USRP send only 512 samples when `issue_stream_cmd` is called. The USRP waits and expects further commands from the application and keeps the RF frontend ready to respond immediately to future requests. The ESC program computes the FFT on the received samples and then computes the average for the FFT bins corresponding to each 10 MHz channel. Once the averages are calculated, the ESC node has two program flow paths based on whether a channel is above a set threshold:

- If the average power in all channels is below the threshold, a HTTPS Post request is made to the OpenSAS with a JavaScript Object Notation (JSON) object that includes all the channel powers and sensor details such their ID and location.

- If the average power for a channel is above a threshold, the ESC application centers the frequency to that channel, changes the sampling rate to 10.24MHz and captures 10 ms worth of IQ samples. It creates a JSON object which contains the captured IQ samples and sensor details. This JSON object is sent to the OpenSAS via a HTTPS Post request for further processing. The ESC application sends more such samples until a time threshold is reached. It then returns to observing the entire spectrum for power.

The ESC application also includes printing debug and timing statistics messages. This can be enabled or disabled by compiling the application with preprocessor directives that enable/disable this functionality. The ESC application requires CA and client certificates to establish a secure connection with OpenSAS. It needs the URL at which the OpenSAS server is listening for requests.
Figure 3.7: ESC Application flowchart.
3.4.4 Incumbent Detection Mechanism

The OpenSAS includes a Neural Network (NN) model to predict if a known incumbent (radar) is transmitting in a detected channel. The ESC node sends 10 ms worth of IQ samples to the OpenSAS for this purpose.

Training on Simulated NIST Dataset

![Image of NIST Radar waveform generator]

Figure 3.8: NIST Radar waveform generator.

The model is trained on simulated radar signal data generated using the NIST simulated waveform generator [2]. The waveform generator supports generating 5 types of radar waveforms described in Table 3.1 at specified sampling rates.

The signal generator is shown in Figure 3.8. The number of waveforms to be generated can be
set as well. The waveform generator also supports adding noise to the generated waveforms to create a dataset with labels. Figure 3.9 shows the UI to add noise to the waveforms. The SNR range and waveform duration can be set. For generating the dataset, 200 waveforms of each type are generated. Noise is added to these waveforms with the range set to 10-20 dB SNR and 1000 noise-only waveforms are generated. The program labels signal waveforms as 1 and noise-only waveforms at 0. Finally, the 2000 waveforms are randomized in the dataset and saved to a .m file.

However, it was observed that the model would overfit the NIST generated data since the dataset only consisted of signal or noise. In this case, when the model was deployed, any signal with power was classified as a radar signal.

![Figure 3.9: Adding noise to generated waveforms.](image-url)
5G NR signal dataset

In order to counter overfitting and generalize the model to reject other signals, 5G NR signals were captured using the ESC sensor node and OpenSAS. An srsRAN gNodeB is started and the ESC node is set to observe the spectrum the gNodeB is transmitting on. When the ESC node detects high average power levels, it centers to the gNodeB channels and sends the sample to the OpenSAS. However, in order to generate a dataset, instead of feeding the data to a model for prediction, it is stored locally in an array with a corresponding label of 0 (since it is not a incumbent radar signal). Once, a threshold number of samples received have been stored in the array, the array is saved as .npz file to be used by the training python script.

A dataset mixing script is created which can be used to mix and randomize two input datasets to create a single dataset. This script to used to mix the NIST simulated waveform dataset and the 5G NR dataset. The script is further used to mix different SNR datasets generated using the NIST waveform generator.

Model Architecture

The model architecture determines its ability to predict outcomes. Generally, a model with more hidden layers and number of neurons per layer is used to predict the outcome when the input samples are complex and have a large number of features. In this case, the following is finalised after experimenting with the number of hidden layers and neurons per layer. Figure 3.10 shows the model summary. The following describes the model in detail.

Input layer: The input layer is a dense layer with 2048 neurons, an input shape of (102400,), a ReLU activation function, HeNormal kernel initializer, and L2 regularization with a weight of 0.001. This layer accepts input vectors of size 102400.
3.4. Implementation

Figure 3.10: Neural network model.

**Dropout layer 1:** A dropout layer with a dropout rate of 0.5 is added to help prevent overfitting by randomly setting a fraction (50%) of input units to 0 during training.

**Hidden layer 2:** Another dense layer with 512 neurons is included, using a ReLU activation function, HeNormal kernel initializer, and L2 regularization with a weight of 0.001.

**Dropout layer 2:** A second dropout layer with a dropout rate of 0.5 is added to reduce overfitting further.

**Hidden layer 3:** This dense layer has 128 neurons, a ReLU activation function, HeNormal kernel initializer, and L2 regularization with a weight of 0.001.

