

Building a Dynamic Spectrum Access Smart Radio With Application to Public Safety Disaster Communications

Mark D. Silvius

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

Charles W. Bostian, Co-Chair
Allen B. MacKenzie, Co-Chair
Tonya Smith-Jackson
Luiz DaSilva
Yaling Yang

August 13, 2009
Blacksburg, Virginia

Keywords: Cognitive Radio, Dynamic Spectrum Access, Protocols, Rendezvous,
Spectrum Sharing, GNU Radio, OMNeT++, Testbed, Public Safety

Copyright 2009, Mark D. Silvius

Building a Dynamic Spectrum Access Smart Radio With Application to Public Safety Disaster Communications

Mark D. Silvius

ABSTRACT

Recent disasters, including the 9/11 terrorist attacks, Hurricane Katrina, the London subway bombings, and the California wildfires, have all highlighted the limitations of current mobile communication systems for public safety first responders. First, in a point-to-point configuration, legacy radio systems used by first responders from differing agencies are often made by competing manufacturers and may use incompatible waveforms or channels. In addition, first responder radio systems, which may be licensed and programmed to operate in frequency bands allocated within their home jurisdiction, may be neither licensed nor available in forward-deployed disaster response locations, resulting in an operational scarcity of usable frequencies. To address these problems, first responders need *smart radio* solutions which can bridge these disparate legacy radio systems together, can incorporate new smart radio solutions, or can replace these existing aging radios. These smart radios need to quickly find each other and adhere to spectrum usage and access policies. Second, in an infrastructure configuration, legacy radio systems may not operate at all if the existing communications backbone has been destroyed by the disaster event. A communication system which can provide a new, temporary infrastructure or can extend an existing infrastructure into a *shaded region* is needed. Smart radio nodes that make up the public safety infrastructure again must be able to find each other, adhere to spectrum usage policies, and provide access to other smart radios and legacy public safety radios within their coverage area.

This work addresses these communications problems in the following ways. First, it applies cognitive radio technology to develop a smart radio system capable of rapidly adapting itself so it can communicate with existing legacy radio systems or other smart radios using a variety of standard and customized waveforms. These smart radios can also assemble themselves into an ad-hoc network capable of providing a temporary communications backbone within the disaster area, or a network extension to a shaded communications area. Second, this work analyzes and characterizes a series of rendezvous protocols which enable the smart radios to rapidly find each other within a particular coverage area. Third, this work develops a

spectrum sharing protocol that enables the smart radios to adhere to spectral policies by sharing spectrum with other primary users of the band. Fourth, the performance of the smart radio architecture, as well as the performance of the rendezvous and spectrum sharing protocols, is evaluated on a smart radio network testbed, which has been assembled in a laboratory setting. Results are compared, when applicable, to existing radio systems and protocols. Finally, this work concludes by briefly discussing how the smart radio technologies developed in this dissertation could be combined to form a public safety communications architecture, applicable to the FCC's stated intent for the 700 MHz Band. In the future, this work will be extended to applications outside of the public safety community, specifically, to communications problems faced by warfighters in the military.

Dedication

I dedicate this dissertation to my family, whose encouragement helped make this endeavor possible.

Acknowledgment

My work was supported by the Air Force Institute of Technology (AFIT), by the National Institute of Justice, Office of Justice Programs, U.S. Department of Justice under Award No. 2005-IJ-CX-K017, and by the National Science Foundation under Grant No. CNS-0519959. In addition, I would like to thank the Software Defined Radio (SDR) Forum for its support of this research. The views, findings, and conclusions or recommendations expressed in this chapter are those of the author and do not necessarily reflect the views of these sponsors or the official policy or position of the U.S. Air Force, the Department of Defense or the U.S. Government. Lastly, I would like to gratefully acknowledge the contributions of my colleagues at Virginia Tech and in the wider cognitive research community, and I would like to thank my graduate student peers in the Center for Wireless Telecommunications (CWT) research laboratory whose daily interaction and assistance made this document possible.

Contents

1. Introduction	1
1.1 Observations: The Public Safety Disaster Communications Scenario	1
1.2 Problem Statement: Radio Discovery and Spectrum Scarcity in Public Safety Operations	3
1.3 Recent Developments	3
1.4 Presented Solution: Smart Radio and Dynamic Spectrum Access Protocols for Improved Public Safety Communications	5
1.5 Methodology	6
1.6 Contributions	6
1.7 Organization of this Dissertation	7
2. Smart Radio Architecture 2007	8
2.1 Problem Statement	8
2.2 Contributions	10
2.3 Previous Work: Historic and Literature Review	11
2.4 Smart Radio 2007 Architecture	13
2.5 Results	23
2.6 Conclusions	24
3. Smart Radio Architecture 2008	25
3.1 Problem Statement	25
3.2 Contributions	28
3.3 Previous Work	29
3.4 Smart Radio 2008 Architecture	30

3.5	Results	39
3.6	Conclusions	40
4.	Smart Radio Network Testbed	42
4.1	Problem Statement	43
4.2	Contributions	44
4.3	Previous Work: Review of Smart Radio Architecture	44
4.4	Testbed Architecture and Setup	46
4.5	Experiments: Validation of Universal Framework Architecture	51
4.6	Results	52
4.7	Discussion	57
4.8	Conclusions	58
5.	Rendezvous Protocol	59
5.1	Contributions	60
5.2	Previous Work	60
5.3	Rendezvous Design Taxonomy	64
5.4	Problem Statement	72
5.5	Experiment	83
5.6	Conclusions	97
6.	Channel Change Protocol	98
6.1	Problem Statement	99
6.2	Contributions	100
6.3	Previous Work	101
6.4	Presented Solution	107

6.5	Methodology for Evaluation	110
6.6	Simulation Setup	111
6.7	Simulation Results	116
6.8	Testbed Experiments Setup	119
6.9	Testbed Experiment Results	120
6.10	Validation of Simulation Results with Experimental Results	125
6.11	Stability of Experimental Results	126
6.12	Summary	131
7.	Conclusions	132
7.1	Summary	132
7.2	Contributions	133
7.3	Publications	135
7.4	Application: The 700 MHz Public Safety Band	136
7.5	Future Work	137
	Bibliography	139
	Appendix – Scripts	147

List of Figures

2.1	Two communicating CWT SR secondary users and a single FRS radio primary user.	11
2.2	Block diagram of the CWT Smart Radio.	14
2.3	User interface showing four states of operation.	17
2.4	FRS database.	18
2.4	A basic SDR system based on GNU Radio and USRP.	20
2.5	Radio Framework Multi-threading Control.	21
2.6	PHY/MAC Radio Framework.	22
3.1	Problem Scenario and Requirements.	28
3.2	CWT Smart Radio 2008 Architecture.	30
3.3	Scan Tab of the User Interface.	32
3.4	Connection Tab.	32
3.5	Configuration Tab.	32
3.6	An example of XML messaging used in radio configuration.	33
3.6	An example of XML configuration for two adapters which are bridged together.	34
3.7	The principal components so the Universal Radio Framework.	37
3.8	Diagram highlight the Bluetooth scatternet formation process.	39
4.1	Scenario and proposed solution.	43
4.2	Architecture of the smart radio.	45
4.3	Detailed architecture of the network testbed.	47
4.4	Screenshot of JPerf.	49
4.5	Remote SSH terminals control network configuration of each test node.	50
4.6	Java-based “NetworkStats” configuration agent.	51

4.7 Summary of Latencies.	53
4.8 Summary of TCP Goodputs.	54
4.9 Summary of UDP Goodputs.	55
4.10 Summary of UDP Jitters.	56
4.11 Summary of UDP Jitters (linear scale).	56
4.12 Summary of UDP Packet Loss.	57
5.1 Node state information for the single channel, synchronous case.	72
5.2 Fraction of undiscovered neighbor $\log_{10}(1-F)$ versus the number of nodes, N , for the single-channel, synchronous case.	76
5.3 Node state information for the multi-channel, synchronous case.	77
5.4 Fraction of undiscovered neighbor $\log_{10}(1-F)$ versus the number of channels, M , for four nodes ($N=4$), for the multi-channel, synchronous case.	79
5.5 Node state information for the multi-channel, asynchronous case.	81
5.6 Node state information for the multi-channel, asynchronous case with $W=2$ slot repetitions.	82
5.7 Detailed architecture of the network testbed.	84
5.8 Multi-Channel, Asynchronous, Experimental Results, $M=1$	89
5.9 Multi-Channel, Asynchronous, Experimental Results, $M=1$	89
5.10 Multi-Channel, Asynchronous, Experimental Results, $M=2$	90
5.11 Multi-Channel, Asynchronous, Experimental Results, $M=2$	90
5.12 Multi-Channel, Asynchronous, Experimental Results, $M=5$	91
5.13 Multi-Channel, Asynchronous, Experimental Results, $M=5$	91
5.14 Multi-Channel, Asynchronous, Experimental Results, $M=10$	92
5.15 Multi-Channel, Asynchronous, Experimental Results, $M=10$	92
5.16 Multi-Channel, Asynchronous, Experimental Results, $M=20$	93

5.17 Multi-Channel, Asynchronous, Experimental Results, M=20.	93
5.18 Multi-Channel, Asynchronous, Experimental Results, M=35.	94
5.19 Multi-Channel, Asynchronous, Experimental Results, M=35.	94
5.20 Multi-Channel, Asynchronous, Experimental Results, M=50.	95
5.21 Multi-Channel, Asynchronous, Experimental Results, M=50.	95
5.22 Comparison of my optimum transmit probability results, Silvius (experimental) to that of Borbash (theoretical).	96
5.23 Comparison of my expected number of discoveries results, Silvius (experimental) to that of Borbash (theoretical).	96
6.1 Testing two secondary users and a single primary user node.	98
6.2 A 802.11 WiFi secondary user' UDP goodput is degraded during the transmission of a primary user with a GMSK waveform, between t=30s to t=40s.	100
6.3 An example of a GNU Radio flowgraph.	102
6.4 Time plot showing the conceptual operation of the CCP.	109
6.5 Modifications to GNU Radio protocol stack to support the CCP.	110
6.6 The FSM representation of the MAC layer in OMNeT++.	113
6.7 OMNeT++ simulation scenario with six secondary users and one primary user.	115
6.8-9 The CCP reduced duration of received interference as measured in four nodes.	117
6.10-11 The CCP reduced latency between packet arrival times	118
6.12 The CCP increased goodput.	118
6.13 The CCP decreased packet loss.	118
6.14 Detailed architecture of the network testbed.	119
6.15 UDP Goodput during GNU Radio Experiment.	121
6.16 UDP Jitter during GNU Radio Experiment.	122

6.17 UDP Packet Loss Rate during GNU Radio Experiment.	122
6.18 Latency during GNU Radio Experiment.	123
6.19 Threshold Detection Reliability for a Given Offered Rate into Secondary User Network	124
6.20 Simplified smart radio network testbed.	126
6.21 Averaged goodput traces.	129
6.22 Averaged jitter traces.	129
6.23 Averaged packet loss curves.	130
6.24 Averaged latency curves.	131

List of Tables

2.1 Operational features provided by CWT Smart Radio.	24
4.1 Equipment used to build network testbed.	46
4.2 Required software to install using Synaptic Package Manager or from web.	48
5.1 Rendezvous Design Taxonomy.	64
5.2 Applying Rendezvous Design Taxonomy to Related Works.	71
5.3 Experiment Configuration Settings.	84
5.4 Optimum transmit probability for a given value of W and M	85
5.5 Expected number of discovered per minislot, $E(h)/W$, for a given combination of p_T , W , and M	86
5.6 Expected number of discovered per slot for a given combination of p_T , W , and M	86
5.7 Average time to discover various fractions of available links per the number of channels, using the identified optimum p_T and W values from Table 5.4.	86
5.8 Optimum transmit probability, p_T , for a given value of W and N single channel, $M=1$	87
5.9 Expected number of discovered per minislot, $E(h)/W$, for a given combination of p_T , W , and N for single channel, $M=1$	87
5.10 Expected number of discovered per slot, $E(h)$, for a given combination of p_T , W , and N for single channel, $M=1$	87
5.11 Comparison of my results (experimental) to that of Borbash (theoretical).	87
6.1 Parameters Used in OMNeT++ Simulations.	116
6.2 QoS Improvements with CCP Technique over Baseline.	125
6.3 Averaged QoS Metrics.	127
6.4 Averaged QoS Metric Change.	128

Chapter 1

Introduction

This chapter introduces the current state of mobile communications in use by public safety first responders. Section 1.1 describes typical operational scenarios faced by first response officers. Section 1.2 summarizes these situations and states the problem which this research addresses. Section 1.3 introduces recent research developments. Section 1.4 states the presented solution to these problems. Sections 1.5 and 1.6 explain the methodology and the contributions of this research. Section 1.7 explains the organization of this dissertation.

1.1 Observations: The Public Safety Disaster Communications Scenario

Recent disasters, including the 9/11 terrorist attacks, Hurricane Katrina, the California wildfires, and the Midwest floods, have highlighted the limitations of current mobile communication systems for public safety first responders. Often, these radio systems are deficient in fundamental features needed in a joint response environment. First, in a point-to-point configuration, legacy radio systems used by first responders from differing agencies are often made by competing manufacturers and may use incompatible waveforms or channels. In addition, first responder radio systems, which may be licensed and programmed to operate in frequency bands allocated within their home jurisdiction, may be neither licensed nor available in forward-deployed federal disaster response locations, resulting in an operational scarcity of usable frequencies. Second, in an infrastructure configuration, legacy radio systems may not operate at all if the existing communications backbone has been destroyed by the disaster event. A communication system is needed to provide a new temporary infrastructure or to extend an existing infrastructure into a *shaded region*. Smart radio nodes that make up the public safety infrastructure again must be able to find each other, adhere to spectrum usage policies, and provide access to other smart radios and legacy public safety radios within their coverage area.

The challenges of a public safety disaster scenario have received significant attention from researchers in the wireless communication area. During the last two years, the Software Defined Radio (SDR) Forum has sponsored a university challenge to address the inadequacies of current public safety communications systems [1]. In the 2007 scenario, a large city experiences a major, crippling earthquake. The city and first responders face a near total loss of the previously existing fixed communication infrastructure. First responders must construct a makeshift command center and establish a temporary communication infrastructure for both point-to-point communications between individual responders, as well as to and from a central mobile command-post. Compounding this obstacle is the challenge of incorporating the communication radios from responding agencies across the country, each with their own assortment of frequency allocations, modulations schemes, and message formats. The competition seeks to demonstrate that these requirements can be addressed by using the capabilities brought to the scene by *smart radio* SDRs. *Smart radios* can overcome the interoperability issue by their ability to adapt to and rendezvous with a vast array of disparate radio systems. They can identify vacancies in the spectrum and can share the use of the spectrum with other radio users. A smart radio is the tool of choice for first responders in this earthquake and other disaster scenarios. In the 2008 challenge scenario, a team of first responders arrives on the scene following a major incident in a city subway. The smart radios' task is to extend communications throughout the subway tunnel. To accomplish this goal, the smart radios must form a mobile ad-hoc network to establish a connection with the *hidden radio* at the end of the tunnel. Data traffic generated by the *hidden radio* at the end of the subway must be relayed, via a series of *repeater radios*, to the tunnel's entrance. These repeater radios form a dynamic, multi-hop, ad-hoc network capable of transmitting pictures, video, and voice messages produced by the first responder. A *command radio* must act as the *ad-hoc extension*, bridging the ad-hoc network within the tunnel to the fixed communications infrastructure present outside of the tunnel.

1.2 Problem Statement: Radio Discovery and Spectrum Scarcity in Public Safety Operations

Current first responder communication systems face two key problems. First, they lack the ability to automatically locate and connect with other compatible radio systems within a disaster site. Second, they lack usable spectrum to carry out their mission due to both current policy and inflexible legacy technology, both of which effectively result in spectrum-scarcity. This dissertation presents a solution to both these problems through research and development of smart radio technologies, experimental smart radio network testbeds, and dynamic spectrum access protocols.

Both communication deficiencies have arisen primarily due to the fixed nature of traditional radio system design. Legacy radio systems are often designed with a singular purpose in mind, and once that radio system leaves the manufacturing plant, it rarely can be changed or reprogrammed by the operator. Contributing factors include available technology, economics, and regulation. Solid state components and non-reconfigurable application specific integrated circuits (ASIC) accomplish signal processing tasks well, but they cannot be modified once manufactured. Until recently, more flexible processing platforms, such as general purpose processors (GPP) and field-programmable gate arrays (FPGA) have been cost prohibitive for use in inexpensive hand-held radio systems. Moreover, government regulation of frequency assignment has reinforced the singular nature of radios. Users must apply for a license from the Federal Communications Commission (FCC) for their systems to operate on a specific frequency range only. Those systems also have to be further accredited to ensure that they do not radiate energy outside their authorized bands.

1.3 Recent Developments

Two recent trends may enable new innovative solutions to public safety's communication deficiencies. The first is rapid expansion of smart radio technologies, and the second is a change in policy direction by the FCC.

1.3.1 Smart Radio Technologies

We consider the term *smart radio* to be a general classification that encompasses SDR, frequency agile radios, cognitive radios, and associated technologies [2-5]. For this research, we say that a smart radio is *any sensing radio, programmed to respond to changes in its environment in an innovative way* [6, 7]. It must build upon its underlying SDR platform by adding clever or intelligent algorithms designed to solve specific spectrum access challenges. A smart radio's awareness of the environment and its understanding of user requirements allow it to adapt to dynamic communications scenarios [6].

Smart radio technology has increased in prevalence within the past decade, thanks in part to the availability of inexpensive GPP technology. Signal processing, which traditionally could only be accomplished in dedicated hardware, can now be accomplished using processors similar to those found in traditional desktop computers. The GNU Radio [8] with the Universal Software Radio Peripheral (USRP) radio frequency (RF) front end [9] is one such SDR architecture that can run on a desktop computer and one which I used extensively in this research.

1.3.2 FCC Policies and the 700 MHz Public Safety Band

Traditionally, licensing regulations by the FCC required that a user maintain a license to operate a radio within a specific frequency band. Operation by the user within only this well-defined band greatly simplified the rendezvous and channel assignment issues [7]. However, the current policy trend at the FCC has changed direction toward allowing one or more users to simultaneously share access to particular spectrum band. The FCC announced that it is now adopting new spectrum regulation policies to allow for the growth of *Secondary Markets* [10] and dynamic spectrum access technologies. In this scheme, secondary users may use unoccupied spectrum owned by a primary user, as long as they do not introduce interference to other users. Secondary users may either be unlicensed users who *squat* in unoccupied bands, or they may be registered users who sublease spectrum from a licensed primary user.

A significant example application of dynamic spectrum access and spectrum sharing is FCC's proposed *700 MHz Public Safety System* [11]. In this scenario, a *National Carrier* has a licensed primary access to a 12 MHz portion of the 700 MHz band allocated for broadband

usage. The National Carrier also has secondary, unlicensed access to the remaining portions of the band allocated for narrowband usage, which is owned and licensed to local public safety jurisdictions. In this system, dynamic spectrum access protocols could be utilized as an effective and efficient way for the National Carrier system to detect narrowband FM public safety radios and to coordinate transfers to alternate, unoccupied portions of the 700 MHz band. [12].

1.4 Presented Solution: Smart Radio and Dynamic Spectrum Access Protocols for Improved Public Safety Communications

This dissertation demonstrates that a dynamic spectrum access smart radio can provide a solution to the interoperability and spectrum scarcity problems faced by public safety first responders. More specifically:

Smart radios, built upon reconfigurable software-define radio architecture, and specifically coupled with dynamic spectrum access protocols, with neighbor rendezvous and spectrum sharing capabilities, provide a solution to the radio discovery and spectrum scarcity problems faced by first responders in public safety disaster response scenarios.

To build and field a dynamic spectrum access smart radio, we research four of its key building blocks. First, we design and construct a *smart radio* platform. The system builds on a SDR architecture which enables rapid reconfiguration, and the ability to span multiple bands, and the use of multiple waveforms and protocols. Second, we use individual smart radio nodes to build a network tested. This testbed enables us to measure the performance for the smart radio architecture and as well as the designed dynamic spectrum access protocols. Third, we research the design and development of *rendezvous* protocols. The rendezvous protocols allow one smart radio to rapidly find another smart radio within a band of interest. These protocols address the current interoperability problem of link formation present in legacy wireless systems, and they facilitate the construction of ad-hoc networks. In this work, we review the design of rendezvous protocols already discussed in existing literature and apply a taxonomy to classify these strategies. The review show that more analysis is needed for the multi-channel asynchronous case. We then implement such a protocol on the smart radio network testbed to empirically measure its performance. Fourth, we research a new *Channel Change* spectrum-sharing protocol

that allows secondary users to opportunistically share spectrum with primary users. This protocol addresses the spectrum scarcity problem in legacy communications systems. We implement the Channel Change protocol on the smart radio network testbed to evaluate its performance. In addition, we conclude by discussing how the Center for Wireless Telecommunication's (CWT) Smart Radio and the Rendezvous and Channel Change protocols could be applied to a prototype solution for FCC's new 700 MHz public safety band.

1.5 Methodology

This research solves problems through the use of communication theory, computer simulation, and laboratory experiments on fully-functional radio prototypes. First, we learn from previous peers, literature, and existing systems regarding the optimal design strategies for smart radios. Then, we hypothesize and formulate new strategies to improve performance. Next, we simulate these ideas, use computer modeling tools where possible, and build a working prototype to test. We run laboratory experiments to evaluate the performance of these prototypes, which are also demonstrated at SDR Forum's Smart Radio Challenges. Afterwards, we collect and analyze results to further refine the designs and conducting future experiments. Last, we form conclusions on the success or failure of the smart radio research and development process.

1.6 Contributions

This research makes four primary contributions to the field of cognitive radio and dynamic spectrum access research. First, it produces a working, affordable smart radio prototype for public safety first responders using commercial off-the-shelf equipment. Second, it assembles these smart radios into a network testbed in which to demonstrate the performance of the radio in an ad-hoc network configuration and to evaluate the developed dynamic spectrum access protocols. The value of this testbed, once devised, is that it also provides an educational tool which can be used not only to evaluate protocol performance, but also to teach undergraduate and graduate students communications and SDR principles. This testbed is also used to evaluate the performance of two dynamic spectrum access protocols, Rendezvous and

Channel Change. Third, this work implements and tests different configurations of the Rendezvous protocol, quantifying the expected discovery probabilities for the multi-channel asynchronous problem. Fourth, it researches the Channel Change protocol for opportunistic spectrum sharing, whose straightforward design can be applied to current wireless medium access control (MAC) protocols, and which improves latency, packet loss, and throughput performance by reducing interference between primary and secondary users. All these contributions can be combined with existing CWT technologies to provide a solution for the FCC 700 MHz public safety band.

1.7 Organization of this Dissertation

Chapter 2 presents a review of previous work in cognitive radio, and detail the architecture and design of the CWT Smart Radio 2007, which was entered in SDR Forum's inaugural Smart Radio Challenge competition. We show how this radio satisfies the requirements of the first responders described in the problem statement. Chapter 3 describes the networking improvements made to this architecture and the design details of the CWT Smart Radio 2008, demonstrated at the following year's radio competition. Chapter 4 describes how the individual smart radio nodes were assembled into Smart Radio Network Testbed. The testbed is used to quantify the results of the performance of the radio systems and to evaluate the performance of the developed dynamic spectrum access protocols. Chapter 5 describes the design and evaluation of the Rendezvous protocol, which enables a collection of smart radios to discover each other and initialize communication links. Chapter 6 describes the development and evaluation of the Channel Change protocol, which enables primary and secondary users to share spectrum and co-exist within a given frequency band. Chapter 7 summarizes the key findings of this research, describes applications to the FCC 700 MHz public safety band, and discusses future research plans.

Chapter 2

Smart Radio Architecture 2007

Before beginning my research into dynamic spectrum access protocols, I needed to first design and develop an underlying cognitive radio architecture. The architecture would provide the framework for running and evaluating the dynamic spectrum access protocols and act as an initial prototype communication system, targeted to addressing the communication deficiencies of public safety first responders. As luck would have it, in the fall of 2006, Center for Wireless Telecommunications (CWT) laboratory team was presented with the opportunity to compete in the Software Defined Radio (SDR) Forum's "Smart Radio Challenge 2007." The SDR Forum is a non-profit international industry association, created with the goal of fostering the development of next-generation SDR technologies [13]. Their 2007 radio challenge specifically detailed the communication problem faced by public safety first responders at the scene of an earthquake disaster and became the perfect case study in which to begin our design process.

The organization of this chapter is the following. First, in Section 2.1, I review the specific public safety communications problem statement for the Smart Radio Challenge 2007. This provides the design requirements and the motivation for the development of our CWT Smart Radio 2007 architecture. Section 2.2 then describes the contributions resulting from development of this architecture. Next, Section 2.3 reviews previous work in the field of software defined radio and cognitive radio to give context for our radio. Section 2.4 describes the specific architecture of the CWT Smart Radio 2007, in block by block fashion. Finally, Section 2.5 discusses the results of the competition, and Section 2.6 concludes with a summary of the radio's functionality.

2.1 Problem Statement

In the Smart Radio Challenge 2007 scenario, a large city experiences a major, devastating earthquake. The earthquake destroys many of the city's structures and kills numerous citizens. Complicating the rescue effort, first responders find that the city's entire fixed communication

infrastructure has been destroyed in the disaster. Large portions of the city's power grid have been destroyed or disabled. Cell towers have been knocked down and fixed land-line communications have been cut. First responders arriving on scene must establish their own temporary communications infrastructure within the disaster scene. The infrastructure must connect a makeshift command post to all of the responders within the field, as well as provide a network in which first responders can communicate amongst each other. First responders need a communication infrastructure that supports not only voice, but data transmission, such as digital pictures, video, high resolution maps, and building floor plans.

Since the disaster is on such a large scale, the rescue effort becomes too complex for the city's own police, fire, and emergency medical services. In response to the mayor's request, public safety agencies from all across the country convene on the disaster site in order to assist. This significantly complicates the communication difficulties. Now, personnel must develop a way to connect and manage a large assortment of different radio types from many different agencies. Each agency's radio may use their own combination of frequency allocations, modulation schemes, and message formats.

