5-1-2012

# A Software Defined Radio Testbed for Research in Dynamic Spectrum Access

David A. Clendenen
*Indiana University - Purdue University Fort Wayne*

Graduate School ETD Form 9
(Revised 12/07)

# PURDUE UNIVERSITY
## GRADUATE SCHOOL
## Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By David A. Clendenen

Entitled
A Software Defined Radio Testbed for Research in Dynamic Spectrum Access

For the degree of Master of Science in Engineering

Is approved by the final examining committee:

Todor Cooklev
Chair

Chao Chen

Yanfei Liu

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Todor Cooklev

Approved by: Carlos Pomalaz Raez 04/24/2012
Head of the Graduate Program Date

Graduate School Form 20
(Revised 9/10)

# PURDUE UNIVERSITY
## GRADUATE SCHOOL

## Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

A Software Defined Radio Testbed for Research in Dynamic Spectrum Access

For the degree of Master of Science in Engineering

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22,* September 6, 1991, *Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

David A. Clendenen

Printed Name and Signature of Candidate

04/24/2012

Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

# A SOFTWARE DEFINED RADIO TESTBED FOR RESEARCH IN DYNAMIC SPECTRUM ACCESS

A Thesis

Submitted to the Faculty

of

Purdue University

by

David A. Clendenen

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Engineering

May 2012

Purdue University

Fort Wayne, Indiana

For my uncle, Jeffrey Davis, who instilled in me from a young age the desire to learn how things work and inspired me to become an electrical engineer.

## ACKNOWLEDGMENTS

I first thank Dr. Todor Cooklev for providing me with the opportunity to expand my engineering background to the world of wireless communications and software defined radio. I appreciate his patience and flexibility in working with me as I balanced my career, graduate courses and this research. Next, I thank my graduate committee; Dr. Chao Chen and Dr. Yanfei Liu and my thesis format advisor, Barbara Lloyd for their support and direction in my thesis writing. I also thank Dr. Elizabeth Thompson for first informing me of the National Science Foundation grant and Dr. Donald Mueller and Dr. Carlos Raez for selecting me as a recipient for the grant. I would also like to thank my colleagues at Regal Beloit and my manager, Dr. Roger Becerra, whose support and flexibility allowed me to accomplish my professional and educational goals. I thank my parents for always encouraging me and instilling in me the value of hard work. Most importantly I thank my wife, Tish, for her encouragement and understanding throughout my graduate studies. Above all else, without her support none of this would have been possible.

TABLE OF CONTENTS

Page

## LIST OF FIGURES

Figure Page

Figure Page

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ADC | Analog-to-Digital Converter |
| API | Application Programming Interface |
| AWGN | Additive White Gaussian Noise |
| COTS | Commercial-Off-the-Shelf |
| CR | Cognitive Radio |
| CRN | Cognitive Radio Network |
| DAC | Digital-to-Analog Converter |
| DSA | Dynamic Spectrum Access |
| FCC | Federal Communications Commission |
| GPS | Global Positioning System |
| GR | GNU Radio |
| GRC | GNU Radio Companion |
| IF | Intermediate Frequency |
| OSA | Opportunistic Spectrum Access |
| PU | Primary User |
| QoS | Quality of Service |
| RF | Radio Frequency |
| SDR | Software Defined Radio |
| SU | Secondary User |
| UHD | Universal Hardware Driver |
| USRP | Universal Software Radio Peripheral |
| VSG | Vector Signal Generator |
| WTC | Wireless Technology Center |

## ABSTRACT

Clendenen, David A. M.S.E., Purdue University, May 2012. A Software Defined Radio Testbed for Research in Dynamic Spectrum Access. Major Professor: Todor Cooklev.

With the rapidly-increasing amount of high data rate wireless devices, technologies and services appearing on the market today, there is an increasing demand for the wireless spectrum. Current wireless networks are characterized by a static spectrum allocation policy, where governmental agencies assign wireless spectrum to license holders on a long-term basis for large geographical regions. Recently, because of the increase in spectrum demand, this policy faces spectrum scarcity in particular spectrum bands. Dynamic spectrum access (DSA) shows promise to increase spectral efficiency. DSA aims at dynamically sharing spectrum that is licensed to primary users (PUs) with non-licensed secondary users (SUs). In order to effectively share spectrum the SUs must be sure to access the spectrum only when the PUs are not utilizing it, otherwise the SUs could cause interference to the PUs. One method to determine when a PU is accessing the spectrum is for a SU to identify if the spectrum is occupied or not through spectrum sensing. Spectrum probing is a key component in spectrum sensing and defines the policy for when the SU will perform a channel scan of the spectrum to collect spectral data to be used for spectrum sensing.

This work describes the development of a software defined radio (SDR) testbed based on the Universal Software Radio Peripheral (USRP) for research in DSA with a focus in spectrum probing methods. Spectrum probing methodology is an often overlooked component of spectrum sensing. Theoretical analysis and simulation results for comparing different spectrum probing methods are presented in [1]. This work expands on the work in [1] by using the developed SDR testbed to collect experimental data and compare the results. Different spectrum probing methods are implemented for the case of an independent SU and for the case of a cooperative network of SUs. Experimental results are compared to theoretical analysis and simulated results. The experimental findings further support the conclusions based on simulation in [1]. In particular, in the independent sensing scenario, periodic probing indeed achieves the smallest probing delay; however, in the cooperative sensing scenario randomization can drastically reduce the probing delay.

# 1. INTRODUCTION

The motivation for this work is to develop a software defined radio (SDR) testbed for research in dynamic spectrum access (DSA). This work begins by providing background information on software defined radio technology and compares and contrasts architectures and testbeds that are commonly used in practice today. Furthermore, the design of the flexible, low-cost SDR testbed that has been developed for this research is presented. The hardware and software elements of the testbed are described and multiple embodiments and use cases for the testbed are presented. The basic description of dynamic spectrum access and spectrum sensing are discussed to provide background information on the functionality and use cases for the testbed.

Specifically, the research is centered on using the developed testbed to implement the different spectrum probing methods proposed in [1] and to evaluate the performance of each method through a performance metric of probing delay. The spectrum probing methods in [1] are presented along with the concept of the performance metric of probing delay.

The experimental probing delay results obtained from the testbed implementation are compared to the theoretical analysis and simulation results in [1] for the case of an independent user and for the case of cooperative sensing in a network of users. This work aims at providing experimental results to further substantiate the conclusions made in [1] that were based on simulation results.  Finally, potential future research work is proposed to expand on the work and concepts presented in this thesis.

# 2. SOFTWARE DEFINED RADIO

## 2.1 Definition

A software defined radio (SDR) is a radio system where components of the radio that have been typically implemented in hardware (e.g. mixers, filters, amplifiers, modulators/demodulators, detectors, etc.) are instead implemented by means of software on a personal computer or embedded computing devices [2]. The primary advantage of an SDR is reconfigurability. By implementing radio functions in software rather than hardware it becomes simpler for a radio device to be reconfigured for different use cases rather needing to redesign the hardware to support new functionality. With the rapid evolution of wireless protocols and standards, a hardware radio could be made obsolete due to the inability to conform to new standards or protocols. An SDR, however, could be reconfigured to support new standards that may not have existed at the time the device was built. Such flexibility is attractive to the manufacturers of the devices as it enables them to update their SDR product through software and not necessarily need to change the hardware of the radio.

## 2.2 SDR Architectures and Testbeds

Recently architectures for rapid prototyping of SDRs have become popular for several reasons. They are being used as an important first step in the development of dedicated systems. These architectures are also important for research and development purposes. The total development time for a dedicated and highly optimized SDR system, including hardware, software, and middleware (device drivers and interfaces), and especially the debugging process can be very long, especially if the system combines different baseband technologies such as FPGAs, DSPs, ASICs. The development time of a new system can be more important than its cost, form-factor, or power consumption. Rapid prototyping is quite important to identify potential problems and reduce the risk of product development. Rapid prototyping is often a “quick and dirty” implementation of the critical components of a system. In a rapid prototyping effort, power consumption and size are typically less important than implementation time.

In an increasing number of cases, consistent with hardware and software re-use principles, SDR systems are being implemented using already-developed testbeds. There are a number of commercially available platforms from companies such as Pentek, Sundance, Spectrum Signal Processing, Lyrtech, and others. Some vendors offer SDR testbeds consisting of hardware and software modules. These solutions are often complex and costly. For academic research and laboratory instruction a simpler and more affordable testbed is desired.

# 3. THE UNIVERSAL SOFTWARE RADIO PERIPHERAL

## 3.1 Introduction to the USRP

The Universal Software Radio Peripheral, or USRP, from Ettus Research is a reconfigurable hardware peripheral that allows general purpose computers to function as high bandwidth software defined radios [3]. There are bus connected USRPs that connect to a host PC via USB2.0 connection, network connected USRPs that connect to a host PC via Ethernet and embedded USRP devices that have an embedded Linux operating system and do not require an external PC. For this research a bus connected USRP is used and henceforth will be referred to simply as "USRP."

The USRP paired with a host computer creates a complete SDR system. In a USRP-based SDR the role of the host computer is to perform all of the baseband and waveform-specific processing such as filtering and modulation and demodulation while the USRP performs high-speed digital up and down conversion from baseband to IF using the on-board FPGA and from IF to RF using an analog RF daughterboard.

## 3.2 Features of the USRP

The USRP is comprised of a motherboard and up to four modular daughterboards. The USRP motherboard implements a digital intermediate frequency (IF) and features four 12-bit, 64MSamples/sec analog to digital converters (ADCs) and four 14-bit, 128MSamples/sec digital to analog converters (DACs). These four input and four output channels are interfaced to the main FPGA. The FPGA is interfaced to a host computer via USB2.0 to create the data link between the USRP and the host computer.

The modular USRP daughterboards provide the USRP access to the RF world. The daughterboards contain the analog RF front-end and IF mixing operations for the SDR. There are several different daughterboards for the USRP each designed for a specific frequency band with a software tunable center frequency. The USRP has a large collection of daughterboards spanning the frequency range from DC to 4.4GHz.

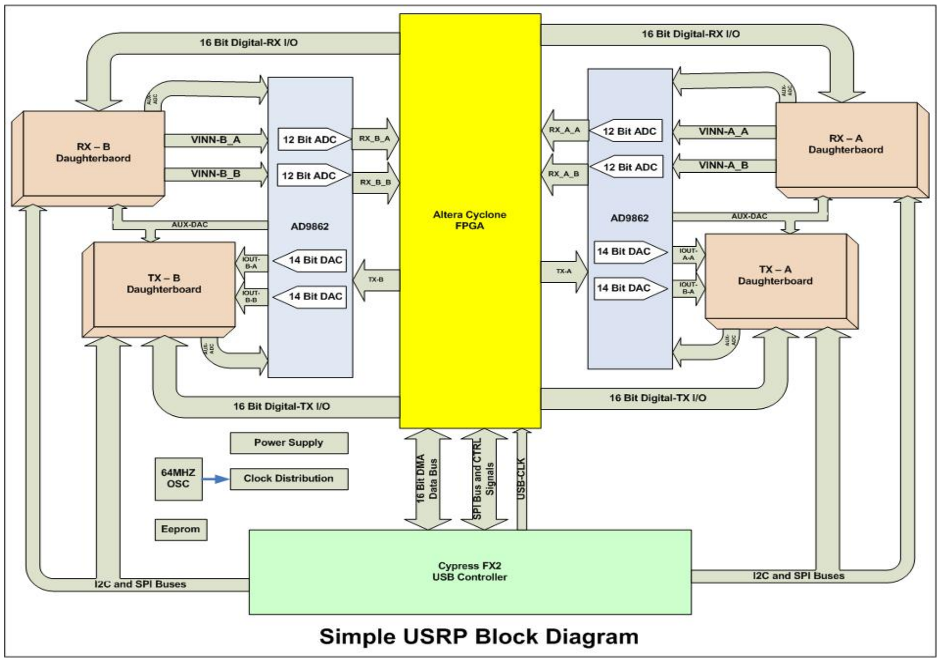


Fig. 1.1. USRP block diagram.

The bandwidth limitations of a bus connected USRP are dictated by the maximum data rate of the connection technology used and the processing power of the host PC. For a USB2.0 connection the USRP can sustain a 256Mbps (32MB/sec) data rate (overhead such as packet headers, time between packets, etc. reduces the USB2.0 maximum of 480Mbps to 256MBps). The format of the complex in-phase and quadrature (IQ) data sent to the PC is 16-bit in-phase and 16-bit quadrature data, which equates to 4 bytes per complex sample. Therefore, the maximum data rate for the baseband IQ data is 8 mega samples per second (32MS/s / 4B = 8MS/s), which translates to a maximum frequency bandwidth of 8MHz by the Nyquist sampling theorem for complex sampling. While the maximum sampling rate across the USB connection is 8MS/s, the true maximum sampling rate that the PC is capable of is dependent on the host PC's data processing capabilities. If the sampling rate is too high for the host PC to properly process the data, the USRP must change the sampling rate through interpolation and decimation.

## 3.3 Compatible Software Packages

The USRP and host computer complete the hardware portion of the SDR system, but in order for them to interoperate the host computer must run a compatible software package. The host computer can control the USRP through the use of software such as the GNU Radio software package or Matlab Simulink. Both software packages offer a means to control the reconfigurable parameters of the USRP such as antenna selection, RF center frequency, gain and sampling rate as well as define the baseband operation of the radio.

## 3.4 GNU Radio

GNU Radio (GR) is a free software development toolkit that provides the signal processing runtime and processing blocks to implement software radios using readilyavailable, low-cost external RF hardware such as the USRP. GNU Radio is natively supported for the Linux operating system with some support for the Mac and Windows operating systems with modifications. GNU Radio has gained popularity in academia due to its open-source nature and diverse user group [4]. GNU Radio applications are in the form of flow graphs, a series of interconnected *blocks* starting from a *source block* and ending in a *sink block*. A *block* in GNU Radio is a piece of software written in the C++ language designed to implement a specific function such as filtering, modulation, signal generation or graphical display. A GNU Radio flow graph is written in the Python programming language to provide a higher level of abstraction from the C++ blocks in order to simplify and streamline the building and implementation of a flow graph.

### 3.4.1 GNU Radio and the USRP

GNU Radio can be interfaced with the USRP to create a software defined radio system. The USRP fulfills the RF and IF functions of an SDR, while GNU Radio performs all baseband functions and reconfigures the USRP. GNU Radio controls the USRP through the Universal Hardware Driver (UHD). The UHD provides a host driver and an application programming interface (API) for the USRP. GNU Radio uses the UHD to modify user-specified parameters such as RF center frequency, antenna selection, gain and interpolation or decimation parameters. These parameters are set by the user in Python and the lower level C++ passes the data to the UHD API which then translates the data for the USRP FPGA.

With the USRP and GNU Radio one can create a transmitter, a receiver or a transceiver system. A block diagram of a USRP-based SDR transmitter built with a

GNU Radio flow graph is shown in Figure 3.1, and a receiver block diagram is shown in Figure 3.2. GNU Radio also offers several graphical sink blocks that can be used to visualize the data throughout the flow graph in the time or frequency domains.

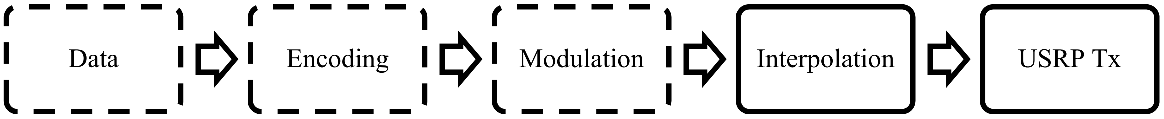


Fig. 3.1. Transmitter architecture. Blocks with dashed outlines are implemented in GNU Radio.

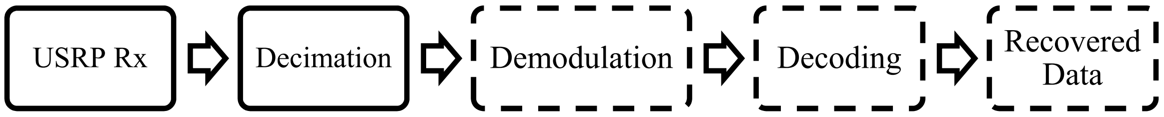


Fig. 3.2. Receiver architecture. Blocks with dashed outlines are implemented in GNU Radio.

### 3.4.2 The GNU Radio Companion

The GNU Radio Companion (GRC) is a graphical environment for building GNU Radio flow graphs and generating the flow graph Python scripts. GRC provides the user with the ability to create a flow graph by connecting graphical blocks that represent the GNU Radio software blocks without the need for writing any software. GRC is a useful tool for quickly building and implementing a USRP-based SDR. More information on the functionality of GRC and a GNU Radio and USRP-based SDR is included in Appendix A.

## 3.5 Simulink-USRP

Another software package that is compatible with the USRP is the Simulink-USRP blockset. Simulink-USRP is used with Matlab Simulink and was developed and made available by the Communications Engineering Lab at the Karlsruhe Institute of Technology [5]. Simulink-USRP equips Matlab Simulink with the capability to control the USRP and transfer data between a host computer and the USRP. Simulink-USRP allows the user to process the data from the USRP in real-time utilizing Matlab and Simulink's extensive list of built-in functions that many engineers are familiar with. Much like with GRC, Simulink provides a graphical environment to build an SDR system with the USRP using USRP source and sink blocks. As with GNU Radio, the USRP's reconfigurable parameters can be set through the software. In Simulink-USRP the USRP source and sink blocks are used to define parameters such as the gain, center frequency and the sampling rate from the USRP through decimation and interpolation, respectively. The USRP source and sink blocks pass data through vectors and therefore the vector size must also be defined.