**Dropout layer 3:** A third dropout layer with a dropout rate of 0.5 is included to maintain the model’s generalization capabilities.

**Output layer:** The final layer is a dense layer with a single neuron and a sigmoid activation function. This layer outputs a value between 0 and 1, representing the probability of the input belonging to a particular class.

The input shape is decided based on the sampling rate of the N310 used to capture the 5G NR signals and process the signal captured to make predictions. The output layer provides the likelihood of it being an incumbent radar signal based on the output value. If the output is 1, it signifies a 100% probability of being an radar, whereas a 0 output means it is, with
100% confidence, not a radar incumbent. Values between 1 and 0 signify the probability lies between 0% and 100%.

Model Training

To create and train the feed-forward neural network model, Tensorflow library [16] is used and installed. A separate library for Intel processors is available [8] that uses Intel architecture-specific instructions to improve the training and prediction time. As the server uses Intel processors, this library is installed in order to achieve faster training time and performance. A training script is created with convenience features that allow it to present a user with a list of the data set, saved model and saved weights to select when starting the script. The model is trained and evaluated using the Adam optimizer with a learning rate of 0.001. The model is compiled with the binary crossentropy loss function, which is defined as:

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)],$$  \hspace{1cm} (3.1)

where $L(y, \hat{y})$ is the binary crossentropy loss, $y$ are the true labels, $\hat{y}$ are the predicted probabilities, and $N$ is the number of samples. The model also uses accuracy (probability of the signal being a radar incumbent) as a performance metric. TensorBoard is employed for logging the training process, and early stopping is implemented as a callback to prevent overfitting.

The user is given the option to load pre-trained model weights or train the model from scratch. If the user opts to train the model, it will undergo 10 epochs with a batch size of 1, and the training process will be monitored with early stopping based on the validation loss. The training and validation loss and accuracy metrics are recorded and plotted at the end of the training process, with the resulting plots saved as .png files. These results and training
plots are presented and discussed in Chapter 4.

### 3.4.5 NRDZ Demo

The OpenSAS functionality is extended for the purpose of a collaborative demo (Contribution 6) with University of Colorado, Boulder and University of California, Berkeley at HCRO. The following main functionalities were added to the OpenSAS to realise this demo. In this demo, the OpenSAS serves the purpose of protecting the radio observatory from damaging interference from CBSD emulators based on Raspberry Pi. The OpenSAS additionally acts as the reservation manager for RA observation by accepting or denying reservations from the RA facility. A Google satellite view of the observatory is shown in Figure 3.11.

![Figure 3.11: HCRO satellite view.](image)

**Dynamic Protected Area (DPA)**

DPAs are used to protect incumbent users such as radio observatories from CBSDs within a given radius of a defined location. DPAs are automatically activated at a defined start time...
and deactivated after a stop time. For the demo, DPA functionality is added by assuming
the DPA as a highest priority grant. A list object of DPAs is defined as shown:

```json
DyanmicProtectionAreas = [
{
    "id": "DPA-HCRO",
    "latitude": 40.817132,
    "longitude": -121.470741,
    "radius": 200000,  # in metres
    "activationTime": "2023-03-15T15:25:00Z",
    "deactivationTime": "2023-03-15T15:26:00Z",
    "maxEIRP": -1
    "active": False,
    "spectrum": [ [3550000000, 3700000000] ]
}
]
```

A thread is implemented that calls a function responsible for checking if any DPA activation
or deactivation times are coming up by iterating through the DPA list. This function is
called periodically at an interval of one second. If a DPA is to be activated, the function
creates a new grant with the DPA flag turned on for that grant. This flag is used by the
grant algorithm when processing new grant requests. Similarly, when a DPA deactivation
time arrives, its grant is terminated. During the creation of the grant by the grant algorithm,
it checks if there exist active CBSDs transmitting in the DPA radius. If there exist CBSDs
with active grants within the DPA radius and using the same spectrum as the DPA, these
3.4. IMPLEMENTATION

grants are terminated. As long as the DPA remains active, incoming grants from CBSDs matching the DPA spectrum and beyond the maximum DPA power limit are denied. Grants in other spectrum and at lower power levels than the DPA maximum are allowed.

Reservation management using SOPP

The SOPP is used by the RA facility to determine if the satellite is causing interference to a radio observation. If the SOPP determines the observation is interference free from orbiting satellites, it uploads the reservation file to a publicly accessible webserver. Figure 3.12 shows the interface diagram between the SOPP and the OpenSAS.