To meet these challenges, the SDR Forum asserted that smart radios, with their underlying software defined radio architecture, could provide the tools capable of providing solutions to the communication deficiencies faced by public safety first responders. In addition to addressing the broad aspects of the earthquake scenario, the SDR Forum enumerated a number of specific requirements which a proposed smart radio needs to address. First, on startup, each radio needs to be able to find the other public safety radios around it. More specifically, the radio must be able to identify available spectrum within a pre-defined band, rendezvous with an intended receiver, and transmit voice and data using a selected quality of service (QoS). It must interoperate with existing family radio service (FRS) radios using a FM waveform on a standardized FRS channel. It must also be able to create a new communication link with other smart radios using a FRS channel selection and the BPSK, QPSK, or 8PSK modulations. With FM, it must support analog voice communications with legacy hand-held FRS radios, and with digital modulations, it must support IP data services, including a CVSD-based VoIP protocol. The system requires the development of two prototype dynamic spectrum access protocols. The first, a rendezvous protocol, facilitates the detection and initialization of communications links with neighboring smart radios. The second coordinates spectrum sharing between analog

primary users and digital secondary users by applying a simple but effective channel-change protocol. The radio must also support expansion, and whenever possible, provide the functionality to be used in a dynamic spectrum test bed for smart and cognitive radio research [6].

2.2 Contributions

In this section, we summarize the contributions made during the development of the smart radio architecture during the Smart Radio Challenge 2007. These developed technologies also contribute to the field of software defined radio and cognitive radio research. First, we built a new smart radio system architecture. Some components are completely new and designed specifically for use in the new system architecture. Others were developed by previous students in the CWT research laboratory, but adapted for this new architecture. An explanation of both is described in this chapter. Second, of the new components, the new Master Control module is a significant contribution. This module provides the *brains* of the smart radio architecture. It responds to the radio's environment through feedback received from the system sensors, as well as feedback from the radio framework. The Master Control module also runs all subsequent dynamic spectrum access protocol logic. Third, another significant contribution was the building and demonstration of an improved radio framework. The new framework supports higher order modulations like 8PSK, as well as a custom CWT developed CVSD-based VoIP protocol. The new radio framework is also capable of running a *rendezvous* protocol, which facilitates the automatic discovery of adjacent radio nodes. More on the Rendezvous protocol is discussed in Chapter 4. In addition, the smart radio architecture is capable of running a *channel change* dynamic spectrum sharing protocol. The protocols allow the smart radios, acting as secondary users, to share spectrum with other legacy-analog or digital secondary users as shown in Figure 2.4. More is discussed on the Channel Change protocol in Chapter 5.

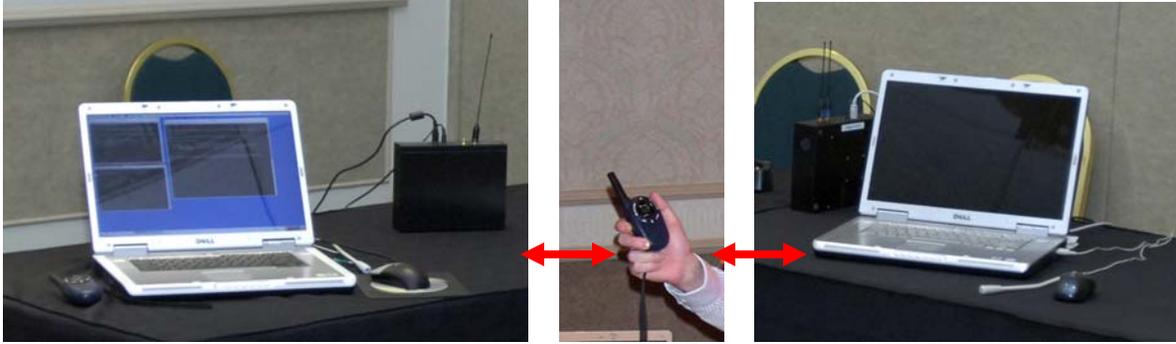


Figure 2.1. Two communicating CWT SR secondary users and a single FRS radio primary user.
[M. Silvius' Image from [6]. Used with permission, © 2008 SPIE.]

2.3 Previous Work: Historic and Literature Review

To understand cognitive radio, we must first understand the origins of software defined radio. There were numerous motivations for moving toward SDR technology particularly in the military and public safety arenas. First, military operations in the 1970s and 1980s began to highlight the stovepipe nature of communications technologies and deficiencies in inter-service communications interoperability. Starting in 1992, the Defense Advanced Research Projects Agency (DARPA) and the US Air Force began a program called *SPEAKeasy*. Its goal was to produce a flexible, reconfigurable platform, incorporating commercial technologies to improve inter-service interoperability issues [14, 15]. *SPEAKeasy* was one of the first attempts by the US military to design a software defined radio. Later, in 1996 Wayne Bonser of AFRL established the SDR Forum, an organization whose charter was to standardize SDR related commercial technologies [15]. The SDR Forum, through its Smart Radio Challenges has been, and continues to be, a sponsor and motivator for much of cognitive radio research conducted at Virginia Tech. In the mid-1990s, the *SPEAKeasy* and subsequent SDRs programs evolved into what is today the Joint Tactical Radios System (JTRS) program. The JTRS continues the US military's work to develop an interoperable software-defined communication platform using commercial technologies. The JTRS uses the Software Communications Architecture (SCA), whose goal is to facilitate an open-software architecture to support legacy components, as well as new, developing commercial technologies and architectures [15]. The development for the SCA is still an active research area for an associated research group, the Mobile and Portable Radio Group (MPRG), here at Virginia Tech.

The next big research breakthrough in the field occurred in 2000 when Mitola presented the first formal description of a *cognitive radio*. He combined the concept of machine learning

and natural language processing with the developing field of SDR technology. He also introduced the Radio Knowledge Representation Language (RKRL) as a means for cognitive radios to communicate with each other and other devices in a network [16, 17]. He envisioned the use of cognitive radio for applications such as spectrum pooling, and explained the etiquette policies that cognitive radio would need to follow to make such a collaborative system possible [18]. This radio etiquette concept provided an inspiration for my own research in dynamic spectrum access protocols for public safety first responders. Mitoloa's work and other developments in cognitive radio are well summarized in [19].

Starting in 2001, the Center for Wireless Communications at Virginia Tech also entered into the area of cognitive radio research. Christian Rieser developed an initial cognitive radio concept for reliable communications in emergency response scenarios [20]. He coined his biologically inspired model of cognition for SDR platforms as the *cognitive engine*. Tom Rondeau continued this work by improving upon the theory and implementation of the cognitive engine [21]. He also further enhanced the implementation of the cognitive engine, by mating it to a SDR architecture built with the GNU Radio [22], an open source software radio package first introduced by Eric Blossom in early 2000. Bin Le's research focused on the complete process of building a cognitive radio node from the ground up [23]. He also conducted extensive research in developing an improved signal detection and classification algorithm for signal awareness within the cognitive radio. He later demonstrated all these contributions by building a prototype radio for emergency first responders, called the *Public Safety Cognitive Radio* [24].

The successful design of our CWT Smart Radio, described in this chapter, has much to thank to the specific work of Tom Rondeau and Bin Le. Tom's vision for a modular architecture, in which his centerpiece cognitive engine connects an assortment of sensors, memory databases, software defined radio framework, to perform the various cognitive radio tasks, forms the model that we emulated when designing our Smart Radio 2007 architecture. We also continued his vision of using standardized XML-based messages to send waveform configurations, commands, and performance feedback data between system blocks over TCP/IP-based sockets. Furthermore, Bin's work in designing a SDR-based radio framework, which transformed XML-based waveform configuration information into real digital signals to be transmitted via USRP, formed the model we used when designing our own radio framework for the Smart Radio 2007. In fact, our radio framework built on many of his contributions, including

his thread-based radio control algorithm and preliminary set analog and digital waveforms, by adding additional digital waveforms and additional provisions for dynamic spectrum access protocol support.

Finally, in 2006, DARPA started the *Next Generation* (XG) program. This military program had goals similar to earlier programs such as SPEAKeasy and JTRS; that is, to refine SDR architecture design and provide improved interoperability. However, XG also included sensors to measure local spectrum usage and to opportunistically borrow or share unused spectrum *whitespace* from secondary users [25].

One of DARPA's most recent communication project out of the Wireless Networks after Next (WNaN) effort is PIRANA [26]. PIRANA stands for "Policy-based Information-centric Reliable Ad hoc Network" [27]. The goal is to develop a hand-held cognitive radio for military personnel whose unit costs falls under \$500 each. The radios will form the cornerstone for the military's next generation, deployable, wireless communication network. The radio designers envision a radio that will be able to identify all cooperative peers in its environment using a rendezvous protocol, and then be able to share spectrum available in that theater of operations with other radio users by employing a series of dynamic spectrum access protocols.

2.4 Smart Radio 2007 Architecture

This section details the design of the smart radio architecture constructed for the SDR Forum Smart Radio Challenge 2007 competition in Denver, Colorado [6]. The CWT Smart Radio contains five main components: the user interface, the master control, the radio framework, the FRS and knowledge databases, and signal detection and classification. In addition, modules for formatting and transferring messages assist in connecting the different components. Figure 2.2 depicts these components and their interactions.

The discussion of the components for the smart radio challenge is divided into two sections. First, we discuss the new components designed specifically for use in new system architecture. Second, we discuss components which were inherited from previous cognitive radio research by graduate students within the CWT laboratory and modified to support the new smart radio architecture.

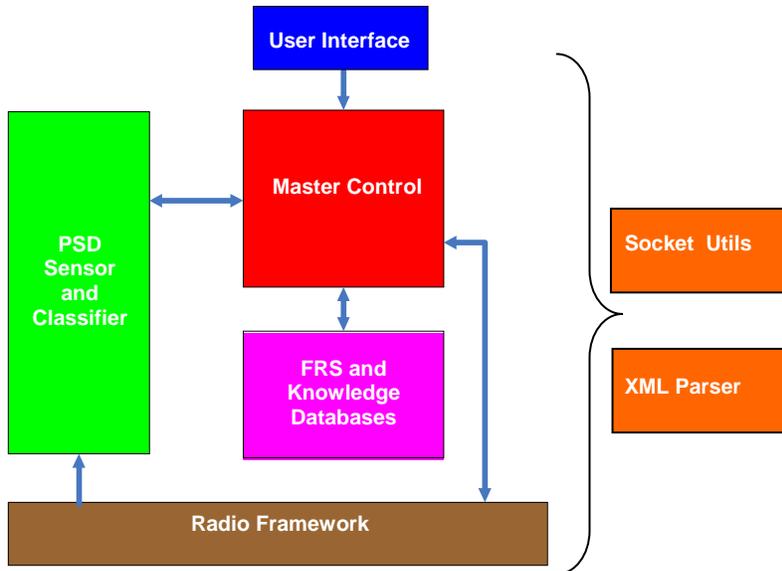


Figure 2.2. Block diagram of the CWT Smart Radio.
 [M. Silvius' Image from [6]. Used with permission, © 2008 SPIE.]

2.4.1 New Smart Radio Components

This section describes new components designed specifically for use in our smart radio system architecture.

2.4.1.1 Master Control

The Master Control (MC) is the *brains* of CWT Smart Radio platform. The MC acts as the centerpiece of the architecture, controlling and coordinating the functions of the entire system. First, it receives basic directions from its operator through the user interface module. Commands from the operator may include scanning the environment to find other radios, or establishing a new communication link with one or more radios using an analog or digital waveform. Upon receipt of these commands, the MC utilizes the other modules to accomplish the operator's objectives. Second, the MC maintains *spectral awareness* of its environment by managing a set of sensors incorporated within the smart radio architecture. The most notable of these sensors is the power spectrum density (PSD) sensor and classifier developed by CWT in a related public safety communication project [24]. The PSD sensor finds sources of transmitted radio signals within range, and the classifier attempts to identify the signal's modulation scheme. Feedback from these sensors is relayed back to the MC for action. Third, the MC enables *spectrum and signal memory* by storing results from the sensors in the knowledge database. By

using this database and a FRS waveform database, the MC can then direct the radio framework to create a new communication link on any designated FRS channel frequency using FM, BPSK, QPSK, or 8PSK modulations. The MC can also direct the initialization of communications links with other neighboring smart radios by running its *rendezvous* protocol. The MC also coordinates spectrum sharing between analog primary users and digital secondary users by running a simple, but effective *channel change* protocol. These two protocols are discussed further in the next section.

2.4.1.2 Dynamic Spectrum Access Protocols

To meet the requirements of first responders, the CWT Smart Radio must be able to rendezvous with other smart radios in its operating environment and share spectrum with legacy FRS users. Moreover, the CWT Smart Radio must automatically switch channels in response to a dynamically changing radio spectrum environment, so that it does not interfere with any other active communication traffic. The CWT Smart Radio accomplished these tasks by using two dynamic spectrum access protocols: *rendezvous* and *channel change*. These protocols' logic runs in the MC module, react to feedback collected by the signal sensors, and provide the innovative logic that transforms its unintelligent SDR platform into a complete *smart* radio.

We first introduced the rendezvous protocol in [7], reviewing related neighbor discovery algorithms, and presenting an initial theoretical framework. The rendezvous protocol provides the logic to allow a CWT Smart Radio to quickly locate another smart radio in the 462-467 MHz spectrum band. The rendezvous protocol facilitates the detection and initialization of communications links with neighboring smart radio nodes by broadcasting a series of frequency-hopped rendezvous beacons which contain node-specific information. Both the rendezvous protocol, and the beacon messages which it transmits, are configured in the radio by using XML setup files.

For the 2007 competition, our radio demonstrated a simplified prototype version of the rendezvous protocol, in which one node was designated as the master, and the other as a slave. The master node, after performing a spectrum scan, would choose a vacant channel, and then begin transmitting beacon messages. The slave node would scan and classify the radio spectrum with the PSD sensor and classifier. Short message frames classified as BPSK would be

identified as rendezvous beacons by the slave node. Both the slave node and the master node, would then tune to the channel used by the rendezvous beacon to begin communications.

The second of the two dynamic spectrum access protocols used in the CWT Smart Radio, is called the Channel Change protocol (CCP). CCP is a spectrum sharing protocol that allows primary and secondary users to co-exist in the same frequency band without interference. Specifically, the CCP enables the CWT Smart Radio to detect the return of an analog FRS primary user, and to orchestrate the radio reconfiguration procedures to change to an alternate, vacant block of spectrum. The CCP is divided into two main components. The first, *primary user detection*, is a set of algorithms that continually monitor the current channel in order to locate the return of the primary user. The second, *fallback execution*, is made of those algorithms which halt the current communications, select a vacant channel from a list of *fallback channels*, reconfigure the smart radio framework to operate that new waveform, and restart and resume communications. More design details and experimental performance results pertaining to the operation of the rendezvous protocol are given in Chapter 4 and more details of the channel change protocol are given in Chapter 5.

2.4.2 Updated Smart Radio Components

This section discusses components which were inherited from previous cognitive radio research within the CWT laboratory and were modified to support the new smart radio architecture.

2.4.2.1 User Interface

The user interface (UI) acts as the control panel from which a public safety first responder can direct the smart radio's operations. At startup, the UI prompts the user to select one of four primary states, which correspond to the radio's principal functions, as shown in Figure 2.3. State 0 puts the CWT Smart Radio in idle mode to save power. State 1 scans and classifies the environment, looking for any active signals such as analog and digital waveforms, in addition to rendezvous beacons transmitted by other CWT Smart Radio nodes. State 2 actively transmits beacons using the rendezvous protocol, so as to initiate communication with CWT Smart Radio nodes. State 3 actively communicates with another FRS or smart radio using a FM, BPSK, QPSK, or 8PSK waveform.

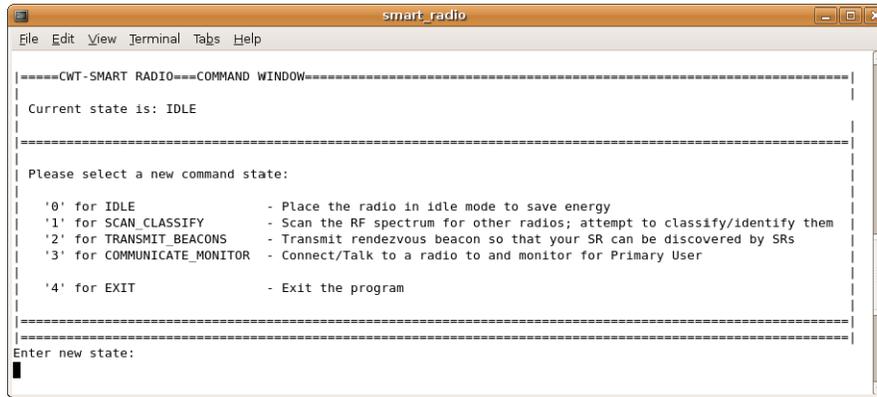


Figure 2.3. User interface showing four states of operation.
 [M. Silvius' Image from [6]. Used with permission, © 2008 SPIE.]

2.4.2.2 FRS and Knowledge Databases

The CWT Smart Radio features two memory structures to store waveform information for transmission and reception. First, the FRS database is used to store the FCC-approved, prototypical waveform definitions for FRS radios. These definitions are employed when the CWT Smart Radio establishes a new communications link with legacy, analog FRS radios. The FRS database accepts a channel number as input and generates an XML file that specifies the waveform associated with the channel. If the smart radio needs to communicate with a legacy FRS system, the MC block can poll the FRS database for specifics to generate a waveform on a desired channel. Figure 2.4 shows a conceptual diagram highlighting the database's purpose. Second, the CWT Smart Radio expands the concept of the knowledge database [24] to store the actual, measured waveform information for radio signals that it has detected and classified in the environment. The knowledge database is similar in design to the FRS database, except that it maintains information on radio signals detected and classified in the environment. To accomplish this, it saves the results of a PSD sensor and classifier into a Python dictionary database. This information can later be recalled by the MC module when it needs to connect to an existing communications link with either an FRS radio or a smart radio node. The MC also uses this database when it needs to quickly identify spectral whitespace to establish a new communications link, or in the case of the channel change protocol, coordinate the relocation to a new vacant channel upon the return of a primary user.

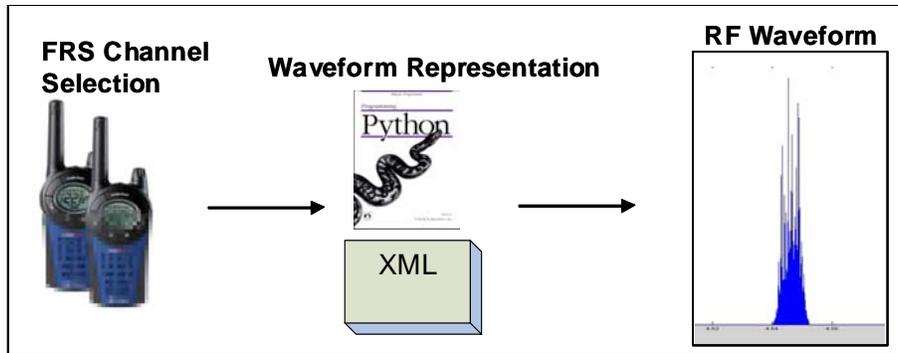


Figure 2.4. FRS database.

[M. Silvius' Image from [6]. Used with permission, © 2008 SPIE.]

2.4.2.3 Inter-Block Data Exchange

The CWT Smart Radio expands on the data standardization and interoperability concepts first introduced in [28, 29]. It relies on XML message formatting and TCP/IP sockets for inter-block communication. It readily supports the addition of future XML file types and a consolidated Python package of socket utilities to assist in communications between CWT Smart Radio components.

2.4.2.4 Radio Framework

The radio framework is the foundation on which all the components rest. The module is responsible for the actual signal processing computations required for converting analog voice or packetized digital data information into physical waveforms capable of being transmitted and received over the air. The radio framework also encompasses all the algorithms associated with the configuration and control of this conversion process. In this section, we review the key aspects of the radio framework used for the Smart Radio Challenge 2007 competition. These include the GNU Radio and USRP SDR platform, the configuration and control algorithms, and the physical and medium access control (MAC) layers.

GNU Radio and USRP

The CWT Smart Radio realizes its radio functions by building on the capabilities of GNU Radio and the USRP device. GNU Radio is an open source toolkit for assembling software radios [22]. It was developed in early 2000 by Eric Blossom and others and has evolved into a mature software-defined radio infrastructure used and supported by a large community of

developers. It was originally designed to run on general purpose processors (GPPs) and minimal analog radio hardware. It allows for software radio development of waveforms, modulations, protocols, signal processing, and other communications functions. The current release of GNU Radio already contains an extensive signal processing library which has been packaged in a set of operational and developmental blocks. These blocks perform such functions as: analog and digital waveform modulators, demodulators, filters, and amplifiers and input and output file operators. Programming in the GNU Radio platform uses a combination of C++ and Python, a simple, high-level language. The computationally intensive processing blocks are implemented in C++, while the control and coordination of these blocks for applications that sit on top are developed in Python [6].

The USRP is an open source, low-priced SDR hardware platform which implements radio front-end functionality, as well as analog-to-digital conversion (ADC) and digital-to-analog conversion (DAC), currently using the Universal Serial Bus (USB2) to connect to the computer that hosts the device. The current USRP device consists of a motherboard containing up to four high-speed 12-bit 64 MSps ADCs, four high-speed 14-bit 64 MSps DACs, an Altera field-programmable gate array (FPGA) and a programmable Cypress FX2 USB 2.0 controller. The ADCs, DACs, and the FPGA together provide support for intermediate frequency (IF) processing. The FPGA on the board provides four digital up-converters (DUC) and four digital down-converters (DDC) to shift frequencies from the baseband to the required frequency. The FPGA can be reprogrammed to provide additional functionality. Radio frequency (RF) front ends are attached in the form of daughter cards which can currently cover all the existing radio bands from 0 Hz to 2.4 GHz. The USRP is fully supported by the GNU Radio library [6]. A combined system of both is shown in Figure 2.4.

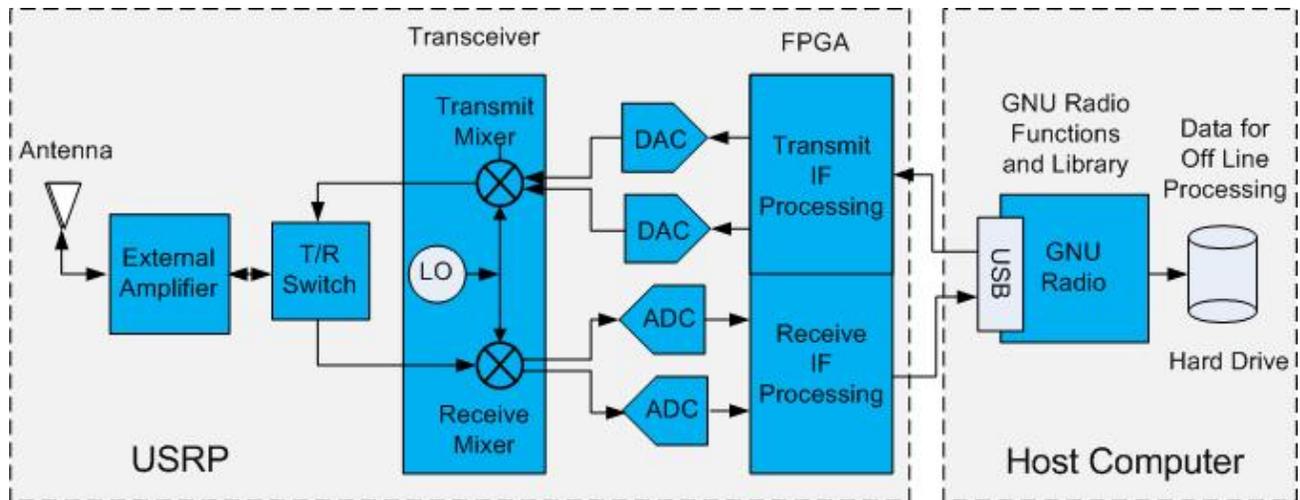


Figure 2.4. A basic SDR system based on GNU Radio and USRP.
 [F. Ge's Image from [5]. Used with permission, © 2008 IEEE.]

Configuration and Control

The smart radio framework improves upon the configuration and control architecture pioneered by Bin Le, an earlier CWT student who worked to apply cognitive and software defined radio technologies to public safety communications [23]. The architecture uses three independent software threads, written in the Python software language, to manage the data flow and signal processing processes within the framework. This can be visualized as the three nested loops in Figure 2.5. The outer thread is responsible for parsing the waveform configuration settings within XML files received from the master control and for initializing all the remaining parameters within the framework. This thread is also responsible for assembling and starting the user-requested medium access control (MAC) and physical layer software threads. The MAC layer is responsible for accepting a digitized voice or data information and packetizing it into the payload of discrete frames. The MAC layer also adds addressing and error-checking fields. The physical layer is responsible for accepting these MAC frames, modulating the information bits into the complex samples of an analog waveform, and relaying this information to the USRP for transmission. This physical layer contains the signal *flowgraph* thread, the chain of GNU Radio blocks which perform the actual signal processing algorithms required in transmission and receptions. For the CWT Smart Radio 2007, we expanded capabilities of the MAC and physical layers and updated the parsing capabilities of the outer configuration thread to support these improvements.

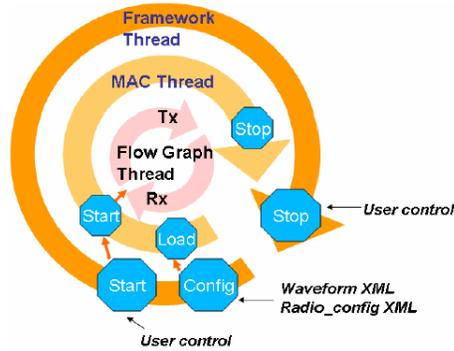


Figure 2.5. Radio Framework Multi-threading Control.
[B. Le's Image from [23]. Used with permission of B. Le.]

MAC Layer

The Smart Radio 2007 uses multiple MAC protocols as needed to perform the communication functions of the radio. First, when the smart radio needs to communicate with other legacy analog radios, the system uses a simple push-to-talk (PTT) MAC. When the user presses the *talk* button on the smart radio user interface, the radio collects digitized voice samples from the computer's sound card and forwards these to the physical layer for FM modulation. For data communications, the smart radio uses the carrier-sense multiple-access (CSMA) algorithm. Prior to sending out packets, the system senses the channel for other transmitting radios. If the channel is open, it forwards the packet to the physical layer for BPSK, QPSK, or 8PSK modulation. Otherwise, if the channel is busy, the MAC protocol *backs-off*, waiting before re-sensing the channel and re-attempting to transmit the packet. The CSMA MAC interacts with higher layers of the protocol stack in the radio through the use of the GNU Radio virtual network interface called the *TUN/TAP* open source library, also used in [24]. During transmission, software applications such digital text chat, VoIP, and other multimedia programs push TCP/IP packets down into the TUN/TAP interface, where they are received and packaged into frames by the MAC layer. During reception, the reverse occurs, where MAC layer frames are de-packaged, and the resulting TCP/IP packets are pushed up through the TUN/TAP interface to the software applications. The location of this MAC layer as a part of the radio framework is shown in Figure 2.6.