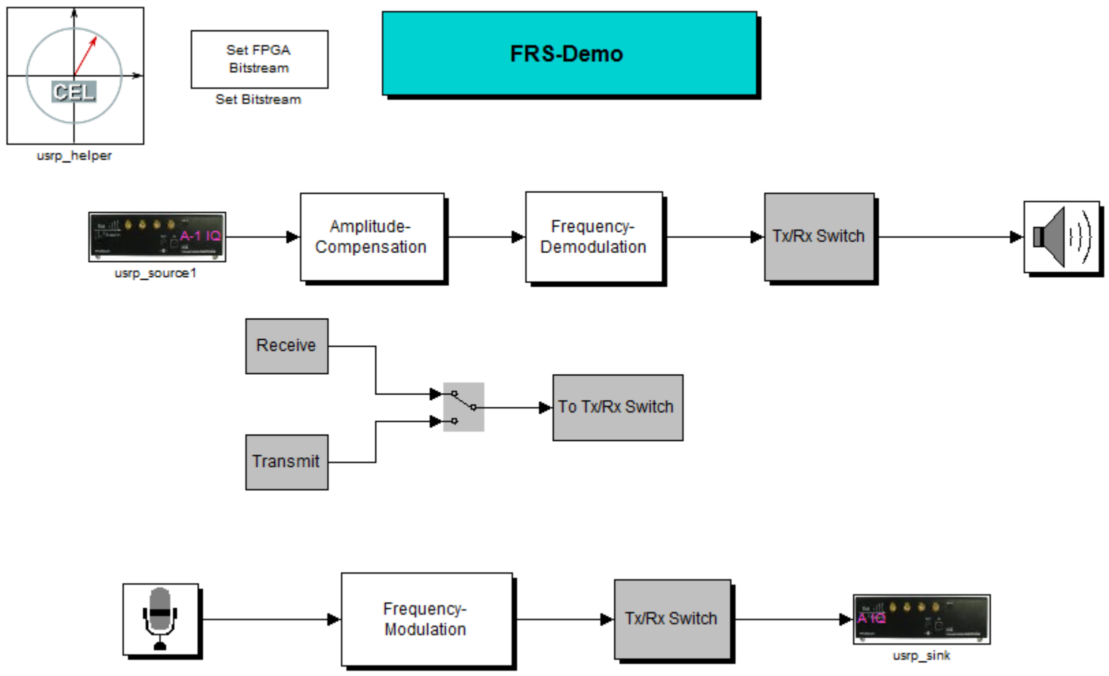


Fig. 3.3. Simulink-USRP demo application for building an FM half duplex transceiver.

Unlike GNU Radio and subsequently GRC, Simulink-USRP contains only USRP source and sink blocks; it does not offer any additional blocks to aid in building a communications system. Functions for synchronization, modulation, filtering, packet framing, etc. must be explicitly implemented in Matlab Simulink to build a true communications system with Simulink-USRP. For example, the Frequency-Modulation block in Figure 3.3 is not a standard Simulink block or a Simulink-USRP block, rather it is a user-defined block built from other Simulink blocks. While implementing a full communications system with Simulink-USRP can be complex, the ability to interface Matlab with real-time RF data is a key feature of Simulink-USRP as it allows for real-time and off-line processing of the IQ data.

The USRP is an excellent foundation for building an SDR testbed given its low cost relative to similar commercial off-the-shelf (COTS) hardware and its ease of use and reconfigurability.  Either the GNU Radio software toolkit or the Simulink-USRP blockset can be used to complete the SDR testbed, depending on the nature of the intended use case of the testbed.

# 4. COGNITIVE RADIO AND DYNAMIC SPECTRUM ACCESS

## 4.1 Spectrum Allocation Models

With the rapidly-increasing amount of high data rate wireless devices, technologies and services appearing on the market today, there is an increasing demand for the wireless spectrum. Current wireless networks are characterized by a static spectrum allocation policy, where governmental agencies assign wireless spectrum to license holders on a long-term basis for large geographical regions. Recently, because of the increase in spectrum demand, this policy faces spectrum scarcity in particular spectrum bands [6].

Some of these static frequency bands can be licensed to certain customers. The licenses limit the use of these frequency bands only to the appropriate licensees. This method of allocation prevents unlicensed users from interfering with licensed users' and ensures a high Quality of Service (QoS) for licensed users; however it can lead to inefficient usage of spectrum. For instance, sporadic usage and geographical variations can lead to spectral utilization of 15% to 85% with a high variance in time [7]. Spectral inefficiencies such as this lead to the creation of spectrum opportunities. Spectrum opportunities are defined in the time, frequency or spatial domains. According to traditional spectral management methods these spectrum opportunities would remain unutilized. However, the increasing demand on the wireless spectrum is fueling research into how this traditional spectrum allocation model can be altered to allow the spectrum opportunities to be utilized by other users.

## 4.1 Cognitive Radio

Cognitive radio (CR) is a breakthrough technology which is expected to have a profound impact on the way radio spectrum will be accessed, managed and shared in the future [8]. It is a particular extension of software radio originally proposed in by Joseph Mitola III in his doctoral dissertation [9]. While there is no formal definition of CR, a popular definition recently adopted by the Federal Communications Commission (FCC) is "*Cognitive radio: A radio or system that senses its operational electromagnetic environment and can dynamically and autonomously adjust its radio operating parameters to modify system operation, such as maximize throughput, mitigate interference, facilitate interoperability, access secondary markets."* [10].

To further expand on the definition, a CR must be able to sense the spectral environment over a wide frequency band and exploit this information to opportunistically provide wireless links that best meet the user communications requirements [11]. CR is currently considered as one of the most promising solutions to the aforementioned spectrum scarcity problem by enabling highly dynamic access to the spectrum in future wireless communication systems [8]. Furthermore, cognitive radio is the enabling technology for dynamic spectrum access [12].

## 4.2 Introduction to Dynamic Spectrum Access

The opportunistic spectrum access (OSA) model is growing in acceptance in the US and the UK [8]. It is a spectrum access model that represents the licensed users as the *primary users* of the licensed spectrum and all other non-licensed users as *secondary users* of that spectrum. The model is based upon the idea that radios known as primary users (PUs) have the spectral right-of-way and cognitive radio users known as secondary users (SUs) can take advantage of the spectrum opportunities that PUs may leave. Since cognitive radios are considered lower priority, or secondary, users of spectrum allocated to a primary user, a fundamental requirement is to avoid interference to potential primary users in their vicinity [11]. This type of spectrum access by the SU is termed dynamic spectrum access (DSA) due to the dynamic behavior the SU must employ in order access the spectrum while avoiding interfering with a PU. Four key components to DSA are spectrum opportunity identification, spectrum opportunity detection, spectrum opportunity tracking and spectrum opportunity exploitation [13].

## 4.3 Spectrum Opportunity Identification

A spectral hole does not necessarily equate to an opportunity for the SU to use the spectrum [6]. Secondary users must determine if a spectral hole is usable through the process of spectral opportunity identification. In order for a spectral hole to be a spectral opportunity for the SU, the SU must determine if the spectral hole is wide enough in frequency to accommodate the bandwidth of the SU’s signal.

## 4.4 Spectrum Opportunity Detection

Once a spectrum opportunity has been identified by a secondary user, the SU must detect when such opportunities exist and when they do not. This method is called spectrum opportunity detection and it can be handled in one of two ways, a priori knowledge of the spectrum or through spectrum sensing [6].

### 4.4.1 The a priori model

A priori knowledge of the spectrum could provide the SU with a location of the spectral holes that are available for use. A simplistic model of such a scenario would be providing the SU with a list of frequencies that are not used by primary users in certain geographical areas. The SU could use a geolocation technology such as global positioning system (GPS) to determine which frequencies are available for the SU to utilize [14-16]. Certain challenges arise when the SU relies solely on a priori knowledge of the spectrum. The SU must be certain that the a priori information that it is basing its decisions on is current and not out of date or else there is risk of interfering with primary users. As more SUs begin accessing the same spectrum in a common geographical location, the a priori knowledge that each SU has must also include the other SUs' activity in order to avoid interference among SUs.

### 4.4.2 Spectrum measurement and sensing

In the event that a priori spectrum information is either not available or not sufficient enough to prevent interference, spectral opportunities must be detected by actively measuring or sensing the spectrum. Spectrum sensing provides the SU with real-time spectral information that can be used to determine if a spectral opportunity exists in the time, frequency and spatial domains. Spectrum sensing enables a higher degree of flexibility to a SU compared to the a priori model in that it is location independent. However, spectrum sensing with no a priori knowledge of the spectrum is inherently less effective at preventing interference with PUs due to detection delay and detection uncertainty. Since one requirement for a spectral opportunity to exist depends on the length of time that the spectrum is vacant, the only way that a SU can know without uncertainty that a spectral opportunity exists is after the opportunity has passed; this illustrates the effect of detection uncertainty. While a SU is accessing a spectrum opportunity, detection delay describes the delay between detecting a PU re-entering the spectrum and the SU recognizing that there is no longer a spectral opportunity and ceasing its use of the spectrum.

### 4.5 Spectrum Opportunity Tracking

In an attempt to minimize interference to primary users while making the most out of the opportunities, cognitive radios should keep track of variations in spectrum availability and should make predictions [17]. The SU can attempt to build a bank of knowledge about the state of the spectrum over time by sensing the spectrum and analyzing the spectral data. The SU can attempt to identify patterns in the data, for example periodic usage by a PU, or attempt to model the spectrum occupancy as a random distribution [18]. Other research has expanded on this area by modeling spectrum states using Markov chains [19].

## 4.6 Spectrum Opportunity Exploitation

Exploiting spectrum opportunities is fairly straightforward once spectrum identification, detection and tracking have been implemented. Once the SU has gained knowledge about the spectrum it can access spectrum opportunities with a lower probability of interference.

# 5. SPECTRUM SENSING

## 5.1 Spectrum Sensing Techniques

As it has been introduced, spectrum sensing is a key issue in cognitive radio and dynamic spectrum access [20]. Spectrum sensing provides the secondary user with knowledge about the spectrum. The SU needs to use this information to attempt to detect spectrum opportunities and to reduce the probability of interference with a PU. In order to successfully detect spectral holes the SU must be able to differentiate between communication signals and noise. Several techniques of varying complexity and resolution can be used to detect the presence of spectrum opportunities.

### 5.1.1 Coherent detection

In the field of wireless communications it is known that the matched filter achieves optimum performance in a receiver. The same principle holds for spectrum sensing and PU detection. If a SU implements a matched filter to perform coherent detection of a PU's signal the SU will achieve optimum detection performance [21].

However, in order to use the matched filter within spectrum sensing, the SU must be synchronized with the PU's signal and must even be able to demodulate the PU's signal. Accordingly, the secondary user has to have prior information about the primary user's signal such as the preamble signaling for synchronization, pilot patterns for channel estimation, and even modulation scheme of the transmitted signal, et cetera.

### 5.1.2 Cyclostationary feature detection

Modulated signals are in general coupled with sine wave carriers, pulse trains, repeating spreading, hoping sequences, or cyclic prefixes which result in built-in periodicity. Even though the data is a stationary random process, these modulated signals are characterized as cyclostationary, since their statistics, mean and autocorrelation, exhibit periodicity. The feature detection technique exploits the cyclostationarity of modulated signals to differentiate between a random signal with a particular modulation type in a background of noise and other modulated signals [11].

Feature detection provides better performance than an energy detector (examined next) however, it can be computationally complex and difficult to implement [22].

### 5.1.3 Energy detection

One of the simplest methods of sensing the spectrum is to measure the energy content in a particular frequency band [23]. When this method is used to detect spectral holes, it is referred to as energy detection. Energy detection is a non-coherent form of detection and therefore does not require any a priori knowledge of other users' signals. The goal of energy detection is to distinguish between two hypotheses [24]:

$$H_0\text{: } Y[n] = W[n] \tag{5.1}$$

$$H_1\text{: } Y[n] = W[n] + X[n] \tag{5.2}$$

$n = 1,\ldots,N$; where N is the observation interval, hypothesis $H_0$ defines that there is no PU present and hypothesis $H_1$ defines that there is a primary user present. $Y$[n] are the digitized received IQ data, $W$[n] represents zero-mean additive white Gaussian noise (AWGN) and $X$[n] is the signal of a primary user. Energy detection selects a hypothesis, either $H_0$ or $H_1$, by computing a decision statistic $T$ and comparing this statistic to a predetermined threshold $\gamma$ [24].

Energy detection can be realized in two different methods [25]. The first method shown in Figure 5.1 uses a low pass filter to remove out of band noise and interference, an analog to digital converter to digitize the analog signal and a square law device to compute the energy and a decision statistic $T$. The second method implements a periodogram shown in Figure 4.1b through means of the magnitude squared of the FFT over the observed spectrum to produce the decision statistic $T$.

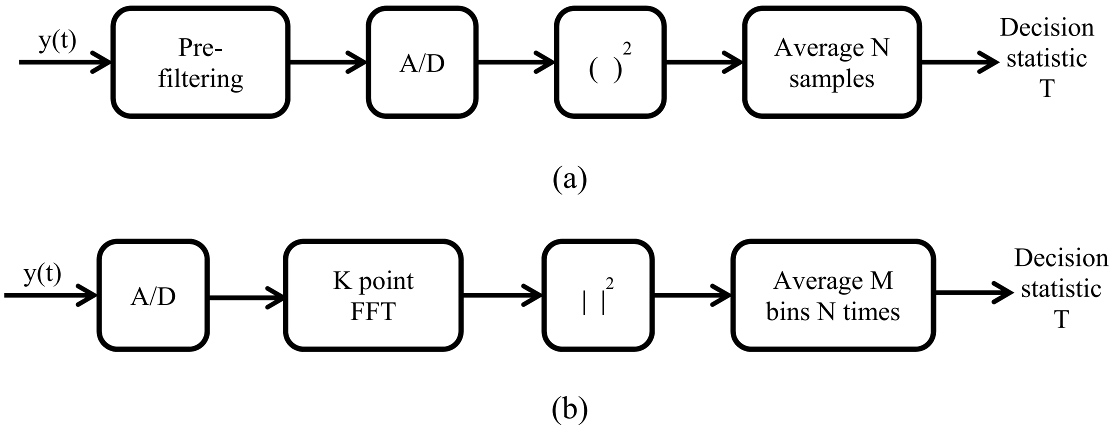


Fig. 5.1. Energy detection method (a) with an analog pre-filter and square-law device (b) with FFT magnitude squared and averaged.

The decision statistic for the energy detector can be mathematically written as:

$$T = \sum_{N} \left| Y[n] \right|^2 \tag{5.3}$$

The hypothesis selection is determined by testing the decision statistic *T*:

$$T < \gamma \rightarrow H_0 \tag{5.4}$$

$$T > \gamma \rightarrow H_1 \tag{5.5}$$

Once the hypothesis has been selected by testing the decision statistic, the energy detector declares the spectrum to be either vacant or occupied and the SU can operate in accordance to this.

While being the simplest sensing method, energy detection also has its limitations. Some of the challenges with energy detector based sensing include selection of the threshold for detecting primary users, inability to differentiate interference from primary users and noise, and poor performance under low signal-to-noise ratio SNR values [26]. The performance of the energy detector is also dependent on the observation period [24]. In cases of low SNR the observation period can be increased to reduce the zero-mean noise by averaging. This can lead to inefficient time usage and missed opportunities by the detector.

## 5.2 Cooperative Spectrum Sensing

When a cognitive radio network (CRN) has multiple SUs, all SUs can collaborate in sensing a channel and share their sensing results with other SUs to determine the presence of PUs. This is defined as cooperative sensing.

The performance of spectrum sensing techniques is limited by the received signal strength which may be severely degraded due to multipath fading and shadowing. In such a scenario cooperative sensing may alleviate the problem of detecting the primary user by reducing the probability of interference to a primary user. In cooperative sensing the variability of signal strength at various locations is relied upon. A large network of cognitive radios with sensing information exchanged between neighbors will have a better chance of detecting the primary user compared to individual sensing [11].

Figure 5.2 shows an example CRN of five SUs in the presence of three PUs. The colored regions *A*, *B* and *C* are the regions in which at least one PU's signal is detectable. In region D there are no PU signals detectable. The SUs which are inside regions *A*, *B* or *C* (*SU 1 – SU 4*) will detect a PU whereas *SU 5* will not. If *SU 5* were sensing the spectrum independently it may decide that the spectrum is free and may begin transmitting. This could be a problem if *SU 5* has the transmit radius depicted by the dotted line in Figure 5.3 and begins transmitting, as it could interfere with PUs that may be in regions *A* or *C*. Through cooperative sensing SUs can share their spectrum sensing results with each other or a common device to obtain a more accurate representation of the spectrum and to prevent this type of interference.

The SUs may exchange information over a reserved common physical channel called a cognitive pilot channel. In some cases the SUs may be connected using a wired network such as Ethernet and use the wired network to exchange information.

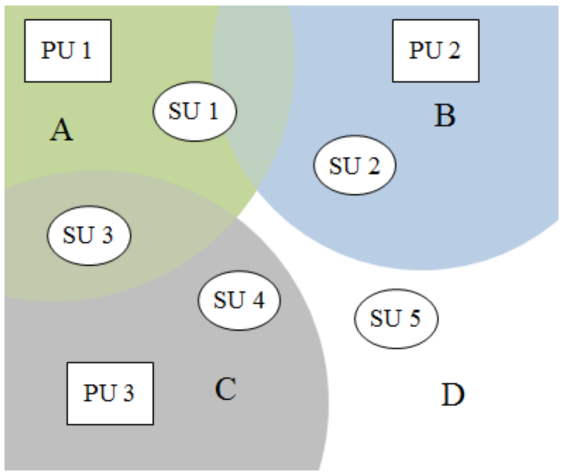


Fig. 5.2. Example CRN with three PU transmitters and five SUs; *SU5* is out of range of all PU transmitters.