![Figure 3.12: SOPP-OpenSAS interface.](image)

The OpenSAS interfaces with the HCRO facility via polling the webserver to read reservations from the facility and confirms or denies the reservation based on the information OpenSAS has about already existing reservations and other incumbents in the area. If the OpenSAS determines no conflict exists with other reservations or incumbent users, it con-
firms the reservation and creates a corresponding DPA entry in the DPA list. To create this entry, the OpenSAS parses the TARDyS3 format reservation file from the facility and determines the start\(\text{time}\), stop\(\text{time}\), spectrum and location of the reservation. This DPA entry enables protection of the RA from CBSD interference during the reservation window.
### Table 3.1: Available signal parameters for NIST signal generator [2]

<table>
<thead>
<tr>
<th>Pulse Modulation</th>
<th>Pulse width (µs)</th>
<th>Chirp Width (MHz)</th>
<th>PRR (pulses per second)</th>
<th>Pulses per Burst (Min to Max)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0N #1</td>
<td>0.5 to 2.5</td>
<td>N/A</td>
<td>900-1100</td>
<td>15 to 40</td>
<td>(\Delta \geq 5)</td>
</tr>
<tr>
<td></td>
<td>(\Delta = 0.1)</td>
<td></td>
<td>(\Delta \geq 10.0)</td>
<td></td>
<td>Similar to currently deployed Radar 1</td>
</tr>
<tr>
<td>P0N #2</td>
<td>13-52</td>
<td>N/A</td>
<td>300-3000</td>
<td>5 to 20</td>
<td>(\Delta \geq 5)</td>
</tr>
<tr>
<td></td>
<td>(\Delta = 13)</td>
<td></td>
<td>(\Delta \geq 10.0)</td>
<td></td>
<td>Simulates possible phase-coded waveforms that could be used in future radar modulations</td>
</tr>
<tr>
<td>Q3N #1</td>
<td>3-5</td>
<td>50-100</td>
<td>300-3000</td>
<td>8 to 24</td>
<td>(\Delta \geq 2)</td>
</tr>
<tr>
<td></td>
<td>(\Delta = 1.0)</td>
<td>(\Delta = 10)</td>
<td>(\Delta \geq 30)</td>
<td></td>
<td>Simulates possible future multifunction Q3N-type radar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>* Short (\tau)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>* Wide Bc</td>
</tr>
<tr>
<td>Q3N #2</td>
<td>10-30</td>
<td>1-10</td>
<td>300-3000</td>
<td>2 to 8</td>
<td>(\Delta \geq 2)</td>
</tr>
<tr>
<td></td>
<td>(\Delta = 1.0)</td>
<td>(\Delta = 1)</td>
<td>(\Delta \geq 50)</td>
<td></td>
<td>Simulates possible future multifunction Q3N-type radar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>* Intermediate (\tau)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>* Intermediate Bc</td>
</tr>
<tr>
<td>Q3N #3</td>
<td>50-100</td>
<td>50-100</td>
<td>300-3000</td>
<td>8 to 24</td>
<td>(\Delta \geq 2)</td>
</tr>
<tr>
<td></td>
<td>(\Delta = 5.0)</td>
<td>(\Delta = 10)</td>
<td>(\Delta \geq 100)</td>
<td></td>
<td>Simulates possible future multifunction Q3N-type radar</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>* Wide (\tau)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>* Wide Bc</td>
</tr>
</tbody>
</table>
Chapter 4

Results

This chapter presents the final results and evaluations for this work. Section 4.1 evaluates the grant algorithm using three test cases involving different CBSDs with varying maxEIRP values in the grant request as specified in [19]. By varying the maxEIRP values, the core functionality of interference protection between grants is evaluated. Experimental evaluations show the algorithm’s ability to process grant conflicts based on distance thresholds and maxEIRP values. The outcome shows that the algorithm passes all the test cases. Section 4.2 measures ESC detection latencies for various operations, highlighting the time required for each task in the ESC application. Section 4.3 describes the training and validation of a feedforward NN model using datasets with varying SNR levels. The model’s performance across different SNR levels is evaluated, and the OTA accuracies are presented. Section 4.4 demonstrates the incumbent detection and protection functionality. OpenSAS is evaluated using three test cases, showcasing its ability to terminate grants and mitigate potential interference to incumbent users. Section 4.5 evaluates the DPA functionality, with the test verifying its successful activation and deactivation. Results show that OpenSAS handles DPA activation and deactivation events as expected.
4.1 Grant algorithm evaluation

4.1.1 Location based interference evaluation

The performance of the grant algorithm is evaluated using three test cases that involve CBSDs with different maxEIRP values. EIRP is used instead of Effective Radiated Power (ERP) as this work assumes a spherical antenna pattern that radiates isotropically for each CBSD. The WINNF specifications [19] provide a way to inform the SAS about the maxEIRP value when making the grant request. The interference distance thresholds for these CBSDs are calculated using the calculateInterferenceRadius function, which takes into account the maxEIRP of a given granted CBSD, the path loss model [10] and an interference threshold power of -96 dBm/10MHz used for PAL Protection Areas (PPAs) [18]. The same frequency is utilized during these tests, since the grant algorithm considers frequency conflicts as well and since we want to test the location/distance based spectrum reuse. Two CBSDs separated by a distance of 751.06 m are used to make grant requests corresponding to the following test cases.

