Investigation of Dynamic Spectrum Access Approaches in Broadcasting

By

Boubacar Abdou Tchoussou
North Carolina A&T State University

Frank Lykes Claytor
Wofford College

Mentor: Dr. Carl Dietrich, Virginia Tech

NSF Research Experiences for Undergraduates: Cognitive Communications
Virginia Polytechnic Institute and State University
Bradley Department of Electrical and Computer Engineering

July 2014

Table of Contents

Table of Figures

Table of Tables

Abstract

Cognitive Radio (CR), a novel radio technology intelligent enough to autonomously change its parameters configuration to adapt to its environment, increases the efficient usage of the underutilized radio spectrum. One proposed technique to achieve this goal is Dynamic Spectrum Access (DSA), a technique in which a secondary (unlicensed) user (SU) senses and accesses the idle frequencies of the spectrum band owned by a primary (licensed) user (PU). The term “idle frequencies” refers to parts of the spectrum that are unoccupied at a certain time and location by the primary user. The main focus of our project is to evaluate the performance of the CRs when performing DSA in broadcasting –transmission to multiple receiving radios. Specifically, a C++ program uses open source Liquid-DSP and Cognitive Radio Test System (CRTS) software to test the performance of a CR in different noise scenarios. DSA metrics such as rendezvous time, evacuation time, false alarm, probabilities of detection/misdetection as well as performance metrics such as bit error rate (BER) and packet error rate (PER) are evaluated.

1. Introduction

In the United States, the frequency allocation chart of the radio spectrum is overcrowded and the radio frequency bands are extremely expensive. A 2002 Federal Communication Commission (FCC) Spectrum Policy Task Force report [5] showed that much of the prized (licensed) spectrum may be idle during the day. Indeed, up to 70% of the spectrum may be idle during the day. This report confirmed a common belief shared by many engineers and scientists for more than a decade now: the radio spectrum is underutilized. Cognitive Radio (CR) is proposed as a new paradigm to alleviate the scarcity of the spectrum and increase its efficient utilization. CR is a novel radio technology intelligent enough to autonomously change its parameters configuration to adapt to its environment. It is composed of Software Defined Radio (SDR) and a Cognitive Engine (CE). SDR, a radio that changes its waveforms using software, provides flexibility and reconfigurability to CR. A main goal of CR is to increase the efficient utilization of the radio spectrum [3]. Dynamic Spectrum Access (DSA) is a proposed technique to achieve this goal. In DSA, the secondary (unlicensed) user (SU) senses and access the idle frequencies of the spectrum own by a primary (licensed) user (PU). Idle frequencies are those parts of the spectrum that are unused at a certain time and location by the PU.

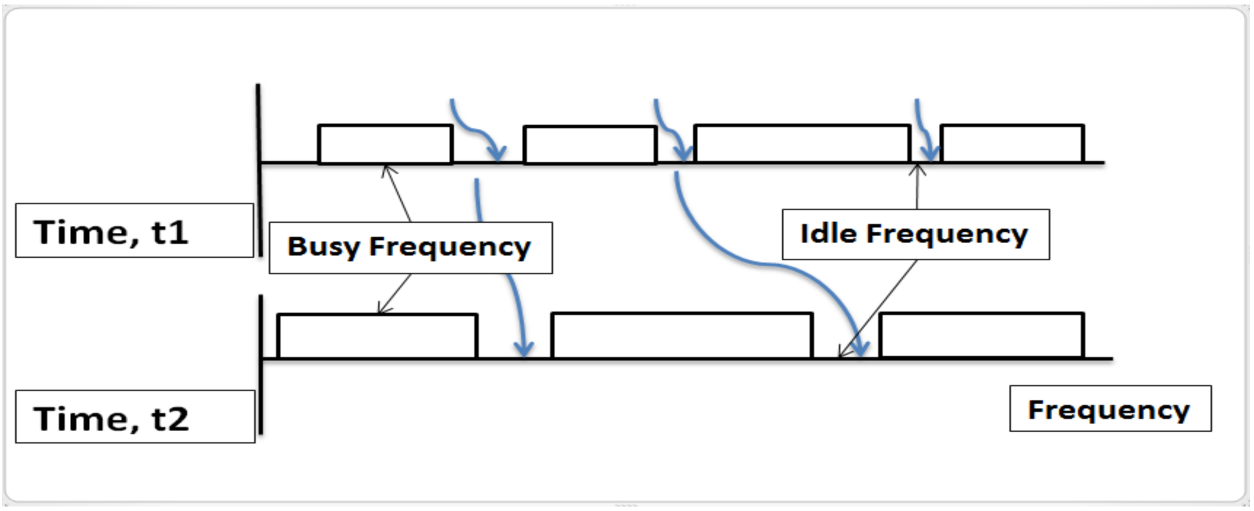


Figure 1: Illustration of idle frequencies

The purpose of this project is to investigate and evaluate the performance of CR when performing DSA in broadcasting cognitive radio networks. Variables such as rendezvous time, evacuation time, false alarm, probabilities of detection/misdetection, and performance metrics such as bit error rate (BER) and packet error rate (PER) are evaluated by a program written in C++. This program uses open source Liquid-DSP and Cognitive Radio Test System (CRTS) software to test the performance of a CR in different noise scenarios such as Additive White Gaussian Noise (AWGN) and Rician fading.

2. Background

Previous work within the following research areas provided critical background for our research project.

2.1. DSA

In contrast to the conventional static spectrum allocation where a part of spectrum band is own and operated exclusively by a licensed user, DSA propose a novel spectrum management approach where the idle frequencies can be dynamically accessed and used by an unlicensed user.

There are three different models of DSA: Dynamic Exclusive Use Model, Open Sharing Model and Hierarchical Access Model [4].

- Dynamic Exclusive Model: the bands are licensed to services at a given time and location for exclusive use (dynamic spectrum allocation) and the licensee is allowed to sell and trade its spectrum band (property rights).
- Open Sharing Model: allow open sharing among peer users.
- Hierarchical Access Model: spectrum underlay approach and spectrum overlay approach.
    - Spectrum underlay approach: the SU can transmit at the same time with the PU but has to satisfy some interference threshold constraint imposed by the PU. The transmission power of the SU has to be below the noise floor of the PU.
    - Spectrum overlay approach: also known as Opportunistic Spectrum Access (OSA), the SU senses and accesses the idle frequencies of the PU. To avoid interference and collision with the PU, the SU must vacates the channel as soon as it detects the presence of the PU.

To access the idle frequencies, the SU performs a full cognitive cycle: spectrum sensing, spectrum analysis, and spectrum decision [7]. The spectrum sensing detects the presence of PUs and determines which portion of the spectrum is idle. The spectrum analysis is then performed to estimate the opportunities. Finally, a spectrum decision is perform to choose the appropriate spectrum band for the transmission.