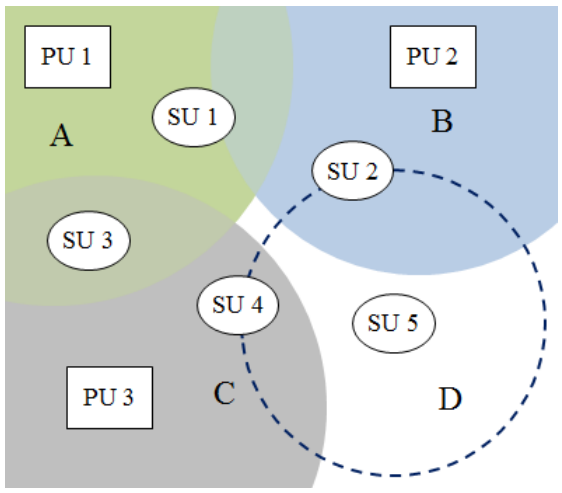


Fig. 5.3. Example CRN with three PU transmitters and five SUs; transmit radius of secondary user *SU 5* represented by dotted line.

# 6. SDR TESTBED DEVELOPMENT

## 6.1 Description of Testbed

The SDR testbed platform developed for the IPFW Wireless Technology Center (WTC) is designed around the USRP and either Simulink-USRP or GNU Radio. Both software platforms are useful tools for research in SDR and DSA, each with different use cases. The GNU Radio platform has been utilized for building wireless transmitter-receiver pairs for use in research in spectrum estimation and classroom exercises. The Simulink-USRP platform has been utilized for research in spectrum estimation and spectrum probing algorithms.

## 6.2 Energy Detector Implementation

This research is focused on the implementation of energy detection with both GNU Radio and Simulink-USRP. As previously presented, energy detection is the simplest method of spectrum sensing. While the implementation of the energy detector differs slightly between the two software packages, the underlying method remains the same. The method of computing the energy that is implemented in the testbed is the FFT magnitude-squared method described above and shown in Figure 5.1b.

### 6.2.1 GNU Radio implementation

The GNU Radio implementation of the energy detector is based on the example GNU Radio program *usrp_spectrum_sense.py* which is used for implementing a wideband spectrum analyzer. The example Python code provides the framework for controlling the USRP and computing FFT-magnitude squared data. By expanding on the framework the remaining components of the energy detector are implemented in Python such as the decision statistic generation and the hypothesis testing. The magnitude-squared data and decided channel state are stored in a file for off-line processing with another software package such as Matlab or Octave.

A key advantage of the GNU Radio implementation is the ability to scan a bandwidth larger than the USRP's limitations. This is done by retuning the USRP to an adjacent frequency band immediately after scanning the previous band, which is controlled by the *usrp_spectrum_sense.py* code, and then combining the data. However, this does not allow real-time scanning of such a wide band and may be too slow for some applications.

### 6.2.2 Simulink-USRP implementation

The Simulink-USRP implementation of the energy detector shown in Figure 6.1 is built in Simulink using the USRP Source block to provide the IQ data to the Simulink environment. Using standard Simulink blocks the IQ data are converted into FFT-magnitude squared data. The data are then input into an Embedded Matlab Function block defined as *Detector block*, which completes the energy detector and controls the behavior of the detector. Data are input into the *Detector block* at the rate that the USRP is providing data, a rate that is much higher than the desired probing rate. The channel scan interval *X* is set as a multiple of the USRP data rate within the *Detector block*, where *X* is typically two orders of magnitude larger than the USRP data rate. An if-loop is used to convert the USRP data rate to the channel scan rate. As data are input into the *Detector* block, data are only processed and stored once every channel scan interval *X* and the other data go unprocessed. This method is used rather than only receiving data from the USRP at the channel scan rate to avoid variable timing overhead associated with initializing the USRP prior to making its first scan.

A key advantage to the Simulink-USRP implementation is the capability for real-time processing of the data. If additional processing of the data is required the data can be directly output to Matlab without the need for being written to a file first. Furthermore, the Simulink model can be controlled from Matlab, enabling automated implementation and data analysis.

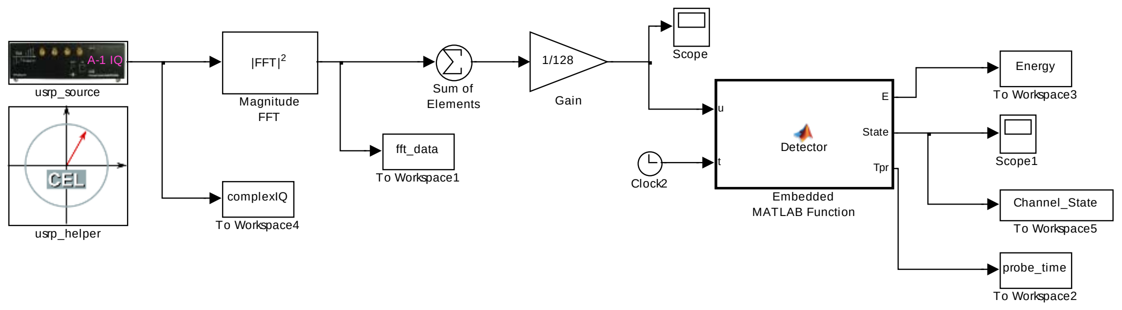


Fig. 6.1. Simulink-USRP implementation of an energy detector.

# 7. SPECTRUM PROBING AND PROBING DELAY

## 7.1 Description of Spectrum Probing

Spectrum sensing describes the process of measuring and estimating the spectrum. One important aspect of spectrum sensing is to determine when and how often to sense the spectrum. The term *spectrum probing* specifically refers to such scheduling policies and is used to avoid confusion with the more general term *spectrum sensing*. Spectrum probing mechanisms should be efficient and fast to avoid harmful interference with primary users (PUs). One of the main purposes of spectrum probing is to detect the absence and the return of PU signals in a channel in a timely manner, so that the SUs can utilize licensed spectrum bands opportunistically without interfering with the PUs [1].

Many existing cooperative sensing schemes consider that all SUs are synchronized in sensing the channel. That is, all SUs probe the channel at the same time. This work investigates the results of spectrum probing when SUs are not necessarily synchronized together and instead probe the channel independently as illustrated in Figure 7.1. Furthermore, randomized probing methods are investigated and compared to periodic methods.

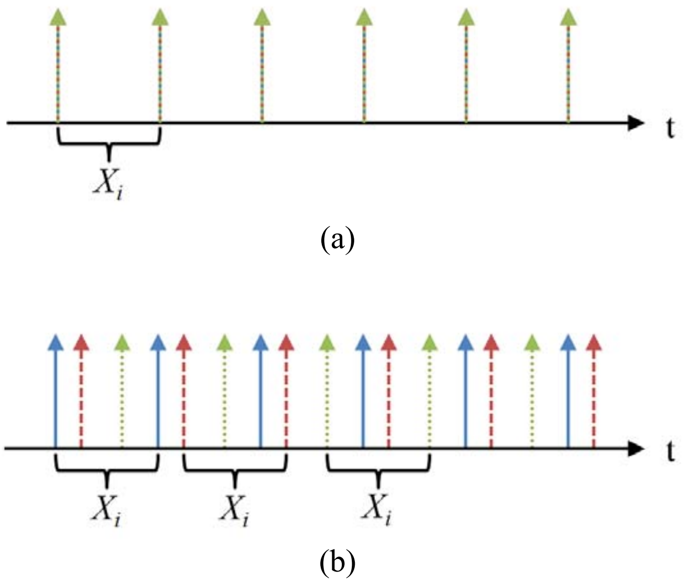


Fig. 7.1. Synchronized periodic spectrum probing (a) and independent periodic spectrum probing by three users (b).

## 7.2 Spectrum Probing Methods

Standards and other works in the area of spectrum sensing do not address the methods used for implementing spectrum probing policies. The most commonly adopted method for spectrum probing is periodic probing. In periodic probing the spectrum is probed at periodic intervals. Other probing methods have been proposed that do not require the spectrum to be probed in a periodic manner, rather in a manner that fits a random distribution such as uniform or Poisson [1].

As in [1] $X_i$ is used to denote the time interval between the SU's *i*th and ($i$ + 1)th channel scan or probe ($i$ = 1,2,…). In the case of periodic probing all $X_i$'s are equal and fixed, *i.e.*, $X_i = \mu$. In the case of randomized probing all $X_i$'s are independent and identically distributed (i.i.d.) random variables with a distribution function $F_X(t)$. For the uniform random probing method the probe interval $X$ follows a uniform distribution within [0, 2μ], *i.e.,*

$$F_X(t) = \begin{cases} 0, & t \le 0 \\ \dfrac{t}{2\mu}, & 0 \le t \le 2\mu \\ 1, & otherwise \end{cases} \tag{7.1}$$

While for Poisson probing the probe interval $X$ follows an exponential distribution with mean $\mu$ *i.e.,*

$$F_X(t) = \begin{cases} 0, & t \le 0 \\ 1 - e^{-t/\mu}, & t > 0 \end{cases} \tag{7.2}$$

Figure 7.2 shows an illustration of the implementations of these different probing methods.

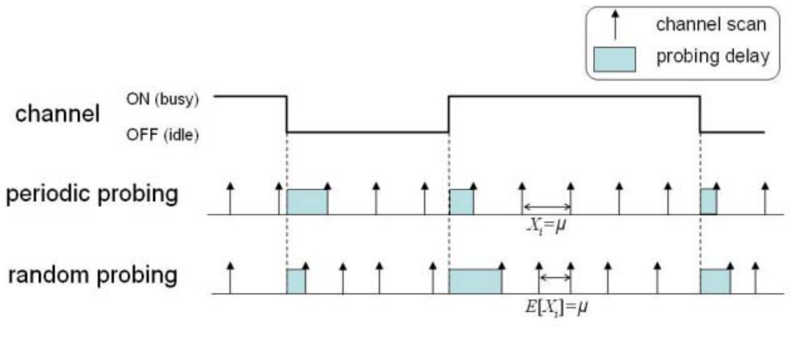


Fig. 7.2. Illustration of periodic and random probing methods.

## 7.3 Description of Probing Delay

To evaluate the performance of a spectrum probing method a performance metric called probing delay, i.e., how quickly a probing method can detect a channel change, is used. Probing delay is also the performance metric used to compare the performance of periodic and random probing methods.

Probing delay can lead to either interference with a PU or a missed opportunity. Probing delay which occurs when the channel changes from the occupied to vacant state results in a missed opportunity. This is due to the receiver declaring the channel occupied when it is actually vacant. Conversely, when probing delay occurs after the channel changes from vacant to occupied results in interference with the PU, due to the receiver declaring the channel vacant when it is actually occupied. Figures 7.3-7.5 show these cases.

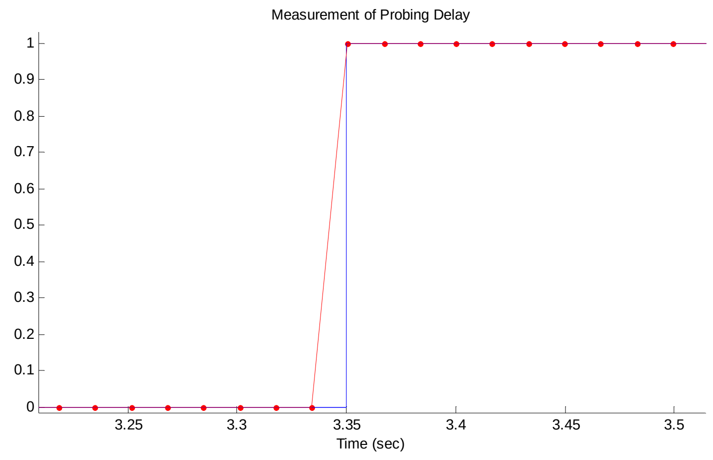


Fig. 7.3. Case of near-perfect detection. Blue – actual channel state, Red – probed channel state. 0=vacant, 1=occupied.

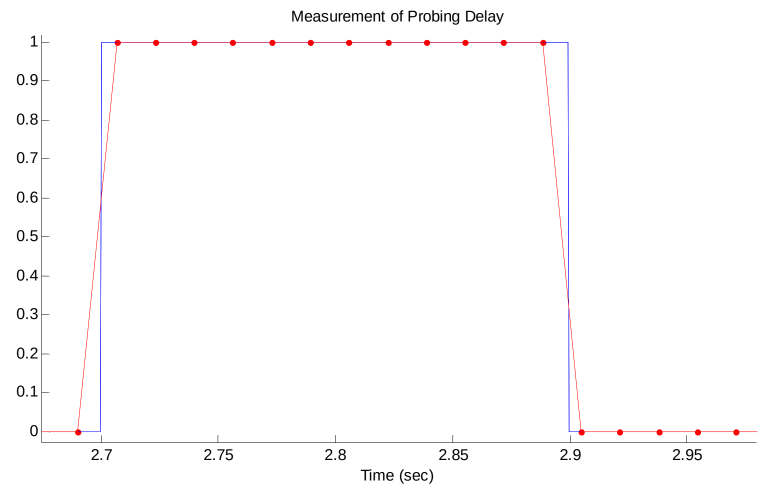


Fig. 7.4. Small probing delay, resulting in interference. Blue – actual channel state, Red – probed channel state. 0=vacant, 1=occupied.

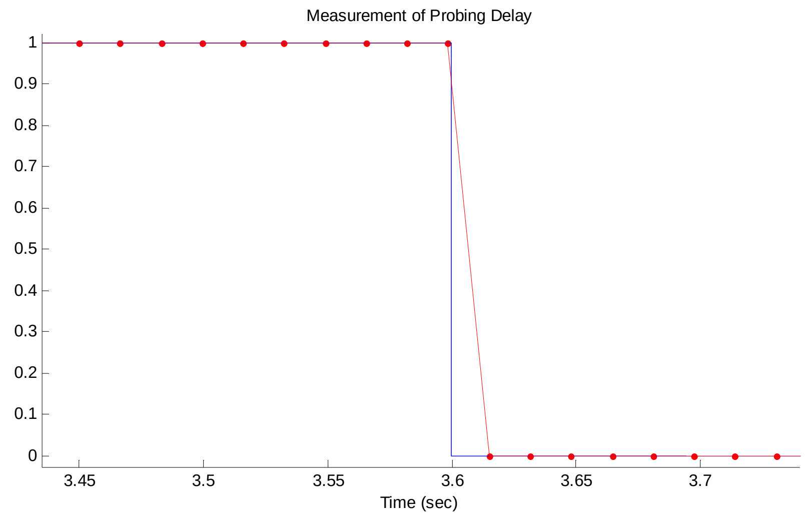


Fig. 7.5. Large probing delay, resulting in missed opportunity. Blue – actual channel state, Red – probed channel state. 0=vacant, 1=occupied.

## 7.4 Theoretical Analysis of Probing Delay

### 7.4.1 Independent perfect detection

If a SU always correctly determines the channel state in the probing time it is considered the case of perfect detection. Let $\{ N(t), t \geq 0\}$ be the number of scans that occurred before time t. Since the time intervals between two successive probes ($X_i$'s) are independent identically distributed (i.i.d.), $N(t)$ can be regarded as a renewal process. The residual time from $t$ until the next scan is further defined as $Y(t)$. Since a change in the channel status can occur at any time $t$, the average probing delay $D$ is equal to the expected value of $Y(t)$, which can be calculated using the renewal reward theorem [27] as:

$$D = E[Y] = \lim_{s \to +\infty} \frac{\int_0^S Y(t)dt}{S}$$

$$= \frac{E[reward\ during\ a\ cycle]}{E[length\ of\ a\ cycle]} = \frac{E[\int_0^X (X-t)dt}{E[X]}$$

$$= \frac{E[X^2]}{2 \cdot E[X]} = \frac{\text{var}[X] + E^2[X]}{2 \cdot E[X]} = \frac{\text{var}[X]}{2\mu} + \frac{\mu}{2} \tag{7.3}$$

Knowing the probing interval $X$ and the fact that $var[X] \geq 0$ and $D \geq \mu/2$, the theoretical minimum average probing delay can be calculated if the average probing interval $\mu$ is also known. For the periodic case where all $X_i$'s are equal, $var[X] = 0$ the average probing delay, $D$ is

$$D = \mu/2 \tag{7.4}$$

For the random probing case, *var*[*X*] is always positive. Therefore the average probing delay is always larger than $\mu/2$. For Poisson probing, *var*[*X*] = μ and the average probing delay *D* is

$$D = \mu \tag{7.5}$$

For uniform probing, $var[X] = \mu^2/3$ and the average probing delay *D* is

$$D = (2\mu/3) \tag{7.6}$$

## 7.4.2 Independent imperfect detection

If a SU does not always detect the channel state successfully in the probing time it is considered the case of imperfect detection or detection with uncertainty. Imperfect detection of a PU results in misdetection or false alarm. The SU then has a probability of misdetection ($P_{MD}$) and a probability of false alarm ($P_{FA}$), where $P_{MD} = P_{FA} = 1 - p$ and $0 \le p \le 1$.

These probabilities of uncertainty must be accounted for in the theoretical average probing delay calculations. From [1] the theoretical independent probing delays in the presence of uncertainty are:

$$\textit{Periodic:}\ D = \left(\frac{1}{2} + \frac{1-p}{p}\right) \cdot \mu \tag{7.7}$$

$$\textit{Uniform:}\ D = \left(\frac{2}{3} + \frac{1-p}{p}\right) \cdot \mu \tag{7.8}$$

$$\textit{Poisson:}\ D=\left(1+\frac{1-p}{p}\right)\cdot\mu \tag{7.9}$$

### 7.4.3 Cooperative perfect detection

Similarly to independent sensing, if a SU always correctly determines the channel state in the probing time it is considered the case of perfect detection. The case of independent cooperative sensing is considered, meaning that all SUs in a CRN are not required to be synchronized in their probing intervals. In the cooperative sensing scenario a simple OR rule [28] is adopted to determine the channel status, meaning that all SUs in a CRN independently probe the same channel and as soon as one SU senses a channel state change it will inform the other SUs. As the number of SUs in a CRN increases the number of inputs into the OR rule increase leading to potential reduction in detection time which is explained in detail in [1] and yields the following results for cooperative probing delay under perfect detection:

$$\textit{Periodic:}\ D=\frac{\mu}{N+1} \tag{7.10}$$

$$\textit{Uniform:}\ D=\frac{2\mu}{2N+1} \tag{7.11}$$

$$\textit{Poisson:}\ D=\frac{\mu}{N} \tag{7.12}$$

In this work, probing delay for the case of independent cooperative sensing with imperfect detection is examined through simulation and experimentation only.