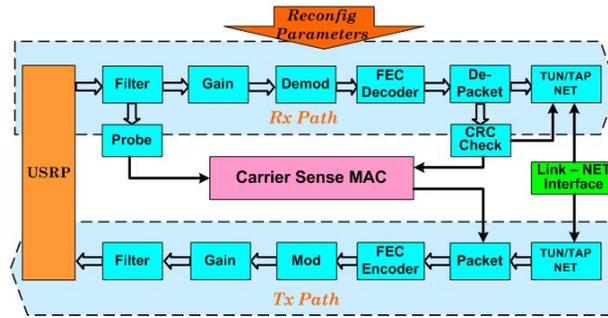


Figure 2.6. PHY/MAC Radio Framework.

[F. Ge' Image from [5], adapted from B. Li's Image from [23]. Used with permission, B. Li, and © 2008 IEEE.]

In addition to the PTT and CSMA MAC protocols, the CWT Smart Radio 2007, contains two additional MAC protocols to support the newly developed Rendezvous protocol and Channel Change protocol (CCP) dynamic spectrum access protocols. First, for the Rendezvous protocol, the smart radio uses a completely new Rendezvous MAC which involves the transmission and reception of pseudo-random frequency-hopped beacon messages. The beacons contain headers including sequence numbers and node address numbers which uniquely identify each smart radio to potential receivers. Upon startup, the Rendezvous MAC runs separately from communications operations for the goal of locating other smart radios within range. When found, these radios' information is added to the knowledge database as introduced in Section 2.5.2.2. Second, for the Channel Change protocol, the smart radio uses a new CCP-MAC. The prototype CCP-MAC used during the 2007 radio competition is based on expansions to the CSMA algorithm used in traditional data communications. However, additional timers and software control logic are added to measure the frame lengths of signals received. Given that the smart radios use a digital frame of a known length, received signals whose frames are measured to be longer are flagged as originating from other primary users of the frequency band. The designs and operation of the Rendezvous protocol and the Channel Change protocol are explained in further detail in Chapters 5 and Chapter 6.

Physical Layer

The physical layer contains the *flowgraph*, the series of blocks responsible for the actual signal processing computations which convert the digitized voice and data information into samples of a modulated waveform for transmission. For the Smart Radio Challenge 2007, the radio framework supported analog voice communication using FM and digital communication

using BPSK, QPSK, or 8PSK. The incorporation of 8PSK marks an improvement over previous framework versions used by CWT [24]. For digital transmissions, these modulations represent two different qualities of service (QoS). The use of BPSK or QPSK at 16 kbps or less defines a low bandwidth digital QoS, while the use of 8PSK at 19.2 kbps or greater defines a high bandwidth digital QoS. CWT has also added a new flowgraph block to the existing GNU Radio library to allow for the use of continuously-variable slope delta modulation (CVSD) vocoder when transmitting voice-over-IP (VoIP) data. All parameters of the physical layer are fully configurable through XML setup file received from the MC and parsed and initialized in the radio framework's outer configuration thread.

2.5 Results

The Smart Radio Challenge 2007 project represented an enormous accomplishment for the Center for Wireless Communications and everyone involved. Although I served as CWT's project captain and primary representative to the SDR Forum, I was assisted by a team of nearly a dozen of graduate and undergraduate students from Virginia Tech's Electrical and Computer Engineering department. Each contributed to the year-long development process by writing a series of monthly research and technical reports exploring optimal SDR designs and authoring numerous lines of software code to help implement each smart radio component. In addition, the team members contributed to our initial written design proposal submitted to the SDR Forum at the beginning of the competition, and this proposal was selected as one of the top ten entries from a pool of over 40 other international universities. The team also contributed in preparing the conference demonstration of the final radio design and in writing the project final report. Together, our efforts and the technical contributions of the CWT Smart Radio 2007, won first place for the specific public safety earthquake disaster problem, as well as the grand prize of entire SDR Forum Smart Radio Challenge 2007 competition. As an addendum, our written proposal for a newer version of the our radio, called the CWT Smart Radio 2008, and submitted shortly thereafter, was also selected by the SDR Forum to compete in its Smart Radio Challenge 2008 competition the following year. The next chapter of this dissertation, Chapter 3, is dedicated to explaining improvements and contributions of this improved CWT Smart Radio 2008 design.

2.6 Conclusions

The CWT Smart Radio 2007 provides a prototype mobile communications package to public safety first responders at the scene a natural disaster. The system can identify available spectrum within a pre-defined band, rendezvous with an intended receiver, and communicate with voice and data by using one of two selected QoS settings. The CWT Smart Radio also features two dynamic spectrum access protocols called *channel change* and *rendezvous*, which enable it to detect and initialize communications links with neighboring smart radio nodes, as well as to share spectrum between analog primary users and digital secondary users. The MC module enables spectrum awareness by characterizing the radio environment with a power spectrum sensor and a signal detection and classification module. The MC also enables spectrum and signal memory by storing sensor results in a knowledge database. By utilizing a FRS waveform database, the CWT Smart Radio can create a new communication link on any designated FRS channel frequency using FM, BPSK, QPSK, or 8PSK modulations. The CWT Smart Radio leverages the GNU Radio toolkit, and its modular architecture provides a dynamic spectrum test bed for future smart and cognitive radio research. The CWT Smart Radio's key operational features are listed in Table 2.1. The CWT Smart Radio meets or exceeds all of the requirements specified by Smart Radio Challenge 2007, and offers public safety first responders a mobile communications prototype for use in disaster relief scenarios.

Smart transceiver 5 MHz FRS Band (462 to 467 MHz) 25 KHz bandwidth
Have FRS compliant spectral properties
Must interoperate with legacy analog FRS users
Demonstrate at least 2 digital QoS settings
At least 16 kbps using BPSK or QPSK CVSD vocoded voice
At least 19.2 kbps using 8-PSK CVSD vocoded voice
Source & destination radios must be able to rendezvous
Will not produce interference to any other active communications traffic
Dynamically change channels without loss of service

Table 2.1. Operational features provided by CWT Smart Radio.
[M. Silvius' Table from [6]. Used with permission, © 2008 SPIE.]

Chapter 3

Smart Radio Architecture 2008

At the end of the SDR Forum Smart Radio 2007 competition, CWT had the opportunity to again compete in the following year's competition. The team worked together in the fall semester to write and submit a proposal to SDR Forum detailing our plan to improve the existing radio design to meet the demands of a new, more complex public safety disaster communications problem. The new scenario required our smart radios to go beyond simple point-to-point link configurations, and it required a large collection of smart radio nodes to automatically assemble themselves into a mobile ad-hoc data network. Our final proposal was again selected, along with a half-dozen other university teams, to compete in the Smart Radio 2008 competition. This chapter describes the details of that proposal, and our efforts to design, build, and test the new CWT Smart Radio 2008.

The organization of this chapter is the following. First, in Section 2.1 reviews the new public safety communications problem statement for the Smart Radio Challenge 2008. This provides the design requirements and the motivation for improvements to the radio's architecture from the previous year. Section 2.2 then describes the contributions resulting from the development of this new architecture. Next, Section 2.3 reviews previous work referenced when designing our system. Section 2.4 describes the specific architecture of the CWT Smart Radio 2008, in block by block fashion. Finally, Section 2.5 discusses the results of the competition, and Section 2.6 concludes with a summary of the radio's functionality.

3.1 Problem Statement

The Smart Radio Challenge 2008 scenario closely parallels the London subway bombings that occurred in July of 2005 [30]. In this situation, a series of terrorist explosions detonated aboard trains within the underground rail system. The initial public safety first responders arriving on scene have to proceed down into the subway tunnels to collect an initial assessment of damage and casualty reports, and relay that information back to the first

responders waiting at street level. Unfortunately, there was no active communication network within the subway tunnels. Although some cities may have cell phone repeaters or micro-cells within their rail terminals, for this scenario, we can assume that these were not present, or the blasts themselves had knocked out power and fixed connectivity to the internal network. The only communication tool available to the first responders were legacy FM-based walkie-talkie radios, which given the electromagnetic shielding of the underground subway tunnels themselves, would be completely cut-off and incapable of transmitting or receiving signals from similar radios above-ground. In order for first responders to relay damage and causality assessments to command-post or triage stations being assembled near the tunnel's entrance, they would need to exit the subway tunnels, or set up a communication bucket brigade, i.e. a manual forwarding of information from one officer to the next, down the tunnel, and out toward the surface.

In this critical public safety communications crisis, the deployment of a system of fully networkable smart radios could have saved precious time, and extended real-time voice and data communications capabilities to first responders in the rush to find and recover the disaster victims. These networkable smart radios could have established an ad-hoc network and temporary infrastructure within the subway tunnel. First responders could access this network either using their existing legacy FM radios, or using newer hand-held version of smart radios. The smart radio network would act as a repeater system, relaying voice and data communications between the first responders within the tunnel. Also, smart radio nodes, situated near and at the entrance of the subway could also connect and interface with the existing fixed communications network at street level. In this way, the existing wide-area communications infrastructure available to public safety responders above-ground, could be *extended* into the communications *shaded region* of subway tunnel itself.

The SDR Forum, in their problem statement for the Smart Radio Challenge 2008, also levied specific requirements for the problem. The smart radio system automatically creates an ad-hoc extension to an existing communications network such that voice communications can be relayed to/from the incident site out of the shaded area to/from the above communications infrastructure. The network extension utilizes peer-to-peer links among radios and reconfigure at least one radio as a repeater. In addition, the smart radio should incorporate and improve upon features introduced in the earlier competitions. Recalling the features of the CWT Smart Radio

2007 from Chapter 2, the system should be to find available spectrum within a pre-defined band using a signal detection and classification module, connect with an intended receiver using a *rendezvous* discovery, and transmit data over that band with a pre-determined quality of service (QoS). The system should also corporately share spectrum between analog FRS primary and digital smart secondary users using a *channel change* dynamic spectrum access protocol.

In terms of the size, the prototype public safety ad-hoc network should incorporate at least six smart radio nodes. There should be one *hidden radio*, which represents the public safety first responder who is the furthest down the subway tunnel and who is situated in the communications-shaded region, completely cut-off the above ground communications infrastructure. There should be at least three *repeater radios* which form an ad-hoc network within the tunnel, and which relay voice and data communications from hidden radio to the tunnel's surface. There should be one *command radio*, which is situated as at the entrance of the tunnel, and acts as the *gateway*, bridging the temporary ad-hoc network within subway, to the fixed infrastructure above ground. There should be one *infrastructure radio* which represents the fixed access point to wide-area communications network above-ground. For our scenario, one could think of this is as fixed cell tower, or a public 802.11 WiFi access point. In addition, each smart radio node must be able to intelligently switch between roles as the scenario warrants. For example, before entering the tunnel, all smart radio nodes maybe be directly connected in a single-hop fashion to the infrastructure radio. However, as they are position within the subway tunnel, they must automatically detect if and when they become disconnected from the infrastructure radio, automatically form an ad-hoc network, assuming the role of a command radio, repeater radio, or the hidden radio. See Figure 3.1 for the picture of this scenario. Lastly, the digital smart radios must support and offer first responders a number of multimedia applications, such as text chat, file transfer, streaming video, voice-over-IP

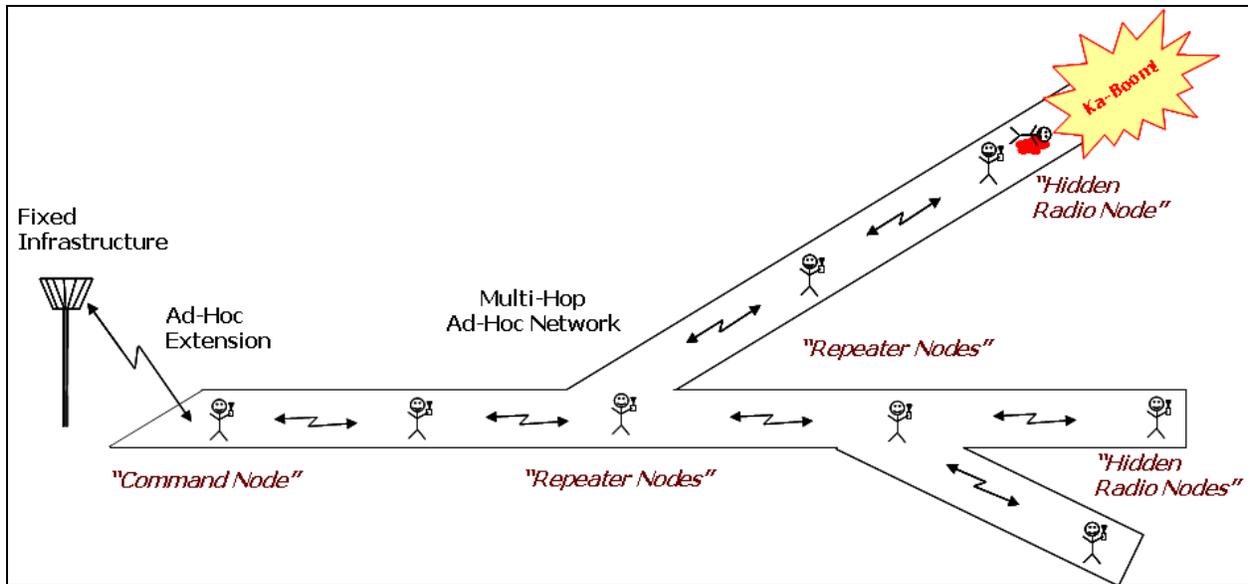


Figure 3.1. Problem Scenario and Requirements.

3.2 Contributions

At a macroscopic level, the CWT Smart Radio 2008 architecture contributes to cognitive radio research by paving the way for the development of mobile, ad-hoc, *heterogeneous* cognitive radio networks. In other words, the technologies developed here now allows for a smart radio platform to simultaneously use multiple radio frequency interfaces, enabling dynamic networks of radios using different, *heterogeneous* waveforms. Furthermore, with the ability to use multiple interfaces, the smart radio architecture also provides the foundational technologies necessary to build a *cognitive gateway*. This way, the smart radio can detect and bridge multiple different networks together, each using distinct waveforms, and route IP-based packet data between them.

At the microscopic level, we now describe the specific technical contributions of the CWT Smart Radio 2008 architecture. First, a new universal radio framework includes configuration and control of multiple adapter types. These include 802.11 WiFi, Bluetooth, Ethernet, as well control over the GNU Radio and USRP interface. Second, the internal design of the radio framework has been redesigned from the 2007 competition to now allow data from connected networks on each of the adapter types to be bridged together. In this way the radio framework can support IP datagram routing between the different networks. The operation of the universal radio framework is described further in the next section. In Chapter 4, we describe

how it can be used to construct a network testbed on which to conduct routing and quality of service (QoS) measurements. Second, we embedded mobile ad-hoc routing protocols into the radio framework for use with the 802.11 WiFi and the GNU Radio/USRP interface. For our initial prototype, we incorporated the On-Demand Link State Routing (OSLR) protocol, as well as a proprietary routing protocol called MobiMesh developed by the MITRE Corporation [31]. Fourth, in the use of Bluetooth, we developed our own, new scatternet formation protocol. The Bluetooth scatternet formation process is similar to that of the ad-hoc network formation in 802.11, with the exception that packet forwarding and routing happens at the MAC layer rather than the NET layer. Fifth, we developed improved network discovery sensors and an improved knowledge database. More specifically, we wrote routines that would have the radio framework periodically poll each of the adapters and search for existing networks. For the 802.11 WiFi and Bluetooth, this involved interfacing with these interface cards' APIs and utilizing the standard's built-in scanning routines. For the GNU Radio and USRP, this involved running the PSD sensor and classifier, as well as the Rendezvous protocol already developed in the CWT Smart Radio 2007. The improved knowledge database now stores network status information for all the adapter types and displays these on the system's improved user interface. Sixth, we developed a set of sensors which would monitor the health of an 802.11 WiFi access point. More specifically, when the CWT Smart Radio 2008 associates with an 802.11 WiFi access point, it monitors that access point's power and connectivity to the global Internet. If the power drops, or the smart radio loose connectivity to the access point, a warning message is sent to the radio's Master Control for action. At this point, the Master Control can search for other access points or ad-hoc nodes which with to re-associate. Sixth, an improved graphical user interface allows users to easily change the role and configuration of each node. The interface also allows users to manually configure the node to bridge one or more networks together. Seven, we physically built the system and were able to run real quality of service experiments using it. More is said on the application of these experiments in Chapter 4.

3.3 Previous Work

The primary source of previous work evaluated in the preparation of the CWT Smart Radio 2008 came from the development of the CWT Smart Radio 2007. However, considerable additional literature review was conducted in such areas as: Linux operating system API calls to

interface attached 802.11 WiFi and Bluetooth adapters; mobile ad-hoc routing protocols; configuration of routing and forward tables within the Linux operating system kernel; as well as Bluetooth scatternet formation. The results of this review are shown in the discussion of the smart radio components in the following section.

3.4 Smart Radio 2008 Architecture

This section details the design of the smart radio architecture constructed from the SDR Forum Smart Radio Challenge 2008 competition in Washington, D.C. The CWT Smart Radio contains five main components: the user interface, the master control, the universal radio framework, the knowledge databases, and the sensors. In addition, modules for formatting and transferring messages assist in connecting the different components. Figure 3.2 depicts these components and their interactions. The overall architecture of the new smart radio closely resembles that of the CWT Smart Radio 2007. The components inherit all the capabilities of this previous design, but many have improved functionality and capabilities. In this section, we detail the improvements over the previous versions.

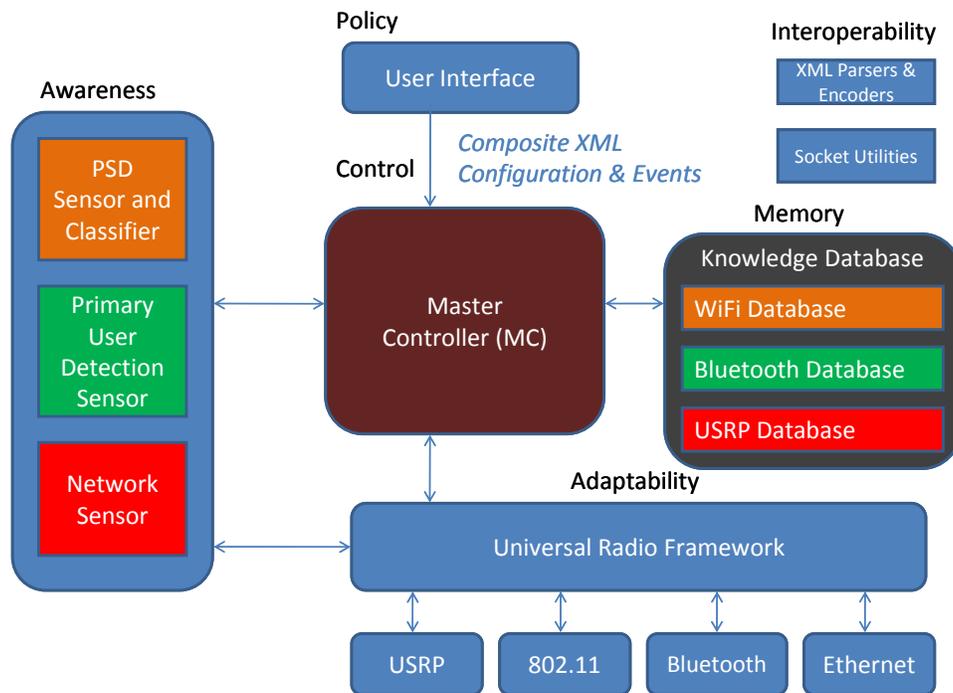


Figure 3.2. CWT Smart Radio 2008 Architecture.

3.4.1 User Interface

The user interface acts as the control panel from which a public safety first responder can oversee the smart radio's operation. When the radio is deployed, the public safety officer can program the radio with basic guidelines or *policies* for operation. This includes identifying the primary waveform or network that all radios should connect with upon initialization. Then, the user should provide the radio with guidelines for types of actions to perform if this primary waveform or network access point fails, or becomes out of range. However, once the radio has received these user's operational communication requirements, its further actions and operations are automatic. The four tabs of the graphical user interface were designed with this purpose in mind.

Scan Tab

The Scan tab provides the user with a visual list of existing and available networks and nodes within its immediate radio environment. This screen contains four tables, which pull their information from the WiFi, Bluetooth, and USRP knowledge databases. The knowledge databases are, of course, populated through scans using the sensor and radio framework components are be described later. See Figure 3.3 for a snapshot of the *Scan Tab*.

Connections Tab

The user can select waveforms or networks listed in the tables under the Scan tab in order to build one more *connections*. Or, a user can manually enter in a new waveform or network if not listed under the Scan tab. A connection can be thought of as the configuration state of the universal radio framework, so it can communicate using one or more interfaces simultaneously, and route IP traffic between interfaces as desired. See Figure 3.4 for a snapshot of the *Connections Tab*.

Configurations Tab

The Configuration tab allows the user to program the radio to switch between *connections* in response to a specific event. For example, in the subway disaster scenario, the public safety officer may want the smart radio to initially connect to a fixed infrastructure above-ground. In this case, the officer should program Connection #1 as the 802.11 access point

network in listed in the WiFi table. In the event that the access point falls out of range, the officer should program Connection #2 to be a Bluetooth scatternet link to the next closest radio. the officer should program Trigger #1 to be “WiFi down.” See Figure 3.5 for a snapshot of the *Configurations Tab*.

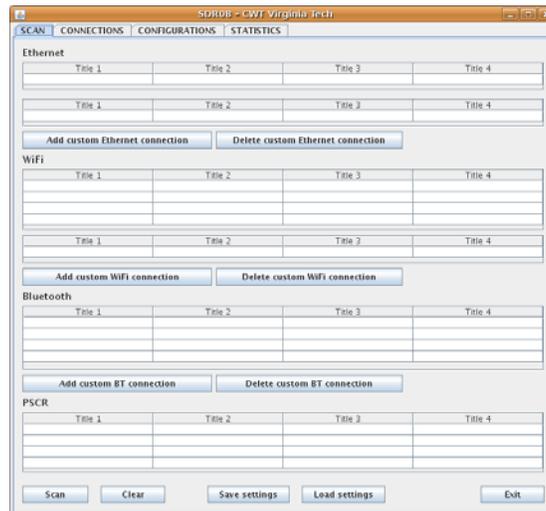


Figure 3.3. *Scan Tab* of the User Interface.

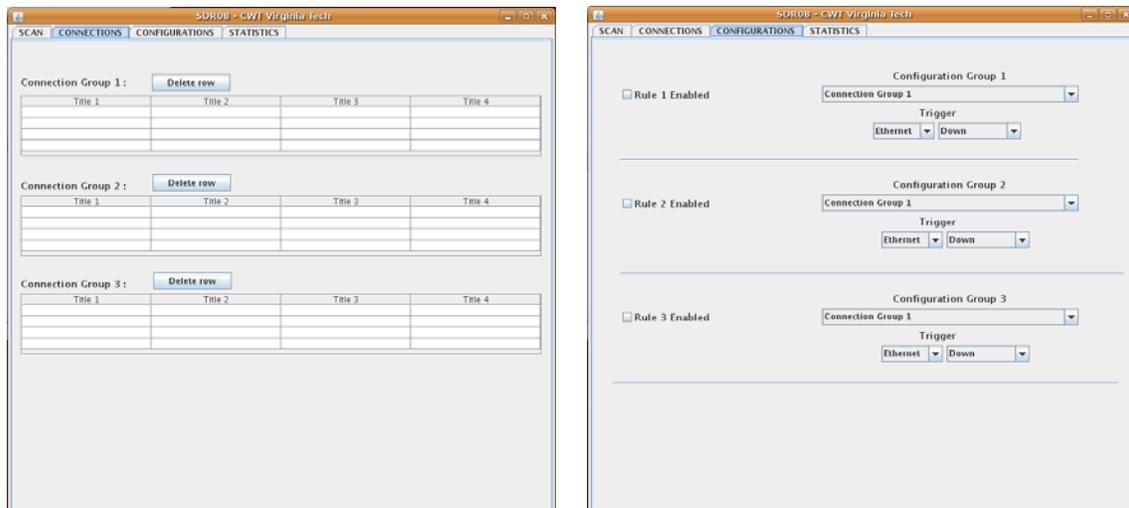


Figure 3.4 and Figure 3.5. *Connections Tab* (left) and *Configurations Tab* (right) of the User Interface.

3.4.2 Inter-Block Data Exchange

The CWT Smart Radio 2008 adheres to the same data standardization and interoperability concepts as its 2007 predecessor. It continues to rely on XML message formatting and TCP/IP sockets for inter-block communication. However, in this version, the XML formatting and parsing routines are more complex and play an even greater role in the configuration of the smart radio components.

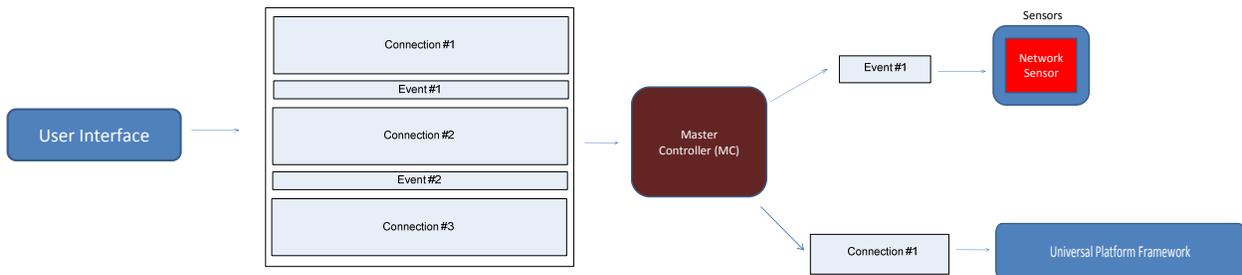


Figure 3.6. An example of XML messaging used in radio configuration.