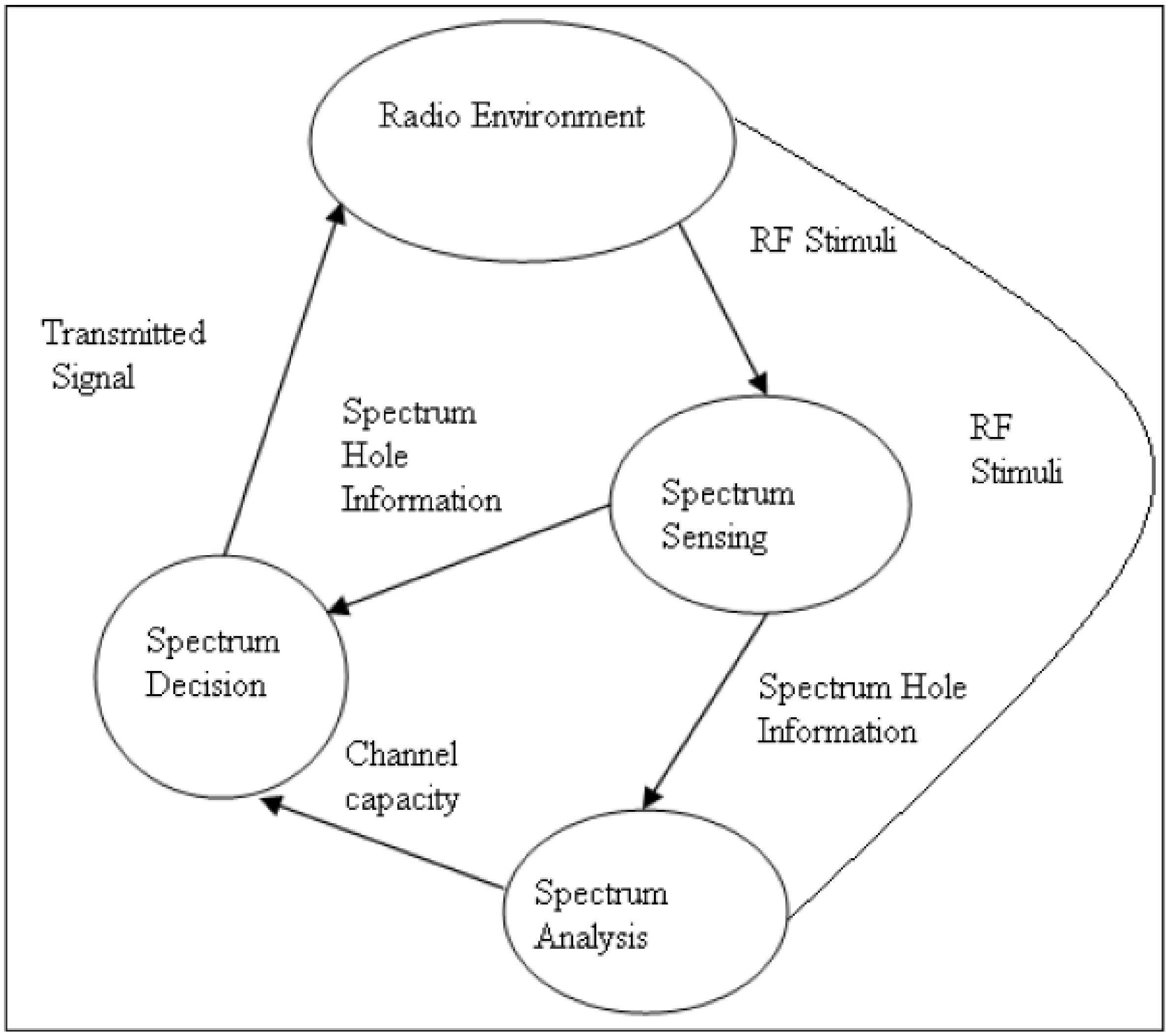


Figure 2: Cognitive cycle [7]

There are five major sensing techniques in DSA: matched-filtering detection, energy detection, waveform-based sensing, spectral correlation (cyclostationarity), and radio identification-based sensing [8]. Matched-filtering requires the knowledge of the PU signal parameters such as bandwidth, modulation type, operation frequency, etc. The radio identification-based sensing identifies the PU transmission technologies (e.g. Bluetooth) and uses this information to obtain the spectrum characteristics. The waveform-based sensing is applicable when the signal patterns of the PU are known. The cyclostationarity exploits the periodicity or the statistics (e.g. mean) of the signal to detect the PU. The energy detection is used to detect the energy of the primary signal. In this technique, the power of the PU is compared to a threshold –chosen value slightly greater than the noise floor (power when there is no transmission).

There are two major hypothesis test approaches applied in sensing: 2-state hypothesis test and 3-state hypothesis test [9]. In the 2-state hypothesis test approach (most used), the PU channel is identified as being busy or vacant. The channel is busy when the power of the PU is greater than the threshold and vacant otherwise. The 3-state hypothesis test adds an intermediary state; underutilize state, between busy and vacant. In this approach, the power of the PU is compared to two thresholds. The channel is underutilized when it is between the lower threshold and the upper threshold.

2.2. CRTS

Cognitive Radio Test System (CRTS) is a system developed at Virginia Tech to evaluate the performance of multiple cognitive radios under a variety of stressful environments [2]. The hardware of the CRTS is composed of a transmitter, a noise/interference source, a receiver and a feedback link. The transmission between the transmitter and the receiver is performed by air while the feedback from the receiver to the transmitter is performed by wire using TCP/IP network as shown in figure 2. The transmitter changes its operating modes in response to the feedback received from the receiver via a TCP connection. The software, written in C++, has two configuration files: cognitive engine configuration files and scenarios configuration files. The first contains the characteristics of each cognitive engine and allow the user to specify the cognitive engine parameters (modulation scheme, transmit power, etc.) without messing with the code. The second contains the scenarios (AWGN, Rician fading, etc.) in which the cognitive engine will be tested.

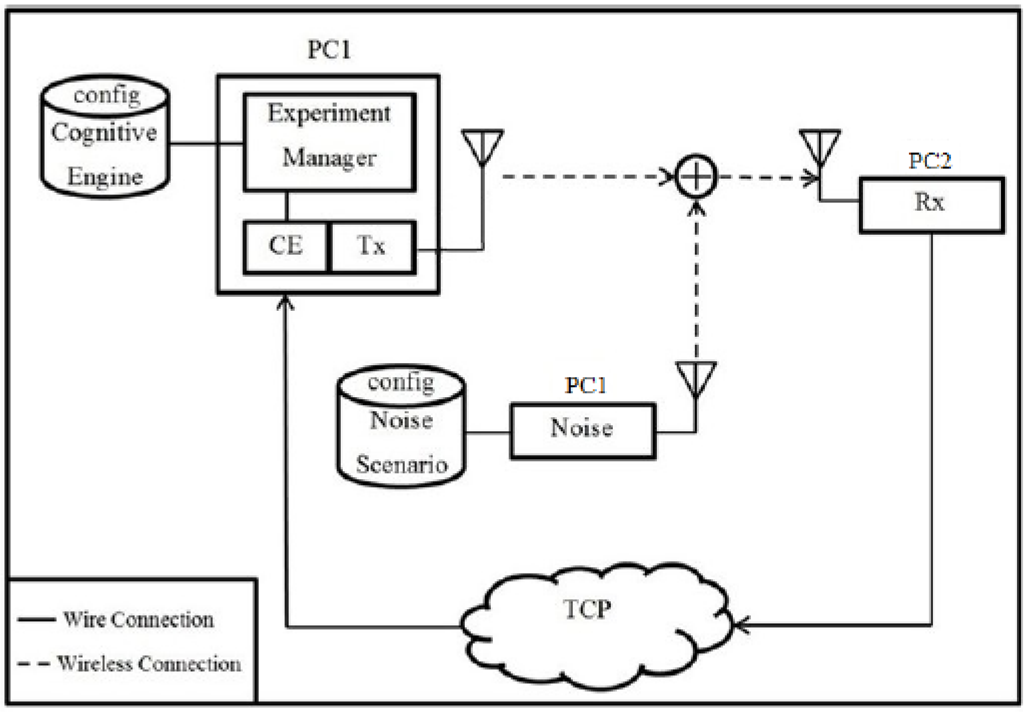


Figure 3: Diagram of CRTS [2]

2.3.Broadcasting

In broadcasting, a transmitter transmits to two or more receivers simultaneously. There are four broadcasting algorithms:

- **Flooding**: Each node re-transmits a received message to all of its neighbors
- **Optimal Algorithm**: Only transmits to unvisited nodes, doesn't consider collisions between two broadcasts
- **Centralized Algorithm**: Schedules nodes with shared neighbors to transmit at different times
- **Distributed Algorithm**: Each node has an unique ID and knows the ID's of nearby nodes

2.4. Cognitive Radio Metrics

The CR metrics can be evaluated at the node, network, and application levels. Node-level and network-level performance metrics can be evaluated from four domains: cognitive functionality, overall performance, complexity, and technical maturity [6].

- Cognitive functionality: situation awareness (SA), and adaptation, reasoning, decision making, planning, and learning capabilities
- Overall performance: spectrum utilization, power efficiency, network reliability, network security
- Complexity: signal processing power requirement, implementation costs, and memory footprint
- Technical maturity: maturity of key technologies (SDR, Analog-to-Digital converters, Artificial Intelligence, policy conformance enforcement, etc.)

3. Methodology

3.1. CRTS

CRTS uses Liquid-DSP to simulate transmitters, noise sources, and receivers. Liquid-DSP allows interaction with the Universal Software Radio Peripheral (USRP) by creating orthogonal frequency-division multiplexing transceiver objects (ofdmtxrx) that can both receive and transmit Liquid frames. Each USRP is hosted on a separate computer and can be used to simulate different kinds of cognitive radios and engines. When CRTS is combined with the Cognitive Radio Network Testbed (CORNET), it can simulate a network of cognitive radios.