# 8. TESTBED IMPLEMENTATION: MEASUREMENT OF PROBING DELAY

## 8.1 Overview

This work implements the spectrum probing methods specified in [1] in the SDR testbed described in this paper. The methods of independent sensing and cooperative sensing are explored with the testbed under the case of perfect detection. The performance metric for each case is the average probing delay. The probing delay for channel state change $m$ is $D_m$, where $m = 1,2,\ldots M$ and $M$ is the total number of channel state changes for a given experiment. Then the average probing delay, $\overline{D}$, is defined as the average of the measured probing delays for each channel state changes.

$$\overline{D} = \frac{\sum_{m=1}^{M} D_m}{M} \tag{8.1}$$

By calculating $\overline{D}$ for different spectrum probing methods, their performance can be evaluated and compared.

## 8.2 Primary User Emulation

For these experiments a vector signal generator (VSG) was used to emulate the behavior of a PU. The VSG was programmed to output a 100 kHz wide test signal at 900

MHz that was toggled on and off at randomly generated intervals to emulate the behavior of a PU entering and exiting the spectrum. The signal ON and OFF intervals were uniform-randomly generated off-line and then programmed into the VSG. A total of 50 channel state changes are included in one cycle of the test signal. This provided a known test signal of finite length that was random in origin but known for the experiment. The channel state (ON/OFF) representation of the test signal is shown in Figure 8.1. To measure the probing delay, the time of the actual channel state must be known along with the time at which the detector has determined the channel status through spectrum sensing. By using the known test signal, the former requirement is satisfied. The challenge then is synchronizing the test signal and the receiver so that the data can be properly analyzed after the experiment and the probing delay be accurately measured.

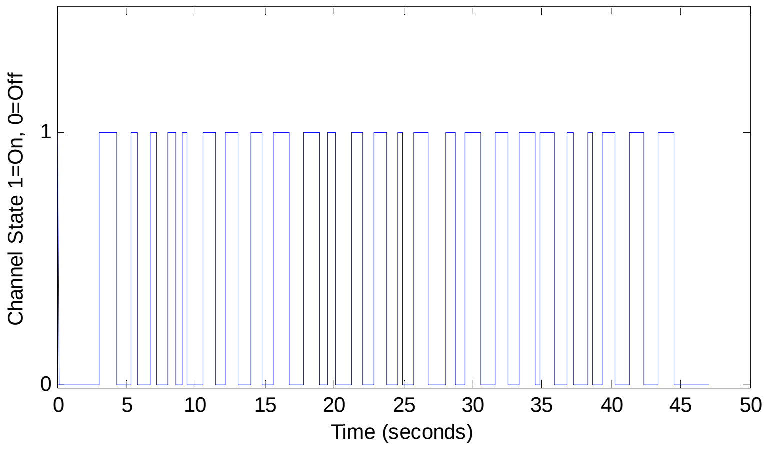


Fig. 8.1. Channel state ON/OFF representation of the test signal.

## 8.3 Signal Synchronization

To perform the synchronization, a *synchronization pulse* is added to the beginning of the test signal. The receiver samples the spectrum at the USRP data rate, a rate of approximately 7.8 kHz (sampling period of 128μs) called the *synchronization frequency*. Once the receiver detects the synchronization pulse it declares that moment as time zero, or $t_0$. The maximum delay between the start of the actual synchronization pulse and when the receiver detected it, called the synchronization delay, is one synchronization period of 128μs. As long as the synchronization period is much less than the spectrum probing period it can be considered negligible and not be factored into the results.

After establishing a time zero, the receiver waits a randomly generated period of time before beginning the actual spectrum probing period. The delay time is uniform random between 1000 and 4000 milliseconds. Figure 8.2 illustrates the synchronization probing process and the transition into the spectrum sensing process.

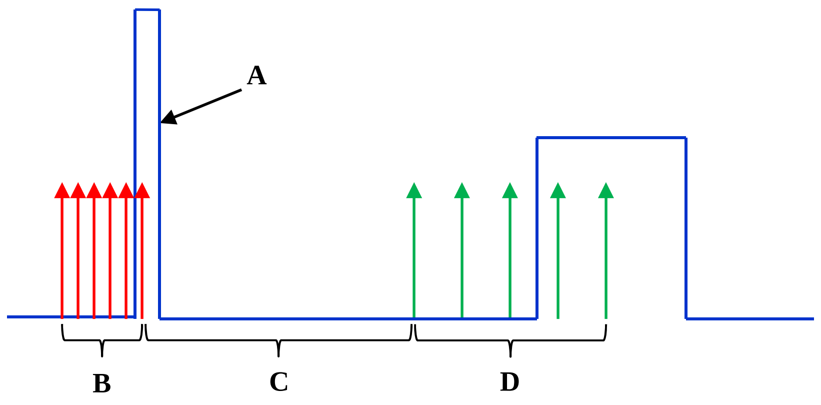


Fig. 8.2. Synchronization probing process; synchronization pulse (A), synchronization probes (B), random delay time (C), spectrum sensing probes (D).

## 8.4 Probing Implementation

The Simulink-USRP energy detector was selected for implementation of the spectrum probing methods. The USRP decimation was set to 64 providing an observable bandwidth of 1MHz (1M Samples/sec). The IQ data are packed in vectors of length 128 which sets the Simulink sample time, $T_{SIM}$ to (1/1MHz)·128=128μs per sample. The probing method is configured to be either periodic, uniform or Poisson. For periodic probing the probing interval, $T_p$, is equal to a constant value that is multiplied by the simulation period, $T_{SIM}$. For random uniform and Poisson probing a Matlab function is used to generate a random number based on the specified mean and distribution which is then multiplied by $T_{SIM}$ to form the probing interval, $T_p$. The FFT magnitude-squared of the IQ data is computed every $T_{SIM}$ and input into the *Detector* block. The *Detector* block increments a count variable that is compared to $T_p$. If the count exceeds $T_p$ then the remaining energy detector functions are implemented. The energy detector computes a running sum of the FFT magnitude-squared data for each consecutive sample until another loop counter variable has exceeded the dwell time, $T_{dwell}$. The running sum is then divided by $T_{dwell}$ in order to compute the average energy over the dwell time. This energy represents the energy of the entire observed spectrum for the observed dwell time. This energy is then compared to a predetermined threshold to determine if the channel state is occupied or vacant. For this work an arbitrary threshold was chosen, however in practice the choice of threshold has a direct effect on the detection and false alarm probabilities and should be set accordingly [21]. Finally, the *Detector* block resets the loop count variables, generates a new $T_p$ (for random probing methods) and then stores the computed energy, determined channel state and the probing time to the Matlab workspace for further analysis of probing delay.

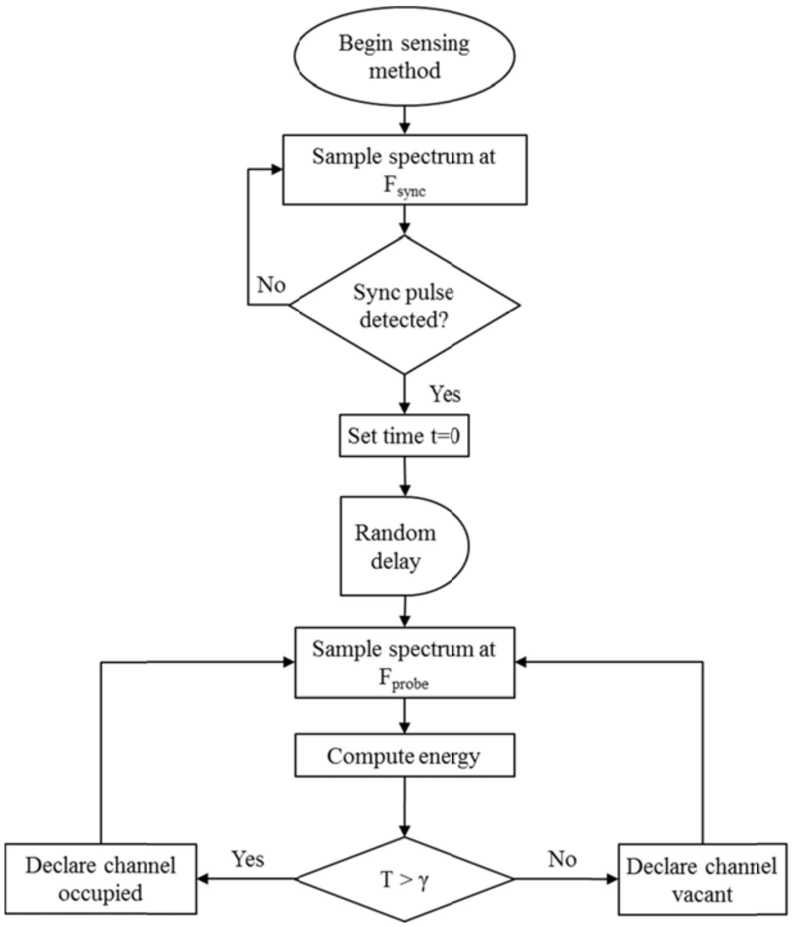


Fig. 8.3. Flowchart of Simulink-USRP energy detector process.

### 8.4.1 Cognitive radio network emulation

To test the cooperative sensing implementation multiple SUs operating in a cognitive radio network are required. However, due to limitations in test equipment the CRN scenario was emulated using a single testbed. The key requirement for non-synchronized cooperative sensing is that the SUs must probe the spectrum independently of each other. Since there is a randomly generated delay between when the synchronization pulse is detected and when the actual probing period starts, each probing experiment can emulate a different SU. By repeating the probing experiment *N* times up to *N* SUs can be modeled and their cooperative probing delays can be computed.

## 8.5 Data Analysis

After completion of a probing period the data required to compute metrics such as the probing delay are stored in the Matlab workspace. The primary measurement of interest is the probing delay. Before the probing delay can be computed in Matlab, the test signal must be stored in Matlab. To accomplish this, the test signal is probed using the energy detector with $T_p$=1 and without a random delay, which digitizes the test signal in Matlab with a resolution of 128μs. The channel state transition points, or edges, of the test signal are identified using a Matlab script file. The times of the edges are stored in a vector *actual_tedges*, which represents the absolute times of the channel state changes of the test signal.

Once the test signal has been stored in Matlab it is used as the reference for computing the probing delay for the different probing methods. The spectrum probing data from the energy detector are stored in Matlab for mathematical analysis of probing delay but can also be plotted to visualize the implementation of the probing method. Figure 8.4 shows the digitized test signal in blue with the spectrum probing events in red. The randomization in probing interval can be clearly seen in Figure 8.6 and Figure 8.7, compared to the periodic probing in Figure 8.5.

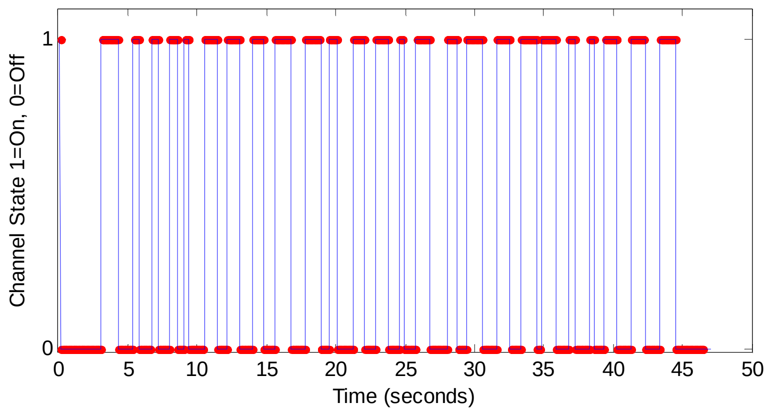


Fig. 8.4.  Digitized test signal (blue) with probing events (red).

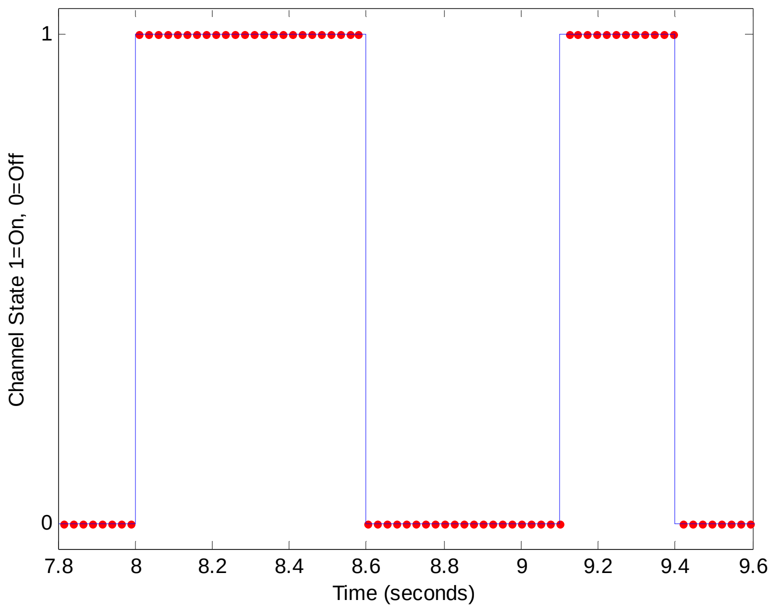


Fig. 8.5.  Periodic spectrum probing, probing events in red.

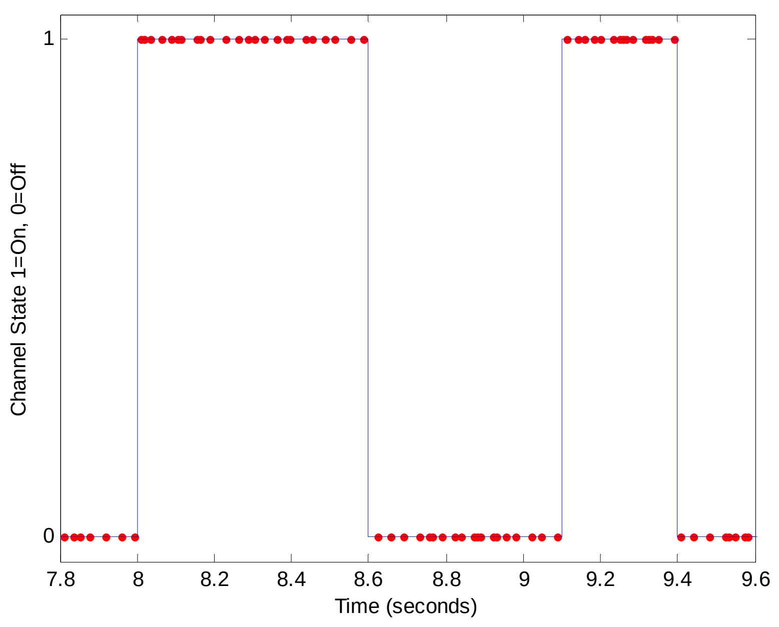


Fig. 8.6.  Uniform random spectrum probing, probing events in red.

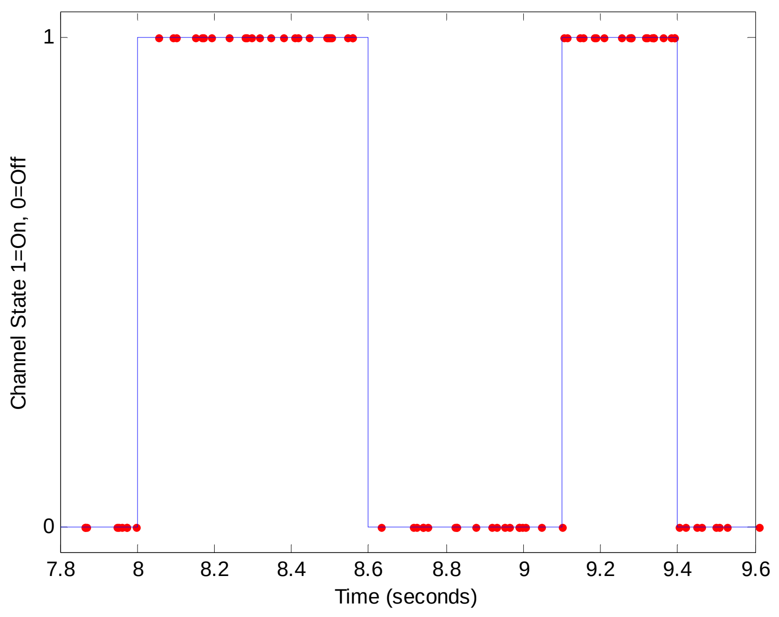


Fig 8.7.  Poisson random spectrum probing, probing events in red.

### 8.5.1 Independent sensing

For the case of independent sensing, a single spectrum probing experiment is analyzed for probing delay. To compute the probing delay of an experiment the probed data that are stored in the Matlab workspace are analyzed using the *simulink_spect_probing_UI* Matlab script by first identifying the synchronization pulse from the probed data and declaring this point as time zero. This allows the remainder of the probed data to be offset to a time relative to this time zero. Next, the channel state transition points, or edges, are identified through a user-defined Matlab function, *edgefind*, and the times of the edges are stored in a vector *tedges*. The vector containing the test signal channel state transition points, *actual_tedges,* is subtracted from the vector containing the probed channel state transition points, *tedges*, to compute the probing delay. User-defined Matlab scripts and functions used are provided in Appendix B.

### 8.5.2 Cooperative sensing

In the case of cooperative sensing the probing delay for a channel state change is equal to the minimum probing delay among all cooperative users in the CRN. The average probing delay then is the average of all of the cooperative delays.

To implement the case of cooperative sensing using the testbed for *n* cooperative users, the experiment must be repeated *n* times as previously described. The probing delay results of the *n* trials of the experiment are computed in the same manner as the independent sensing case; however the probing delay results are stored in a matrix, *delays*. The matrix *delays* is an *M* x *n* matrix where *M* is the total number of channel state changes and *n* is the total number of cooperative users.

$$delays = \begin{bmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{M1} & d_{M2} & \cdots & d_{Mn} \end{bmatrix} \tag{8.2}$$

The element $d_{ij}$ is defined as the probing delay for *i*th state change computed by the *j*th cooperative user.