Figure 3.6 demonstrates the operation of one configuration. First, the first responder enters in the desired *configurations* setting in the user interface. The waveform, network, and trigger event information is converted to XML and forward to the Master Control module. The Master Control parses this information, and sends the first *connection* to the universal radio framework module for instantiation. Next, the Master Control interacts with the sensors block, waiting for an event trigger to occur. Both the sensors and the universal radio frameworks can then send feedback to the Master Control, also using XML formatted messages. After the event occurs, the Master Control sends a new, appropriate connection setting to the universal radio framework in response. Figure 3.7 shows an example XML file. In this *connection*, the universal radio framework is programmed to bridge and route packets between a fixed Ethernet network to a mobile Bluetooth scatternet.

```

<?xml version="1.0" encoding="utf-8"?>
<waveform type="cots_network">
  <connection type="double">
    <side0>
      <adapter>ethernet</adapter>
      <sensor>None</sensor>
      <app>
        <service>None</service>
        <nat>True</nat>
        <nat_client>bluetooth</nat_client>
      </app>
      <net>
        <ip>192.168.10.3</ip>
        <netmask>255.255.255.0</netmask>
        <default_gateway>192.168.10.1</default_gateway>
        <dns_1>192.168.10.1</dns_1>
        <dns_2>192.168.10.1</dns_2>
        <routing>None</routing>
      </net>
    </side0>
    <side1>
      <adapter>bluetooth</adapter>
      <sensor>None</sensor>
      <app>
        <service>GN</service>
        <nat>False</nat>
        <nat_client>None</nat_client>
      </app>
      <net>
        <ip>192.168.1.1</ip>
        <netmask>255.255.255.0</netmask>
        <default_gateway>192.168.10.1</default_gateway>
        <dns_1>192.168.10.1</dns_1>
        <dns_2>192.168.10.1</dns_2>
        <routing>None</routing>
      </net>
      <mac>
        <connection_type>service</connection_type>
        <connect_to>GN</connect_to>
        <connection_retries>20</connection_retries>
      </mac>
    </side1>
  </connection>
</waveform>

```

Figure 3.6. An example of XML configuration for two adapters which are bridged together.

3.4.3 Master Control

The Master Control (MC), as in the previous competition, is the brains of the smart radio system. It receives basic directions from the public safety operator through the user interface. From these, it is responsible for directing the operation of all the other modules in the architecture. First, it obtains *spectral and network awareness* by scanning the environment with the attached sensors and network interfaces. The results of these scans are forwarded and stored in the knowledge database, which provides *signal and network memory*. The MC then directs the dynamic reconfiguration and initialization of the universal radio framework, which includes control over the Ethernet, 802.11 WiFi, Bluetooth, and GNU Radio/USRP interfaces. The MC also aids in the assembly of the dynamic routing and forwarding tables, which allow IP data traffic from a network on one interface to be bridged with a network on another interface. One of the most significant new responsibilities of the MC in the new 2008 architecture is to monitor the health of these attached networks through the use of the smart radio's network sensors. When the MC detects that a network associated with an interface has become idle, disconnected, or is out of range, the MC reacts to this *trigger event* by automatically directing the universal radio framework to re-associate with a new network on a different interface. For example, in the

London subway scenario, when MC detects that the 802.11 WiFi infrastructure access point is out-of-range of the smart radio, i.e. the smart radio has entered a *shaded region*, it automatically directs the universal radio framework to shut down the 802.11 interface, and start-up and form a peer-to-peer scattered network on the Bluetooth interface. The MC can also direct the framework to stop the peer-to-peer network and re-associate with the WiFi infrastructure access point, should the radio become within range of the access point again. Finally, like in the previous smart radio design, the MC is responsible for directing the operation and execution of the dynamic spectrum access Rendezvous and Channel Change protocols, which aid in the discovery of other smart radios and in the sharing of the operational spectrum band with other primary user radios.

3.4.4 Sensors

As highlighted in Figure 3.2, the CWT Smart Radio 2008 uses three sets of sensors in order to maintain *awareness* about the status of the radios and networks in its environment. First, it uses the power spectral density (PSD) sensor, in conjunction with the GNU Radio/USRP interface of the universal radio framework, to locate and capture signals transmitted from other radios within range. Its classifier then works to determine the signal's modulation scheme, and the results are stored in the knowledge database and displayed in the user interface. The MC can then use this signal information when attempting to connect to one or more of these radios, or in identifying spectrum whitespaces in which to establish new communication links. This process was discussed in Chapter 2. Second, the primary user detection sensor, shown in Figure 3.2, is actually just the encapsulation of the primary user detection routines previously embedded in the universal radio framework, which aid the Channel Change protocol in identifying the return of primary users to the operation channel. It works with the carrier-sensing routines within the GNU Radio/USRP interface, and it uses a set of timers to measure the frame lengths of received signals. Frames whose lengths exceed a threshold length are classified as having been transmitted by primary user radios. This process is explained in further detail in Chapter 6. Third, the network sensor is the newest of the sensors, specifically designed for the CWT Smart Radio 2008. It works in conjunction with the MC to monitor the health of networks attached to the interfaces within the universal radio framework. The sensors can determine if a network has become idle, disconnected, or is out of range, and then it sends a *trigger event message* to

MC. The MC can then decide whether to direct the universal radio framework to re-associate with a new network on a different interface. In order to build the network sensor we utilized the APIs and built-in scanning routines of the interface cards within the universal radio framework.

3.4.5 Knowledge Databases

The knowledge database stores information on the discovered waveforms and networks within range of the smart radio. The database is partitioned into three tables, called the WiFi, Bluetooth, and USRP databases, as shown in Figure 3.2. These represent the discovered waveforms and networks with which the smart radio is capable of connecting to, using the WiFi, Bluetooth, and USRP interfaces within the universal radio framework respectively. The contents of the knowledge database are displayed in the user interface, as shown in Figure 3.3.

3.4.6 Universal Radio Framework

The universal radio framework is responsible for the configuration and control of all the attached interfaces. A single interface can be enabled for communications over a single waveform or network, or two or more interfaces can be enabled so as to bridge communications between multiple waveforms or networks. The universal radio framework and adapters are configured using XML, just like all other modules in the smart radio architecture. The structure and principal components of the universal radio framework are shown in Figure 3.7. First, the XML parser receives configuration information from the MC in XML format and translates this into parameter settings for the rest of the framework. Second, like the radio framework used in the 2007 competition and described in Chapter 2, the universal radio framework uses a series of nested software threads to configure, manage, and perform any signal process associated with each of the attached interfaces. These threads are generally divided into two categories: the first of these inherited from the CWT Smart Radio 2007 which operate the GNU Radio/USRP interface; and the second, are the new set of threads which control the commercial off-the-shelf (COTS) interfaces, such as the 802.11 WiFi, Bluetooth, and Ethernet. Third, the universal radio framework contains a module responsible for the ad-hoc network routing and bridging. This module uses ad-hoc network routing protocols such as OLSR to accomplish routing within a single network associated with a single interface, as well as algorithms to route IP-based packet traffic between networks associated with different interfaces.

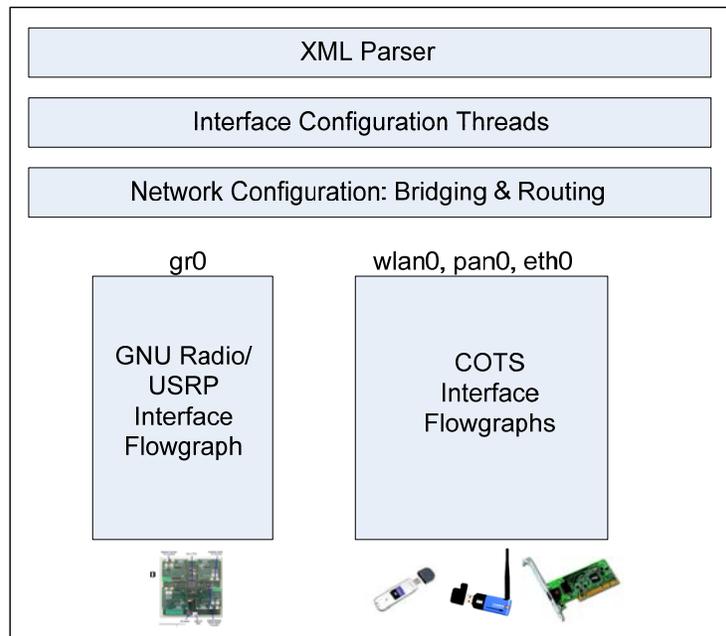


Figure 3.7. The principal components of the Universal Radio Framework.

3.4.6.1 Network Routing

As introduced in the previous section, the universal radio framework uses ad-hoc network routing protocols for routing within the network associated with each interface. To accomplish this, we incorporated the On-Demand Link State Routing (OSLR) protocol, as well as a lesser known protocol called MobiMesh, developed by the MITRE Corporation [31]. Although available to any interface, these protocols work best with 802.11 WiFi and GNU Radio/USRP. For the Bluetooth interface, we created a custom scatternet formation and routing protocol, described in the next section.

3.4.6.2 Bluetooth Scatternet Formation

Figure 3.8 details the operation of the custom Bluetooth scatternet formation protocol used in the Smart Radio Challenge 2008. The goal of the protocol is to assemble all of the smart radio nodes into a chain-like topology using their Bluetooth interfaces, for the purpose of extending communications connectivity from the *infrastructure radio* at street-level, to the *command radio* at the entrance of the subway tunnel, down a line of *repeater radios* in the subway tunnel, and ultimately to the *hidden radio* at the end of the tunnel.

For the purpose of this scenario, we can model the *infrastructure radio* as an 802.11 WiFi infrastructure access point. Initially, at the beginning of the disaster scenario, all the radios nodes are located at street-level. The public safety operator, using the radios' user interfaces, programs all the smart radios nodes to use this 802.11 infrastructure access point as their primary connection; however, if they should enter a shaded region and lose this connection, the operator programs the radios to form an ad-hoc scatternet using their Bluetooth interfaces. The MC receives these directions, encoded in XML from the user interface, and then sends the first *connection* setting to the universal radio framework. The universal radio framework initializes each radio's 802.11 WiFi interface and associates it to the infrastructure access point. The MC also starts up the network sensor to monitor the health of this infrastructure access point. Next, as the radios enter the subway tunnel, i.e. the communications shaded area, the network sensors detect that the health of the infrastructure access point is deteriorating. When the connectivity to the access point completely fails, the MC sends the alternate *connection* setting to the universal radio framework, activating each radio's Bluetooth interface. The MC then runs the Bluetooth scatternet protocol detailed in Figure 3.8.

For the purpose of the scenario, we also assume that the public safety operator identifies one smart radio node to be the *command radio*, which is permanently positioned at the entrance of the tunnel. The public safety operator uses the user interface on this radio to program it to act as the *gateway*, bridging connectivity from the infrastructure node using 802.11 WiFi to the first *repeater radio*, using Bluetooth. The MC forwards the connection setting for this gateway role to the universal radio framework, which simultaneously starts up both 802.11 WiFi and the Bluetooth interfaces, with the appropriate routing settings to forward IP packets between the two. This Bluetooth interface is configured to *always* accept incoming connections from other Bluetooth nodes.

The basic premise of the Bluetooth scatternet formation protocol is that each node continually tries to connect to other Bluetooth nodes; however, it only *accepts* connections from other Bluetooth nodes, once it has regained communications connectivity to the outside world. Note that in terms of Bluetooth, accepting connections requires the node to switch from a personal area network user (PANU) to group ad-hoc network controller (GN). The first node to become a GN and begin accepting connections is the *command radio*, since by definition, this node is still within line-of-sight of *infrastructure radio* and it has connectivity to the outside

world. The *repeater nodes* then form a chain, starting from the *command radio*, down the tunnel to the *hidden radio* at the end of the tunnel, as each radio gains connectivity to the outside world and begins accepting connections from other Bluetooth nodes. It is also not necessary for the chain to be static. The change can operate in a dynamic fashion. For example, if a radio moves out range of an associated peer, it can establish a link and re-associate with the next closest node within range.

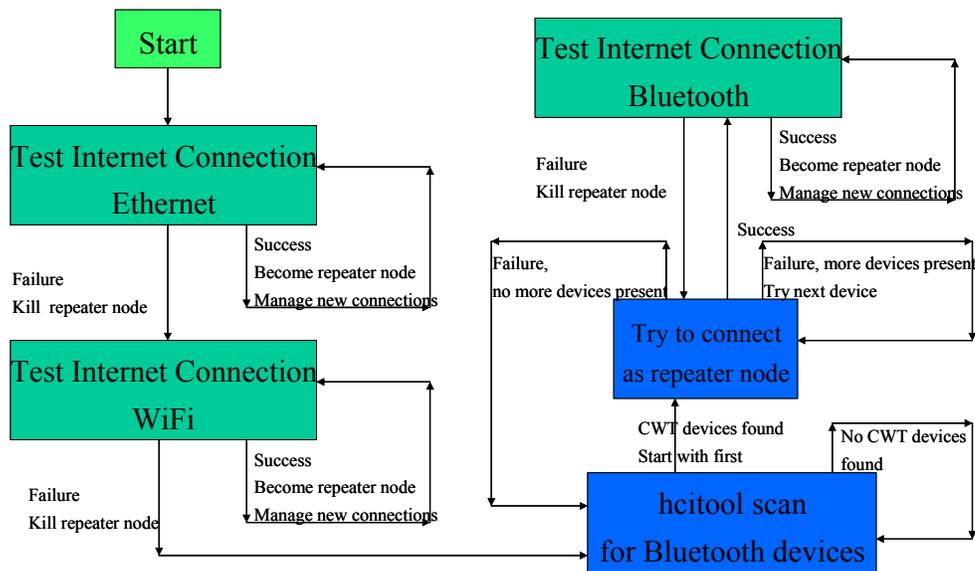


Figure 3.8. Diagram highlight the Bluetooth scatternet formation process. [T. Brisebois' Image. Used with permission of T. Brisebois.]

3.5 Results

During the Smart Radio Challenge 2008, the judges asked the CWT team to demonstrate the ability of their smart radios to detect the loss of connectivity from a fixed infrastructure point, and to automatically form an ad-hoc network extension. The judges had us imitate the subway scenario setup by positioning our smart radios in the basement hallway of the conference hotel. This setting was ideal, because the radio propagation characteristics of the thick concrete walls of the hallway were similar to that of a subway tunnel.

At the beginning of the demonstration, the team set up all the smart radios in a conference room near the entrance of the hallway. They then configured the *infrastructure radio* to imitate an 802.11 WiFi infrastructure access point. Then, they configured all the other smart radios to initially associate with this fixed infrastructure point. Next, the team members carried

the smart radios one-by-one from the conference room and positioned them along the hotel's hallway in a chain-like topology. Just as designed, each of the smart radios detected their entrance into the communications shaded region and their loss of connectivity from the fixed infrastructure point. Each enabled their Bluetooth interferences, ran the Bluetooth scatternet formation protocol described in Section 3.4.6.2, and automatically established an ad-hoc scatternet. This Bluetooth scatternet provided the ad-hoc extension necessary to bridge communications from the *command radio* at the entrance of the hallway, through three *repeater radios*, to the *hidden radio* at the end of the hallway. After this successful demonstration, the judges awarded the CWT team first place for the subway disaster scenario problem.

3.6 Conclusions

In this chapter, we presented a conceptual design and initial prototype of the CWT Smart Radio 2008, capable of automatically creating an ad-hoc extension to bridge an existing public safety infrastructure to a shaded incident response site. The smart radio continuously monitors the health and availability of one or more network access points. Upon the disappearance of the primary access point, the cognitive radio gateway directs the formation of an ad-hoc peer cognitive radio network. The smart radio node in proximity to the next available access point becomes the gateway command radio, bridging internal network traffic to the external public safety network through this available access point. The contributions of CWT's smart radio included, first, a new architecture for the configuration and control of radio interfaces called the *universal radio framework*. The universal radio framework allows an intelligent master control module—also known as a cognitive engine—to control one or more heterogeneous radio front end adapters. In the prototype system, these include a Universal Software Radio Peripheral (USRP) board, as well as commercial off-the-shelf 802.11 WiFi, 802.15.1 Bluetooth, and 802.3 Ethernet adapters. Configuration and cognitive behavioral functions are expressed in the eXtensible Markup Language (XML) to provide interoperability between modules. Second, a suite of radio frequency and interface sensors continuously monitor the radio spectrum and locate the availability of infrastructure access points and other cognitive or legacy radio nodes. Moreover, these sensors monitor the health of infrastructure access points, and allow the cognitive engine to select the topology and waveform used in the cognitive radio network to optimize bandwidth, latency, and packet loss performance. Third, additional contributions

include a radio knowledge database to maintain neighbor and topology information, dynamic spectrum access protocols to allow the smart radio network to share available spectrum with other primary users, a Bluetooth network formation algorithm, and a graphical user interface for radio configuration, topology and network statistics monitoring. This chapter also presents CWT's initial performance results, showing the available bandwidth, latency and packet loss statistics across multiple waveforms, adapters and network topologies, both in available infrastructure and ad-hoc configuration modes.

Chapter 4

Smart Radio Network Testbed

This chapter details our efforts to quantify the communications performance of the CWT Smart Radio 2007 and the CWT Smart Radio 2008 architectures. Furthermore, it describes how we used these smart radio architectures to design and assemble the Smart Radio Network Testbed. The testbed served four objectives. First, we documented the details of assembling and using the testbed, and turned this into an educational networking tutorial. The tutorial describes how to build the network testbed using commercial off-the-shelf equipment and open-source software, and is suitable for undergraduate and graduate students to follow for conducting basic network experiments in a laboratory class. Second, the performance data collected during these experiments gives us specific quality of service (QoS) metrics, which represent the baseline operational performance of an ad-hoc network of four CWT Smart Radios. Third, the network testbed is used to quantify the performance of the Rendezvous protocol and the ability of four smart radio nodes to locate each other in an operational setting. These results are discussed in Chapter 5. Fourth, the network testbed is used to quantify the performance of the Channel Change protocol and the ability of the protocol to improve network QoS of a secondary user network when it co-exists with a primary user network. These results are discussed in Chapter 6.

The remainder of the chapter is organized as follows. Section 4.1 reviews the public safety communications problem statement of the Smart Radio Challenge and helps to motivate the construction of the testbed. Section 4.2 reviews the testbed's contributions. Section 4.3 reviews the relevant details of the SDR architecture developed by our research lab during the smart radio competitions. Section 4.4 describes how we assembled an ad-hoc network testbed using four desktop and laptop computers, Ubuntu Linux, 802.11 and Bluetooth adapters, the Universal Software Radio Peripheral (USRP) radio frequency front end, and GNU Radio open source software. Section 4.5 and Section 4.6 explain how we created network topologies using the 802.11, Bluetooth, and USRP and presents the results from measuring the performances of these networks using open-source tools. Section 4.7 discusses the significance of the experimental results and Section 4.8 concludes.

4.1 Problem Statement

As a review, the motivation to develop an educational testbed stemmed from the CWT research group's entry into the SDR Forum Smart Radio Challenge [1]. During the past two years, the Forum has focused much of its attention on the communication problems faced by first responders at the scene of public safety disasters. This 2008 challenge closely mimicked the real-world scenario of the London subway terrorist bombing [30]. The competition required our team to design, build, and demonstrate a collection of smart radios which could form a *network extension*, capable of bridging a shaded disaster location back to a fixed communication infrastructure. Moreover, upon arriving at the disaster location, each smart radio had to be able to detect the loss of connectivity to the fixed infrastructure, locate its peer radios, and automatically form ad-hoc network in a chain-like topology. The temporary public safety network had to maintain a QoS which supported the transfer of digital images, video, and two-way voice.

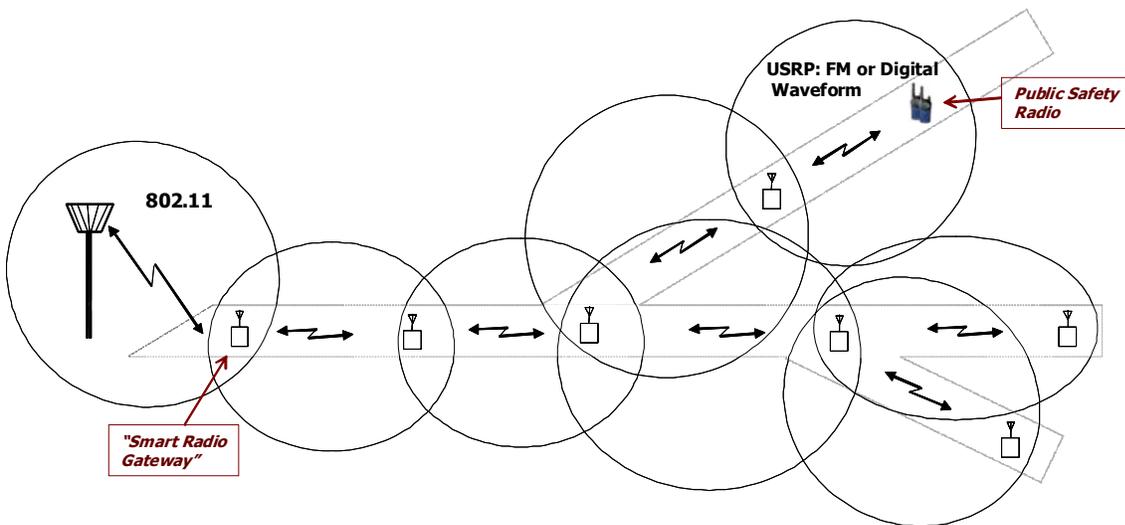


Figure 4.1. Scenario and proposed solution.

After reviewing this problem, our team hypothesized that the QoS requirements could be met by utilizing commercial off-the-self technologies (COTS) such as 802.11 WiFi and Bluetooth, in combination with the GNU Radio [8] and USRP [9] SDR platform, as shown in Figure 4.1. To test our hypothesis, the team proceeded to develop the CWT Smart Radio 2008 architecture to dynamically control the COTS and USRP radio interfaces, as well as the Smart

Radio Network Testbed to evaluate the resulting attainable network QoS using these interfaces. This chapter presents an overview our new architecture, and details the construction and experimental results of our network testbed. Furthermore, this system provides undergraduate students with an educational testbed to conduct future networking experiments and measurements in a laboratory environment.

4.2 Contributions

The principal contribution of this chapter is the design of a network testbed, which allowed us to specifically quantify the performance of the CWT Smart Radio 2007 and 2008 architectures. The construction of the testbed also forms the basis of an educational laboratory tutorial for undergraduate and graduate students illustrating ad-hoc networking and software defined radio principles. After performing the experiments described in this chapter, a student reader will acquire the following skills. First, a reader will gain the basic proficiency for using the Ubuntu Linux operating system. This includes downloading and installing programs using Ubuntu's *Synaptic Package Manager*, as well as downloading, compiling, and installing software such as GNU Radio from source code. Second, the reader will learn basic networking principles, and be able to create simple ad hoc network topologies with 802.11 and Bluetooth by executing a series of commands at the Linux terminal. Third, the reader will learn how to collect network performance measurement data on the networks' latency, bandwidth, jitter, and packet loss using open-source monitoring tools. The results will then be plotted in Matlab.

4.3 Previous Work: Review of Smart Radio Architecture

Before building the Smart Radio Testbed, let us first review the previous details of the smart radio architecture. Our smart radio architecture provides a framework for the configuration and control of the COTS and USPR radio interfaces, as well as the essential components in the system. These components are shown in Figure 4.2. The user conveys the tasks and mission policies to the radio through its graphical user interface. The on-board sensors give the radio awareness of its radio frequency (RF) environment, as well as the ability to monitor the health of fixed communication infrastructures. The databases work hand-in-hand with the sensors to store scan results in the radio's memory. The master control enables the radio to adapt to the user's direction and environmental changes by providing new configuration

settings to the universal radio framework. The universal radio framework dynamically reconfigures the attached COTS and USRP radio interfaces.

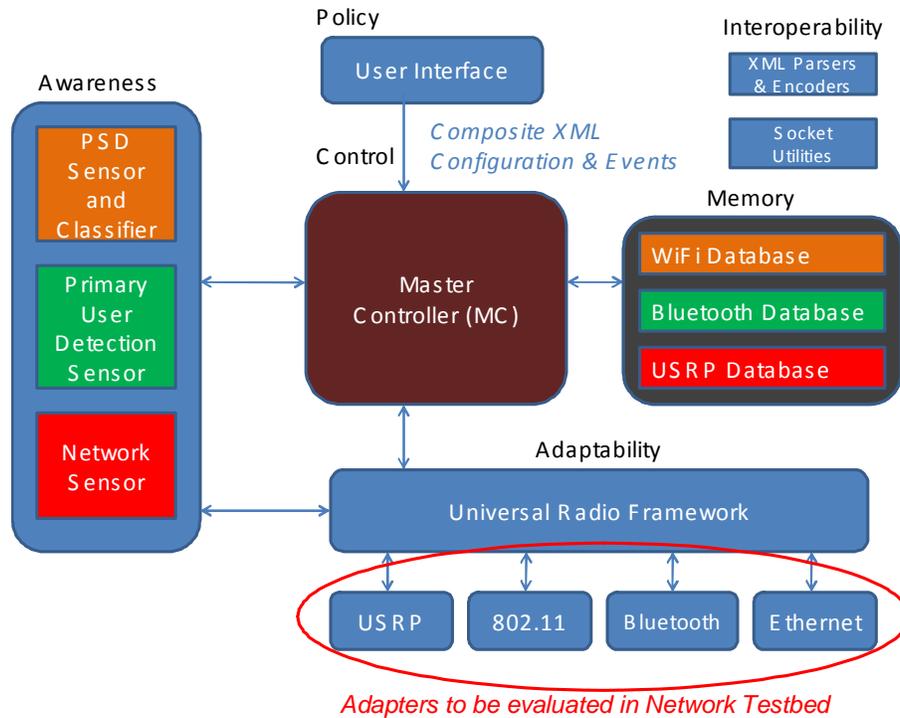


Figure 4.2. Architecture of the smart radio.

In order to determine if a smart radio using COTS network interfaces, such as 802.11 WiFi and Bluetooth, as well as the USRP, could meet the QoS for the Smart Radio Challenge, our team built the dedicated network testbed shown in Figure 4.3. This allowed us to evaluate such metrics as latency, goodput, jitter, and packet loss in a four node, multi-hop, ad-hoc network. The testbed also proved to be an ideal educational platform for our undergraduate students to use to perform network experiments. The remainder of the chapter is dedicated to explaining the step-by-step process of building this testbed, as well as performing four experiments using COTS and USRP interfaces.

4.4 Testbed Architecture and Setup

After obtaining the necessary hardware, the two primary setup tasks included setting up and installing needed software, and configuring the networking equipment to form the testbed and its control and data collection backbone.

4.4.1 Equipment

All of the experiments conducted in this chapter were conducted using COTS equipment that would be available in most wireless laboratories. Table 4.1 gives a detailed listing of the equipment used in the testbed, as shown in Figure 4.3. The most exotic device is the USRP [9]. The USRP is the RF front end to be used with GNU Radio SDR [8] to create an ad-hoc network using customized waveforms.