Test Case 1 (No conflict): In this test case, the maxEIRP value for Request A is set to 0 dBm, resulting in a distance threshold of approximately 229.1 meters. For Grant B, the maxEIRP value is set to 5 dBm, leading to a distance threshold of approximately 316.2 meters. If the distance between Request A and Grant B is greater than the sum of their distance thresholds (545.3 meters), there will be no conflict. Figure 4.1 shows this case on the OpenSAS map view.

Test Case 2 (No conflict, close call): For this test case, the power values are set such that the interference radii for each CBSD come extremely close to each other but do not overlap. The maxEIRP value for Request C is set to 3 dBm, resulting in a distance threshold of
Figure 4.1: Grant success – OpenSAS UI.
approximately 277.8 meters. For Grant D, the maxEIRP value is set to 11 dBm, leading to a distance threshold of approximately 467.1 meters. If the distance between Request C and Grant D is more than the sum of their distance thresholds (744.9 meters), there will be no conflict. In this case, the conflict is avoided by 6.16 m. Figure 4.2 depicts this case on the OpenSAS map view.

Test Case 3 (Conflict): In this test case, the maxEIRP value for Grant E is set to 20 dBm, resulting in a distance threshold of approximately 823.45 meters. For request F, the maxEIRP value is set to 10 dBm, leading to a distance threshold of approximately 436.5 meters. As the distance between Grant E and Request F is less than the sum of their distance thresholds (1259.95 meters), there will be a conflict. The interference radius of Request E reaches
beyond the location of the CBSD with Grant F, leading to an obvious `GRANT_CONFLICT` response. Figure 4.3 depicts this case on the OpenSAS map view and the grant interference radius is seen engulfing the CBSD that is requesting Grant F.

![OpenSAS UI](image)

Figure 4.3: Grant conflict – OpenSAS UI.

These test cases demonstrate the grant algorithm’s ability to process grant conflicts between CBSDs based on their `maxEIRP` values and distance thresholds computed using these `maxEIRP` values.
4.2 ESC Detection Latencies

The ESC application experiences inherent latencies between the commencement of incumbent transmissions and the detection of increased power levels by the ESC. Additional latency occurs while capturing sufficient IQ samples for transmission to OpenSAS for incumbent prediction processing. The ESC standalone application calculates these latencies using its internal clock by noting the number of cycles before and after operations specified in Table 4.1. The time required for each operation is determined by subtracting the difference between these cycle counts. If the pre-processing directive STATS is set to 1, the application output displays the time taken for each operation. The application output is then used to calculate the average latency and SD value for each operation using around 500 sample points, as presented in Table 4.1. The center frequency shift and return latencies are significant at 120.847 ms and 121.132 ms, respectively. The frequency operation requires changing the voltage input to the voltage controlled oscillator in the USRP RF frontend, which explains the significant time. The sample rate change latency is relatively low at 4.524 ms, as the sample rate is derived from the master clock, and only the divider value is altered when changing the sample rate. The HTTPS request has the highest latency at 1207.284 ms, contributing the most to the overall latency. The HTTPS latency involves sending the entire captured IQ data to the OpenSAS. Thus, this latency depends on the network latency as well as the bandwidth, causing it to be the highest. Finally, the FFT compute performance is notably better on the Intel server (4.847 ms) compared to the embedded N310 (22.362 ms). Since the compute capability of the server is higher (due to a higher core clock speed at 5.1 GHz), it outperforms the N310 (1.2 GHz ARM core) in FFT compute time.
### Table 4.1: Average latencies and Standard Deviation (SD) for each operation using 500 sample points.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Latency (ms)</th>
<th>SD (ms)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center frequency shift</td>
<td>120.847</td>
<td>2.60</td>
<td>Time required to change the USRP center frequency to the detected channel frequency.</td>
</tr>
<tr>
<td>Sample rate change</td>
<td>4.017</td>
<td>1.310</td>
<td>Time required to change sampling rate to 10.24 MS/s from 122.88 MS/s to capture IQ samples for sending to OpenSAS.</td>
</tr>
<tr>
<td>IQ sample acquisition</td>
<td>10.866</td>
<td>0.491</td>
<td>Time required to received the samples once the USRP is commanded to send the 102400 IQ samples.</td>
</tr>
<tr>
<td>HTTPS request</td>
<td>1209.486</td>
<td>76.228</td>
<td>Time required to make the HTTPS request to the OpenSAS containing the captured IQ samples.</td>
</tr>
<tr>
<td>Center frequency return</td>
<td>121.132</td>
<td>2.99</td>
<td>Time required to return to the center frequency (3650 MHz) to observe the entire 100 MHz spectrum.</td>
</tr>
<tr>
<td>FFT compute (N310)</td>
<td>22.362</td>
<td>0.21</td>
<td>Time required to compute FFT when the ESC application is running on the N310.</td>
</tr>
<tr>
<td>FFT compute (Server)</td>
<td>4.847</td>
<td>0.102</td>
<td>Time required to compute FFT when the ESC application is running on a Intel server virtual machine.</td>
</tr>
</tbody>
</table>
4.3 Model training and OTA results