The set of all cooperative users is defined as *S*, where *S*={1,2,…,*n*} corresponds to the columns of the matrix *delays*. From the matrix *delays,* probing delay can be computed for any set of *k* cooperative users where $k = 1,2,...,n$.

To evaluate the average cooperative probing delay for *k* cooperative users, the average probing delay for each subset of *S* which contains *k* users must be computed and averaged together. Since order does not matter these subsets are combinations of the elements of *S*. The total number of combinations for *k* cooperative users is computed using the binomial coefficient:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{8.3}$$

The set which contains all combinations of *k* users is defined as

$$S_k = \{S_{k1}, S_{k2}, ..., S_{ki}\} \tag{8.4}$$

where $i = 1,2,...,\binom{n}{k}$ and $S_k \subseteq S$. The $S_{ki}$ elements are enumerated combinations of length *k* of the elements of *S* that define which users*, i.e.* which columns of the matrix *delays*, should be used to compute the average probing delay.

The average probing delay for *k* cooperative users can then be computed as

$$\overline{D}_k = \min(\overline{D}_{k1}, \overline{D}_{k2}, ..., \overline{D}_{ki}) \tag{8.5}$$

where $\overline{D}_{ki}$ is defined as the average cooperative probing delay between the *k* users in subset $S_{ki}$.

The Matlab script *coopsensing* shown in Appendix B was written to determine and enumerate all possible *k* combinations of cooperative users from $S_k$ and to compute $\overline{D}_k$ for $k$=1,2,…,$n$.

# 9. SIMULATION AND EXPERIMENTAL RESULTS

This chapter presents the results of the experimental measurement of probing delay obtained from implementation of the testbed for three probing methods, periodic, uniform random and Poisson random. Experimental probing delays are compared to the theoretical analysis of probing delays presented in chapter 7. For completeness, simulation of the probing methods is also performed and the probing delay results are presented and compared to the theoretical analysis and experimental results. Both the independent and cooperative scenarios are evaluated and all results are presented for both the perfect and imperfect detection scenarios.

In order to compare performance of probing methods it is assumed that all SUs spend the same average amount of power in sensing the channel, *i.e.*, the average probing interval of all SUs is the same ($E[X] = \mu$). The minimum time for the channel to remain at a given state, ON or OFF, was 300ms and the maximum time was 1200ms.

## 9.1 Simulation Results of Probing Delay

For completeness, all three probing methods were simulated using the same simulation method as in [1] with adjustments for time scaling. The simulation results are compared against the theoretical analysis results in figures 9.1-9.6. It is evident from the figures that the simulation results are a near perfect fit with the theoretical analysis. The error bars in the figures represent the variability of the mean of the data within one standard deviation.

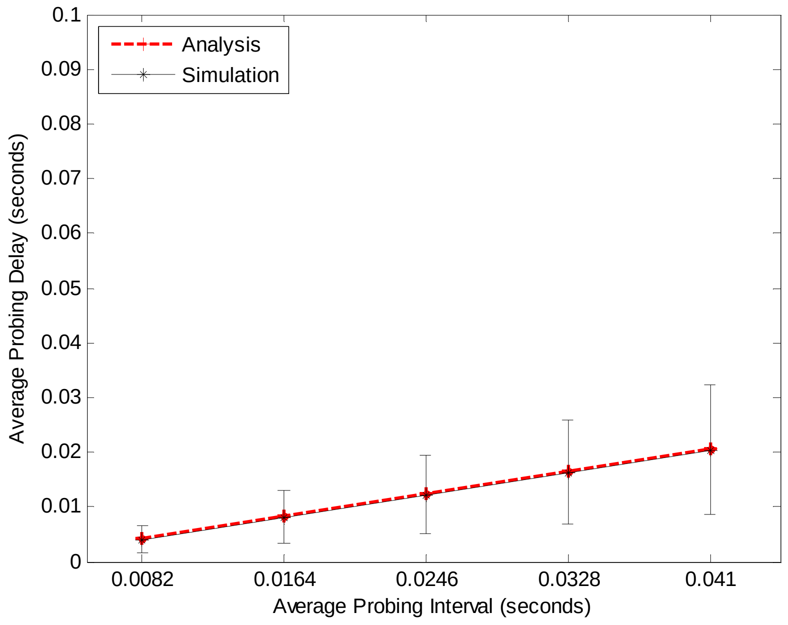


Fig. 9.1. Periodic probing. Average probing delay, simulation versus theoretical, for the independent sensing scenario.

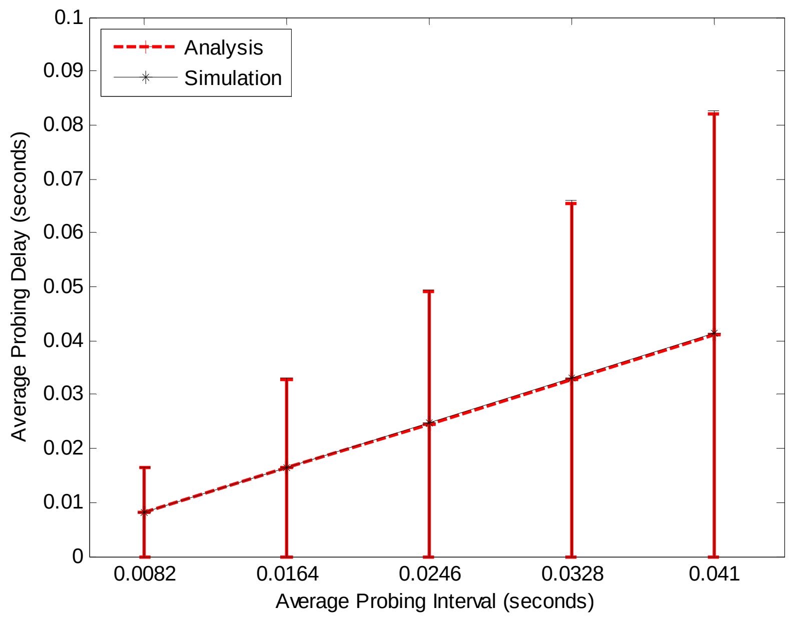


Fig. 9.2. Poisson probing. Average probing delay, simulation versus theoretical, for the independent sensing scenario.

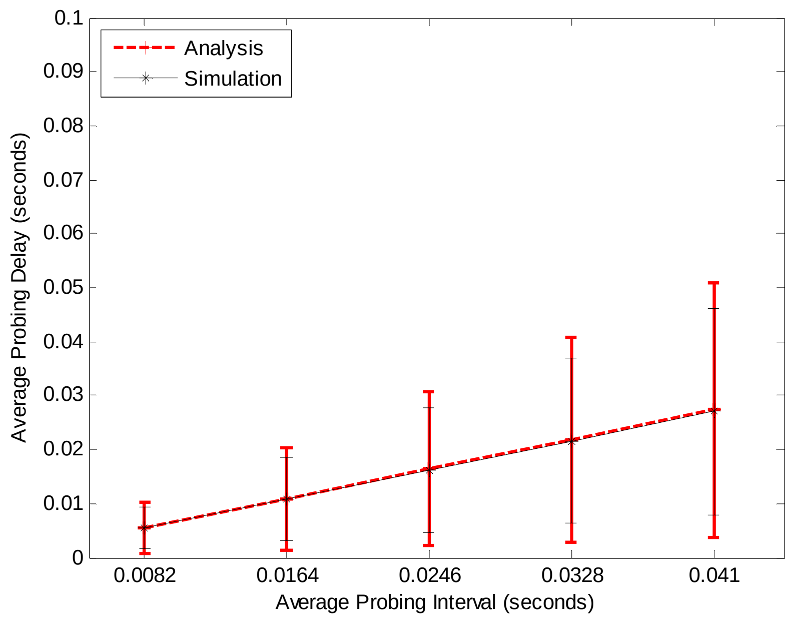


Fig. 9.3. Uniform probing. Average probing delay, simulation versus theoretical analysis, for the independent sensing scenario.

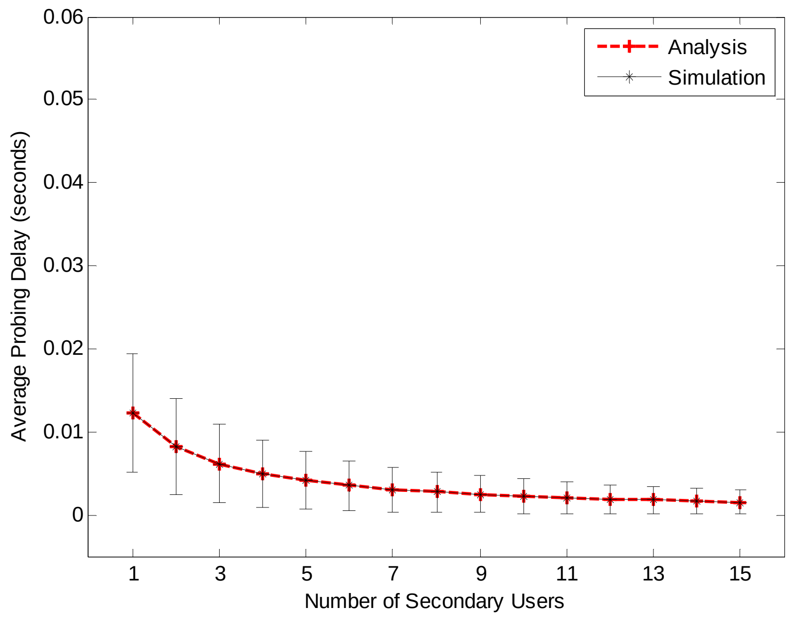


Fig. 9.4. Periodic probing. Average probing delay, simulation versus theoretical, for the cooperative sensing scenario.

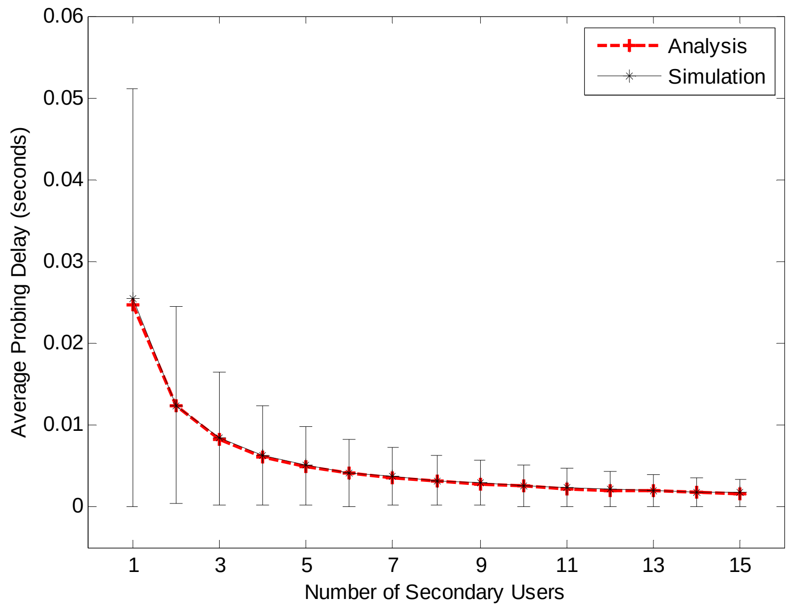


Fig. 9.5. Poisson probing. Average probing delay, simulation versus theoretical analysis, for the independent sensing scenario.

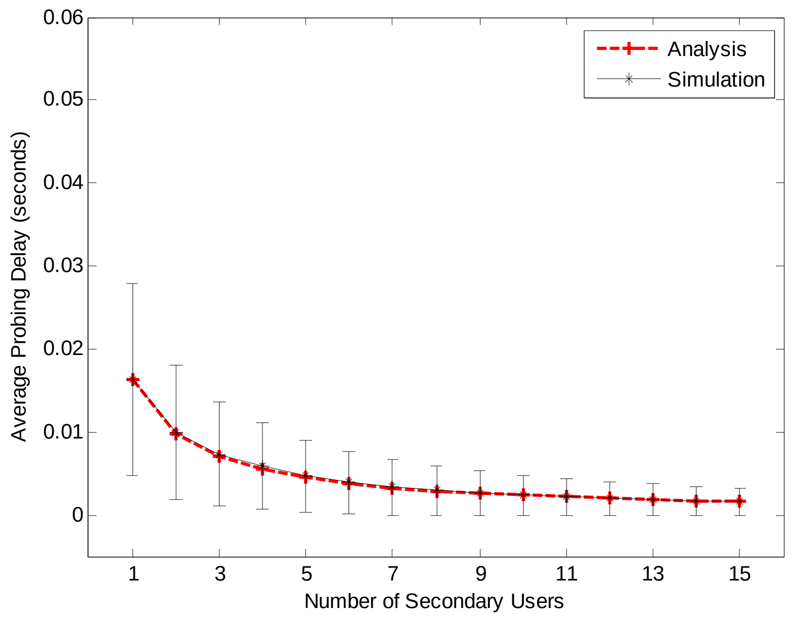


Fig. 9.6. Uniform probing. Average probing delay, simulation versus theoretical analysis, for the independent sensing scenario.

## 9.2 Experimental Results of Probing Delay: Perfect Detection

### 9.2.1 Independent sensing results

Independent sensing was performed for a single SU implementing periodic, uniform random and Poisson random spectrum probing methods. Each method was implemented for five different average probing intervals to examine the performance of the probing method with respect to the probing interval.

First, the theoretical analysis of the probing delays for the three probing methods was perform and plotted to compare the theoretical performance of the different methods. It can be seen from the results of Equations 7.4-7.6 and the plot of Figure 9.7 that periodic probing provides the smallest average probing delay for a single user for all average probing intervals.

Finally, the experimental probing delays of the three probing methods were measured and analyzed. To obtain a statistical representation of the average probing delay, each experiment was repeated 50 times for each probing period and each probing method. The average probing delay for each of the 50 trials was computed and averaged to obtain the average probing delay used as the experimental result.

The comparison of the experimental results for the three probing methods is shown in Figure 9.8. Comparison of the experimental results to the simulation results is found in Figures 9.11, 9.13 and 9.15. In addition to the average, the data for each of the 50 trials of the experiments are shown in Figure 9.10, 9.12 and 9.14. Figure 9.9 shows the average probing delay normalized to the probing period. It can be seen from Figure 9.10 that independent of the probing interval, the average probing delay closely matches the theoretical analysis derived in Equations 7.4-7.6.

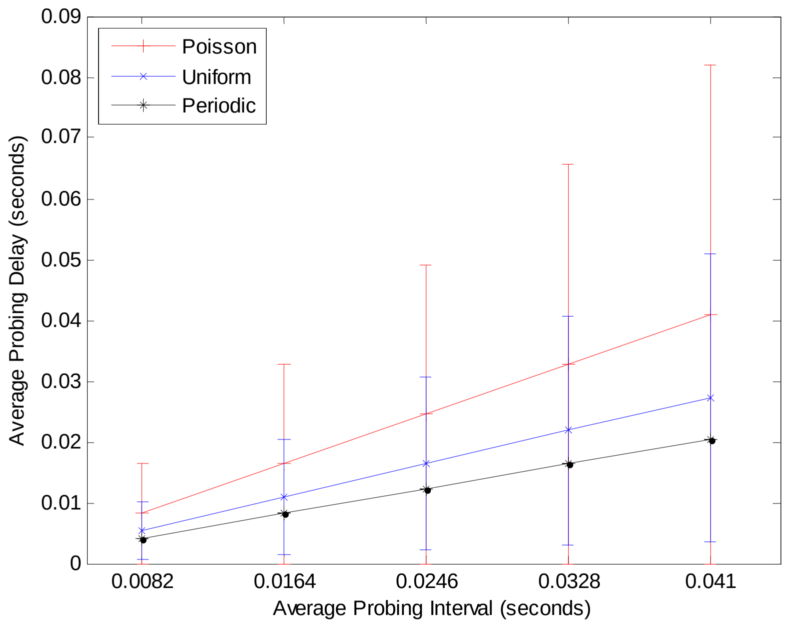


Fig. 9.7. Theoretical analysis of average probing delay versus average probing interval.

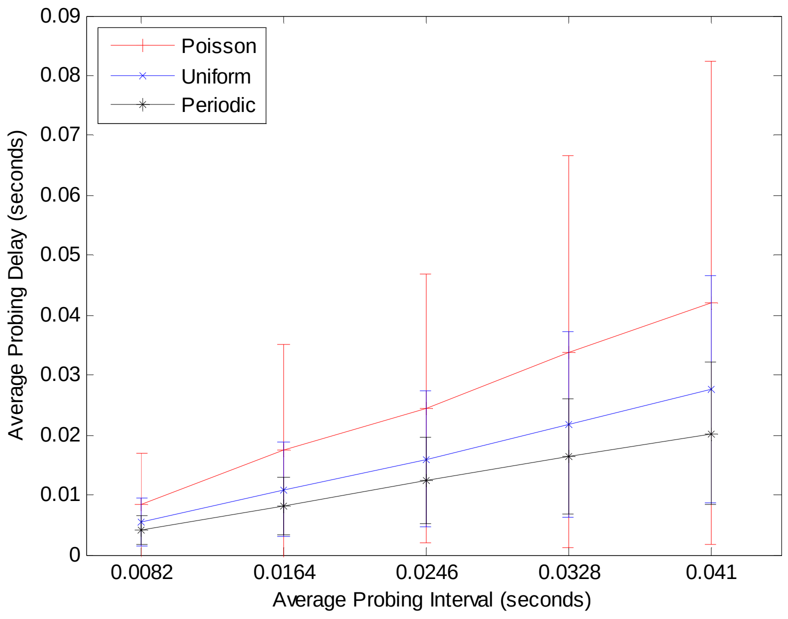


Fig. 9.8. Experimental average probing delay versus average probing interval.