CRTS uses TCP links to send feedback between different nodes to find the error rates and other metrics related to each transmitted frame. When testing DSA algorithms error feedback is still received but additional TCP messages are also sent. The receivers will send feedback to the transmitters. The transmitters will average the feedback together and send them to the controller.

The transmitters will also tell the controller when they turn on and off and when the test is over. When each node in CRTS is initialized one of the command line arguments must include the address of the node above them in the TCP hierarchy.

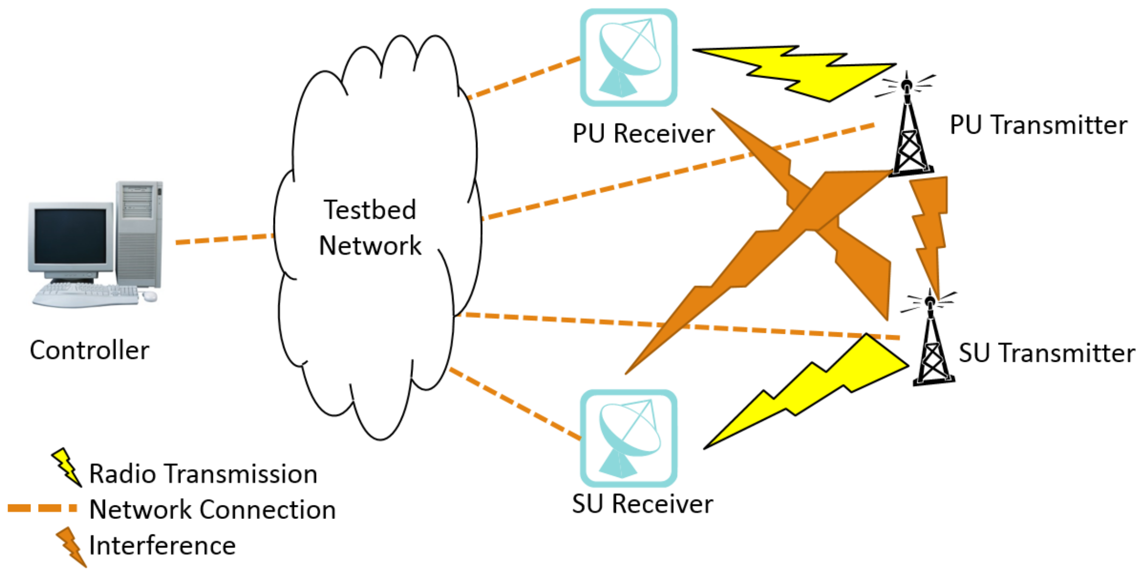


Figure 4: DSA testing diagram

Files		**Descriptions**
crts.cpp		Main C++ program
ceconfigs		CE configuration files
1	ce1.txt	Default CE
2	primary.txt	CE of PU; set parameters such as frequency, gain, adaptation, bandwidth, etc.
3	secondary.txt	CE of PU; set parameters such as frequency, bandwidth, gain, BER threshold, etc.
4	userEngine.txt	CE that can be modified while the program is running
scconfigs		Scenarios configuration files
1	AWGN.txt	Additive White Gaussian Noise (AWGN) scenario: enable the AWGN and choose the SNR
2	AWGNAndRicianFading.txt	AWGN + Rician Fading: enable AWGN and Rician fading; set Rician fading factor and the max Doppler frequency values
3	Interferer_CW.txt	Continuous Waves interference scenario
master_dsa_file.txt		File containing DSA parameters such as sensing type and FFT parameters
master_scenario_file.txt		Allow to define and choose the number of scenarios
master_cogengine_file.txt		Allow to select the number of CE and specify which CE will be tested

Table 1: Description of CRTS files

3.2. CORNET

The Cognitive Radio Network Testbed (CORNET) is an open-source platform developed at Virginia Tech for education and research in CR. Its 48 SDR nodes, remotely accessible, are deployed within a four-story building on Virginia's Tech main campus [10]. In this project, CORNET is used to test DSA by hosting different parts of the DSA network on different nodes. A node can be set to be a primary transmitter/receiver, a secondary transmitter/receiver, an energy detection receiver, or a controller. It is also used to visualize the signal transmitted in 3D or waterfall. The figure below is the CORNET nodes map on the four floors. The green nodes are those that are free and can be used. The red indicate nodes that are not in working order. For more information on how to use CORNET, refer to [10].

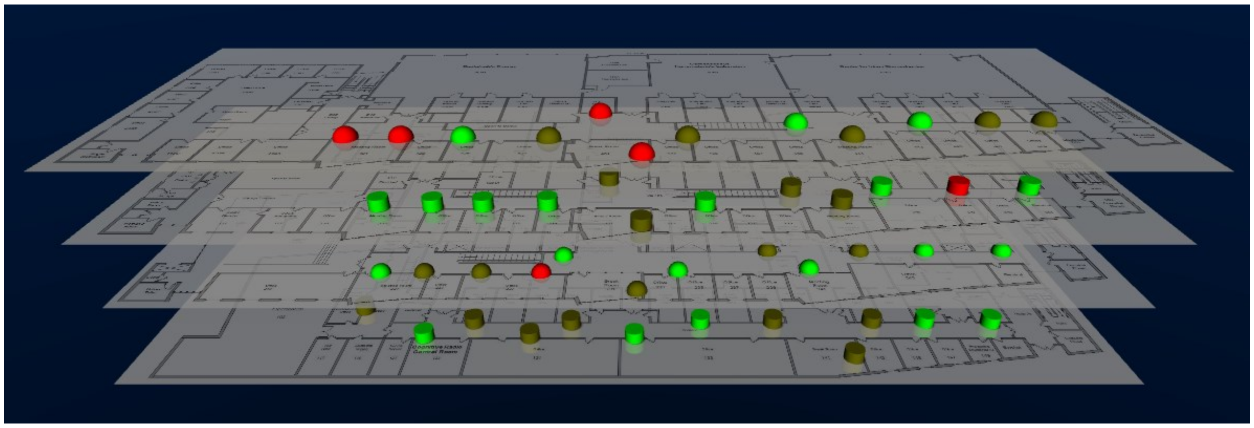
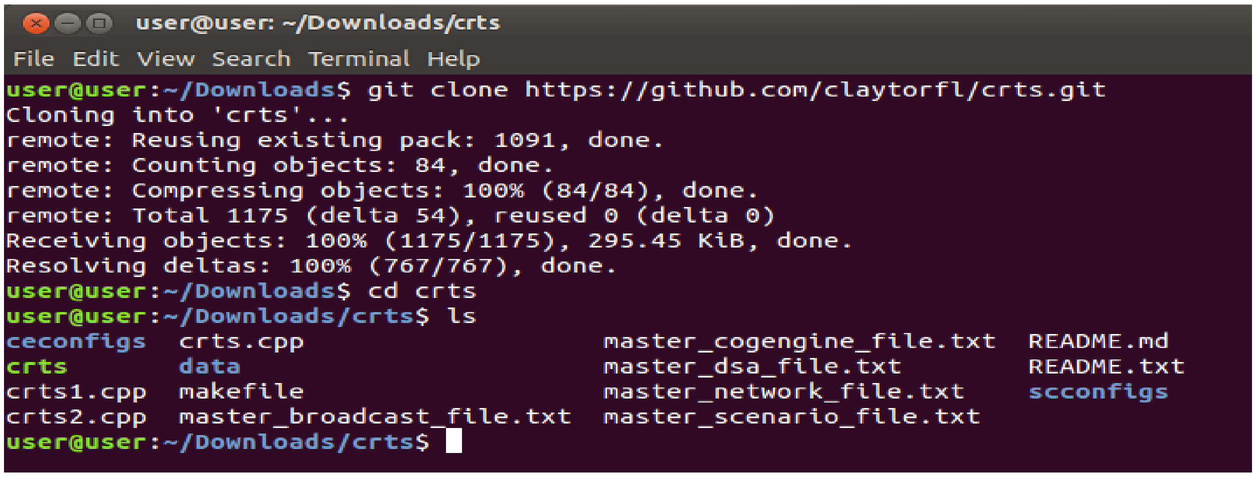