Item	Quantity	Approximate Cost
Laptop/Desktop (Intel Duo-Core or better recommended)	4	\$1500
USRP Version 1 Mainboard	4	\$700
Flex2400	4	\$275
Linksys Bluetooth USB (USB BT100)	1*	\$85
Linksys WiFi USB Card (WUSB54GC)	4	\$40
NetGear WiFi Router (WGT624)	1	\$40
Linksys WiFi USB Card (WUSB54GC)	4	\$40
Total:		\$~10,185
* One only card used for desktop computer.		

Table 4.1. Equipment used to build network testbed.

In our experiments, our team used three Dell Inspiron and XPS laptops. For these, we used the laptops' on-board Bluetooth hardware. The implementation was quite robust and operated well under Linux. However, we decided to go with external Linksys USB WiFi adapters, disabling the internal 802.11 chipset in the laptops BIOS. The Linksys USB adapters proved to be more reliable and easier to configure. For the Dell desktop computer, we used both

and Linksys USB WiFi and Bluetooth adapters, since our desktop did not have native wireless support. As shown in Table 4.1, the entire testbed can be assembled for approximately \$10,000, or less, if this equipment is available in the lab.

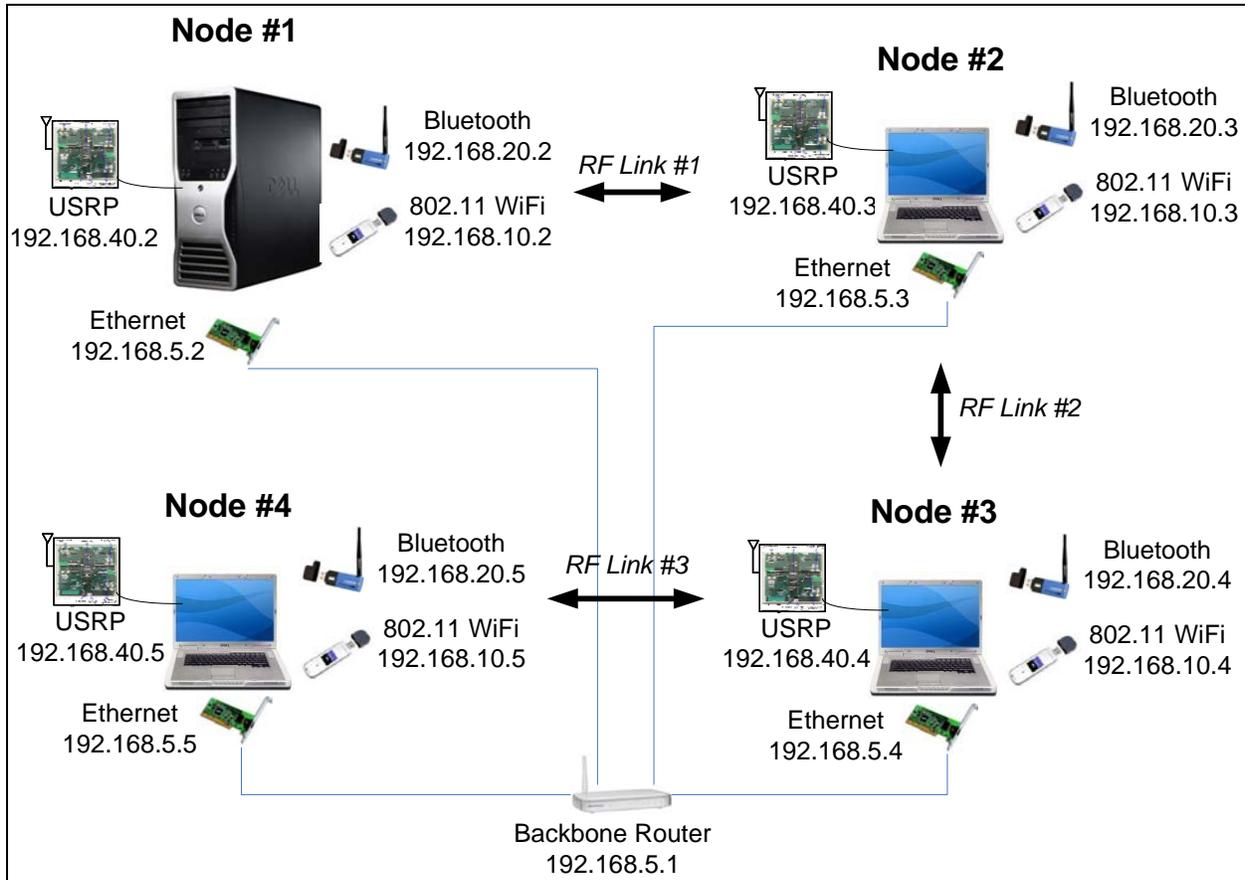


Figure 4.3. Detailed architecture of the network testbed.

4.4.2 Installing Software

First, we recommend installing Ubuntu Linux version 7.10 [32] on all the laptops and desktops. We have found Ubuntu to be the most user-friendly Linux distribution available, as it has good native Bluetooth support, in addition to being tested and verified to work with GNU Radio.

Package(s)	Purpose
bluetooth gnome-bluetooth bluez-gnome bluez-utils	Bluetooth drivers and utilities. Includes commands such as “pand” and “hcidool.”
bridge-utils	MAC layer bridging utility, used to interconnect multiple Bluetooth links when forming an ad-hoc mesh.
IPerf	IP layer bandwidth measuring tool for evaluating network performance.
JPerf	A graphical interface for IPerf. Can be downloaded from [33].

Table 4.2. Required software to install using Synaptic Package Manager or from web.

Second, we download and install the most recent stable release of GNU Radio (called the *trunk* version). There are a number of well written tutorials on how to accomplish this [34]. Third, we use Ubuntu’s “Synaptic Package Manager” to install the required additional tools, drivers, and dependencies, as shown in Table 4.2. Fourth, we download and install JPerf from [33]. We make sure the computer has Java Runtime Environment installed.

Although all the performance measurement statistics can be collected using the command-line utility IPerf, as detailed in Script 5 in Appendix 1, a student may want to download and experiment with its graphical front-end, JPerf, as shown in Figure 4.4

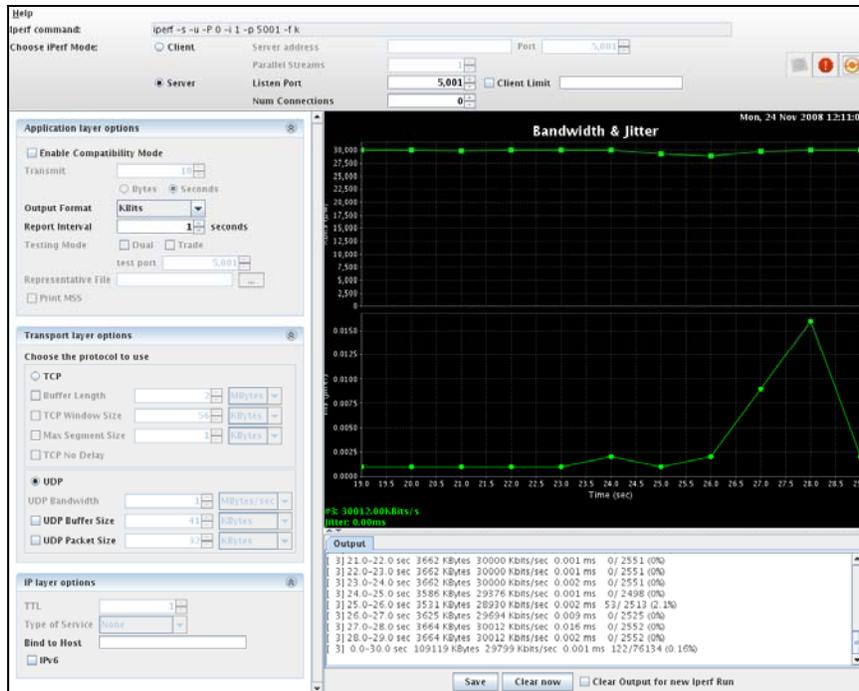


Figure 4.4. Screenshot of JPerf.

4.4.3 Assembling Testbed Backbone

Next, we started our experiments by constructing the test network shown in Figure 4.3. The four computers are physically connected together using each system’s standard 802.3 wired Ethernet adapters. Script 1 in Appendix 1 gives the example Linux commands to accomplish this. Also, we consulted our wireless router’s documentation [35] on how to disable DHCP and allow for the static assignment of IP addresses.

The wired network forms the communication backbone which is used for collecting performance data from each computer. This also allows the operator, sitting at Node 1, to remotely control the other nodes through a series of SSH terminal screens, as shown in Figure 4.5.

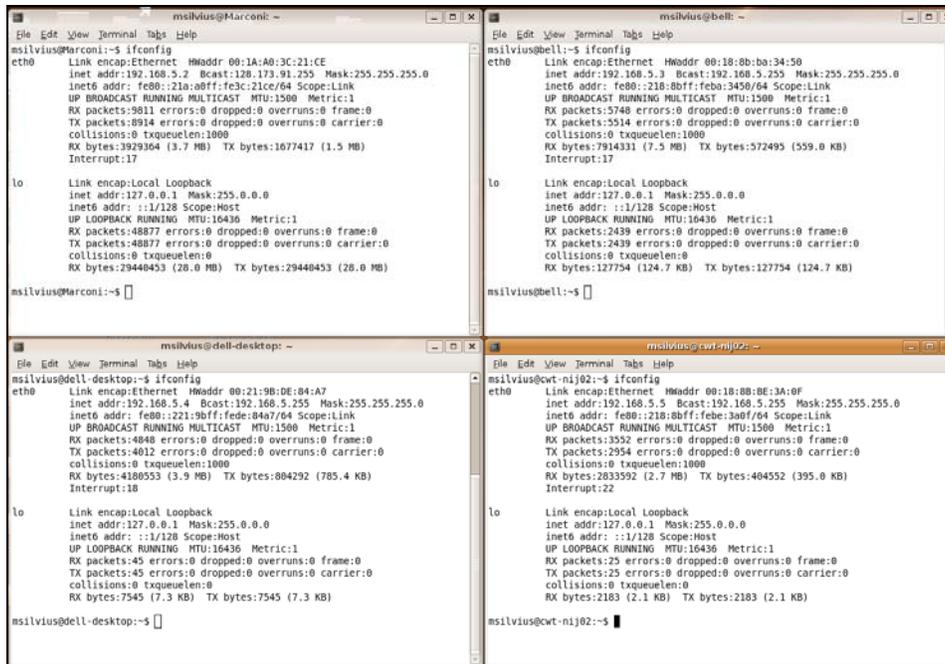


Figure 4.5. Remote SSH terminals control network configuration of each test node.

To streamline testing in the future, we also wrote a Java based control agent, which allowed us to push out configuration and test commands to multiple nodes simultaneously, as shown in Figure 4.6.

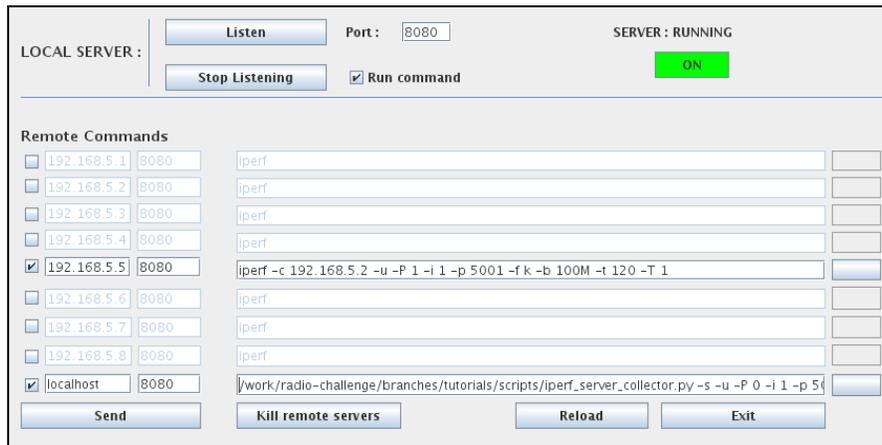


Figure 4.6. Java-based “NetworkStats” configuration agent.

4.5. Experiments: Validation of Universal Framework Architecture

First, this section details the metrics collected by the network testbed, as well as explain their significance in determining a network's QoS. Second, this section describes four experiments in which we evaluated the performance of 802.3 wired Ethernet, 802.11 WiFi, Bluetooth, and GNU Radio/USRP networks.

4.5.1 Performance Metrics

The metrics collected include: latency, goodput, jitter, and packet loss. First, we use the standard Ping command built into Linux to measure the *round-trip-time (RTT) latency*—also know as the *response time*—between the a pair of nodes. This metric records the total time for a ping packet to travel from the server to the client and for the client to send a response packet back to the server. Second, for the remainder of the data collection, we use the Linux package called “Internet Protocol Bandwidth Measuring Tool” (IPerf). To measure bandwidth, we use IPerf to send a stream of Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) packets from the client Node 4 to the server Node 1, and measure their maximum effective received rate.

TCP uses a reliable transfer protocol, which waits for acknowledgements back from the recipient before it transmits out new packets, while UDP uses an unreliable transfer protocol, which streams out packets without acknowledgements or feedback from the recipient. Since our bandwidth measurement records the *effective* rate of new original data traveling from the client to server, not counting bits associated with dropped packets or retransmissions, this metric is also commonly referred to as *goodput*.

To measure packet loss and jitter, we use IPerf to send a stream of UDP packets, from the client to the server, at the maximum possible offered rate supported by the network, and record the number of lost packets that fail to arrive at the server. IPerf also measures jitter, which is the difference in arrival times of received packets, resulting from queuing and forwarding in the network. Large fluctuations in inter-packet arrival times often results in poor performance of real-time applications like Voice-over-IP (VoIP) telephony. Data collected using the Ping and IPerf programs are manipulated using a Python script, and then imported and plotted in Matlab.

4.5.2 Experiments for Network Performance Collection

Experiment I: 802.3 Ethernet, Wired Infrastructure Mode

First, to test the readiness of our testbed and our ability to collect and manipulate data, we first checked the performance of our 100 Mbps 802.3 wired Ethernet backbone network. See Script 1 in Appendix 1 for setup instructions. The experiment measured baseline results from the 1-hop travel of data from Node 1 to Nodes 2, 3, and 4 respectively through the wired router. This configuration produced the lowest latency, jitter, and packet loss, and the highest goodput, and we use this as the benchmark to compare all other network types.

Experiment II: 802.11 WiFi, Wireless Ad-Hoc Mode

Experiment III: Bluetooth, Wireless Ad-Hoc Mode

Experiment IV: GNU Radio/USRP at 100Kbits/s DBPSK, Wireless Ad-Hoc Mode

Then, we measured the performance of our three hop ad-hoc 802.11 wireless network, our Bluetooth network, and our GNU Radio/USRP network with 100 Kbit/s DBPSK links. Script 2, Script 3, and Script 4 in the Appendix list these setup instructions respectively. An example resulting static routing table for the computers is shown in Figure A.1 in the Appendix. The experiment measured the QoS metrics between Node 1 and Node 2 (1-hop); between Node 2 and Node 3 (2-hops); and between Node 1 and Node 4 (3-hops).

4.6 Results

This section summarizes the results of all data collected for the 802.3 wired, 802.11 wireless, Bluetooth, and GNU Radio/USRP four node networks. All curves represent test traffic from Node 1 to Node 4. All wireless networks are configured in the three-hop ad-hoc topology of Figure 4.3. These results give insight into the performance of the smart radio architecture.

4.6.1 Latency

Figure 4.7 shows the relative latencies recorded for 802.11 WiFi, Bluetooth, GNU Radio/USRP and 802.3 wired Ethernet. 802.3 Ethernet's latency was flat and stable, but GNU Radio/USRP's, 802.11's and especially Bluetooth's latency varied up and down. Variations at the high end of the logarithmic scale also correspond to large amplitude fluctuations and these variations become more apparent in Section 4.6.3 where we discuss jitter.

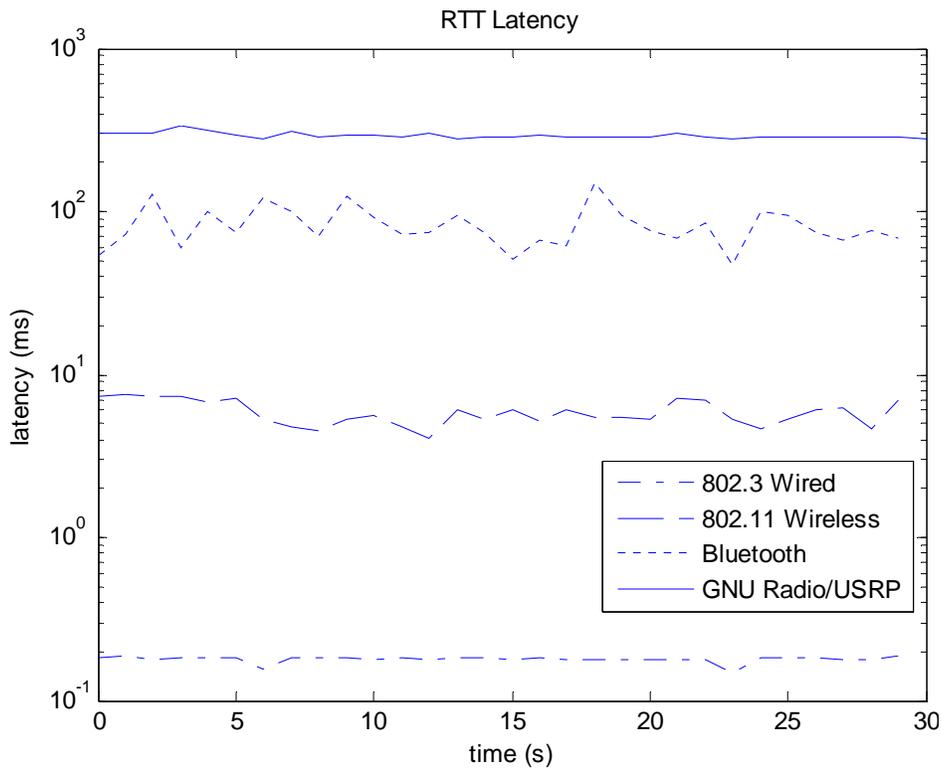


Figure 4.7. Summary of Latencies

4.6.2 Goodput

Figure 4.8 and Figure 4.9 show the measured TCP and UDP goodputs respectively. The TCP curves for 802.11 and GNU Radio/USRP contained discontinuities where goodput dropped to zero, resulting from dropped packets, retransmissions, and TCP flow control window fluctuations. However, the UDP curves were smooth, since retransmissions were not required. These two curves show that 802.11WiFi's goodput was one-and-a-half orders of magnitude

lower, Bluetooth was nearly three orders of magnitude lower, and GNU Radio/USRP was nearly four orders of magnitude lower than 802.3 Ethernet.

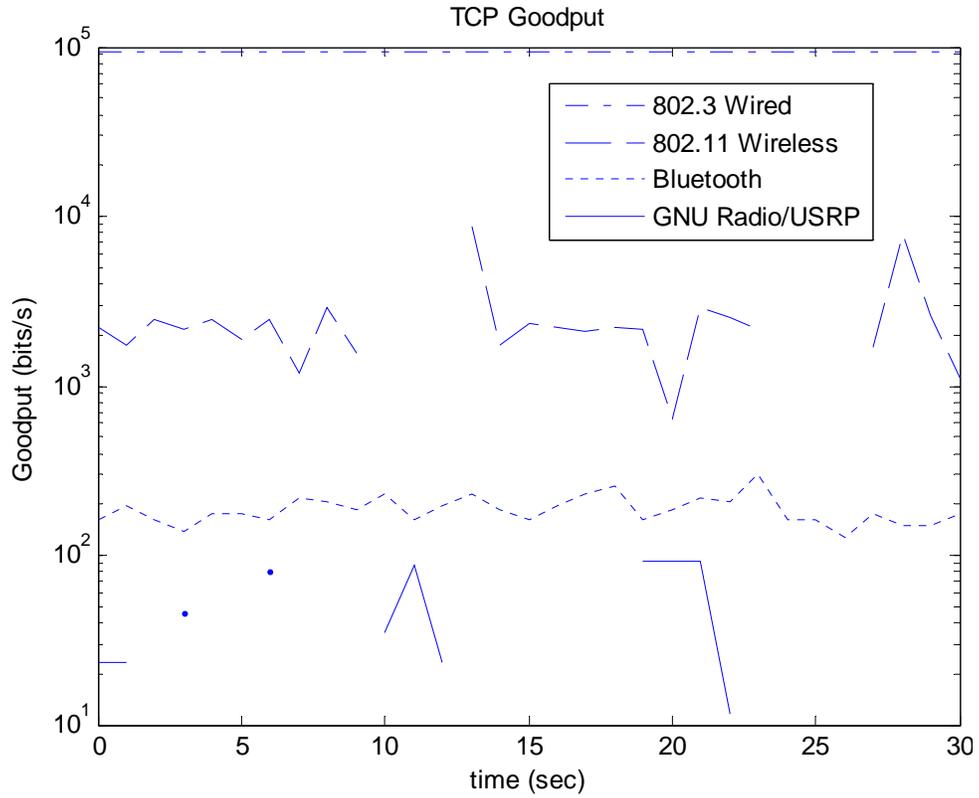


Figure 4.8. Summary of TCP Goodputs.

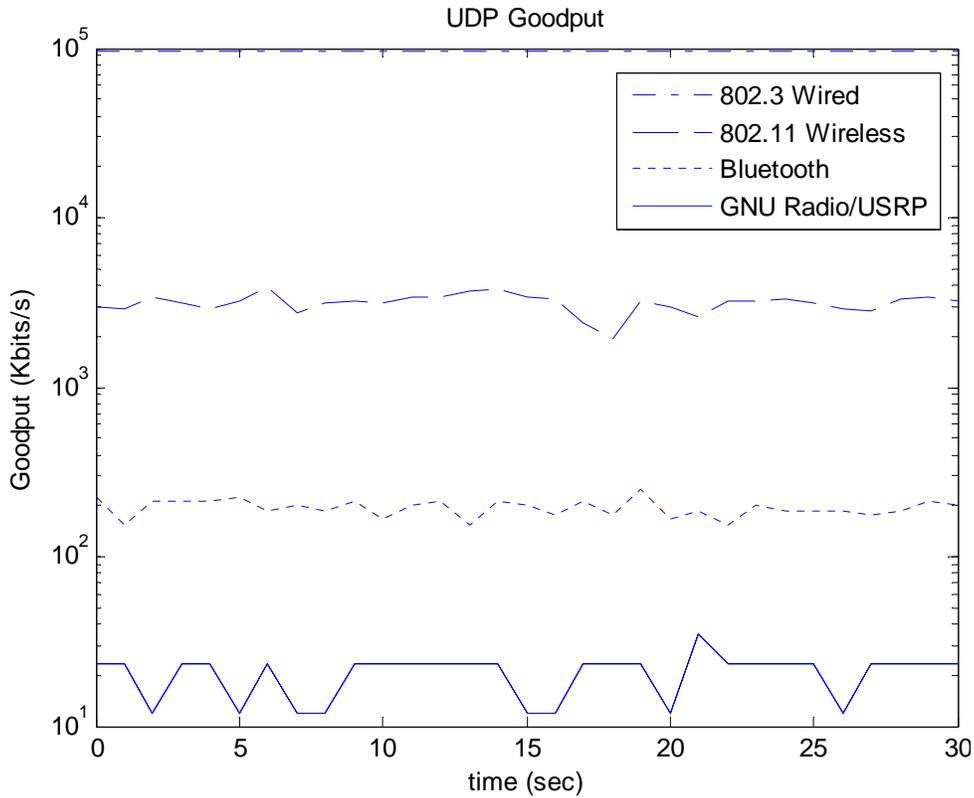


Figure 4.9. Summary of UDP Goodputs.

4.6.3 Jitter

Figure 4.10 shows the jitter for the three networks. GNU Radio/USRP's and Bluetooth's jitter was four orders of magnitude greater, and 802.11 WiFi's were three orders of magnitude greater than 802.3 Ethernet. Furthermore, 802.3 Ethernet was the most stable network, except for two random spikes at 20 and 26 ms. The Bluetooth and GNU Radio/USRP networks were the least stable, and their jitter curves contained regular oscillating ripples. The relative magnitude of these jitter oscillations is best observed when the vertical axis is linearly scaled, as in Figure 4.11.

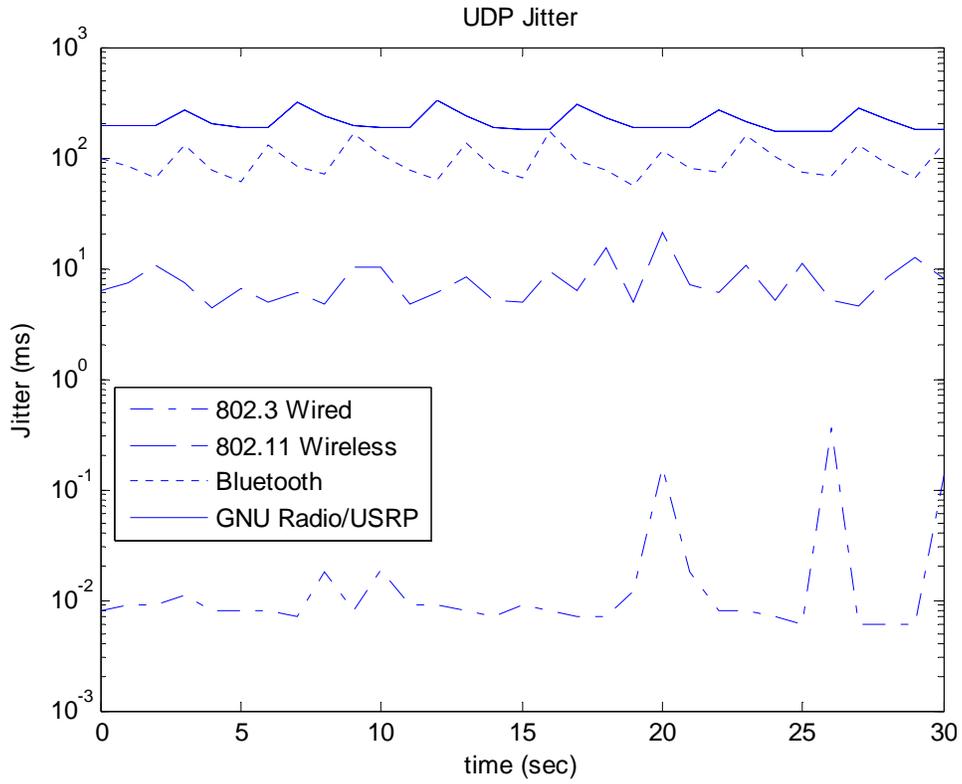


Figure 4.10. Summary of UDP Jitters.