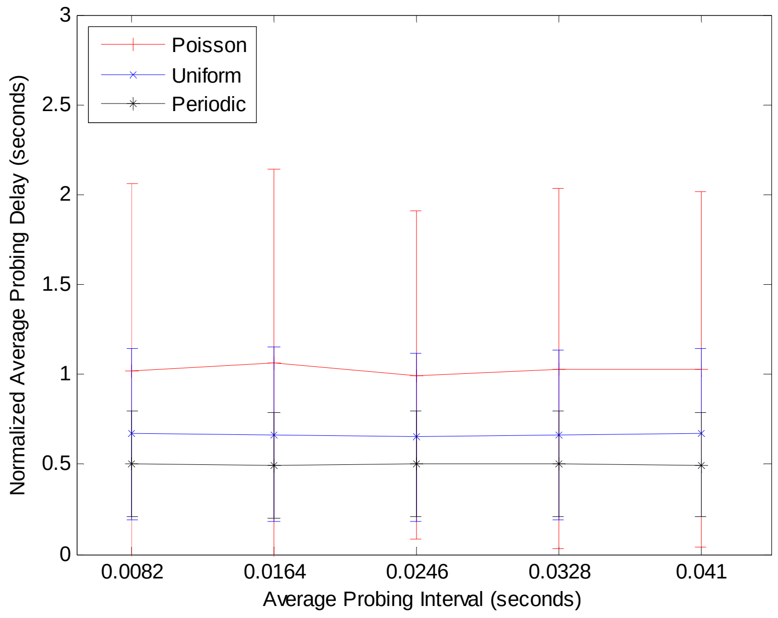


Fig. 9.9. Normalized probing delay versus average probing interval.

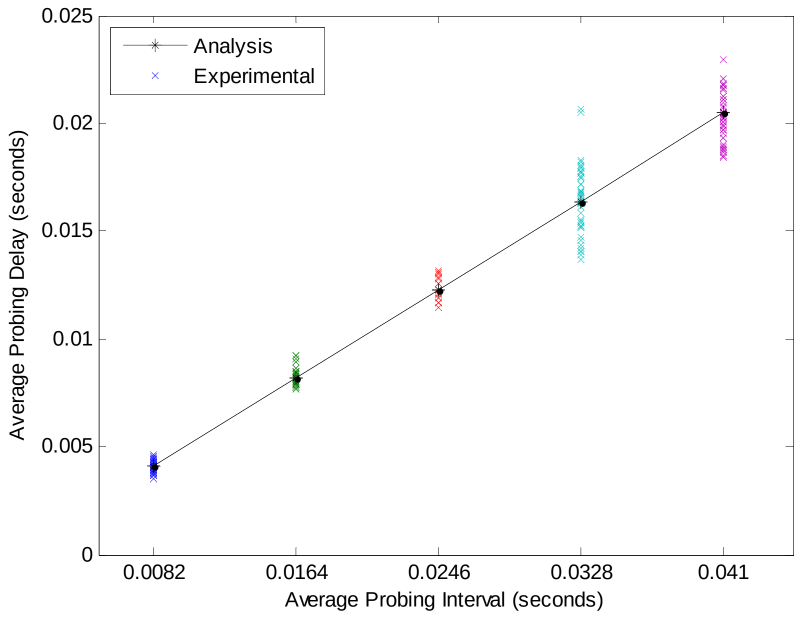


Fig. 9.10. Periodic probing. Average probing delay, experimental versus theoretical analysis; data from 50 users.

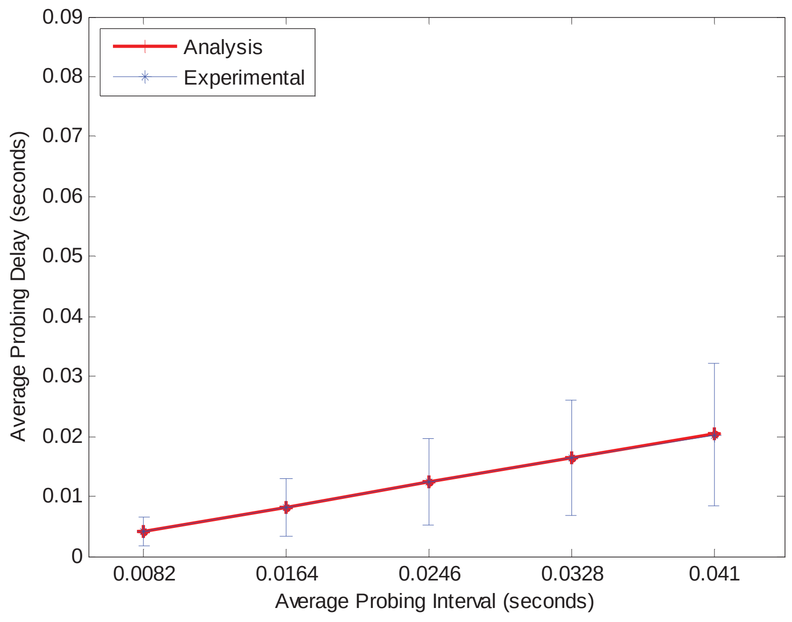


Fig. 9.11. Periodic probing. Average probing delay, experimental versus theoretical analysis, average of 50-user data set.

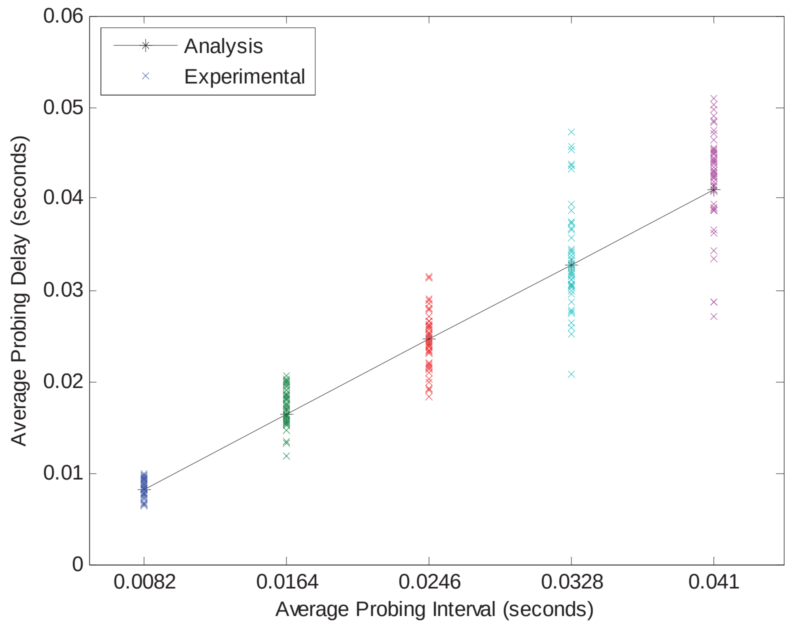


Fig. 9.12. Poisson probing. Average probing delay, experimental versus theoretical analysis; raw data from 50 users.

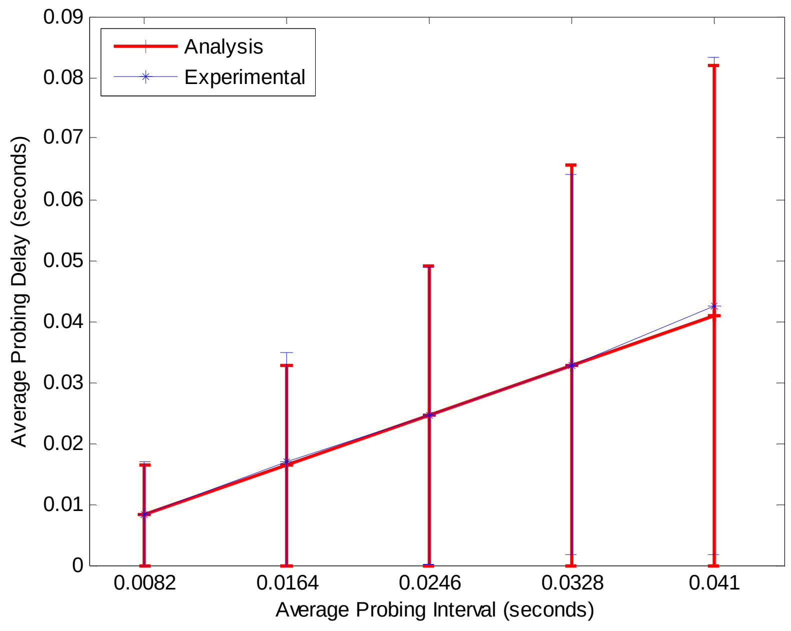


Fig. 9.13. Poisson probing. Average probing delay, experimental versus theoretical analysis.

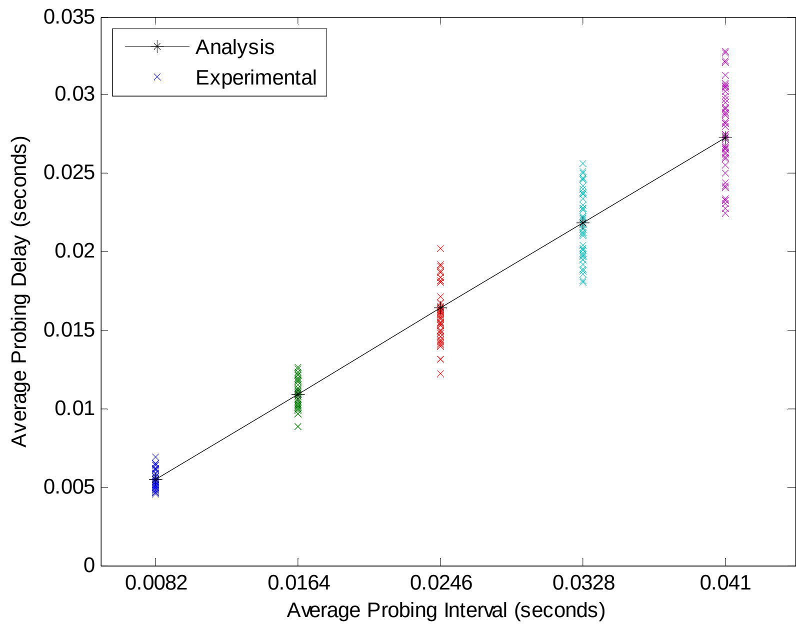


Fig. 9.14. Uniform probing. Average probing delay, experimental versus theoretical analysis; data from 50 users.

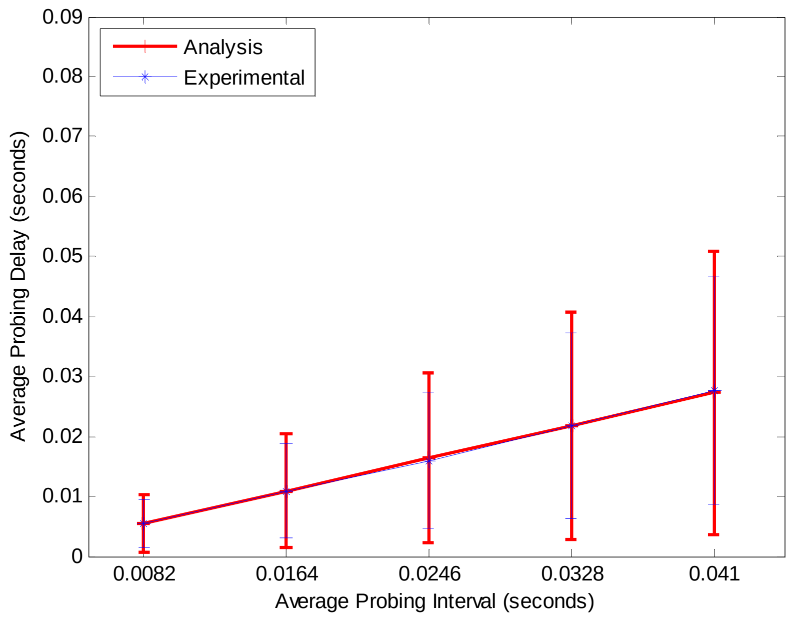


Fig. 9.15. Uniform probing. Average probing delay, experimental versus theoretical analysis.

### 9.2.2 Cooperative sensing results

As *k* grows larger, the number of combinations required to be enumerated and computed grows exponentially as seen in Figure 9.16. For simplicity and efficiency of the cooperative probing experiments the number of cooperative users was limited to 15.

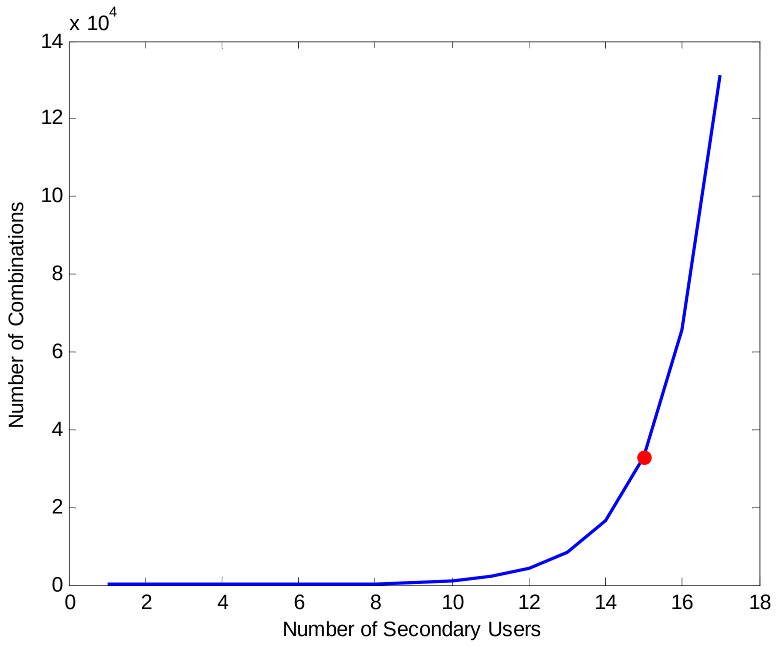


Fig. 9.16. Exponential growth of SU combinations required to compute, combinations of 15 SUs indicated.

Figure 9.17 shows the theoretical performance comparison between periodic, uniform random and Poisson random probing methods. It can be seen from Figure 9.17 that periodic probing provides the smallest average probing delay for a small number of cooperative users. However as the number of cooperative users increases, the random probing methods provide similar performance to periodic probing.

Figure 9.18 shows the experimental results comparing the performance of cooperative CRNs employing periodic, uniform random and Poisson random probing methods. Following the trend of the theoretical results, the experimental results suggest that while periodic probing provides the smallest probing delay for a small number of cooperative users, the random probing methods provide similar performance. The performance comparisons between the experimental and theoretical probing delays for each individual probing method can be seen in figures 9.19-9.21.

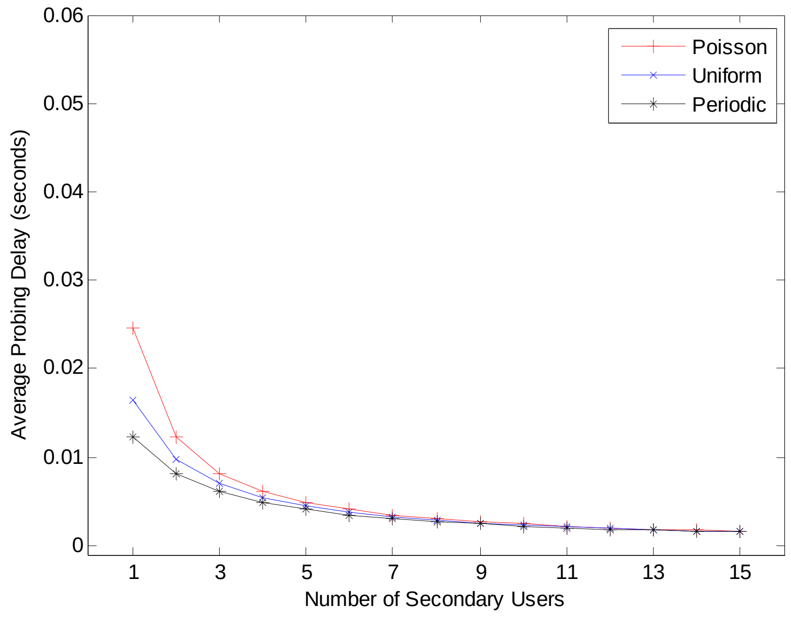


Fig. 9.17. Theoretical analysis of average probing delay of spectrum probing methods under cooperative sensing scenario.

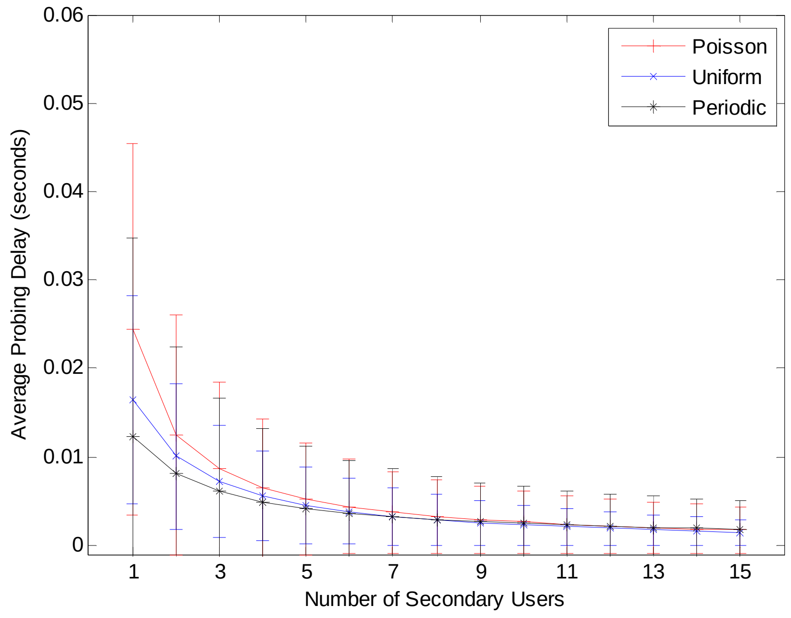


Fig. 9.18. Experimental average probing delay of spectrum probing methods under cooperative sensing scenario.