The feedforward NN model was trained using five datasets with varying SNR levels generated by the NIST simulated waveform generator. These datasets, characterized by the following SNR ranges, were used to assess the model’s training and validation accuracy: 0–10 dB, 10–20 dB, 20–30 dB, 30–40 dB, and 40–50 dB.

In Figures 4.4, 4.5, and 4.6, the validation accuracy demonstrates a positive correlation with the SNR. For example, at epoch 10, the validation accuracy is 0.78 in Figure 4.4b, which increases to 0.92 in Figure 4.5b and further rises to 0.95 in Figure 4.6b. This increase in validation accuracy with higher SNR levels can be attributed to the improved signal quality, as a higher SNR corresponds to a stronger signal in relation to the noise. With clearer signal representation, the model can more effectively learn and identify patterns, leading to better performance and higher accuracy.

![Figure 4.4: SNR 20-30 dB dataset training metrics.](image)

The loss metrics in Figures 4.4a, 4.5a, and 4.6a provide insight into the model’s learning process across epochs. A substantial decrease in loss value is observed during the initial epochs, indicating significant learning. However, the loss value stabilizes around the 4-6
CHAPTER 4. RESULTS

(a) Loss metrics

(b) Accuracy metrics

Figure 4.5: SNR 30-40 dB dataset training metrics.

(a) Loss metrics

(b) Accuracy metrics

Figure 4.6: SNR 40-50 dB dataset training metrics.

epoch mark, suggesting that the model’s learning has plateaued.

Table 4.2 presents the training and validation accuracy at different SNR levels, highlighting the peak accuracy achieved by the model during the training process.

A few observations can be made from the table:

As the SNR range increases, both training and validation accuracy tends to improve. This suggests that the model is better at learning from and generalizing to datasets with higher SNR values (20-30 dB, 30-40 dB, and 40-50 dB), where the signal is clearer and less affected
by noise. The training accuracy is consistently higher than the validation accuracy across all SNR ranges (0-10 dB, 10-20 dB, 20-30 dB, 30-40 dB, and 40-50 dB), which is expected as the model is trained to fit the training data. However, there is a significant gap between the training and validation accuracies in the 0 - 10 dB SNR range (92.08% and 66.83%, respectively), indicating that the model might be overfitting to the noisy training data in this range. The smallest gap between training and validation accuracies is in the 40 - 50 dB SNR range, with 95.71% training accuracy and 95.83% validation accuracy. This indicates that the model generalizes well to unseen data with high SNR values (30-40 dB and 40-50 dB).

To calculate the accuracy of the model in predicting the detected signal transmitted using an incumbent emulator over the air, a Python script is employed to iterate over the dataset. The UHD example program, `tx_samples_from_file`, is utilized to transmit IQ samples from a file. This file is updated with a new sample every 5 seconds by the iterator script, ensuring that OpenSAS receives each sample at least once. Each prediction value is logged to a CSV file, which is subsequently used to calculate the average prediction accuracy for the given dataset. Table 4.3 presents incumbent prediction accuracy data for three SNR (20-30 dB, 30-40 dB, 40-50 dB) levels when tested over the air. The OTA accuracy results show that
Table 4.3: Over the air (OTA) accuracy.

<table>
<thead>
<tr>
<th>Dataset (SNR)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 - 30 dB</td>
<td>68.92</td>
</tr>
<tr>
<td>30 - 40 dB</td>
<td>79.56</td>
</tr>
<tr>
<td>40 - 50 dB</td>
<td>85.35</td>
</tr>
</tbody>
</table>

Table 4.4: OTA 5G NR rejection accuracy.

<table>
<thead>
<tr>
<th>5G NR signal (SNR)</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 10 dB</td>
<td>80.390</td>
</tr>
<tr>
<td>10 - 20 dB</td>
<td>91.308</td>
</tr>
</tbody>
</table>

real-world effects on the received signal affect the accuracy adversely. The real-world effects include the RF characteristics of the USRP, antenna radiation pattern, the environment (channel) used to transmit the signal and multipath fading. These phenomena affect the received signal reducing the model accuracy for OTA incumbent detection. Comparing the training validation accuracies in Table 4.2 to OTA accuracies in Table 4.3, it is evident that for signals with SNR 2-30 dBm, the accuracy drops from 78.83% to 68.92%. Similarly, for the other two datasets (30 - 40 dB and 40 - 50 dB) the accuracy drops by 12.44% and 10.48% respectively.