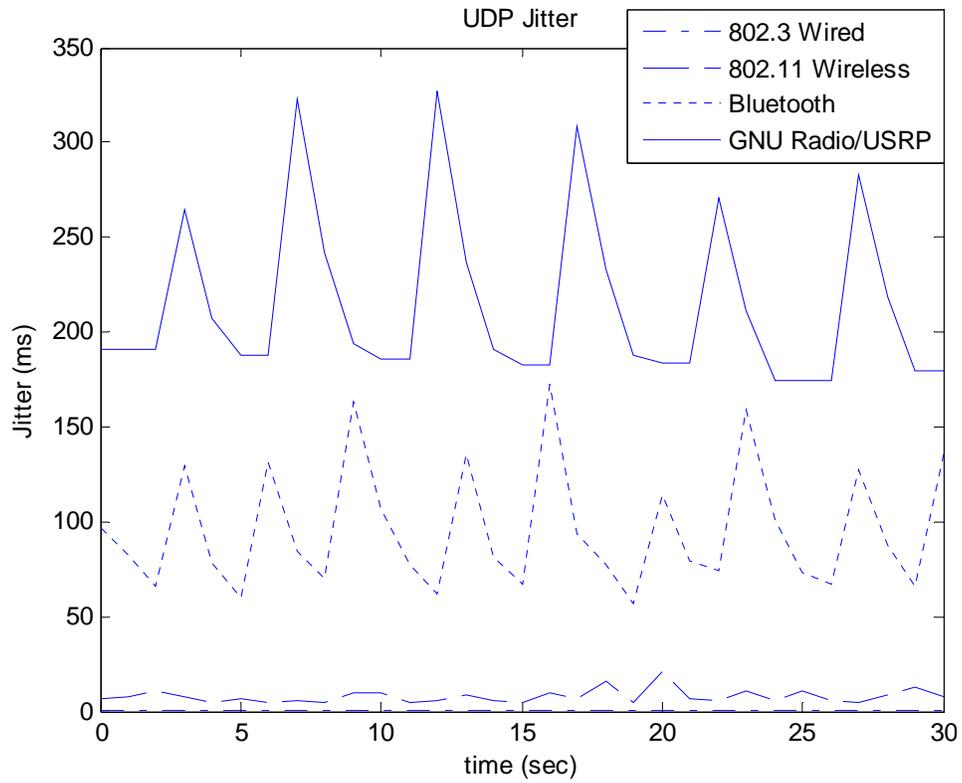


Figure 4.11. Summary of UDP Jitters (linear scale).

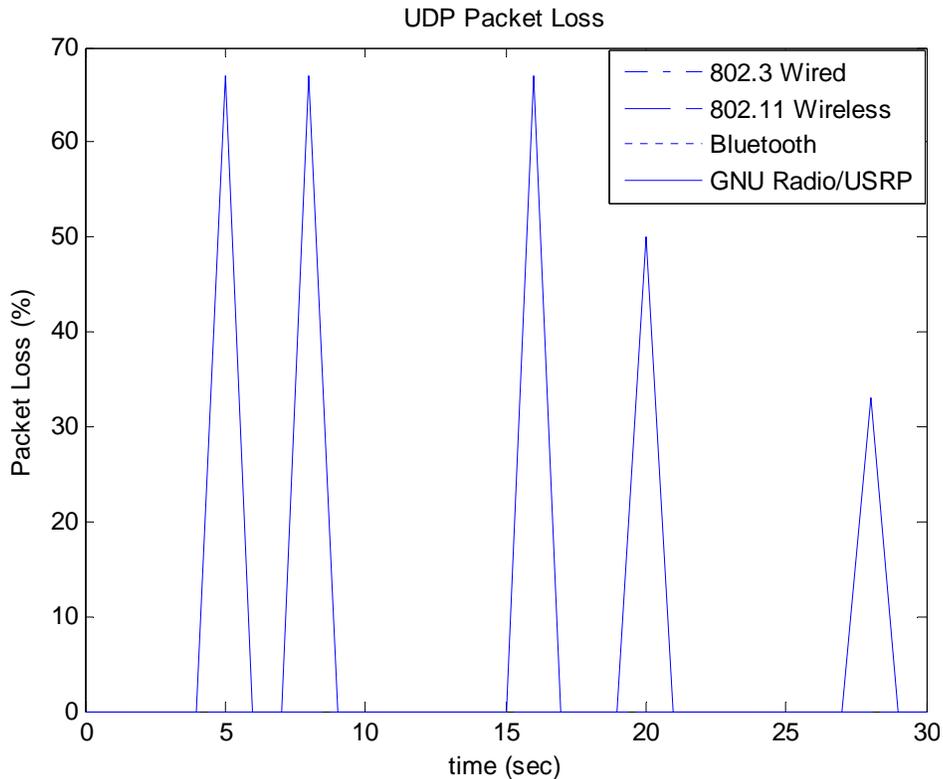


Figure 4.12. Summary of UDP Packet Loss.

4.6.4 Packet Loss

Figure 4.12 shows the packet loss for the three networks. Surprisingly, 802.3 Ethernet, 802.11 WiFi, and Bluetooth suffered little to no packet loss. GNU Radio/ USRP incurred noticeable loss, as depicted by the large spikes at times 5, 8, 16, 20, and 28 ms.

4.7 Discussion

First, with regards to all the wireless standards, we discovered that 802.11 WiFi had the best overall QoS. It had the highest goodput, and lowest latency, jitter, and packet loss. Its only downfall could potentially be the noticeable gaps in its TCP goodput curve, resulting from dropped packets, retransmissions, and fluctuations in the TCP flow control window. 802.11 would be the best choice for real-time applications, like streaming video or VoIP. Second, the Bluetooth standard had the second best QoS with middle-of-the-road scoring in latency, goodput, jitter, and packet loss. Bluetooth's TCP goodput was also much more constant than for 802.11; however, Bluetooth demonstrated significant variations in its jitter. Bluetooth would be useful

for non-real-time applications like file transfers and short bursty transmissions like push-to-talk audio. Third, in our experiments, the GNU Radio/USRP network interface, configured at 100 Kbits/s DBPSK, generally faired worse than any other network interface standard. It had the smallest goodput and the greatest latency, jitter, and packet loss. Its jitter also varied greatly with time. This interface would most likely have the most difficulty supporting real-time applications like VoIP, as well as non-real-time file transfers. In terms of Smart Radio Challenges, all waveforms and adapters tested here, when incorporated and controlled by the universal radio framework, would provide acceptable QoS in when used in an ad-hoc network.

4.8 Conclusions

This chapter described how to build an ad-hoc network testbed using commercial off-the-shelf equipment and open-source software, suitable for conducting basic network experiments, by students in a laboratory environment. The work constitutes a sampling of the core technologies used by our lab in the construction of smart radios for public safety disaster response scenarios. The performance data collected during these experiments also give us QoS metrics, which represent the baseline operational performance of an ad-hoc network of four CWT Smart Radios, when assembled with this same commercial off-the-shelf equipment. Next, we use this Smart Radio Network Testbed to evaluate two dynamic spectrum access protocols, Rendezvous and Channel Change, in Chapters 5 and 6 respectively.

Chapter 5

Rendezvous Protocol

Recent disasters including the 9/11 terrorist attacks, Hurricane Katrina, and the California wildfires, have highlighted the limitations of current mobile communication systems for first responders. Responders from differing agencies often possess radio systems made by competing manufacturers, which may be programmed for different frequency bands, modulations or protocols. This represents the *interoperability* problem, where the physical limitations of existing legacy radios prevent radios from directly talking with each other [36]. Recent cognitive radio research and prototypes have made progress in developing stand-alone cognitive radios which can quickly reconfigure themselves to talk directly with competing legacy public safety radios, or cognitive radio *gateways* which can dynamically locate competing public safety radios, reconfigure themselves, and automatically bridge them together [5, 24, 37]. These cognitive radio solutions effectively join otherwise inoperable systems, and allow the use of available existing public safety infrastructures.

A more complex design challenge is when all previous communication infrastructures are completely destroyed or non-existent, as would occur at the scene of a large-scale natural disaster. Here the challenge for communications engineers is to setup a temporary, fixed communications infrastructure for public safety first responders. In recent efforts, researchers are working to apply cognitive radios to solve this problem [38]. Cognitive radios provide ideal temporary backbone infrastructures, particularly because of their ability to scan and locate available spectrum, or to dynamically share existing spectrum. Their envisioned cognitive radio infrastructure network backbone would be completely digital and would use high-speed digital modulations, such as OFDM, between each deployed infrastructure node. Infrastructure nodes would also support the transmission and reception of existing public safety frequencies, modulations, and protocols, so as to interoperate with legacy radios at the end-points.

One challenge is how to quickly deploy and setup this backbone cognitive radio infrastructure. How do these cognitive radios find each other, and how do they determine which

channels to use for their initial links? Can this be accomplished in an automatic and systematic way that does not need the intervention of or pre-coordination by communications network designers? Some authors suggest the answer lies in sophisticated spectrum scanning and classification strategies [39, 40]. We view this approach as well suited to a heterogeneous network consisting of legacy, fixed, or non-cognitive nodes which are incapable of adhering to neighbor discovery etiquette. However, if the system is homogeneous, cognitive, or capable of following neighbor discover etiquette, then this approach may be over-kill. In this case, we believe the answer lies with novel sets of active, MAC-layer rendezvous protocols. Reliable, timely, and energy efficient rendezvous algorithms enable cognitive radio nodes to build neighbor information and link connection information, and they are the foundational tool in building well-connected cognitive radio networks.

5.1 Contributions

This chapter makes the following contributions. First, we review recent work in the development of rendezvous protocols. Second, we present a taxonomy which enables us to classify these authors' proposed protocols based on its design assumptions. Our efforts reveal the need for more research in asynchronous, multi-channel MAC rendezvous protocols, beyond that which has already been done for the Bluetooth system. Third, we restate the multi-channel, asynchronous problem, showing analytical models currently available. Fourth, we implement a few candidate rendezvous protocols on a cognitive radio testbed, to support our mathematical analysis and to validate previous authors' simulation results. Fifth, we discuss these protocols' incorporation within a cognitive radio, with respect to setting up a cognitive radio network infrastructure backbone for use by public safety first responders at the scene of a natural disaster.

5.2 Previous Work

The process of rendezvous, also commonly referred to as neighbor discovery, has been the topic of a number of recent papers. Early authors proposed mobile communication systems using a pre-defined number of *control channels* or *common pilot channels* in their systems for

use in rendezvous. This is satisfactory when the frequency band is relatively small or when the system has exclusive license to the frequency band. This also works fairly well if the system essentially acts as the only primary user in the band, and the risk of one of the pre-defined common pilot channels being occupied or unavailable due to a secondary user is low or non-existent. Cell phones, pagers, household wireless phones, etc. all use this concept. To a great extent, even the current 802.11 wireless internet standards use this construct [41-43]. For example, in 802.11b, all the nodes, including the wireless access point (WAP) and the individual network interface cards (NIC), use one of 11 overlapping channels, each 30 MHz wide, in the frequency range of 2.400-2.483GHz. Upon startup, and periodically throughout its operation, each NIC scans each of these 11 channels, searching for beacon frames from nearby WAPs.

More recent authors have attempted to improve upon the concept of the *common pilot channel* by allowing this location of the pilot to be flexible and mobile within a particular frequency band [44-47]. However, these systems still require knowledge about the location and characteristics of the pilot channel, or require the use of a centralized infrastructure server to manage the pilot channels. These systems only approach, but do not solve, the dynamic spectrum allocation requirements necessitated by the FCC's move toward spectrum re-use [10]. The solution genuinely needed is a rendezvous method that uses a truly dynamic pilot channel—one in which users need little or no a priori knowledge about its location within the frequency spectrum.

In [48], McGlynn presents a set of neighbor discovery protocols which he calls *birthday protocols*, coined after the probability of two or more people in a room having the same birthday. He describes a scenario where nodes and their link topology are static, as would be the case in a sensor network dispersed randomly throughout a region. Nodes are assumed to be in one of three states—transmit, receive, or sleep—in a given time slot, with fixed probabilities. He computes the fraction of the expected number of links that can be formed as a function of the number of nodes present, the number of time slots waited, and their state probabilities. Energy efficiency is obtained by varying the probability of nodes sleeping. This work forms a foundation for rendezvous analysis, but it is limited in that it assumes a single broadcast channel for rendezvous, and it assumes that time slots are fixed and synchronized across all nodes in the neighborhood.

In [49], Alonso also analyzes the problem of rendezvous for a single broadcast channel with synchronized time slots. However, he adds rules for beaconing and listening beyond the purely random state switching of McGlynn. These include the *answering protocol*, where nodes must immediately respond with an acknowledge message after locating a rendezvous beacon, and the *listen after talking protocol*, which conversely requires a node to listen for an acknowledgement message after it transmits each beacon. In [50], Alonso then goes on to provide one of the first mathematical analyses where the problem is extended to multi-channel broadcast channels. In [51], Angelosante also analyzes this multi-channel problem, with emphasis on physical layer demodulation and decoding. In [52, 53], Galluccio also addresses the multi-channel problem; however, he analyzes the problem of mobile nodes moving in space with velocity through the use of Markov chains. He emphasizes the importance of real systems needing to be multi-channel, as is the case for Bluetooth.

Bluetooth [54] was one of the first, widely-used wireless technologies to employ a multi-channel, frequency-hopped discovery algorithm based on probabilistic discovery models. Discovery occurs in an asynchronous manner using frequency-hopped beacons transmitted in slots determined by pseudo-random sequences. However, discovery is asymmetric. Some nodes must be designated as masters and others as slaves, and the widths of their time slots, as well as the algorithm in which beacons are transmitted and received in those slots, are specific to the node's role. Numerous authors have analyzed Bluetooth's performance for rendezvous and personal area network establishment [55, 56].

In [57, 58], Salonidis discusses how to overcome the limitation of the asymmetric nature of the Bluetooth discovery algorithm, by allowing nodes to randomly switch between master and slave roles, so that discovery happens in a symmetric, distributed, and more scalable fashion. Salonidis, and as well as Law [59], describes how this modification improves the ability to form Bluetooth scatternets.

Other authors focus their attention on developing their own unique multi-channel, symmetric rendezvous strategies, instead of attempting to modify the existing Bluetooth system. Avoiding the single-channel limitation of McGlynn, Balachandran [60] describes a multi-channel, symmetric, frequency-hopped, rendezvous protocol. The protocol transmits and listens for beacons on channels, as determined by a set of pseudo-random codes. He presents an analytical model for computing the probability of a collection of nodes all finding each other on

or before a particular discovery time slot, as a function of the number of nodes in the neighborhood, and the number of channels used. Balachandran also introduces the concept of dynamic spectrum access to neighbor discovery by limiting frequencies used by the pseudorandom hops so as to keep interference to primary users less than a given interference fraction.

Alternatively, Hamida [61], uses a completely different strategy which involves beacons transmitted at random time offsets, rather than at random hop frequencies. The operation of his *random protocols* is modeled after Aloha, in which nodes transmit beacons at random times within a fixed frame, and either receive or sleep for the remainder of the time in that frame. The main limitation of this analysis is that Hamida assumes a single broadcast channel, and nodes must have time synchronization in order to align these discovery frames. In [62], Fang performs an in-depth comparison and contrast of the benefits and performance of both the birthday protocols of McGlynn, and the random protocols of Hamida, with regards to protocol duration, energy consumption of effective messages, total energy consumption, and energy efficiency. In [63], Yang finds a general solution relating optimal energy savings to the probability of occurrence of the sleep state for the rendezvous between two nodes.

In [64], Borbash presents one of the first analytical solutions to the general asynchronous rendezvous problem, in which nodes transmit and receive beacon messages in slots that are not aligned in time. Fang suggests that this is a generalized combination of the birthday and random protocols [62]. Unfortunately, Borbash's model is only for single broadcast channels, and has not yet been extended to the multi-channel problem. We explore extending Borbash's analysis in Section 5.3.

In [65], DaSilva introduces a rendezvous algorithm for use in dynamic spectrum access cognitive radios. Similar to Balachandran, as well as in our own previous work [6], nodes transmit and listen for beacons on channels whose frequencies are determined by the use of pseudo-random hop-sequences. However, DaSilva investigates the selection of the pseudo-random sequences so as to improve the probability of rendezvous. He concludes that completely random sequences result in *blind rendezvous*, which has no upper bound on the time-to-rendezvous (TTR); however, carefully chosen sequences with specific patterns of periodicity result in *sequence-based rendezvous*, which does have an upper bound on the expected TTR. Further, by scanning the spectrum band and by pruning channels which are occupied, the radio

can adhere to dynamic spectrum access policies and avoid interference to primary users. For his analysis, DaSilva assumes CR nodes are synchronized and rendezvous slots are aligned in time.

In [66], Arachchige introduces an asynchronous rendezvous algorithm to be applied to cognitive radio networks. The scheme requires that a node be designated as the single leader. The leader performs the task of neighbor discovery and global channel set selection, and then it relays this information to the other nodes which it has discovered in the neighborhood. This scheme eliminates the requirement for nodes to be time synchronized, but the centralized approach might present limitations when scaled to larger collections of nodes.

In [67, 68], Horine presents and evaluates a completely different approach to rendezvous based on frequency domain statistics. Instead of transmitting packet data via a digital modulation scheme, Horine's nodes transmit and scan for simple carriers encoded with a number of analog sidetones. The system is built and tested using GNU Radio, and it may prove to be a useful alternate strategy in special applications requiring beacons with minimal power, RF bandwidth, or risk of unintentional interference to adjacent users.

5.3 Rendezvous Design Taxonomy

The design and implementation of rendezvous protocols can be categorized by nine significant design attributes. These considerations are listed in Table 5.1. We use these when discussing configuration and operation of the protocols in Section 5.3.

TABLE 5.1. RENDEZVOUS DESIGN TAXONOMY

Parameter	Possible Values
Channels	Single / Multiple
Collection	Independent / Shared
Timing	Asynchronous / Synchronous
Direction	1-Way / 2-Way
Messages	BCN / BCN-ACK
Slot Widths	Equal / Varying
Roles	Ad-Hoc / Master-Slave
Collision Avoidance	None / CSMA
Stochastic Aspects	State / Frequency / Time / None

5.3.1 Channelization: Single or Multiple

Channelization describes the number of frequencies in use by the radios during execution of the rendezvous protocol. While most systems reviewed assume a single radio transceiver operating in half-duplex mode, designers vary in their allocation and use of channels. Some rely on a single channel broadcast operation, as has been customary for sensor networks or early wireless communications systems, like 802.11. Others use multiple channels, typically involving frequency-hopping via a set of pseudo-random codes, as in Bluetooth, or newer cognitive radio architectures.

5.3.2 Neighbor Information Collection: Independent or Shared

In an independent neighbor information collection design, the responsibility for radio discovery and rendezvous rests solely with the end nodes. Nodes are not required to share this information with their neighbors. This strategy works well when all nodes are nearby in a 1-hop configuration, or in a sensor or energy-limited network, where the overhead of neighbor table coordination and sharing would be prohibitive. In a shared collection scheme, nodes collaborate in collecting and sharing neighbor discovery information. This can either happen in a centralized fashion, where a single host or server directs and manages the operations of radio discovery for the entire network, or in a distributed fashion, where nodes exchange local discovery information with their nearby peers until the information is eventually disseminated throughout the neighborhood.

5.3.3 Discovery Timing: Synchronous or Asynchronous

A synchronous rendezvous protocol requires that all nodes have a precisely aligned clock, and that all of the slots for the transmitted and received message frames are in alignment. This requires that the radio system can maintain sufficient timing requirements, has minimum latency issues, and can adhere to rendezvous slot boundaries. Asynchronous systems do not require precisely aligned system clocks and levy less strict requirements on hardware precision, timing, and latency.

5.3.4 Discovery Direction: 1-Way or 2-Way

Broadly speaking, discovery direction addresses the question of symmetry in the discovery process and the nature of the communications between the locating and located nodes. In an one-directional scheme, Node A may be able discover Node B by recovering its beacon; however, since the discoverer's communication link is essentially one-directional, B will not be aware of A's presence. Conversely, if a discovery communication is two-directional, then when A recovers B's message, B is also notified of A's presence. In this way, mutual discovery is made regardless of which node initially transmitted or received a beacon message. Practically speaking, two-directional discovery requires the transmission of an acknowledge message at the MAC or high layer in the protocol stack after the initial discovery. We say more about message types in the next section.

5.3.5 Discovery Messages

During the Rendezvous protocol's execution, we envision the use of at least two types of MAC-layer packets, depending on the required discovery direction. In the 1-Way discovery strategy, the locating nodes only need to transmit out beacon (BCN) messages. However, in a 2-Way discovery process, the located node may also send a MAC-layer acknowledgement (ACK) response message back to the locating node after successfully recovering the original BCN message. Discovery messages must have a header which includes the message type, sequence number, transmission identifier, and optionally, a channel number. Sequence numbers are required if the discovered node is configured to transmit ACK messages. An ACK message will contain the sequence number of the BCN packet which it is acknowledging. The BCN messages, as well as the ACK response message, may contain node-specific information such as MAC and IP addresses, etc.

5.3.6 Discovery Slots Widths: Equal or Varying

In the rendezvous process, the term *slots* refers to the internal time scheduling of the radio node, during which it performs its rendezvous operations. For example, during each time slot, the radio may be in either a transmit mode (Tx), when it is actively sending out beacon messages; receive mode (Rx), when it is actively receiving or scanning for other beacons; or in

an idle mode, when it is essentially asleep to conserve energy. In a fixed-slot system, all nodes in the system use slots of *equal*, fixed length. In a variable-slot system, nodes use slots with *unequal*, variable lengths.

5.3.7 Nodal Roles: Master-Slave or Ad-Hoc

During the rendezvous process, systems can opt to assign two types of roles to individual nodes in the network. In the Master-Slave configuration, a small fraction of the nodes is designated as masters, while the remainder is designated as slaves. Masters typically initiate the rendezvous process by actively transmitting beacon messages. Slaves typically scan, or passively monitor the spectrum, and they only respond when they receive a beacon message from a master. In the Ad-Hoc configuration, all nodes have equal responsibility for transmitting and receiving beacons in the rendezvous process. They essentially must perform all tasks normally associated with both master and slave nodes.

5.3.8 Collision Avoidance: None or CSMA

Rendezvous protocols can also be classified by the degree to which they attempt to prevent packet collisions when actively transmitting beacon and acknowledgement messages. Excessive packet collisions reduce the effectiveness of the discovery protocol and reduce energy efficiency through wasted transmissions. Furthermore, excess packet collisions introduce undue interference into the operational frequency band, and in dynamic spectrum access scenarios, this may result in degraded performance for other primary users in that band. Some rendezvous protocols have little or no provisions for collision avoidance. This approach may be satisfactory if neighborhood sizes are small, the number of discovery channels is large, the nodes transmit beacons with a relatively low duty cycle, or the length of the rendezvous process is sufficiently long so as to account for any lost transmissions. Other systems use a more active carrier-sense, multiple-access (CSMA) scheme to sense the channel for activity prior to transmitting beacon messages.

5.3.9 Stochastic Aspects: State, Frequency, Time, or None

The majority of the rendezvous protocols reviewed rely on probabilistic strategies to accomplish rendezvous. These tend to be more scalable than deterministic schemes for larger networks. However, systems vary on which elements of the system are random. First, many let the transmission state, e.g. transmit, receive, or sleep, be determined randomly via pre-determined probabilities. Second, if the system is multi-channel, some designers may let the operating channel for each rendezvous slot be determined randomly, or by a set of pseudo-random sequences. Third, other systems model their rendezvous scheme after Aloha, and transmit beacons at random times within a fixed frame structure.

5.3.10 Applying to Related Work

Applying these classifications to the related work discussed in Section 5.1 results in the Table 5.2. The table allows us to determine trends and identify which aspects of Rendezvous need more research attention. The key findings are:

1. The foundational papers on rendezvous, such as McGlynn, Hamida, Borbash, focused only on the *single channel* instance. This highlights the need for more *multi-channel* analysis, particularly since many real world systems, as well as upcoming cognitive radio systems, will require operation on multiple channels.
2. Most rendezvous schemes are based on *independent* neighbor information collection, meaning that each node is responsible for scanning and building up its neighbor table. This leaves open the possibility for future research in *collaborative* rendezvous, regardless of whether that uses a centralized or decentralized approach.
3. The large majority of rendezvous algorithms currently available require at least some degree of time *synchronization* between the nodes in the networks. In our opinion, this is a tenuous assumption that may be difficult or impossible to achieve in real systems. This is further evidenced by the fact that one of best well known and successful wireless networking

systems, Bluetooth, is *asynchronous*. More research and analysis is needed for alternate *asynchronous* strategies.

4. Few systems have analyzed or experimentally tested *varying-width* rendezvous slots. Nearly every scheme assumes *fixed slots of equal length*, with the exception of Bluetooth, where the slave's slots are M times the lengths of the master's slot, where M is the number of hop frequencies used in discovery. This gap warrants more exploration.
5. Most systems have aimed to be *symmetric* rather than *asymmetric*. This is mostly like the appropriate trend, for ease of deployability and scalability. Bluetooth uses asymmetric rendezvous, but authors frequently have cited this as a limitation, and many have suggested modifications to make the algorithm symmetric.
6. Few rendezvous strategies use active *collision avoidance* schemes like CSMA or CSMA/CA. Instead, most allow collisions to occur, but assume that rendezvous will still succeed on average, with an expected duration. This opens the door to experimentally evaluating protocols that use active collision avoidance schemes to see if the performance increase is worth the carrier sensing and processing overhead.
7. Last, most algorithms only use *stochastic* and probabilistic methods to configure the *state* of the node, i.e. transmit, receive, or sleep, as well as the current *operating frequency*. However, few have used probabilistic methods to vary the *time positioning* of the rendezvous beacons within their discovery slots. This likewise warrants more research; however, we assert that designing a system using beacons with random time-offsets is equivalent to designing an *asynchronous* system. Relaxing the requirement for time synchronization of rendezvous slots naturally results in random time-variation in the transmission and reception of beacons.

In the remainder of this chapter, we address Area 1 and Areas 3 for more research in multi-channel asynchronous, MAC rendezvous protocols. We also experiment with using varying-

width rendezvous slots. We now detail the multi-channel, asynchronous problem, showing analytical models currently available.