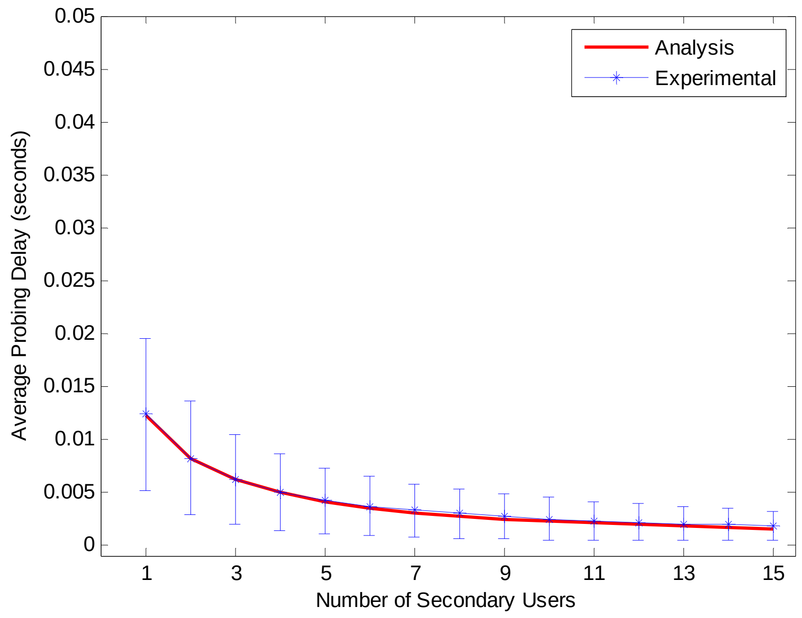


Fig. 9.19. Periodic probing. Average probing delay, experimental versus theoretical analysis, for the cooperative sensing scenario.

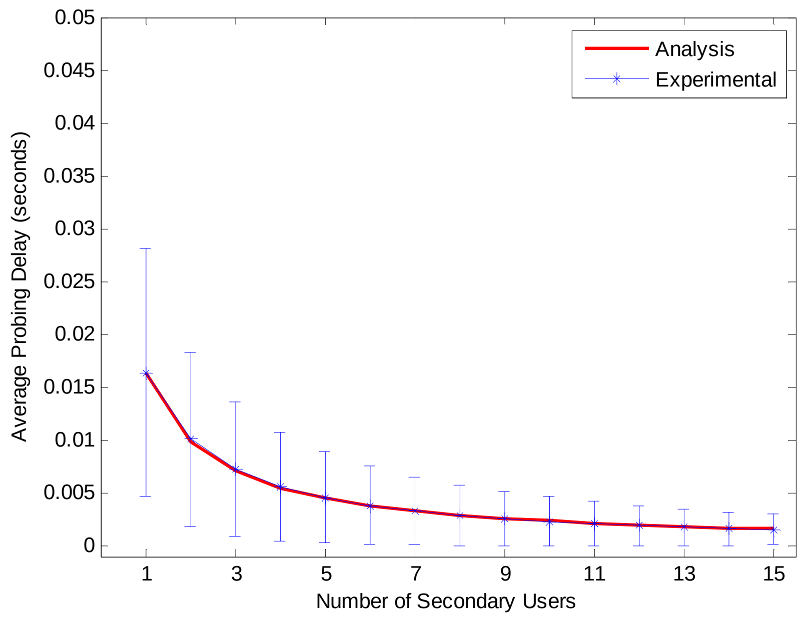


Fig. 9.20. Uniform probing. Average probing delay, experimental versus theoretical analysis, for the cooperative sensing scenario.

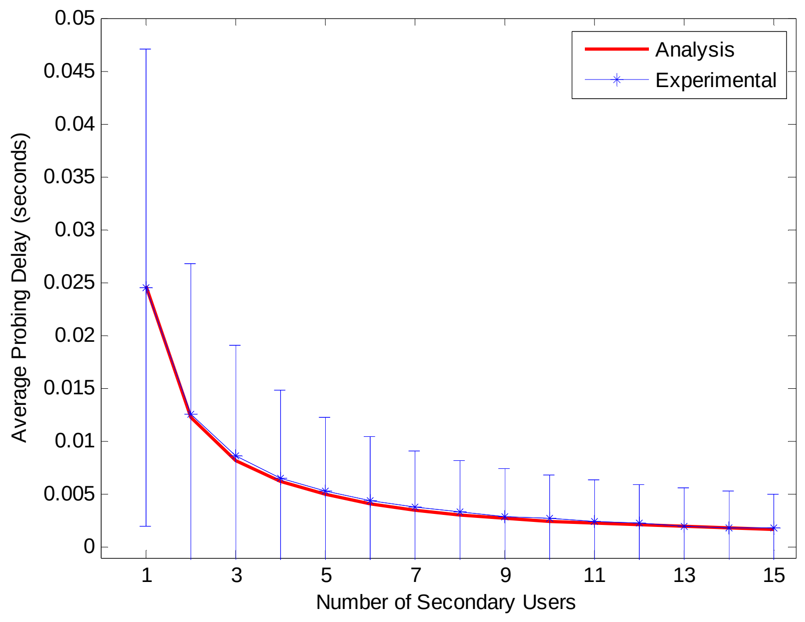


Fig. 9.21. Poisson probing. Average probing delay, experimental versus theoretical analysis, for the cooperative sensing scenario.

Figure 9.22 shows the average probing delay of the different spectrum probing methods normalized to the average probing delay of the periodic method. From Figure 9.22 it is shown that as the number of cooperative users exceeds seven, uniform random probing performs equal to or better than periodic probing. For Poisson random probing, 13 or more cooperative users are required to perform equal to or better than periodic probing. It should be noted that depending on the probing period used and the detection time requirements, performance differences between the three probing methods may be considered negligible with just a few cooperative users.

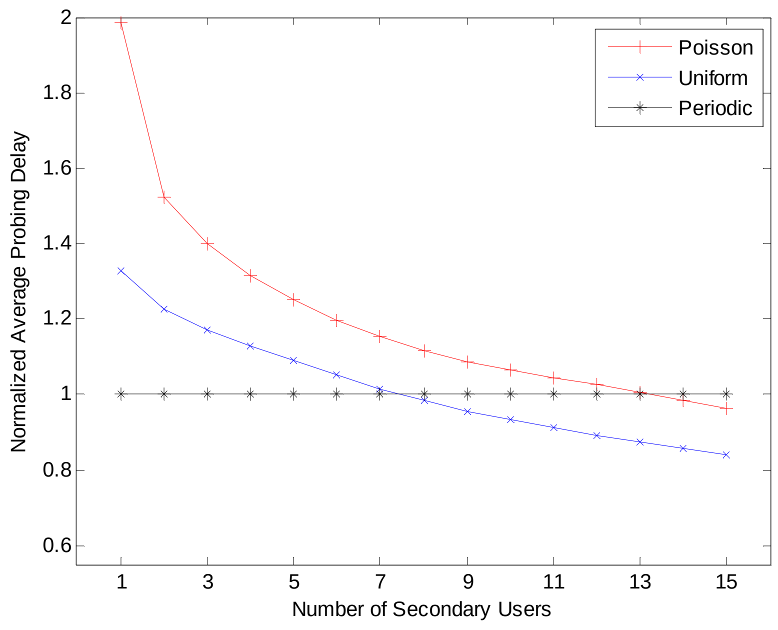


Fig. 9.22. Average probing delay of cooperative spectrum probing methods normalized to the average probing delay of the periodic method.

# 10.CONCLUSIONS AND FUTURE WORK

A software defined radio testbed has been developed for the IPFW Wireless Technology Center enabling future experimental research in cognitive radio. This work has detailed the hardware and software design of the testbed and provided background information on resources used to develop the testbed.

Compared to other popular USRP energy detector implementations such as the *usrp_spectrum_sense.py* implementation, the Simulink implementation of this testbed has been optimized for use in energy detection. The total processing time for energy detection in 1 MHz of spectrum is less than 128µs (per unit dwell time) for the Simulink implementation compared to the 3ms (per unit dwell time) processing time of *usrp_spectrum_sense.py*. This improvement in processing time is due to the reduction in overheard in Simulink and the elimination of non-essential processing. This overhead can be reduced in GNU Radio through modification of the lower level C++ code in the GNU Radio blocks; however, making these modifications requires much more time and coding effort and reduces the overall design flexibility.

The optimized Simulink energy detector implementation of the testbed has been used for research in spectrum probing mechanisms, specifically in the measurement of probing delay. The testbed's performance has been verified by comparing the results of the experimental data collected with the testbed to analytical and simulation results. The implementation of the testbed in this spectrum probing research further supports and substantiates the claims and findings in [1]. Specifically that under the same power budget on spectrum probing, periodic probing allows a SU to detect a channel change with the minimum delay compared to Poisson and uniform random probing methods. However, if SUs in a CRN collaborate in detecting the channel change probing mechanisms with some randomization can reduce the probing delay, especially when the number of SUs is large, without an increase in power consumption.

## 10.1 Future Work

The development of this SDR testbed enables the IPFW WTC to perform experimental research on software defined radio, cognitive radio and dynamic spectrum access. The IPFW WTC testbed has been designed to be a platform for future research topics in the area of cognitive radio and dynamic spectrum access. This section describes opportunities for future research utilizing the IPFW WTC testbed.

### 10.1.1 Probing delay under imperfect detection

The testbed in this work has been used to evaluate the performance of spectrum probing methods for single SUs and for a network of SUs under the scenario of perfect detection. The scenario of imperfect detection has been introduced in this work but has not been sufficiently studied. The imperfect detection scenario is where the case of cooperative sensing provides the greatest performance advantage over individual sensing. The testbed can be implemented under the imperfect detection scenario in order to further study this performance advantage.

### 10.1.2 A DSA-enabled CR transceiver

This work has focused on the spectrum opportunity identification and detection aspects of DSA. Future work can expand on this and implement the remaining aspects of DSA, spectrum opportunity tracking and exploitation, to build a DSA-enabled cognitive radio transceiver. The GNU Radio design of the testbed can be used as the foundation for building such a transceiver. Implementation of a DSA-enabled CR using the testbed could also be expanded to include and ontology.

### 10.1.3 Advanced spectrum sensing methods

The spectrum sensing method implemented in this work is the energy detector. As previously stated, while the energy detector is the simplest spectrum sensing method to implement it has performance limitations. Two additional popular spectrum sensing methods have been presented in this work, coherent detection and cyclostationary feature detection. Implementation of either of these spectrum sensing methods could be achieved using the testbed.

### 10.1.4 Spectrum sensing with radio learning

One of the most important aspects of CR is the ability for the radio to learn [9]. The testbed can be used to implement a method of learning from spectrum sensing data to improve spectrum probing delay and spectral hole utilization. For instance, the testbed could be used to sense the spectrum and attempt to fit the channel state data to a probability model in an effort to add a feed-forward element to the CR testbed. Additionally, the testbed could attempt to learn PU activity patterns and potential “quiet” periods to provide another source of information for determining when to access the spectrum.

## LIST OF REFERENCES

# LIST OF REFERENCES

[1] C. Chen, Z. Chen, T. Cooklev, and C. Pomalaza-Ráez. "On Spectrum Probing in Cognitive Radio Networks: Does Randomization Matter?," Proc. of IEEE International Conference on Communications, Cape Town, South Africa, May 2010.

[2] M. Dillinger, K. Madani and N. Alonistioti, *Software Defined Radio: Architectures, Systems and Functions.* Wiley & Sons, 2003, pp. xxxiii.

[3] F. Hamza, (2008). *The USRP Under 1.5X Magnifying Lens* http://gnuradio.org/redmine/attachments/129/USRP_Documentation.pdf.

[4] T. Rondeau, *GNU Radio the open source software radio* http://gnuradio.squarespace.com/about.

[5] Karlsruhe Institute of Technology, *Simulink-USRP: Universal Software Radio Peripheral (USRP) Blockset,* http://www.cel.kit.edu/english/usrp.php.

[6] I. Akyildiz, W.-Y. Lee, M. Vuran, and S. Mohanty, "Survey on Spectrum Management in Cognitive Radio Networks" *IEEE Communications Magazine*, vol. 46, issue 4, 2008.

[7] I. Akyildiz, W.-Y. Lee, M. Vuran and S. Mohanty, "NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey" *Computer Networks 50*, 2127-2159, 2006.

[8] M. Nekovee, "Impact of Cognitive Radio on Future Management of Spectrum," *International Conference on Cognitive Radio Oriented Wireless Networks and Communications*, pp. 1-6, 2008.

[9] J. Mitola, *Cognitive Radio - An Integrated Agent Architecture for Software Defined Radio*, Ph.D. dissertation, The Royal Institute of Technology, Kista, Sweden, 2000.

[10] Federal Communications Commission, "Notice of proposed rule making and order: Facilitating opportunities for flexible, efficient, and reliable spectrum use employing cognitive radio technologies," ET Docket No. 03-108, Feb. 2005.

[11] D. Cabric, S. Mubaraq Mishra, R. Brodersen, "Implementation Issues in Spectrum Sensing for Cognitive Radios" *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, 2004, pp. 772-776.

[12] Y.-C. Liang, H.-H. Chen, J. Mitola III, P. Mähönen, R. Kohno, J. Reed, "Cognitive Radio: Theory and Application," *IEEE Journal on Selected Areas in Communications*, vol. 26, issue 1, 2008.

[13] Q. Zhao and B. Sadler, "A Survey of Dynamic Spectrum Access: Signal Processing and Networking Perspectives," *IEEE International Conference on Acoustics, Speech and Signal Processing,* vol. 4, pp. IV-1349-IV-1352, 2007.

[14] Federal Communications Commission, "Notice of proposed rule making: Unlicensed operation in the TV broadcast bands," ET Docket No. 04-186 (FCC 04-113), May 2004.

[15] M. Marcus, "Unlicensed cognitive sharing of TV spectrum: the controversy at the federal communications commission," *IEEE Communication Magazine,* vol. 43, no. 5, pp. 24-25, 2005.

[16] Y. Zhao, L. Morales, J. Gaeddert, K. K. Bae, J.-S. Um, and J. H. Reed, "Applying radio environment maps to cognitive wireless regional area networks," in *Proc. IEEE Int. Symposium on New Frontiers in Dynamic Spectrum Access Networks*, Dublin, Ireland, Apr. 2007, pp. 115-118.

[17] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," *IEEE Communications Surveys & Tutorials*, vol. 11, issue 1, pp. 116-130, 2009.

[18] H. Kim and K.G. Shin, "Efficient Discovery of Spectrum Opportunities with MAC-Layer Sensing in Cognitive Radio Networks," *IEEE Transactions on Mobile Computing*, vol. 7, issue 5, pp. 533-545, 2008.

[19] Z. Sun, J. N. Laneman, and G. Bradford, "Sequence Detection Algorithms for Dynamic Spectrum Access," presented at IEEE Symposium on New Frontiers in Dynamic Spectrum Access, 2010.

[20] A. Mariani, A. Giorgetti and M. Chiani, "Energy Detector Design for Cognitive Radio Applications," *International Waveform Diversity and Design Conference*, 2010, pp. 53-57.

[21] H. Wang, G. Noh, D. Kim, S. Kim and D. Hong, "Advanced Sensing Techniques of Energy Detection in Cognitive Radios," *Journal of Communications and Networks*, vol. 12, issue 1, pp. 19-29, 2010.

[22] S. Ziafat, W. Ejaz, H. Jamal, “Spectrum sensing techniques for cognitive radio networks: Performance analysis,” presented at IEEE MTT-S International Microwave Workshop Series on Intelligent Radio for Future Personal Terminals, 2011.

[23] Beaulieu, N.C., Y. Chen, “Improved Energy Detectors for Cognitive Radios with Randomly Arriving or Departing Primary Users,” *IEEE Signal Processing Letters*, vol. 17, issue 10, pp. 867-870, 2010.

[24] H. Urkowitz, “Energy detection of unknown deterministic signals,” *Proceedings of the IEEE*, vol. 55, issue 4, pp. 532-531, 1967.

[25] D.D. Ariananda, M.K. Lakshmanan, H. Nikoo, “A survey on spectrum sensing techniques for Cognitive Radio,” *Second International Workshop on Cognitive Radio and Advanced Spectrum Management*, pp. 74-79, 2009.

[26] H. Tang, “Some physical layer issues of wide-band cognitive radio systems,” *Proc. IEEE Int. Symposium on New Frontiers in Dynamic Spectrum Access Networks*, pp.151–159, 2005.

[27] S. Ross, “Introduction to Probability Models,” eighth edition, Academic Press.

[28] E. Peh and Y.-C. Liang, “Optimization for Cooperative Sensing in Cognitive Radio Networks,” *IEEE Wireless Communications and Networking Conference*, pp. 27-32, 2007.

APPENDICES

# A. GNU RADIO SUPPLEMENTAL INFORMATION

## A.1 A Transmitter with GNU Radio and the USRP

A basic USRP-based GNU Radio transmitter system is comprised of a source block, an encoding block, a modulator block, an interpolating block and a USRP sink block. GNU Radio must provide the source data to be transmitted. A variety of source blocks in GR allows the data that is to be transmitted to be imported from many different file types including binary files, text files and audio files. Alternatively the data can be generated from within GR as a vector of data or as a signal such as a discrete-time sine wave. GR has blocks that can be used to encode or packetize the source data prior to modulation if required. Data must be modulated prior to transmission. Several modulation block sets are available that include varieties of frequency-shift keying (FSK), phase-shift keying (PSK), quadrature amplitude modulation (QAM), and orthogonal frequency division multiplexing (OFDM) that can be used to modulate the data. If the baseband sampling rate is not compatible with the USRP the data must be interpolated to a rate that is compatible. Finally the modulated in-phase and quadrature (IQ) data are transferred to the USRP via USB.