Figure 5: CORNET nodes and floors map [10]

3.3. DSA

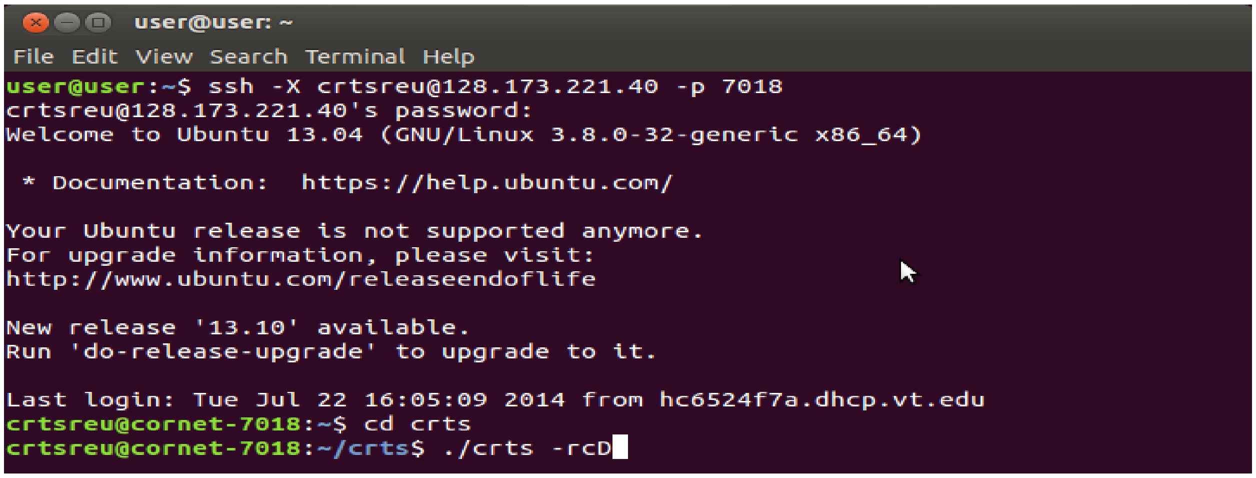
3.3.1. Controller

A controller collects metrics and error feedback. Evacuation time and rendezvous times are important metrics to measure when testing a DSA system. But if the primary and secondary users are running on separate computers their clocks might not be synchronized and the evacuation and rendezvous times they calculate might be distorted. The controller node corrects this error by doing all of the timings in the same place. When a primary or secondary transmitter is initialized on a different node it will link up to the controller so it can use TCP to send feedback. Whenever a transmitter turns on or off it will send a message to the controller. The controller will then record the time and use its collection of time data to find the evacuation time, rendezvous time, probability of detection, and other useful metrics. The transmitters will also send the averaged error feedback from their receivers for each frame they transmit. The controller uses all of this feedback to calculate the average total feedback, average primary feedback, average secondary feedback, and the average primary and secondary feedback when both of the transmitters are on and interfering with each other.



```
user@user:~/Downloads$ git clone https://github.com/claytorfl/crts.git
Cloning into 'crts'...
remote: Reusing existing pack: 1091, done.
remote: Counting objects: 84, done.
remote: Compressing objects: 100% (84/84), done.
remote: Total 1175 (delta 54), reused 0 (delta 0)
Receiving objects: 100% (1175/1175), 295.45 KiB, done.
Resolving deltas: 100% (767/767), done.
user@user:~/Downloads$ cd crts
user@user:~/Downloads/crts$ ls
ceconfigs  crts.cpp                   master_cogengine_file.txt  README.md
crts       data                       master_dsa_file.txt        README.txt
crts1.cpp  makefile                   master_network_file.txt    scconfigs
crts2.cpp  master_broadcast_file.txt  master_scenario_file.txt
user@user:~/Downloads/crts$ 
```

Figure 6: Command line to download files



```
user@user:~$ ssh -X crtsreu@128.173.221.40 -p 7018
crtsreu@128.173.221.40's password:
Welcome to Ubuntu 13.04 (GNU/Linux 3.8.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife

New release '13.10' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Jul 22 16:05:09 2014 from hc6524f7a.dhcp.vt.edu
crtsreu@cornet-7018:~$ cd crts
crtsreu@cornet-7018:~/crts$ ./crts -rcD
```

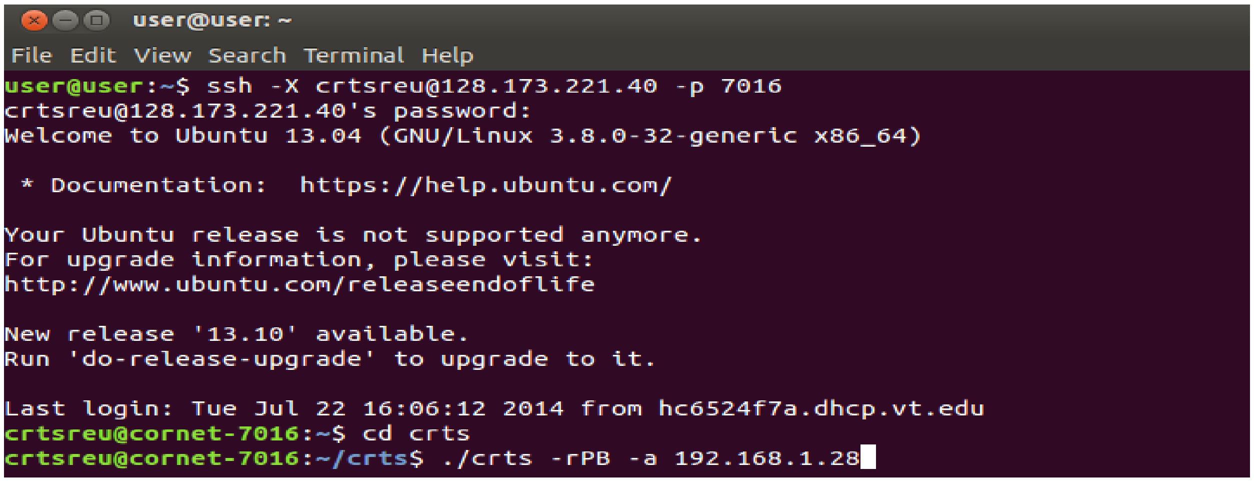
Figure 7: The command line arguments to initialize the Controller

3.3.2. PU Transmitter

The PU can transmit at any time since it's the licensed user. It is the SU's responsibility to accurately sense and access the spectrum without interfering with the PU. The DSA CRTS primary transmitter reflects this. It has no complicated spectrum sensing algorithms. It only transmits and rests for random intervals. After initialization, the primary transmitter will begin transmitting for a random amount of time then turn off and rest for a random amount of time. The primary transmitter transmitting and resting once constitutes one cycle. The primary transmitter will continue with the cycle of transmitting and resting until it completes the number of cycles indicated in the configuration file master_dsa_file.txt.

While it transmits, the primary transmitter will use a TCP link to receive feedback from the primary receivers about the error rate and other statistics of its transmissions. The transmitter will take the feedback related to each frame transmitted and average them together before using a different TCP link to send the averaged feedback to the controller.

The primary transmitter's transmitted frames always have a header of all 1's. This allows the receivers to easily identify whether a frame came from a primary user or a secondary user so they know what kind of feedback to send. The header serves as a kind of access code, preventing a secondary receiver from sending feedback on a primary transmission and vice versa.



```
user@user:~$ ssh -X crtsreu@128.173.221.40 -p 7016
crtsreu@128.173.221.40's password:
Welcome to Ubuntu 13.04 (GNU/Linux 3.8.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Your Ubuntu release is not supported anymore.
For upgrade information, please visit:
http://www.ubuntu.com/releaseendoflife

New release '13.10' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue Jul 22 16:06:12 2014 from hc6524f7a.dhcp.vt.edu
crtsreu@cornet-7016:~$ cd crts
crtsreu@cornet-7016:~/crts$ ./crts -rPB -a 192.168.1.28
```

Figure 8: The command line arguments to initialize the primary transmitter (including address of the Controller)

3.3.3. PU Receiver

The primary receiver job is to wait and receive primary transmissions. It then analyzes these transmissions before sending back error feedback using a TCP link. CRTS uses Liquid-DSP, and with Liquid-DSP every receiver has a callback function. Whenever a frame is received the callback function is called and the frame is analyzed. The primary receiver's callback function allows it to identify primary transmissions by the header as well as find various statistics about the frame.