Author	Channels		Collection		Timing		Direction		Messages		Slot-Widths		Roles (Symmetry)		Collision Avoidance		Randomnicity				
	Single	Multi	Independent	Shared	Synchronous	Asynchronous	1-Way	2-Way	BCN	BCN-ACK	Equal	Varying	Adhoc (Symmetry)	Master-Slave (Assymetric)	None	CSMA	State	Frequency	Time	None	
McGlynn	X	X	X		X		X		X		X		X		X		X				
Alonso	X	X	X		X			X	X		X		X		X		X	X			
Angelosante		X	X		X				X		X		X		X		X	X			
Galluccio		X	X		X						X		X		X		X	X			
Bluetooth		X	X			X		X			X			X			X	X			
Salonidis		X	X		X			X	X		X		X			X	X	X			
Law		X	X		X			X	X		X		X			X	X	X			
Balachandran		X	X		X			X		X		X		X		X	X	X			
Hamida	X	X	X		X				X		X		X		X		X		X		
Fang	X	X	X		X		X		X		X		X		X		X	X			
Yang	X	X	X		X						X		X		X		X		X		
Bordash	X		X			X		X		X		X		X		X				X	
DaSilva		X	X		X				X								X	X			
Silvius		X	X		X			X		X			X		X		X	X			
Arachchige		X		X		X				X				X			X	X			
Horine		X	X			X	X		X				X		X						X

Table 5.2. Applying Rendezvous Design Taxonomy to Related Works.

5.4 Problem Statement

In this section, we present the multi-channel, asynchronous problem, and develop an analytical model to predict its behavior. For simplicity, our presentation assumes independent neighbor information collection, 1-way discovery direction, only beacon (BCN) messages, ad-hoc roles (i.e. symmetric operation), with no active collision avoidance system enabled. First, our strategy is to review the derivation of an analytic model for the single-channel problem, and then extend this to the multi-channel case. Second, we take the synchronous cases and extend these solutions to the asynchronous cases.

5.4.1 Single-Channel, Synchronous

The single-channel, synchronous model here parallels that which is presented in McGlynn [48] and in the first half of Borbash [64]. First, assume that we observe a Node X and compute the number of neighbor discoveries it makes with $N-1$ other nodes around it. These other nodes are labeled Y_1 through Y_{N-1} . We assume each node can either be in transmit (Tx) mode with probability p_T , or a receive mode (Rx) with probability $p_R=(1-p_T)$ in any given rendezvous slot. There are a total of S rendezvous time slots. We can view the states of Node X and the other Nodes Y_i , at a given instance using a table, as shown in Figure 5.1.

Node								
X	Tx, Ch1	Rx, Ch1	Tx, Ch1	Tx, Ch1	Rx, Ch1	Tx, Ch1
Y1	Rx, Ch1					
Y2	Rx, Ch1	Tx, Ch1	Rx, Ch1	Rx, Ch1	Rx, Ch1	Tx, Ch1
Y3	Rx, Ch1	Rx, Ch1	Tx, Ch1	Rx, Ch1	Rx, Ch1	Rx, Ch1
Y4	Tx, Ch1	Rx, Ch1	Tx, Ch1	Tx, Ch1	Rx, Ch1	Rx, Ch1
...	...							
...								
...								
...								
Y(N-1)	Rx, Ch1	Rx, Ch1	Rx, Ch1	Rx, Ch1	Tx, Ch1	Tx, Ch1
Time	1	2	3	4	5	6	...	S

Figure 5.1. Node state information for the single channel, synchronous case.
[M. Silvius' Image from [69]. Used with permission, © 2009 IEEE.]

Since this scheme uses a single broadcast channel, any time more than two nodes are in the Tx state in the same time slot (i.e. column), there will be a collision of beacon messages. Although real systems may still be able to receive beacon information if there is a significant

different in power levels between the transmitting nodes, for our analysis, we assume that the information will be corrupted and rendezvous will not occur. Also, if we assume a 1-Way Discovery, the only way that Node X can find another node, is for it to be listening in the Rx state, with one, and only one other of the (N-1) Node Y_i's transmitting a beacon message in the Tx state.

First, we let h equal the number of discoveries X hears in a given time slot. A number of h values with their associated scenarios are given below in (1).

$$h = \left\{ \begin{array}{l} 0, \quad \text{if } (X == Rx) \text{ and } \sum_{i=1}^{N-1} (Y_i == Tx) = 0 \\ 1, \quad \text{if } (X == Rx) \text{ and } \sum_{i=1}^{N-1} (Y_i == Tx) = 1 \\ 0, \quad \text{if } (X == Rx) \text{ and } \sum_{i=1}^{N-1} (Y_i == Tx) > 1 \\ 0, \quad \text{if } (X == Tx) \end{array} \right\} \quad (1)$$

As we can see, the number of discoveries equals zero for the instances when no Y_i are transmitting, when more than one Y_i is transmitting, or when X is transmitting. Next, we compute the expected number of discoveries per rendezvous slot, $E(h)$. This computation is set up in (2)

$$E(h) = 1 \cdot \Pr \left\{ (X == Rx) \text{ and } \sum_{i=1}^{N-1} (Y_i == Tx) = 1 \right\} + 0 \cdot \Pr \{ \dots \} \quad (2)$$

To compute this probability, remember that the number of nodes transmitting down a column of Figure 5.1 follows Binomial distribution, since it has a discrete probability distribution consisting of a series of N-1 independent events, each with a two states, transmit (Tx) and receive (Rx), and with fixed probabilities, p_T and $p_R=(1-p_T)$ respectively.

$$\begin{aligned} E(h) &= 1 \cdot p_R \cdot \binom{N-1}{1} p_T^1 p_R^{N-2} \\ E(h) &= 1 \cdot (1-p_T) \cdot \binom{N-1}{1} p_T^1 (1-p_T)^{N-2} \\ E(h) &= (N-1) p_T (1-p_T)^{N-1} \end{aligned} \quad (3)$$

Since we know from (1) that we can only make one discovery in any particular rendezvous slot, expression (3) also represents the probability that Node X finds any other Node Y_i in a specific slot, or

$$E(h) = \Pr\{X \text{ finds any } Y_i \text{ in a specific slot}\} \quad (4)$$

As discussed in [64], since all nodes act identically, and due to the symmetry of problem, the probability that Node X finds a specific Node Y_i is $1/(N-1)$ of this value, or

$$p_1 = \Pr\{X \text{ finds a specific } Y_i \text{ in a specific slot}\} = \frac{E(h)}{N-1} \quad (5)$$

Now, the goal is to compute how many of Node Y_i 's will be discovered by Node X during the duration of the rendezvous process across all S slots. We start by computing the number of times, k , Node X successfully discovers a specific Node Y_i . This follows the Binomial distribution,

$$\Pr[K = k] = \binom{S}{k} p_1^k (1 - p_1)^{S-k} \quad (6)$$

However, since S is typically large and the probability p_1 is small, the Binomial distribution can be approximated using the Poisson distribution,

$$\Pr[K = k] \approx \frac{\alpha^k}{k!} e^{-\alpha} \quad (7)$$

where the average number of the successes in a specific time interval, α , is given by

$$\alpha = S \cdot p_1 = \frac{S \cdot E(h)}{N-1} \quad (8)$$

The probability that a specific Y_i is discovered in at least one of the slots is one minus the probability of not being found, or

$$\begin{aligned} & \Pr\{X \text{ finds a specific } Y_i \text{ at least once}\} \\ &= 1 - \Pr[K = 0] \\ &= 1 - e^{-\alpha} \\ &= 1 - e^{-\left[\frac{S \cdot E(h)}{N-1}\right]} \end{aligned} \quad (9)$$

As suggested in [64], one metric for the rendezvous process is the fraction of links discovered during its execution, or

$$F(t) = E \left\{ \frac{d(t)}{z} \right\} \quad (10)$$

where z is the total number of links available and $d(t)$ are those links discovered during a duration of time, t . In our case, $z=(N-1)$. Due to the symmetry of the problem, we can assume that the probability of finding a specific Node Y_i is equal to finding any other Node Y_j . Therefore, the total number of discovered links is

$$E \{d(t)\} = (N - 1) \cdot \Pr \{X \text{ finds a specific } Y_i \text{ at least once}\} \quad (11)$$

Substituting this back into (10), and assuming that each of the S slots has duration, T , we have,

$$F(ST) = 1 - e^{-\left[\frac{S \cdot E(h)}{N-1} \right]} \quad (12)$$

Plotting expression (12) versus the number of nodes, N , leads to Figure 5.2. The first two traces, when $p_T=0.10$ and $p_T=0.14$ are identical to the ones depicted in [64]. The last two are those with the higher transmit probabilities $p_T=0.25$ and $p_T=0.50$. Note that these higher transmit probabilities work well only for small collections of nodes, e.g. $N \leq 3$ and $N \leq 5$ respectively.

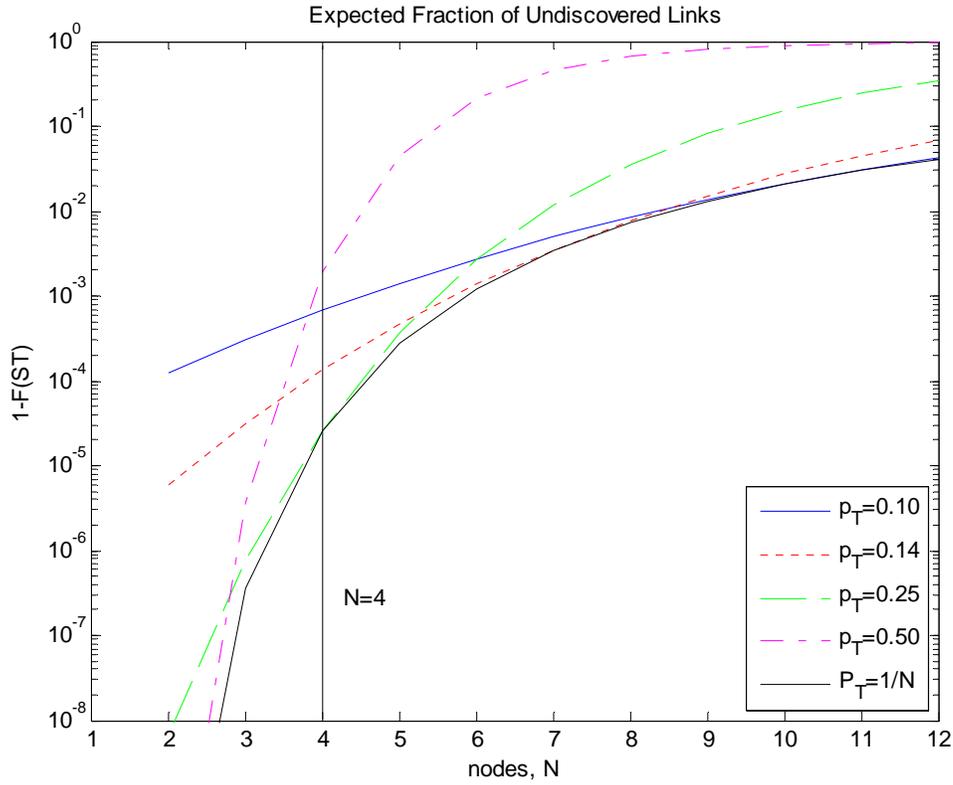


Figure 5.2. Fraction of undiscovered neighbor $\log_{10}(1-F)$ versus the number of nodes, N , for the single-channel, synchronous case.

Furthermore, we can compute the transmit probability, p_T , that maximizes $E(h)$, the probability that Node X finds a Node Y_i within a specific slot by setting the first derivative of (3) and solving for p_T .

$$\frac{dE(h)}{dp_T} = \frac{d}{dp_T} \left((N-1)p_T(1-p_T)^{N-1} \right) = 0 \quad (13)$$

which yields,

$$p_T = \frac{1}{N} \quad (14)$$

Therefore, the transmit probability that maximizing discovery is equal to one over the number of nodes, N , or if not known, the estimated number of nodes, \hat{N} . In Figure 5.2, we can see that the lowest fraction of remaining nodes is achieved when set $p_T=1/N$. This is shown by the bottom, innermost trace of tightest curvature.

5.4.2 Multi-Channel, Synchronous

This multi-channel, synchronous case has setup similar to that described in Section A, but in this case, each node can be on one of M channels. Rendezvous occurs when Node X is Rx mode on a channel, and only one other $(N-1)$ Node Y_i 's are in Tx on that same channel, as shown in Figure 5.3.

Node								
X	Tx, Ch1	Rx, Ch2	Tx, Ch6	Tx, Ch2	Rx, ChM	Tx, Ch8
Y1	Rx, Ch2	Rx, Ch1	Rx, Ch8	Rx, Ch9	Rx, Ch3	Rx, Ch2
Y2	Rx, Ch3	Tx, Ch2	Rx, ChM	Rx, Ch3	Rx, Ch4	Tx, Ch6
Y3	Rx, Ch4	Rx, Ch3	Tx, Ch7	Rx, Ch4	Rx, Ch7	Rx, Ch4
Y4	Tx, Ch5	Rx, Ch5	Tx, Ch3	Tx, Ch8	Rx, Ch5	Rx, Ch5
...	...							
...								
...								
...								
Y(N-1)	Rx, ChM	Rx, Ch7	Rx, Ch2	Rx, Ch1	Tx, ChM	Tx, Ch3
Time	1	2	3	4	5	6	...	S

Figure 5.3. Node state information for the multi-channel, synchronous case. [M. Silvius' Image from [69]. Used with permission, © 2009 IEEE.]

Again, we let h equal the number of discoveries X hears in a given time slot. A number of h values with their associated scenarios are given below in (15).

$$h = \left\{ \begin{array}{l} 0, \quad \text{if } (X == Rx) \text{ and } \sum_{i=1}^{N-1} [(Y_i == Tx) \text{ and } (f_{Y_i} == f_X)] = 0 \\ 1, \quad \text{if } (X == Rx) \text{ and } \sum_{i=1}^{N-1} [(Y_i == Tx) \text{ and } (f_{Y_i} == f_X)] = 1 \\ 0, \quad \text{if } (X == Rx) \text{ and } \sum_{i=1}^{N-1} [(Y_i == Tx) \text{ and } (f_{Y_i} == f_X)] > 1 \\ 0, \quad \text{if } (X == Tx) \end{array} \right. \quad (15)$$

Next, we compute the expected number of discoveries per rendezvous slot, $E(h)$. This computation is set up in (16)

$$E(h) = 1 \cdot \Pr \left\{ \text{if } (X == Rx) \text{ and } \sum_{i=1}^{N-1} [(Y_i == Tx) \text{ and } (f_{Y_i} == f_X)] = 0 \right\} \quad (16)$$

To compute this probability, remember that the number of nodes transmitting on a channel down a column of Figure 5.3 follows Binomial distribution, with probabilities p_T and p_R , times the probability of residing on a specific channel m , or p_M . If we let,

$$p_2 = \Pr\{(Y_i == Tx) \text{ and } (f_{Y_i} == f_X)\} \quad (17)$$

$$p_2 = p_T p_M = p_T \left(\frac{1}{M}\right) \quad (18)$$

then, we have,

$$E(h) = p_R \cdot \binom{N-1}{1} p_2^1 (1-p_2)^{N-2} \quad (19)$$

Simplifying yields the equation involving the number of nodes, N , the number of channels M , and the transmit probability of each node $p_T = (1-p_R)$.

$$E(h) = \left(\frac{1}{M}\right) (N-1) (1-p_T) p_T \left[1 - p_T \left(\frac{1}{M}\right)\right]^{N-2} \quad (20)$$

It is important to note, that when only one channel is used, i.e. $M=1$, expression (20) simplifies to expression (3), as would be expected. Likewise, as in (5),

$$\Pr\left\{\begin{array}{l} X \text{ finds a specific } Y_i \text{ in a} \\ \text{specific slot on channel } m \end{array}\right\} = \frac{E(h)}{N-1} \quad (21)$$

The remainder of the derivation follows exactly the same as in the single-channel case, with the end result being again that after S slots with duration, T , we have the expected number of discovered links being,

$$F(ST) = 1 - e^{-\left[\frac{S \cdot E(h)}{N-1}\right]} \quad (22)$$

Using expression (22), we plot the expected fraction of undiscovered links (i.e. one minus the expected discovered links), to produce Figure 5.4. Unlike in Figure 5.2, where we plotted this fraction as a function of the number of nodes, here, we fix the nodes at $N=4$, and we vary the number of channels, M . As expected the fraction of undiscovered nodes increases as the number of channels increase. However, unlike in Figure 5.2, large transition probabilities, such as

$p_T=0.25$ and $p_T=0.50$, produce better results than lower probabilities, $p_T=0.10$ and $p_T=0.14$, for a given number of channels.

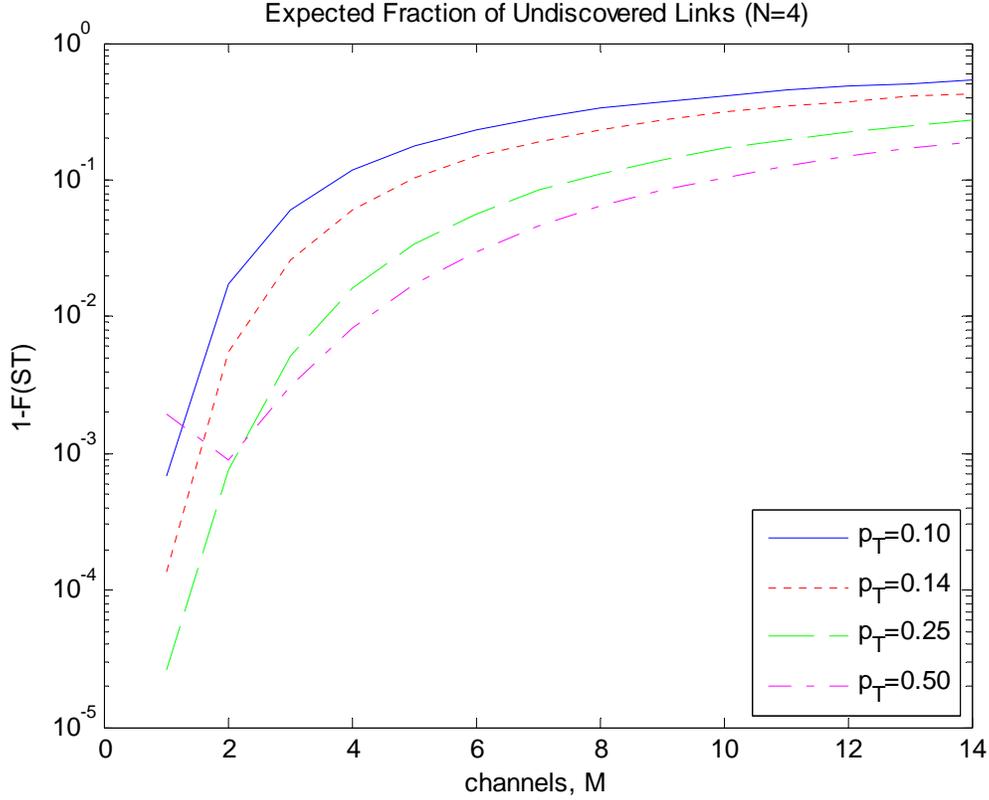


Figure 5.4. Fraction of undiscovered neighbor $\log_{10}(1-F)$ versus the number of channels, M , for four nodes ($N=4$), for the multi-channel, synchronous case.

Furthermore, we can compute the optimum transmit probability, p_T , that maximizes $E(h)$, the probability that Node X finds a Node Y_i within a specific slot and channel m by setting the first derivative of (20) to zero and solving for p_T .

$$\frac{dE(h)}{dp_T} = \frac{d}{dp_T} \left\{ \left(\frac{1}{M} \right) (N-1) (1-p_T) p_T \left[1 - p_T \left(\frac{1}{M} \right) \right]^{N-2} \right\} = 0 \quad (23)$$

yields, the expression,

$$p_T = \frac{1}{2N} (N-1) + \frac{M}{N} \pm \frac{M}{2N} \left\{ \left[\frac{1}{M} (N-1) + 2 \right]^2 - \frac{4N}{M} \right\}^{1/2} \quad (24)$$

When M is large, the optimum transmit probability tends toward $p_T \approx 1/2$, but when M is small, the optimum transmit probability tends toward $p_T \approx 1/N$.

At this point, it's worthwhile to discuss other synchronous derivations and how they compare. In particular, we consider the derivation of the multi-channel synchronous case in Balachandran [60]. In his derivation, he presents an expression for the probability of discovery between two nodes. In a formula that resembles our expression for $E(h)$ in (20), he states,

$$\begin{aligned} & \Pr\{X \text{ finds } Y \text{ in a specific slot}\} \\ & = p_d = 2p_T p_R (1 - I_f) / M \end{aligned} \quad (25)$$

where p_T and p_R are the transmit and receive probabilities of a specific node, and M is the number of discovery channels in use. The factor of two is introduced, because Balachandran also assumes a 2-Way discovery strategy, since rendezvous success occurs regardless of which node originates and which receives the beacon message. I_f is a quantity that addresses deferring transmission of beacons on channels occupied by primary users. This expression holds only when Nodes X and Node Y are the only two nodes running the rendezvous protocol. If neighborhood sizes increase beyond $N > 2$, excessive collisions make this estimate overly optimistic. Likewise, our initial attempts to use expression (25) to compute the rendezvous of entire neighbors of nodes, $N \gg 2$, did not address the degradation of collisions, and may be overly optimistic [6]. One possible way to solve this issue would be to include an active collision avoidance technique like CSMA or CSMA/CA.

5.4.3 Multi-Channel, Asynchronous

In this section, we consider the problem of rendezvous when neighboring nodes do not have time synchronization. Most of the configuration information stated in Section 5.4.2 still applies. We observe a Node X and compute the number of neighbor discoveries it makes with N other nodes around it. These other nodes are labeled Y_1 through Y_{N-1} . We assume each node can either be in transmit (Tx) mode with probability p_T , or a receive mode (Rx) with probability $p_R = (1 - p_T)$ in any give rendezvous slot. Each can be in one of M channels, which are equally likely. There are a total of S rendezvous time slots. We can again view the states of Node X and the other Nodes Y_i , at a given instance using a table, but this time, the alignment of the slots in time is random. This is shown in Figure 5.5.

Node									
X	Tx, Ch1	Rx, Ch2	Tx, Ch6	Tx, Ch2	Rx, ChM	Tx, Ch8	
Y1	Rx, Ch2	Rx, Ch1	Rx, Ch8	Rx, Ch9	Rx, Ch3	Rx, Ch2	
Y2	Rx, Ch3	Tx, Ch2	Rx, ChM	Rx, Ch3	Rx, Ch4	Tx, Ch6	
Y3	Rx, Ch4	Rx, Ch3	Tx, Ch7	Rx, Ch4	Rx, Ch7	Tx, Ch4	
Y4	Tx, Ch5	Rx, Ch5	Tx, Ch3	Tx, Ch8	Rx, Ch5	Rx, Ch5	
...									
...									
...									
...									
Y(N-1)	Rx, ChM	Rx, Ch7	Rx, Ch2	Rx, Ch1	Tx, ChM	Tx, Ch3	
Time	1	2	3	4	5	6	...	S	

Figure 5.5. Node state information for the multi-channel, asynchronous case.

Unlike in Figure 5.1, the difficulty in the rendezvous results not only from the random nature of the transmit/receive states and the operating frequency, but also the alignment in time. In the simple scenario shown above, rendezvous can also only occur if Node Y_i 's time slot is correctly aligned with Node X. Otherwise, part of the transmitting node's beacon will be truncated when it arrives at Node X. For example, if Node Y_i leads Node X in time, the beacon packet's headers may not be received, and if it lags, the payload and CRC error-check information will not be received. Unless the design of a real system builds in some excess width, and sizes the slots a little larger than the actual beacon packet length, perfect alignment is difficult and improbable to achieve.

To solve this problem, designers often configure the system to transmit or receive multiple-slots on the same frequency in a row. For example, each slot is doubled or tripled back-to-back. This would look like the scenario in Figure 5.6. For example, if Node X had two receive slots in a row on a channel, it could start receiving a beacon message transmitted by Node Y_i in the first slot, and successfully finish receiving the beacon in a second slot. This loosens the requirement that slots have to be aligned for rendezvous to occur and improves discovery access.

Node									
X	Tx, Ch1	Tx, Ch1	Rx, Ch2	Rx, Ch2	Tx, Ch3	Tx, Ch3	
Y1	Rx, Ch6	Rx, Ch6	Tx, Ch8	Tx, Ch8	Rx, Ch4	Rx, Ch4	
Y2	Rx, Ch9	Rx, Ch9	Rx, Ch7	Rx, Ch7	Rx, Ch4	Rx, Ch4	
Y3	Tx, Ch5	Tx, Ch5	Tx, Ch8	Tx, Ch8	Rx, Ch5	Rx, Ch5	
Y4	Tx, ChM	Tx, ChM	Rx, Ch1	Rx, Ch1	Rx, Ch9	Rx, Ch9	
...									
...									
...									
...									
Y(N-1)	Rx, ChM	Rx, ChM	Tx, Ch2	Tx, Ch2	Tx, Ch3	Tx, Ch3	
Time	1	2	3	4	5	6	...	S	

Figure 5.6. Node state information for the multi-channel, asynchronous case with $W=2$ slot repetitions. [M. Silvius' Image from [69]. Used with permission, © 2009 IEEE.]

Borbash [64] uses this strategy when presenting a solution to the single-channel, asynchronous problem. Here, he divides each main rendezvous slot into W *minislots*. Each of the transmissions or receptions occurs W times in a row. In this case, the total number of slots, S , now includes all minislots, and is given by,

$$S = \left\lceil \frac{t}{WT_m} \right\rceil \approx \frac{t}{WT_m} \quad (26)$$

Where t is the total duration of the rendezvous protocol's execution, and T_m is the duration of a *minislot*. In this case, Borbash [64] asserts that expression (22) still holds, given that (26) is substituted use for S , or

$$F(t) = 1 - e^{-\left[\frac{t}{T_m} \cdot \frac{E(h)}{W} \cdot \frac{1}{N-1} \right]} \quad (27)$$

The challenge now is to compute $E(h)$ appropriately for the asynchronous slot alignment. Borbash does a very thorough job of this in [64] for the single-channel case. Our goal now is to determine $E(h)$ for the multi-channel, asynchronous case. We opt not to derive a new mathematical expression due to the complexity. Instead, in the remainder of this chapter, we measure $E(h)$ through the use of laboratory experiments

At this point, it's worthwhile to discuss how other asynchronous systems handle this scenario. One of the best known multi-channel, asynchronous systems now in use is Bluetooth. Bluetooth also uses slot repetitions to increase the probability of successful rendezvous. However, unlike in Borbash's model, where all nodes use W repetitions for both transmit and

receive states, Bluetooth uses only repetitions for the receive state. In this case, Bluetooth nodes in the receive mode, repeat their receive slot on a specific channel $W_R=M$ times before hopping to the next channel. M is the number of channels used for rendezvous. On the other hand, Bluetooth nodes in the transmit state, only repeat their transmit slot $W_T=I$ time, before hopping to the next channel. This maximizes the probability that the Bluetooth node in the receive state successfully intercepts a beacon message from another transmitting node, since transmitting nodes are guaranteed to *sweep* through the receiving nodes operating channel at least once per cycle [54-56].