4.4 Incumbent detection and protection demonstration

The scenario laid out in Figure 4.7 serves the purpose of evaluating, testing, and demonstrating the incumbent detection functionality developed in this work. When the incumbent
4.4. Incumbent Detection and Protection Demonstration

Figure 4.7: Timeline of the Incumbent Detection and protection demonstration.

starts transmitting, and the signal is detected and validated beyond a certain threshold, OpenSAS activates incumbent protection measures. This involves identifying and terminating grants for CBSDs that may be operating within the spectrum and creating potential interference for the incumbent user.

The evaluation of incumbent protection mechanisms in OpenSAS was conducted using two distinct test cases, yielding expected results as described below.

Figure 4.7 displays numbered events from 1 to 15. Relevant events are referenced in the test cases using round brackets.

In the first test case, a CBSD 1 was operating (1) in a spectrum conflicting with the incumbent user (3) (3.60 - 3.61 GHz) and located near the ESC sensor. This CBSD could potentially interfere with the incumbent user due to its proximity and shared spectrum. The ESC sensor node detects the incumbent’s channel power (5) and sends the IQ data
to OpenSAS (5,6). Upon receiving the IQ data, OpenSAS confirms the incumbent user’s presence (7, 8) and terminates the grants (9) for CBSD 1, mitigating potential interference. The detected user for a specific channel is displayed on the OpenSAS sensing page, as shown in Figure 4.8. CBSD 1 enters a registered state and periodically requests new grants (11). As long as the incumbent is detected, OpenSAS denies conflicting grants, including grants from CBSD 1. When the incumbent stops transmitting (12), OpenSAS ceases receiving IQ from the ESC sensor node. However, OpenSAS waits (13) for a threshold time (10 seconds for the demo) before terminating the incumbent grant (14). Finally, CBSD 1 receives the grant (15) as it no longer conflicts with the incumbent grant.

Figure 4.8: Detected incumbent user – OpenSAS UI.

In the second test case, CBSD 2 is operating (2) in close proximity to the ESC sensor
4.5. DPA functionality evaluation

location, but using a different spectrum band (3.55 - 3.56 GHz) compared to the incumbent user and CBSD 1 (3.60 - 3.61 GHz). Although this CBSD was close, it did not conflict with the incumbent user’s spectrum. The incumbent grant created (8, 9) in the OpenSAS did not cause a conflict to the grant of this CBSD due to non-overlapping spectrum. CBSD 2 is unaffected by the incumbent detected by OpenSAS.

These expected outcomes in the above test cases demonstrate the incumbent detection and protection mechanisms implemented in this work.

4.5 DPA functionality evaluation

To evaluate the DPA functionality added to OpenSAS in collaboration with Wireless@VT and University of Colorado, Boulder, the following test is considered.

A DPA is added to the DynamicProtectedAreas list manually with an activation time set to an arbitrary time. The DPA deactivation time is set to activationTime + 5 minutes to test the DPA functionality for 5 minutes. The DPA spectrum is set to 3600 MHz to 3700 MHz. Four simulated CBSDs are considered to be transmitting and are placed in the following locations: two CBSDs (HCRO1 and HCRO2) are placed inside the DPA, and two CBSDs are placed outside the DPA. All the CBSDs are programmed to keep trying to get a grant if their existing grant is terminated. However, one of the CBSDs inside the DPA (HCRO1) is programmed to switch frequency to 3550 - 3560 MHz in case it receives a GRANT_CONFLICT response when retrying to acquire a grant. The observed result as seen in Figures 4.9 and 4.10, successfully evaluates the activation and deactivation of the DPA functionality.

It is observed during the test and as seen in Figure 4.10 that at the DPA activation time, all the active grants interfering with the DPA are dropped and only HCRO1 is able to
receive a new grant at a different frequency. The CBSDs with grants terminated following the DPA activation receive a heartbeat response with $\text{TransmitExpiryTime}$ set to -1 and a non-zero response code, leading to them changing to the registered state. CBSD HCRO1 inside the DPA is programmed to retry the grant request with the frequency set to 3.55 - 3.56 GHz, while the other CBSD (HCRO2) keeps trying the same grant. OpenSAS denies the grant from HCRO2 as long as the DPA is active since it is conflicting with the DPA grant. However, HCRO1 receives the grant since it retries with a non-conflicting spectrum and starts transmitting. At the deactivation time, the DPA grant expires, and HCRO2 receives the grant as well. This concludes the DPA functionality evaluation.