3.3.4. Spectrum Sensing

In this project, the spectrum overlay approach under the hierarchical access model is used to perform the DSA. Since only one PU is used, the spectrum analysis and spectrum decision of the cognitive cycle are not necessary. Among the sensing techniques discussed above, two are implemented. The selection of the preferred sensing technique is made in the master_dsa_file.txt. The first technique is header matching. It is based on the matched-filtering detection in that it requires a great familiarity with the structure of the primary transmissions. The secondary transmitter transmits frames with headers of all 0's, while the primary transmitter frames have headers of all 1's. The secondary transmitter can receive Liquid frames and call a callback function to analyze the frame and find where they came from. When a frame is received that has a header of all 1's then the secondary transmitter knows that the primary user is transmitting.

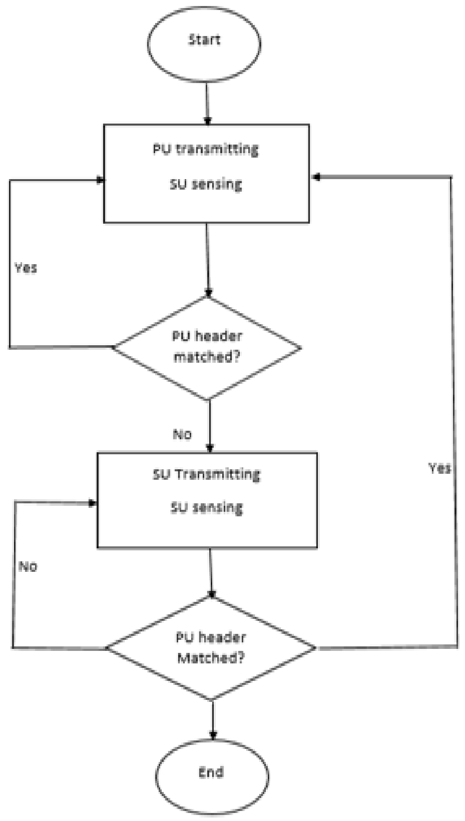


Figure 9: Header matching sensing flowchart

The second technique is energy detection. The secondary transmitter will use its receiver to find the power of the channel it is on by using the FFTW3 library. The current algorithm for energy detection is receiving samples from the USRP into an input buffer, using FFT to transform the input and put them into the output buffer, normalize the output buffer, and add together the first 20 entries. The first entries are the one closest to the central frequency, thus measuring a range of measurements from the center of the FFT will find the power of the central channel being measured. When only it is transmitting the power is at one level, but when the primary user starts transmitting the power will go up. When the secondary transmitter is first initialized it will run a function to find the power of the channel when the PU is not transmitting (noise floor). It will then add to the noise floor to create a threshold. If the power of channel rises above the threshold then the primary user is transmitting. The number added to the noise floor and the power of the secondary user will need to be adjusted so that the primary user is properly detected while the secondary user can transmit above the noise floor but with a low enough power to not pass the threshold.

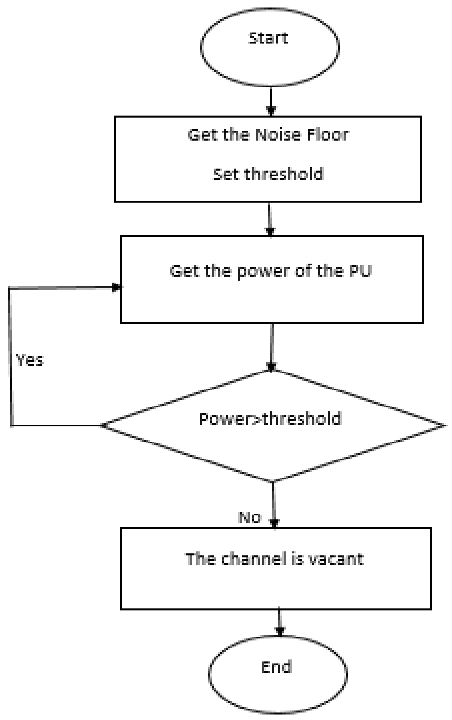
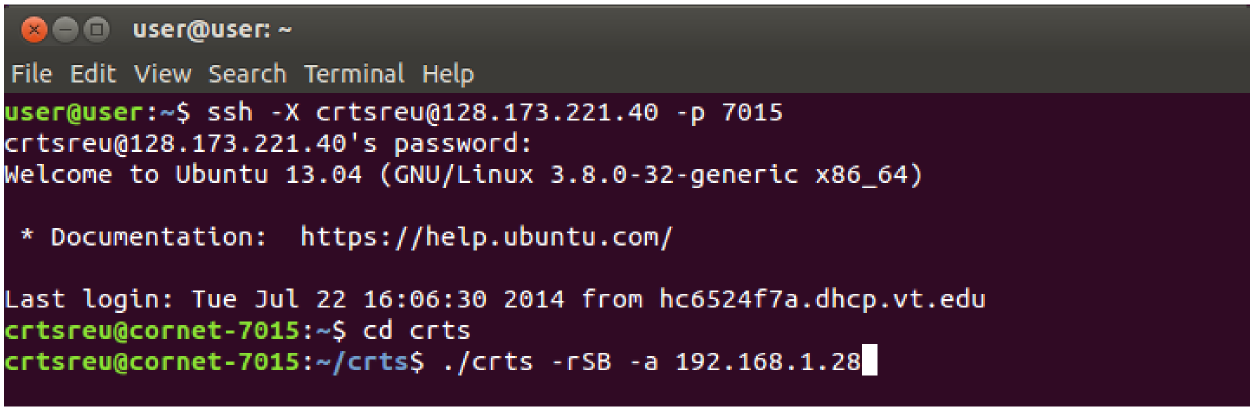


Figure 10: Energy detection sensing flowchart

3.3.5. SU Transmitter

The secondary transmitter is the transmitter that must use spectrum sensing techniques to check for the presence of primary user transmissions to make sure the owner of the spectrum suffers as little interference as possible. The secondary user has two states: transmitting and sensing. When the secondary transmitter is transmitting then that means it thinks the spectrum is clear and free to use. The transmitter will transmit for a burst of specified length and then stop to sense the spectrum to see if conditions have changed. If no primary user is detected the secondary transmitter transmits for another burst before checking again. If a primary user is detected the transmitter stops transmitting and switches to sensing mode. In sensing mode the transmitter will transmit nothing and continually sense the spectrum environment. Once it hasn't detected the primary user for a long enough time it assumes that it has shut off and it switches back to transmitting mode. Whenever it switches between modes it will send a TCP message to the controller so that it can calculate timing metrics. The secondary transmitter will also send averaged feedback from its receivers to the controller, just like the primary transmitter. To initialize the secondary transmitter on a node the command line argument –rS is used as well as the address of the controller.



```
user@user:~$ ssh -X crtsreu@128.173.221.40 -p 7015
crtsreu@128.173.221.40's password:
Welcome to Ubuntu 13.04 (GNU/Linux 3.8.0-32-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

Last login: Tue Jul 22 16:06:30 2014 from hc6524f7a.dhcp.vt.edu
crtsreu@cornet-7015:~$ cd crts
crtsreu@cornet-7015:~/crts$ ./crts -rSB -a 192.168.1.28
```

Figure 11: The command line arguments to initialize the secondary transmitter (including address of the Controller)

3.3.6. SU Receiver

The secondary receiver behaves just like the primary receiver. It does nothing but wait to receive secondary transmissions and use its callback function to create error feedback. It will also send a TCP message to the secondary transmitter if it detects a primary user. The secondary transmitter will only pay attention to that if it is set to a DSA strategy that relegates the sensing to the receivers. Due to limitations of the USRPs and CORNET secondary receivers cannot receive and interpret Liquid frames and perform energy detection at the same time without causing overflows. Thus a different kind of receiver, energy detector, is initialized for when the secondary transmitter wants its receivers to spectrum sense using energy detection.