## A.2 A Receiver with GNU Radio and the USRP

Similarly to a transmitter system, a USRP-based receiver system can be built in GNU Radio. In a receiver system the USRP is the source of the data. The in-phase and quadrature (IQ) samples are transferred across USB from the USRP to the host computer. The host computer receives the IQ samples and all signal processing on the data is performed on the host computer. A simple receiver is comprised of a USRP source block, a band-limited filter block, a decimator block, a demodulator block, a decoding block and a sink block.

## A.3 The GNU Radio Companion

The GNU Radio Companion (GRC) is graphical environment for building GNU Radio flow graphs and generating flow graph source code. GRC provides the user with the ability to create a flow graph by connecting graphical blocks that represent the GNU Radio software blocks without the need for writing any software. GRC is a useful tool for quickly building and implementing a USRP-based SDR. In Figure A.1 a differential quadrature phase-shift keying (DQPSK) receiver is implemented. The receiver flow graph blocks function as follows:

**UHD: USRP Source** – Configures a USRP as the source of data and sets the USRP center frequency, gain, antenna selection and sampling rate.

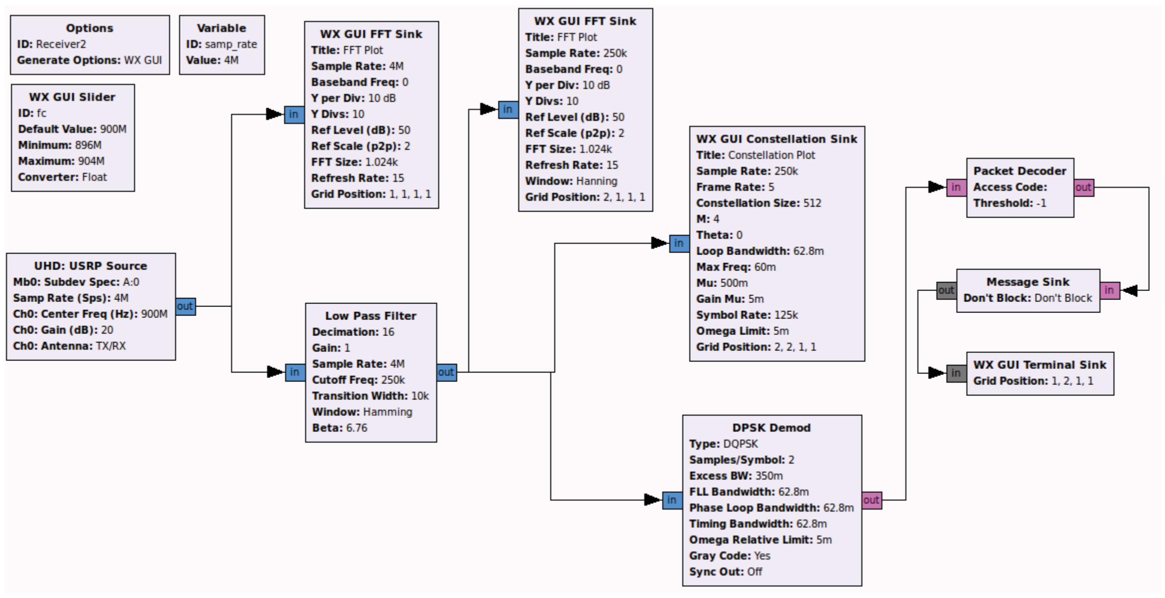
**Low Pass Filter** – Filters out of band interference and decimates the signal by a factor of sixteen to reduce the processing load on the host PC.

**DPSK Demod –** Synchronizes with the modulated data and demodulates the data packets.

**Packet Decoder –** Extracts the source data from the packetized data that was transmitted.

**Graphical Sinks –** The FFT Sink, Constellation Sink and Terminal Sink blocks are graphical user interface (GUI) sinks that display data to the user.

Figure A.2 shows an example of the GUI outputs while the flow graph is running.



**Options**
**ID:** Receiver2
**Generate Options:** WX GUI

**Variable**
**ID:** samp_rate
**Value:** 4M

**WX GUI Slider**
**ID:** fc
**Default Value:** 900M
**Minimum:** 896M
**Maximum:** 904M
**Converter:** Float

**UHD: USRP Source**
**Mb0: Subdev Spec:** A:0
**Samp Rate (Sps):** 4M
**Ch0: Center Freq (Hz):** 900M
**Ch0: Gain (dB):** 20
**Ch0: Antenna:** TX/RX

**WX GUI FFT Sink**
**Title:** FFT Plot
**Sample Rate:** 4M
**Baseband Freq:** 0
**Y per Div:** 10 dB
**Y Divs:** 10
**Ref Level (dB):** 50
**Ref Scale (p2p):** 2
**FFT Size:** 1.024k
**Refresh Rate:** 15
**Grid Position:** 1, 1, 1, 1

**Low Pass Filter**
**Decimation:** 16
**Gain:** 1
**Sample Rate:** 4M
**Cutoff Freq:** 250k
**Transition Width:** 10k
**Window:** Hamming
**Beta:** 6.76

**WX GUI FFT Sink**
**Title:** FFT Plot
**Sample Rate:** 250k
**Baseband Freq:** 0
**Y per Div:** 10 dB
**Y Divs:** 10
**Ref Level (dB):** 50
**Ref Scale (p2p):** 2
**FFT Size:** 1.024k
**Refresh Rate:** 15
**Window:** Hanning
**Grid Position:** 2, 1, 1, 1

**WX GUI Constellation Sink**
**Title:** Constellation Plot
**Sample Rate:** 250k
**Frame Rate:** 5
**Constellation Size:** 512
**M:** 4
**Theta:** 0
**Loop Bandwidth:** 62.8m
**Max Freq:** 60m
**Mu:** 500m
**Gain Mu:** 5m
**Symbol Rate:** 125k
**Omega Limit:** 5m
**Grid Position:** 2, 2, 1, 1

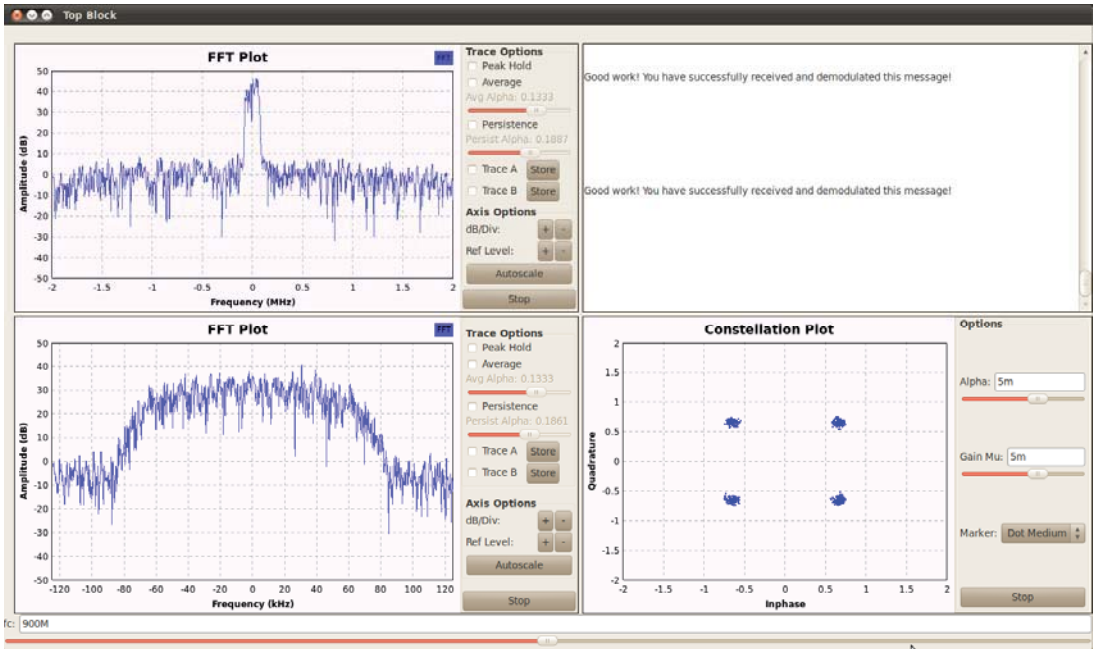
**DPSK Demod**
**Type:** DQPSK
**Samples/Symbol:** 2
**Excess BW:** 350m
**FLL Bandwidth:** 62.8m
**Phase Loop Bandwidth:** 62.8m
**Timing Bandwidth:** 62.8m
**Omega Relative Limit:** 5m
**Gray Code:** Yes
**Sync Out:** Off

**Packet Decoder**
**Access Code:**
**Threshold:** -1

**Message Sink**
**Don't Block:** Don't Block

**WX GUI Terminal Sink**
**Grid Position:** 1, 2, 1, 1

Fig. A.1. DQPSK USRP-based SDR receiver.



Good work! You have successfully received and demodulated this message!

Good work! You have successfully received and demodulated this message!

Avg Alpha: 0.1333
Persist Alpha: 0.1887

Avg Alpha: 0.1333
Persist Alpha: 0.1861

Alpha: 5m
Gain Mu: 5m
Marker: Dot Medium

fc: 900M

Fig. A.2. GUI outputs from the DQPSK receiver in Figure A.1.

# B. USER-DEFINED MATLAB SCRIPTS AND FUNCTIONS

## B.1 Matlab Embedded Function: *Detector*

The *Detector* embedded Matlab function is used to complete the energy detection process for the Simulink energy detector model. The scaled FFT magnitude-squared data are input into the *Detector* block in order to perform the averaging and threshold comparison operations of the energy detector. Data are input to the *Detector* block at a rate equal to the USRP sampling rate, divided by the USRP decimate factor, divided by the data vector length. For this work a decimation factor of 64 and a vector length of 128 were used which results in a Simulink data rate of 7.8125k samples per second ([64e6/64]/128). The *Detector* block decides whether to implement the remaining energy detector operations on the data or to ignore the data based on an iterative loop which is determined by the probing interval. In order to maintain proper flow, Simulink requires the data that are to be stored in the Matlab workspace to be assigned a value every sampling period. The *Detector* block assigns the value of *NaN* (not a number) to data that it does not process through the energy detector operations so that the Simulink flow of data is not broken and the stored data of interest are simple to filter out in post processing. If the iterative loop count limit has been reached the *Detector*

```
function [E,State,Tpr]= Detector(u,t)
%#eml
persistent loopcnt Tp Tdwell sync delay temp Esum
if isempty(loopcnt)
    loopcnt=0;
    Tp = 16;
    Tdwell = 8;
    sync=0;
    delay=0;
    temp=0;
    Esum=0;
end

threshold = 2e+8;
if delay == 0
    if loopcnt > Tp
        Esum = Esum + u;
        if sync == 0
            Tp = 1;
            if u >= 2.5e7
                E=u;
                sync = 1;
                delay = temp;
                State=1;
                Tpr=t;
            else
                E = NaN;
                Tpr = NaN;
                State = NaN;
            end
        else
            if loopcnt > Tp+Tdwell
                %Tp = -log(rand())*192; % Poisson Random
                Tp = randunifd(1,15)*32;  % Uniform Random
                %Tp = 256;                % Periodic
                loopcnt=0;
                E = Esum/Tdwell;
                Esum = 0;
                Tpr = t;
                if E >= threshold
                    State = 1;
                else
                    State = 0;
                end
            else
                E = NaN;
                Tpr = NaN;
                State = NaN;
            end
        end
    else
        E = NaN;
        Tpr = NaN;
        State = NaN;
    end
```

```
else
    delay = delay-1;
    E = NaN;
    Tpr = NaN;
    State = NaN;
end

temp = randunifd(1000,4000);
loopcnt = loopcnt+1;
```

## B.2 Matlab Script: *simulink_spect_probing_UI*

The *simulink_spect_probing_UI* script is used to automate the data collection process, process the data from the Simulink model and compute the probing delays.

```
% FIRST RUN THE ENERGY DETECTOR WITH Tp=1 AND periodic. THIS WILL BE
% USED AS THE act_time AND act_state DATA.
% ONCE THE actual DATA ARE COLLECTED, CHANGE Tp TO THE PROBING PERIOD
% AND RUN THE EXPERIMENT. RETAIN THE act DATA THROUGHOUT THE ENTIRE
% EXPERIMENT
clearvars -except act_state act_time act_tedges
itt=1; % ITERATION COUNTER
for a=1:20
a
tic
sim('usrp_energy_detector_03152012',50);

t = nanstrip(probe_time); % STRIP ALL NaN ENTRIES IN DATA
state = nanstrip(Channel_State);
E = nanstrip(Energy);
datastart = find(E>2e7,1);
state=state(datastart:end);
t=t(datastart:end) - t(datastart);
E=E(datastart:end);

%state = glitchfix(state); % ONLY USE FOR IMPERFECT DETECTION

% FIND THE EDGES IN THE PROBED DATA AND ALIGN THE SYNC PULSE TO ZERO
[edges,tedges,state_aligned,t_aligned]=edgefind(state,t);

% DEFINE THE actual DATA. RUN ONLY ONCE WITH MINIMUM Tp
% act_time = t_aligned;
% act_state = state_aligned;
% act_tedges = tedges;
% PLOT THE PROBED DATA VS. ACTUAL CHANNEL DATA
 scatter(t_aligned,state_aligned,'r','.')
 hold on
 plot(act_time,act_state)
 axis([0 50 -.01 1.5])
 ylabel('Channel State 1=On, 0=Off')
 xlabel('Time (sec)')
 title('Measurement of Probing Delay')

% COMPUTE AND STORE THE PROBING DELAYS FOR EACH ITERATION
delays(:,itt) = (abs(tedges-act_tedges))';
mean_delays = mean(delays);
mean_Tprobe(itt) = mean(diff(t_aligned(80:end)));
itt = itt+1;

save('FILENAME)
clearvars -except act_state act_time act_tedges delays start_delay
mean_Tprobe itt

toc
pause(52-toc)
end
```

## B.3 Matlab Function: *edgefind*

The *edgefind* function is used to remove all probed data prior to the synchronization pulse and align the time component of the data accordingly. Finally the script identifies and returns the times of the state transitions or edges in the data.

```
function [edges,tedges,state_aligned,t_aligned] = edgefind(state, t)
if nargin == 1  % IF ONLY INPUT IS state, JUST RETURN THE EDGES
    b=1;
    for a=1:length(state)-1
        if state(a) ~= state(a+1)
            edges(b)=a+1;
            b=b+1;
        end
    end
else
    b=1;
    ind = find(state,1,'first');
    ind=1;
    state_aligned=state(ind:end);   % REMOVE ALL LEADING DON'T CARE
DATA
```

```
    t_aligned = t((ind):end)-t(ind);% REMOVE ALL LEADING DON'T CARE
DATA

    % RETURN THE EDGES, TIME OF EDGES AND TRUNCATED AND
    % ZERO-ALIGNED TIME AND STATE DATA
    for a=1:length(state_aligned)-1
        if state_aligned(a) ~= state_aligned(a+1)
            edges(b)=a+1;
            tedges(b)=t_aligned(a+1);
            b=b+1;
        end
    end
    edges(1)=[];
    tedges(1)=[];
end
```

## B.4 Matlab Function: *coopsensing*

The *coopsensing* function is used to enumerate all combinations of SUs and return the cooperative average probing delay and standard deviation for *N* cooperative users.

```
function [D var] = coopsensing(delays, N)
% FUNCTION USED TO ENUMERATE ALL COMBINATIONS OF SUs AND COMPUTE THE
% COOPERATIVE AVERAGE PROBING DELAY AND STANDARD DEVIATION (STDEV)

if nargin == 1
    N = 15;
end

C = delays(:,1:N);
B = mean(C);
A=1:length(B);
combo = [];

for a = 2:length(A)     % NUMBER OF USERS
    nck = combnk(A, a); % COMPUTE THE COLUMN COMBOS OF USER SUBSETS
                        % FROM TOTAL DATA
                        % nck CONTAINS THE COMBINATIONS OF columns
                        % of C TO USE
    [combs, nums] = size(nck); % NUMBER OF COMBOS AND SIZE OF COMBOS

    for b = 1:combs     % DO FOR ALL COMBOS
        for c = 1:nums  % GRABS THE APPROPRIATE COLUMNS
                        % OF DATA FOR THE COMBO
            combo(:,c) = C(:,nck(b,c)); % nck(b,c) IS A POINTER TO
                                        % WHICH COLUMN IN C TO USE
        end

        mins(:,b) = min(combo,[],2);    % FINDS THE MIN PROBING DELAY
                                        % FOR EACH EDGE OUT OF THE
                                        % COLUMNS IN THE COMBO
        combo=[];
    end
    D(a) = mean(mean(mins));            % AVERAGE THE MINIMUM PROBING
                                        % DELAY FROM ALL EDGES
    var(a) = mean(std(mins));           % AVERAGE THE STDEV
    mins = [];
end
D(1) = mean(mean(C));
var(1) = mean(std(C));
```

## B.5 Matlab Function: *glitchfix*

The *glitchfix* function is used to analyze probing data in the imperfect detection scenario. The metric of interest for this work is the probing delay, which only exists at actual channel state transitions. Therefore, if the energy detector makes a misdetection other than at a channel state transition, it can be considered a misdetection or a “glitch” in the data and not of particular interest. Furthermore, these glitches cannot be properly analyzed by the *edgefind* function and prevent proper calculation of the probing delay. Therefore for this work it was decided to eliminate these glitches.

```
function [filtered,glitches,Pr] = glitchfix(state)
filtered=state;
glitches=0;
detections = length(find(state==1));
 for a=(2:length(state))
    if filtered(a) ~= filtered(a-1)     % TEST FOR CHANGE IN STATE
        if filtered(a) == 0             % TEST IF CHANGE IS TO A ZERO
            if sum(filtered(a:a+8)) ~= 0   % IF THE NEXT 5 ARE NOT ALL
                                           % ZERO, THIS IS A GLITCH
                glitches = glitches+1;
                filtered(a) = 1;
            end
        end
    end
end
Pr = glitches/detections;
```