Figure 4.9 and Figure 4.10 show the CBSD state in OpenSAS for all the CBSDs before and after the DPA activation, respectively. The light red highlighted area represents the interference distance of the CBSD based on its $\text{maxEIRP}$. The light blue area in Figure 4.10 represents the DPA area.
Figure 4.9: DPA inactive, HCRO1 & HCRO2 transmitting – OpenSAS UI.
Figure 4.10: DPA active, HCRO1 transmitting after shifting frequency – OpenSAS UI.
Chapter 5

Conclusions & Future work

This work aimed to develop and evaluate an incumbent detection and protection system for spectrum sharing in the context of OpenSAS (Contribution 2). A feedforward neural network model was trained on a dataset generated using the NIST radar waveform generator, reflecting different levels of SNR. Furthermore, 5G NR data was captured using the OpenSAS and an ESC sensor node, with the SDR-based CBSD as the signal source. The collected 5G NR data was mixed with the NIST generated datasets. The training results showed better performance at higher SNR values, as expected. The highest validation accuracy of 95.83% was achieved for the portion of the dataset with signals in the SNR range 40-50 dB. Over the air (OTA) incumbent detection is achieved by transmitting the signals in the dataset OTA and results are presented. For OTA results, the highest accuracy is achieved is 85.35%. The grant algorithm enhancements (Contribution 1) are evaluated using test cases to validate whether the correct interference distances are computed for incoming grants requests based on the path loss model [10] and interference threshold of -96dBm/10MHz [18]. The test cases pass with the correct responses from the OpenSAS based on the maxEIRP from the CBSDs. Two test cases were employed to demonstrate the incumbent detection and protection mechanisms. The first cases concluded that CBSDs interfering with incumbent are commanded to stop transmissions and the second verified that non-interfering CBSDs are unaffected. The work extended the open source SAS [20] by improving the web interface framework (Contribution 4), moving to event-based communication between the OpenSAS and the
UI for real-time updates. Lastly, the DPA functionality developed in collaboration with University of Colorado, Boulder and Wireless@VT research group (Contribution 6) was evaluated through a simulated scenario. The scenario demonstrates the activation of a DPA to protect a RA observatory from the damaging interference from CBSD emulators based on Raspberry Pi. The results show that the OpenSAS is able to determine the CBSDs inside the DPA zone and terminate their grants based on the spectrum each CBSD is using.

5.1 Grant Algorithm

The functionality of the updated grant algorithm was evaluated through three test cases involving CBSDs with different maxEIRP values. The algorithm’s effectiveness in handling location-based interference was demonstrated by accurately identifying and processing grant conflicts based on maxEIRP values and distance thresholds, computed using interference radius calculations and path loss models.

Test Cases 1 and 2 showed the algorithm’s ability to recognize non-conflicting situations, allowing successful grant requests. Test Case 3 illustrated the algorithm’s proficiency in detecting conflicts and responding with a GRANT_CONFLICT, thereby preventing interference.

5.2 Incumbent Detection Performance

The feedforward neural network model was trained using five datasets with varying SNR levels, generated by the NIST simulated waveform generator. The validation accuracy demonstrated a positive correlation with the SNR, as higher SNR levels led to improved signal quality, enabling the model to learn and identify patterns more effectively. As the SNR range increased, both training and validation accuracy tended to improve, with the model
5.3 Future work

Future research can build upon the findings of this study in several ways:

- Further refine the deep learning model by addressing the real-world factors affecting the OTA accuracy, potentially leading to improved performance in real-world scenarios.

- Investigate the model’s performance at lower SNR levels to understand its limitations better and develop strategies for enhancing its robustness.

- Explore alternative deep learning architectures and techniques to improve the incumbent detection and protection system’s accuracy.
In conclusion, this study has made significant strides in developing and evaluating an incumbent detection and protection system for spectrum sharing. The results demonstrate the potential of deep learning techniques in spectrum management and pave the way for future research and development in dynamic spectrum access.

5.3.1 ESC sensor framework

As described in Chapter 2, the work presented by Troglia et al. [17] can be considered an improvement to the OpenSAS ESC. It would involve distributing the incumbent prediction model to each sensor node and framework to periodically update the central model based on the sensor node models. The model prediction can be run on each processor or the built-in FPGA on most USRPs. However, not all USRPs will have abundant logic elements in the FPGA for this processing capability since the firmware will use most of this resource. Offloading the model processing and the FFT processing to the FPGA is beneficial since it leads to lower latencies in both processes. Additionally, in the case of the USRP N310, it frees the processor for other tasks.