3.3.7. Energy Detecting Receiver

The energy detector is a secondary receiver that only detects if a primary user is transmitting. When first initialized it starts by finding a noise floor and threshold. Then it starts sensing and whenever it senses a power above its threshold it sends a warning TCP message to the secondary transmitter. The receiver of the USRP is occupied by energy detection so the energy detector is unable to receive Liquid frames or give feedback on Liquid frames.

3.3.8. Broadcasting

In a broadcasting situation one secondary transmitter could have many receivers, giving it a wide area that it can scan the spectrum of. There might be a PU that is far away from the SU transmitter but close to one of its receivers. This receiver can sense the PU spectrum and relay the information to its PU transmitter. The receivers can both use header matching and energy detection. Once a secondary receiver senses a primary user it will send a warning message through TCP. In these situations the secondary transmitter will be doing no sensing and will only change states when prompted by the receivers or if it hasn’t received a warning message in a set amount of time.

4. Results

The figures and tables below illustrates the results of our simulations. Figures 13 and 14 are the waterfall graph of the signals of both PU and SU transmitting on the same frequency of 460 MHz. In tables 2 and 3, some of the computed metrics are presented. For full results with all the metrics refer to the DSA CRTS Github repository (given in the appendix).

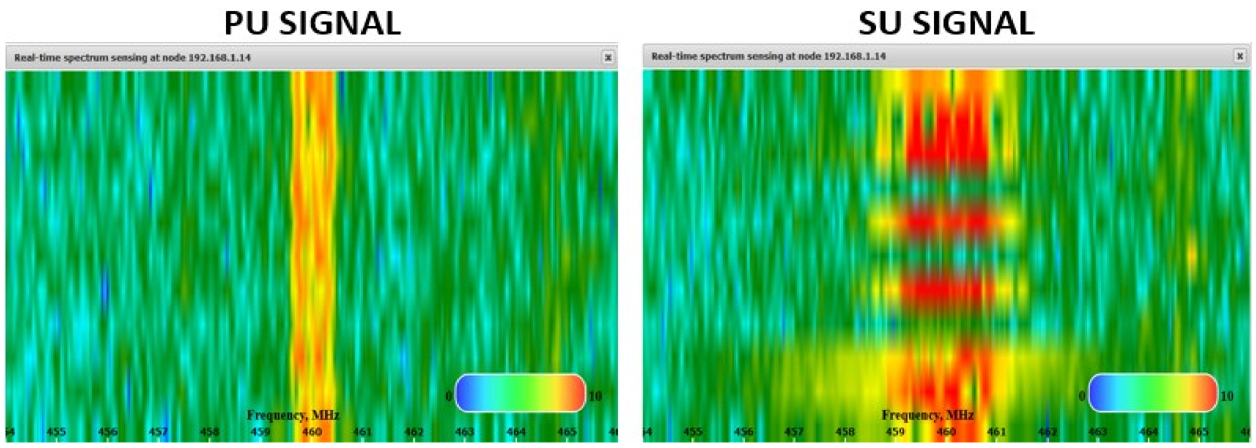


Figure 13: PU and SU signals using waterfall

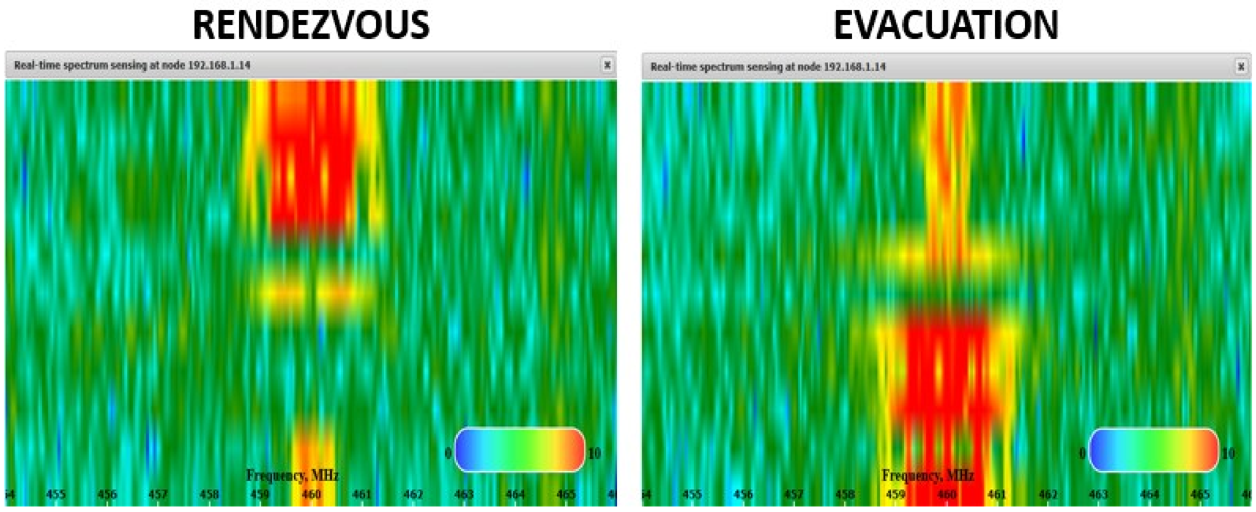
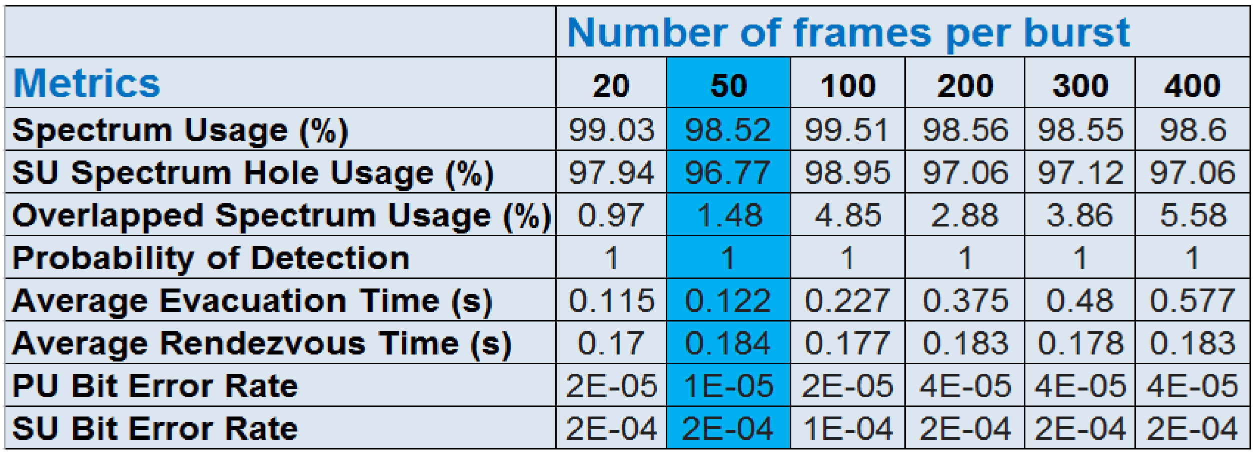
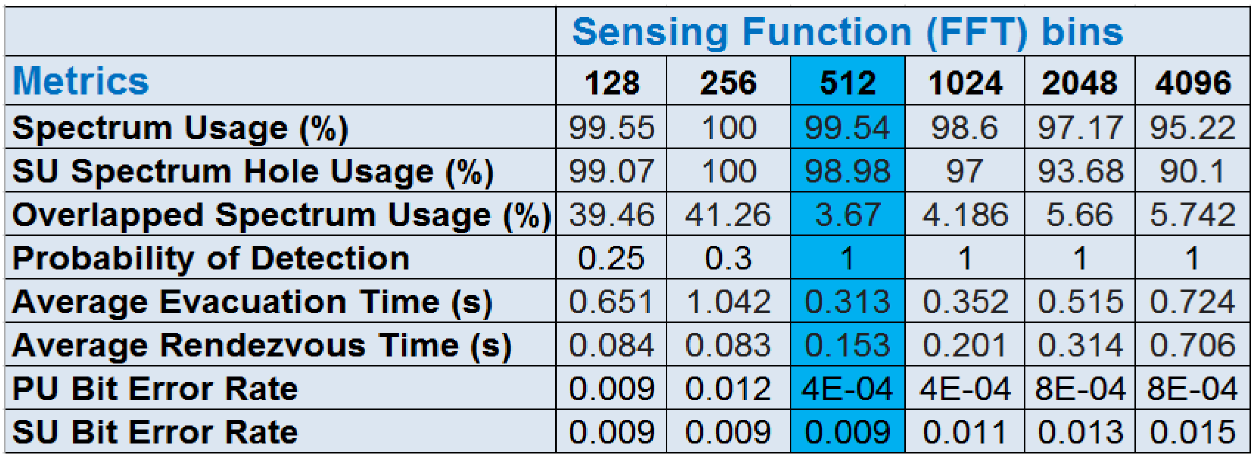


Figure 14: Rendezvous and Evacuation of the SU



	Number of frames per burst					
Metrics	**20**	**50**	**100**	**200**	**300**	**400**
Spectrum Usage (%)	99.03	98.52	99.51	98.56	98.55	98.6
SU Spectrum Hole Usage (%)	97.94	96.77	98.95	97.06	97.12	97.06
Overlapped Spectrum Usage (%)	0.97	1.48	4.85	2.88	3.86	5.58
Probability of Detection	1	1	1	1	1	1
Average Evacuation Time (s)	0.115	0.122	0.227	0.375	0.48	0.577
Average Rendezvous Time (s)	0.17	0.184	0.177	0.183	0.178	0.183
PU Bit Error Rate	2E-05	1E-05	2E-05	4E-05	4E-05	4E-05
SU Bit Error Rate	2E-04	2E-04	1E-04	2E-04	2E-04	2E-04

Table 2: Metrics Evaluation based on the number of frames per burst (512 FFT bins)



	Sensing Function (FFT) bins					
Metrics	**128**	**256**	**512**	**1024**	**2048**	**4096**
Spectrum Usage (%)	99.55	100	99.54	98.6	97.17	95.22
SU Spectrum Hole Usage (%)	99.07	100	98.98	97	93.68	90.1
Overlapped Spectrum Usage (%)	39.46	41.26	3.67	4.186	5.66	5.742
Probability of Detection	0.25	0.3	1	1	1	1
Average Evacuation Time (s)	0.651	1.042	0.313	0.352	0.515	0.724
Average Rendezvous Time (s)	0.084	0.083	0.153	0.201	0.314	0.706
PU Bit Error Rate	0.009	0.012	4E-04	4E-04	8E-04	8E-04
SU Bit Error Rate	0.009	0.009	0.009	0.011	0.013	0.015

Table 3: Metrics Evaluation based on the number of FFT bins (200frames/burst)

5. Discussion

The left part of Figure 13 illustrates the signal of the primary transmitter on a 460 MHz frequency with a bandwidth of 1 MHz and a gain of 20dB. On the right is the secondary transmitter transmitting at different time on the same frequency of 460 MHz. It has a bandwidth of 2 MHz and a gain of 2 MHz. The PU transmits on a narrow bandwidth with much more power than the SU.

Figure 14 illustrates the transition between the PU and the SU. On the left, the PU stopped transmitting and after some time (rendezvous time) the SU accesses the channel and starts transmitting. On the right, the SU detect the presence of the PU and evacuate the channel. The time it takes to the SU to vacate the channel after detecting the presence of the PU is the evacuation time.

Table 2 shows some metrics evaluated with different SU frames. The simulation was performed using the sensing function with 512 FFT bins. The blue column indicates the best choice of number of frames per burst (50) when using 512 FFT bins.

In table 3, the number of frames per burst is set to 200. The simulations are then performed with different numbers of FFT bins, changing the speed and accuracy of the energy detection. The blue column indicates the optimal choice of FFT bins (512) that gives higher spectrum usage and lower evacuation time.

6. Conclusion

Table 2 shows that changing the length of the payload of the secondary user can have adverse effects on the primary transmissions. If the secondary user transmits more frame at one time it may slightly boost the overall spectrum usage and utilize the spectrum holes a bit more efficiently, but it also causes an increase in evacuation time and more overlap with the primary user. Since the secondary user must transmit its entire payload before evacuating it can't evacuate as fast, and that can cause collisions with secondary and primary users that can increase the bit error rate of both transmissions. The table indicates that the best option is to use a small payload size, like 50 frames, to avoid unnecessary interference with the primary user.

Table 3 shows how changing the number of FFT bins used can change the performance of the secondary user. The entries for the large number of bins such as 4096 and 2048 show that using many bins to compute your FFT can slow down the spectrum sensing abilities of the secondary user and increase evacuation and rendezvous times. As the number of bins decreases the evacuation and rendezvous times also decrease. But after a certain point decreasing the number of FFT bins can cause a lack of accuracy and precision. The entries for 128 and 256 bins show that using too few FFT bins can make the primary user difficult to detect and drop the probability of detection, causing large amounts of overlap. The table indicates that the optimal FFT size is somewhere in between overly large and slow and small and inaccurate. 512 bins had the best metrics for that set of tests.

Tables 2 and 3 illustrate the power of a cognitive radio test system that can test DSA metrics. The original CRTS could already find the bit error rate, received signal strength indication, and many other metrics. It used these metrics to test how well a cognitive radio could intelligently adapt to different scenarios. Adding the DSA capabilities allows it to expand and test more of the cognitive processes that a cognitive radio must do. Once CRTS is updated it may even be able to test adaptability and DSA at the same time, greatly streamlining the process of testing different cognitive engines and DSA strategies.

7. Future Work

There are many ways in which this project can be extended. Some examples are:

- Application to multiple PU and/or SU
- Include adaption capabilities
- Evaluate the metrics in many more scenarios such as Signal to Noise Ratio (SNR), modulation scheme, etc.
- Implement new sensing techniques