5.3.2 Grant algorithm improvement

This work considers the antenna radiation pattern as spherical and assumes a fixed antenna height [10] when calculating the interference distance of the CBSD. However, the CBSD registration request provides details such as antenna type, antenna azimuth and height. This information can be used to determine the interference distance precisely, leading to more spectrum reuse as future work. If a directional antenna is used, the spherical model leads to wasted spectrum in the direction where the antenna is not pointing. In this ideal model of the transmission area, two CBSDs separated by a certain distance and pointing in
the opposite direction might be able to reuse the same spectrum. The OpenSAS map UI can be extended to plot this antenna angle and height-based interference model to visualize the CBSD transmission area.

5.3.3 Dataset & model improvement

The performance of the ML model is directly proportional to the quantity and quality of the dataset. Ideally, a dataset should consist of all possible signals that the ESC might encounter during sensing. The future work could be on improving the model performance by adding more types of non-incumbent signals such as LTE, 5G NR and other proprietary signals in the CBRS band. The presence of more non-incumbent signals improves the incumbent learning since the diverse and contrasting dataset helps in better classification. Additionally, more incumbent signals such as fixed earth stations can be included in the incumbent dataset to make the model distinctively separate variety of non-incumbent and incumbent signals.
Bibliography


Appendices
Appendix A

First Appendix

A.1 Grant algorithm and related code

def defaultGrantAlg(self, grants, request, IsPAL, CbsdList, SpectrumList, socket):
    gr = WinnForum.GrantResponse()
    gr.grantId = None
    #gr.cbsdId = request["cbsdId"]
    gr.grantExpireTime = self.calculateGrantExpireTime(grants, None, gr, True)
    gr.heartbeatInterval = self.getHeartbeatIntervalForGrantId(gr.grantId)
    gr.measReportConfig = ["RECEIVED_POWER_WITH_GRANT", "RECEIVED_POWER_WITHOUT_GRANT"]
    conflict = False
    for grant in list(grants):
        if hasattr(grant, 'status'):
            if grant.status == "TERMINATED":
                print("Skipping terminated grant")
                continue
        rangea = self.getHighFreqFromOP(grant.operationParam)
        rangeb = self.getLowFreqFromOP(grant.operationParam)
        freqa = self.getHighFreqFromOP(request.operationParam)
        freqb = self.getLowFreqFromOP(request.operationParam)
        dist = self.calculateDistance((grant.lat, grant.long), (request.lat, request.long)) * 1000
        print("Distance: " + str(dist))
        print("Grant Distance: " + str(grant.dist))
        print("Request Distance: " + str(request.dist))
if self.frequencyOverlap(freqa, freqb, rangea, rangeb):
    if IsPAL:
        if (dist < grant.dist or dist < request.dist or dist < (grant.dist + request.dist)):
            grant.status = "TERMINATED"
            for i, item in enumerate(SpectrumList):
                if item['cbsdId'] == grant.cbsdId:
                    item['state'] = 1
                    item['stateText'] = "Registered"
            SpectrumList[i] = item
            for i, cbsd in enumerate(CbsdList):
                if cbsd['cbsdId'] == grant.cbsdId:
                    cbsd['state'] = 1
                    cbsd['stateText'] = "Registered"
            CbsdList[i] = cbsd
            socket.emit('spectrumUpdate', SpectrumList)
            socket.emit('cbsdUpdate', CbsdList)
            time.sleep(i)
        else:
            if (dist < (grant.dist + request.dist)):
                conflict = True
            else:
                print("Distance greater than threshold")
    if conflict:
        print("Conflict detected")
        delattr(gr, 'grantExpireTime')
        delattr(gr, 'heartbeatInterval')
        delattr(gr, 'measReportConfig')
        delattr(gr, 'grantId')
        delattr(gr, 'channelType')
        gr.response = self.generateResponse(401)
        print(gr.response)
    else:
        gr.response = self.generateResponse(0)
        gr.operationParam = request.operationParam
gr.channelType = "GAA"

return gr

def calculateDistance(self, a, b):
    lat1, lon1 = a
    lat2, lon2 = b
    radius = 6371  # km

    dlat = math.radians(lat2 - lat1)
    dlon = math.radians(lon2 - lon1)
    a = (math.sin(dlat / 2) * math.sin(dlat / 2) +
         math.cos(math.radians(lat1)) * math.cos(math.radians(lat2)) *
         math.sin(dlon / 2) * math.sin(dlon / 2))
    c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
    d = radius * c

    return d

def calculateInterferenceRadius(self, request):
    transmission_power_dBm = request.operationParam.maxEirp
    # Calculate the interference radius based on the request
    # PL(d)=15+36log10d (3.5 GHz)
    path_loss = transmission_power_dBm - self.interferenceThresholddBm
    log10_distance = (path_loss - 15) / 36
    distance = 10 ** log10_distance
    return distance