8. Appendix

8.1. Master DSA file

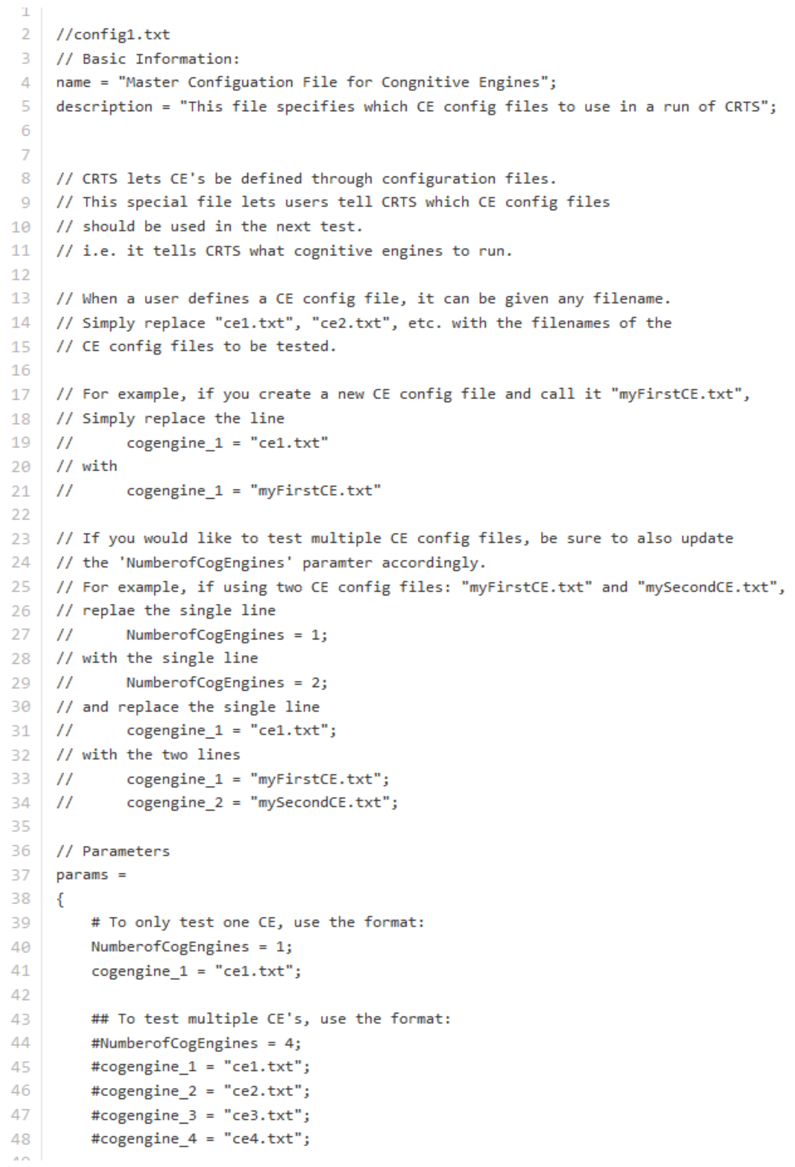
```
//master_dsa_file.txt

params=
{

//The total number of times the primary user will
//transmit and rest. After the primary user has finished
//all of its cycles it will send a message to the controller
//and the test will end
totalcycles = 20

//The DSA technique being used to detect the primary user
//"m" = Header Matching
//"e" = Energy Detection with FFT
//"r" = Receivers do the sensing and communicate with transmitter using TCP
//"t" = Energy Detection using Receivers to sense and transmit simultaneously
detectiontype = "t";
};

//Parameters used by the FFT function during Energy Detection
fft=
{

//Bandwidth scanned by the FFT function
bandwidth = 1000000.0;

//Bandwidth of each channel in the full bandwidth (doesn't do anything right now)
channelbandwidth = 500000.0;

//Sampling rate of the FFT usrp
rate = 195312;

//Number of bins measured by FFT
numbins = 128;

//Antennae of the FFT usrp
antennae = "RX2";

//Float added to the noise floor to create a threshold
noiseadder = 0.0;

noisemult = 1.2;

noisefloormeasuredbins = 3;

noisefloorrepeat = 30;

noisefloortestnumber = 1;

//Number of bins to be added together to find the power
//of the center frequency
measuredbins = 3;

//Number of times the secondary user will run fft function
//before reaching a decision
testnumber = 1;

//Number of times the fft function will use FFT before averaging together the samples
repeat = 30;
//If debug=1 every time the fft function runs it will print out the values of the boolean cantransmit,
//which determines if the secondary user can transmit, the measured power of the center frequency, and
//the threshold value the measured power is being compared to
//If debug=0 nothing is printed out
debug = 1;

//The receiver gain of the FFT USRP
gain=90.0;
};


//Parameters of the primary user
PU=
{

//Base burst time (float) How long the primary user transmits before stopping
bursttime = 5.0;

//Every cycle the base burst time will have a random int between 0 and burstrandom added to it
//to find the burst time for that cycle
burstrandom = 4;

//Base rest time (float) How long the primary user is inactive before it starts a new cycles
//and begins transmitting again
resttime = 8.0;

//Every cycle the base rest time will have a random int between 0 and burstrandom added to it
//to find the rest time for that cycle
restrandom = 5;

//The file name in ceconfigs that corresponds to the primary user's cognitive engine
ce = "primary.txt";

//The number of frames the primary user will send in a row before stopping and checking if it
//has exceeded its burst time
uninterruptedframes = 1;

//adapt = 1 if it uses postTxTasks to adapt based on receiver feedback. adapt = 0 if the primary user doesn't adapt
adapt = 0;

//usescenario = 1 if a scenario is applied to the signal before it is transmitted
//if usescenario = 0 no scenario is applied
usescenario = 0;

//The scenario file name in scconfigs that will be used is usescenario=1
scenario = "sc1.txt"
};
```

```
//Parameters for the secondary user
SU=
{

//How many seconds the secondary user will stay in sensing mode without detecting a primary transmission
//Only used with Header Matching
scantime = 2.0;

//Filename for file in ceconfigs that corresponds to the cognitive engine for the secondary user
ce = "secondary.txt";

//How many frames the secondary user will send in a row before stopping to check for the primary user
uninterruptedframes = 50;

//adapt = 1 if it uses postTxTasks to adapt based on receiver feedback. adapt = 0 if the secondary user doesn't adapt
adapt = 0;

//usescenario = 1 if a scenario is applied to the signal before it is transmitted
//if usescenario = 0 no scenario is applied
usescenario = 0;

//The scenario file name in scconfigs that will be used is usescenario=1
scenario = "AWGN.txt"
};
```

8.2. Master Scenario file

```
//config1.txt
// Basic Information:
filename = "Master Configuation File for Scenarios";

// Parameters
params =
{
NumberofScenarios = 3;
scenario_1 = "AWGN.txt";
scenario_2 = "AWGNAndRiceanFading.txt";
scenario_3 = "Interferer_CW.txt";
};
```

8.3. Master Cogengine file



```

//config1.txt
// Basic Information:
name = "Master Configuation File for Congnitive Engines";
description = "This file specifies which CE config files to use in a run of CRTS";


// CRTS lets CE's be defined through configuration files.
// This special file lets users tell CRTS which CE config files
// should be used in the next test.
// i.e. it tells CRTS what cognitive engines to run.

// When a user defines a CE config file, it can be given any filename.
// Simply replace "ce1.txt", "ce2.txt", etc. with the filenames of the
// CE config files to be tested.

// For example, if you create a new CE config file and call it "myFirstCE.txt",
// Simply replace the line
//      cogengine_1 = "ce1.txt"
// with
//      cogengine_1 = "myFirstCE.txt"

// If you would like to test multiple CE config files, be sure to also update
// the 'NumberofCogEngines' paramter accordingly.
// For example, if using two CE config files: "myFirstCE.txt" and "mySecondCE.txt",
// replae the single line
//      NumberofCogEngines = 1;
// with the single line
//      NumberofCogEngines = 2;
// and replace the single line
//      cogengine_1 = "ce1.txt";
// with the two lines
//      cogengine_1 = "myFirstCE.txt";
//      cogengine_2 = "mySecondCE.txt";

// Parameters
params =
{
    # To only test one CE, use the format:
    NumberofCogEngines = 1;
    cogengine_1 = "ce1.txt";

    ## To test multiple CE's, use the format:
    #NumberofCogEngines = 4;
    #cogengine_1 = "ce1.txt";
    #cogengine_2 = "ce2.txt";
    #cogengine_3 = "ce3.txt";
    #cogengine_4 = "ce4.txt";
```

8.4. Others

For the full content of the project including the C++ file go to the DSA CRTS Github repository:

https://github.com/dsacrts/dsacrts

9. Acknowledgements

We would like to acknowledge all the cognitive communications REU faculty and staff and Freddie Romano. The Cognitive Communications REU site at Virginia Tech is supported by NSF award 1156503. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

10. References

[1]. C.B. Dietrich, E.W. Wolfe, and G. Vanhoy, “Cognitive Radio Testing using Psychometric Approaches,”SDR11/WinComm, 2011.

[2]. J. Choi, S. Guha, F. M. Romano, T. L. Smith-Jackson, C. B. Dietrich, “A Flexible and Extensible Cognitive Radio Test System (CRTS)”, SDR-WinnComm, 2014.

[3]. S. Haykin, “Cognitive Radio: brain-empowered wireless communications,” IEEE journal on Selected Areas in Communications, vol. 23, no. 2, pp. 201–220, 2005.

[4]. Q. Zhao and B. M. Sadler, "A Survey of Dynamic Spectrum Access," IEEE Signal Processing Magazine, vol. 24, no. 3, pp. 79-89, 2007.

[5]. FCC Spectrum Policy Task Force, “Report of the spectrum efficiency working group,” Nov. 2002. [Online]. Available: http://www.fcc.gov/sptf/reports.html

[6]. Y. Zhao, S. Mao, J. O. Neel, J. H. Reed, “Performance Evaluation of Cognitive Radios: Metrics, Utility Function, and Methodology,” Proceedings of the IEEE, Vol. 97, No. 4, 2009

[7]. A. Garhwal, P. P. Bhattacharya, “A survey on Dynamic Spectrum Access Techniques for Cognitive Radio,” International Journal of Next-Generation Networks (IJNGN) Vol.3, No.4, 2011

[8]. T. Yucek and H. Arslan, “A survey on Dynamic Spectrum Access Algorithms for Cognitive Radio Applications,” IEEE Communications Surveys and Tutorials, vol. 11, no. 1, 2009

[9]. S. A. Hanna, “A 3-State Hypothesis Test Model for Cognitive Radio Systems,” IEEE International Symposium on Dynamic Spectrum Access Networks (DYSPAN), 2014

[10]. Cognitive Radio Network Testbed (CORNET). http://cornet.wireless.vt.edu/