Note that in Bluetooth, the roles of the rendezvous process are divided asymmetrically among Slaves and Masters. Slaves generally remain in the receive state, except when transmitting an acknowledgment message back to a Master. Masters generally remain in the transmit state, except when pausing to listening for an acknowledgement message from a Slave after each beacon transmission. As mentioned earlier, authors such as Salonidis [57, 58] have suggested modification strategies to overcome these limitations and allow nodes to operate in a symmetric fashion.

5.5 Experiment

In this section, we determined the number of expected discoveries, $E(h)$, for the multi-channel, asynchronous case through experiments on a four node laboratory testbed. This allowed us to generate tables with empirically based values for $E(h)$ for permutations of the design parameters M , W , and p_T , with $N=4$. These experiments also allowed us to determine which values maximized the performance for a given configuration. We used $N=4$ here, because this was the size of the CWT Smart Radio testbed described in Chapter 4.

5.5.1 Setup

All experiments were conducted using the four node cognitive radio network testbed using asynchronous timing, shown in Figure 5.7. Nodes were setup on a laboratory bench in linear fashion with separations between 5-10 feet. Each node consists of a Dell desktop or laptop computer running Ubuntu 7.10 Linux, in combination with GNU Radio 3.1 [8] and an Universal Software Radio Peripheral (USRP) [9]. Table 5.3 summarizes the PHY layer configuration

variables used in conjunction with the MAC rendezvous parameters. We used USRP daughterboards designed for the 400 MHz frequency band, and we assigned 100 KHz wide channels. We also used GMSK modulation with a bitrate of 100 kbit/s. Experiment configuration and control was accomplished via a backbone Ethernet network, connected to each test node. The experiments focused on independent neighbor information collection with node X collecting statistics on the arrival and decoding of rendezvous beacons from nodes Y_1 , Y_2 , and Y_3 .

Table 5.3. Experiment Configuration Settings
 [M. Silvius' Table from [69]. Used with permission, © 2009 IEEE.]

Variable	Value
Number of Nodes, N	4
Run Time, t_x	25s
Band	400 Mhz
Channels	100 KHz
Modulation	GMSK
Bit Rate	100 kbits/s
Beacon Message Size	1024 bytes

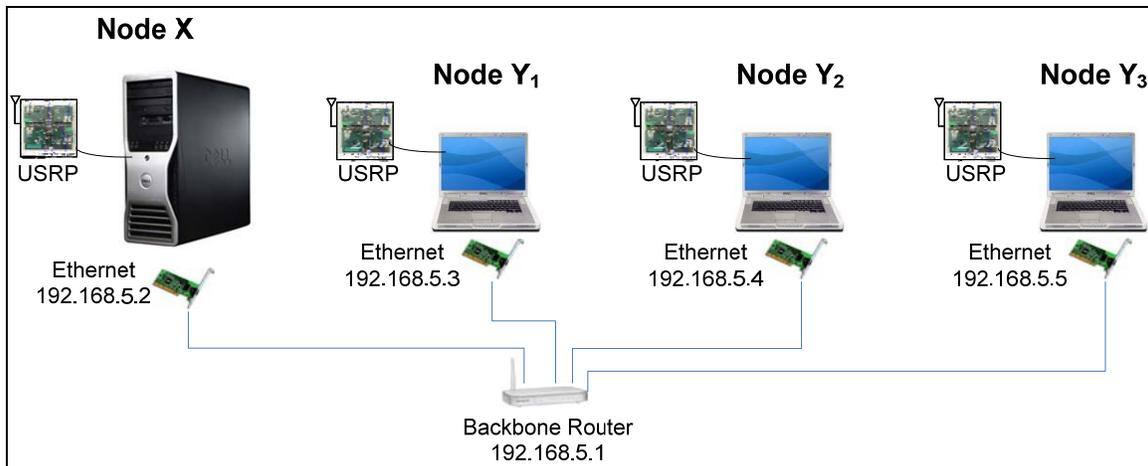


Figure 5.7. Detailed architecture of the network testbed.
 [M. Silvius' Image adapted from [69]. Used with permission, © 2009 IEEE.]

5.5.2 Procedure

We began by starting up node X and nodes Y₁, Y₂, and Y₃ using the Ethernet control network. We then executed a test script which iterated over the range of design parameters, M , W , and p_T . Each experiment lasted $t=25s$, and started every half-minute. For each combination of parameters, we averaged the results of 10 experiments. We first iterated over the transmit probabilities $p_T=[0.1, 0.2, 0.3, 0.4, 0.5]$, then the number of channels $M=[1, 2, 5, 10, 20, 35, 50]$, and finally, the slot repetition factor $W=[1, 2, 5, 10, 15, 20]$, for a total of 2100 experiments. We counted the number of beacon messages that node X successfully received from nodes Y₁, Y₂, and Y₃ and divided this by S to approximate $E(h)$. Remember that the number of slots is given by $S \approx (t/WT_M)$, and S decreases as repetition factor increases. The optimum transmit probability, p_T , which produced the largest $E(h)/W$ was recorded in Table 5.4, with the corresponding values of $E(h)/W$ and $E(h)$ recorded in Table 5.5 and Table 5.6 respectively. Note that in case where $W > 1$, the expected number of discoveries per slot, $E(h)$, could be greater than one. By dividing the probability of detection by $E(h)$ by the slot repetition factor, W , and selecting the largest quotient, we determined the optimum W for a given value of M . This value is highlighted in Table 5.5. The quantity, $E(h)/W$, can be thought of as the expected number of times node X finds any other node Y in a given *minislot*, which we would also like to maximize through the optimum selection of W . The corresponding values of p_T and $E(h)$ for the optimum value of W were then highlighted in Tables 5.4 and Table 5.6. Last, we recorded the first time node X finds another node Y_i, and displayed the results for the run with these optimum parameter settings in Table 5.7.

Table 5.4. Optimum transmit probability, p_T , for a given value of W and M . [M. Silvius' Table from [69]. Used with permission, © 2009 IEEE.]

pT_optimum in {0.1, 0.2, 0.3, 0.4, 0.5} for N=4							
		W=1	W=2	W=5	W=10	W=15	W=20
M=	1	0.1	0.2	0.2	0.2	0.2	0.2
	2	0.0	0.1	0.3	0.2	0.2	0.2
	5	0.0	0.1	0.3	0.3	0.3	0.3
	10	0.0	0.3	0.3	0.3	0.4	0.3
	20	0.0	0.1	0.3	0.3	0.4	0.3
	35	0.0	0.2	0.3	0.2	0.4	0.3
	50	0.0	0.3	0.4	0.1	0.4	0.2

Table 5.5. Expected number of discovered per minislot, $E(h)/W$, for a given combination of p_T , W , and M . Largest value of $E(h)/W$ for a given channel, M , (i.e. largest value in a given row) is highlighted here, and used to identify the corresponding tables entries containing the optimum parameters in Table 5.4 and resulting expected number of discoveries, $E(h)$, in Table 5.5.

Eh/W(p_T_optimum,M,$N=4$)							
		W=1	W=2	W=5	W=10	W=15	W=20
M=	1	0.0003	0.0549	0.1409	0.1815	0.2329	0.2803
	2	0.0000	0.0199	0.0544	0.1098	0.1609	0.2148
	5	0.0000	0.0085	0.0358	0.0500	0.0804	0.1007
	10	0.0000	0.0045	0.0237	0.0403	0.0635	0.0502
	20	0.0000	0.0037	0.0119	0.0227	0.0284	0.0370
	35	0.0000	0.0030	0.0059	0.0108	0.0154	0.0249
	50	0.0000	0.0015	0.0056	0.0083	0.0144	0.0114

Table 5.6. Expected number of discovered per slot, $E(h)$, for a given combination of p_T , W , and M . [M. Silvius' Table from [69]. Used with permission, © 2009 IEEE.]

Eh(p_T_optimum,M,$N=4$)							
		W=1	W=2	W=5	W=10	W=15	W=20
M=	1	0.0003	0.1097	0.7047	1.8145	3.4933	5.6053
	2	0.0000	0.0398	0.2718	1.0977	2.4140	4.2951
	5	0.0000	0.0171	0.1788	0.4997	1.2067	2.0135
	10	0.0000	0.0090	0.1183	0.4029	0.9518	1.0033
	20	0.0000	0.0074	0.0596	0.2275	0.4266	0.7393
	35	0.0000	0.0060	0.0296	0.1085	0.2306	0.4985
	50	0.0000	0.0031	0.0281	0.0828	0.2161	0.2286

Table 5.7. Average time to discover various fractions of available links per the number of channels, using the identified optimum p_T and W values from Table 5.4. A value of 25s indicated that the link was not found during the experiments. [M. Silvius' Table from [69]. Used with permission, © 2009 IEEE.]

Ave. Discovery Time in Seconds, $N=4$, $t=25s$				
		1/3 Links	2/3 Links	3/3 Links
M=	1	2.3445	7.3102	12.4989
	2	2.1974	5.6615	13.8959
	5	3.0547	12.9175	21.2930
	10	4.2382	10.1417	21.8081
	20	11.6389	18.9586	25.0000
	35	11.7182	22.9695	25.0000
	50	18.3236	21.2095	25.0000

In a subsequent experiment, we conduct additional investigation of the asynchronous case, in order to compare to the theoretical results presented by Borbash in [64]. The procedure is identical as before, except that we fix the number of channels to $M=1$, and we varied the number of nodes, N , operating in simultaneously in the testbed of Figure 5.7 over $N=[2, 3, 4]$.

We also use smaller step sizes for the transmit probability for greater resolution, $p_T=[0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50]$, and we varied the number of slot repetitions over $W=[1, 2, 3, 4, 5]$. Like before, we list the optimum transmit probability, p_T , for a given combination of N and W in Table 5.8. We compute and list the corresponding values of $E(h)/W$ and $E(h)$ in Table 5.9 and Table 5.10 respectively. The largest $E(h)/W$ value is highlighted in Table 5.9, and this table cell location is used to identify the optimum transmit probability, p_T , and slot repetition fraction, W , in Table 5.8. The corresponding $E(h)$ value for these optimum parameters is listed in Table 5.10. The results of this experiment are compared against Borbash's theoretical results in Table 5.11.

Table 5.8. Optimum transmit probability, p_T , for a given value of W and N single channel, $M=1$.

p_T_optimum in {0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5} for $M=1, N$						
		W=1	W=2	W=3	W=4	W=5
N=	2	0.00	0.25	0.35	0.35	0.25
	3	0.15	0.25	0.15	0.15	0.25
	4	0.15	0.15	0.20	0.20	0.15

Table 5.9. Expected number of discovered per minislot, $E(h)/W$, for a given combination of p_T , W , and N for single channel, $M=1$.

$Eh/W(p_T$_optimum,$M=1,N$)						
		W=1	W=2	W=3	W=4	W=5
N=	2	0.0000	0.0388	0.0625	0.0781	0.1061
	3	0.0004	0.0408	0.0885	0.1017	0.1315
	4	0.0004	0.0490	0.0724	0.1092	0.1328

Table 5.10. Expected number of discovered per slot, $E(h)$, for a given combination of p_T , W , and N for single channel, $M=1$.

$Eh(p_T$_optimum,$M=1,N$)						
		W=1	W=2	W=3	W=4	W=5
N=	2	0.0000	0.0776	0.1874	0.3124	0.5307
	3	0.0004	0.0815	0.2655	0.4067	0.6577
	4	0.0004	0.0981	0.2173	0.4367	0.6642

Table 5.11. Comparison of my results (experimental) to that of Borbash (theoretical).

Silvius (experimental)				Borbash (theoretical)		
N	W	p_T_optimum	E(h)	W	p_T_optimum	E(h)
2	5	0.25	0.5307	2	0.423	0.385
3	5	0.25	0.6577	2	0.261	0.434
4	5	0.15	0.6642	2	0.189	0.454

5.5.3 Results

The results of the experiments were presented in Tables 5.4-5.6. A few general trends can be seen on inspection. First, proceeding vertically downward on Table 5.6, we can see that the expected number of discoveries per slot, $E(h)$, decreased as the number of channels in use, M , increased, as predicted. However, the transmit probability, p_T , required for optimum performance tended to increase as M increased, though not in all cases. It is informative to note that rendezvous did not occur for the multi-channel cases with no slot repetitions ($W=1$, $M>1$), for any value of transmit probability, p_T , for the reasons discussed in Section 5.4.3. Second, proceeding horizontally across Table 5.6, we can see that increasing the repetition factor, W , in every instance, increased the number of expected discoveries. This occurred even though doing so decreased the total number of slots, S , used during the rendezvous process. Third, Table 5.7 shows that the time necessary to establish a given fraction of links during each run generally increased as the number of channels increased. The results of Tables 5.4-5.6 can be plotted. Figures 5.8-5.21 show these results for $M=[1, 2, 5, 10, 20, 35, 50]$. The maximum peak value for the set of traces in each figure correspond to the highlighted optimum values identified in Tables 5.4-5.6.

The results of the subsequent single-channel experiments were presented in Tables 5.8-5.11. Graphical plots of this data are also shown in Figures 5.22-5.23. My determined values of $E(h)$ tended to be larger than those predicted by Borbash. I believe that this is due to practical limitations of the underlying GNU Radio and USRP SDR architecture used in the smart radio nodes of the testbed in Figure 5.7. Higher $E(h)$ may be due to the systems correctly receiving beacon messages, even during what should be considered *collision* conditions. One possibility is that there may have been sufficient differences in USRP powers in transmitting nodes for the receiver to discriminate one of them over the others. Or, another possibility is that due to clock time errors, the slot widths for transmitting and receiving of beacons may not have been equal. For example, the *receive slots* might have actually been longer than *transmit slots*, allowing for more beacons to be received in each *receive slot* than theoretically predicted.

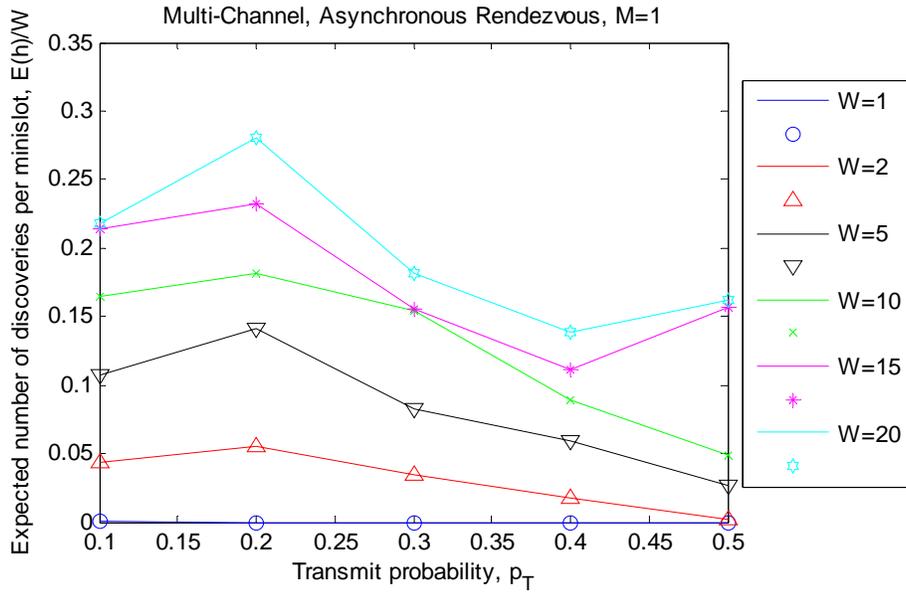


Figure 5.8. Multi-Channel, Asynchronous, Experimental Results, M=1.

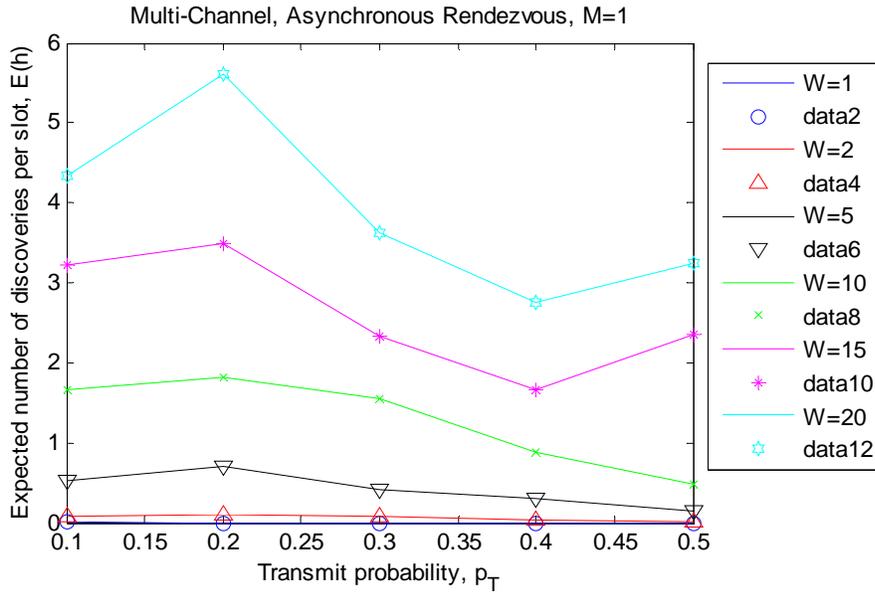


Figure 5.9. Multi-Channel, Asynchronous, Experimental Results, M=1.

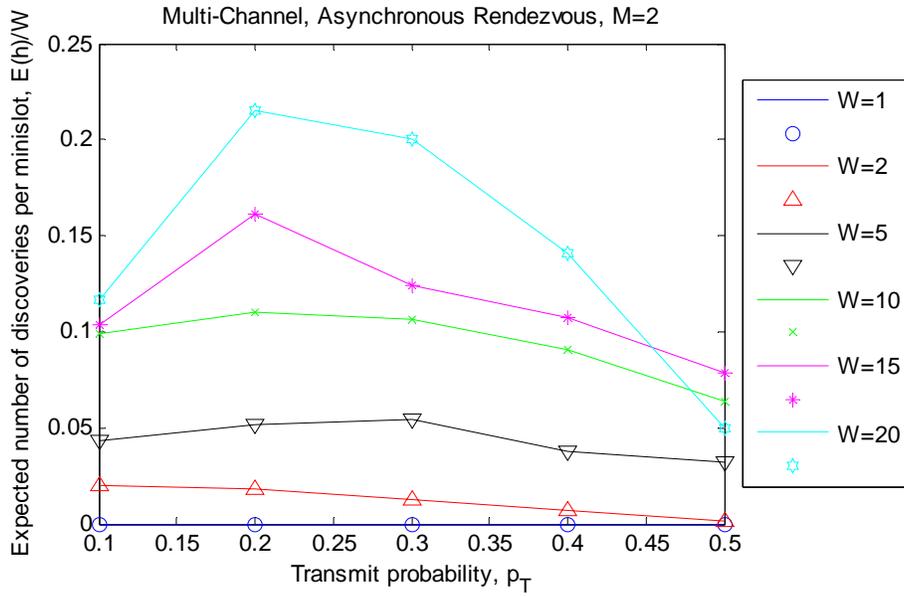


Figure 5.10. Multi-Channel, Asynchronous, Experimental Results, $M=2$

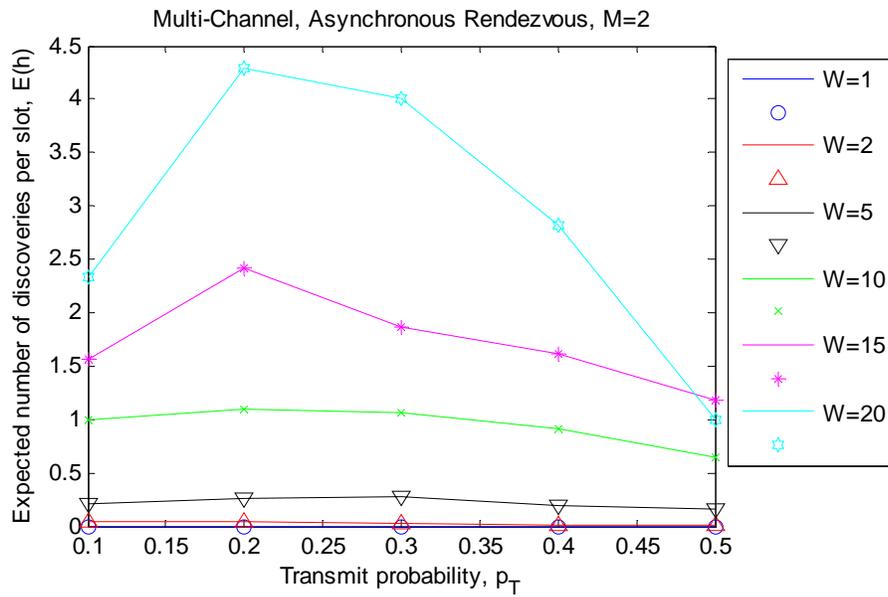


Figure 5.11. Multi-Channel, Asynchronous, Experimental Results, $M=2$

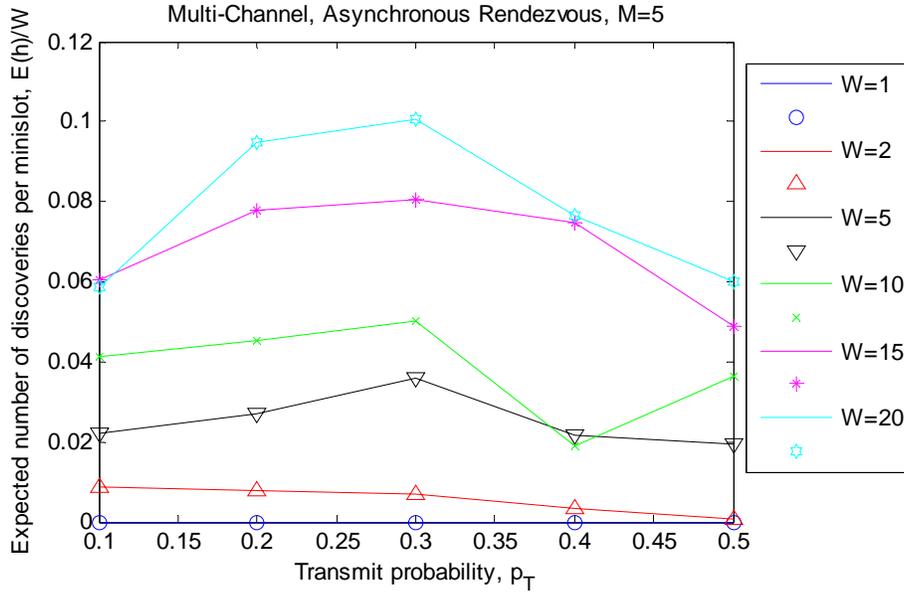


Figure 5.12. Multi-Channel, Asynchronous, Experimental Results, M=5.

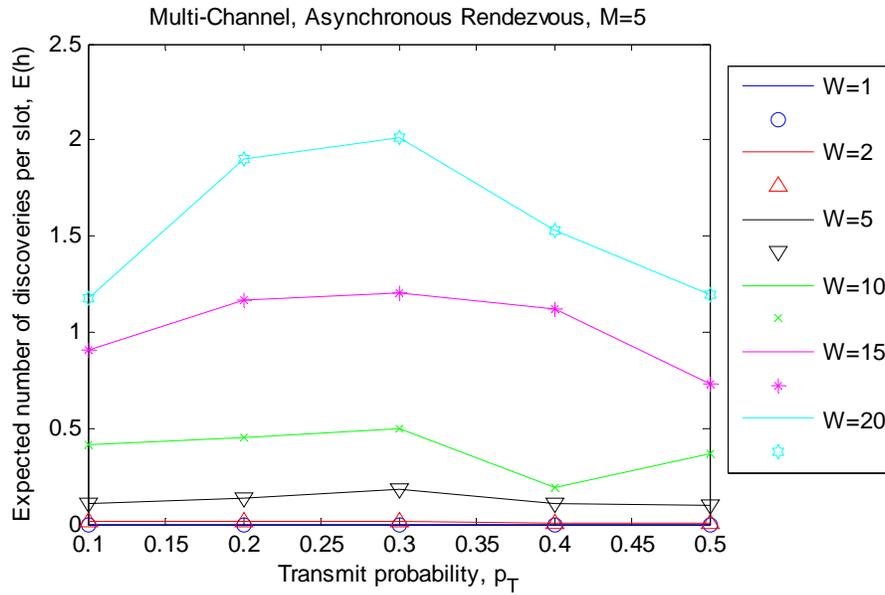


Figure 5.13. Multi-Channel, Asynchronous, Experimental Results, M=5.

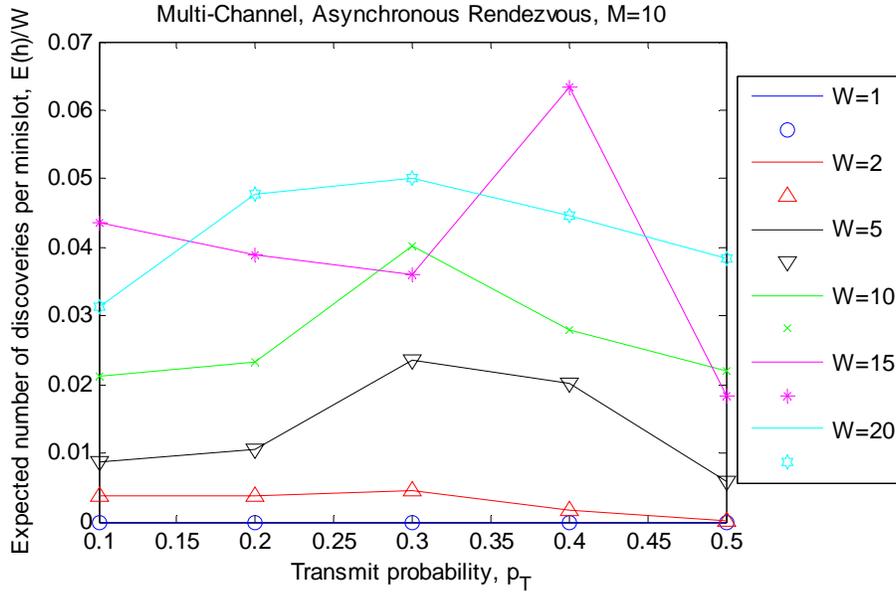


Figure 5.14. Multi-Channel, Asynchronous, Experimental Results, $M=10$.

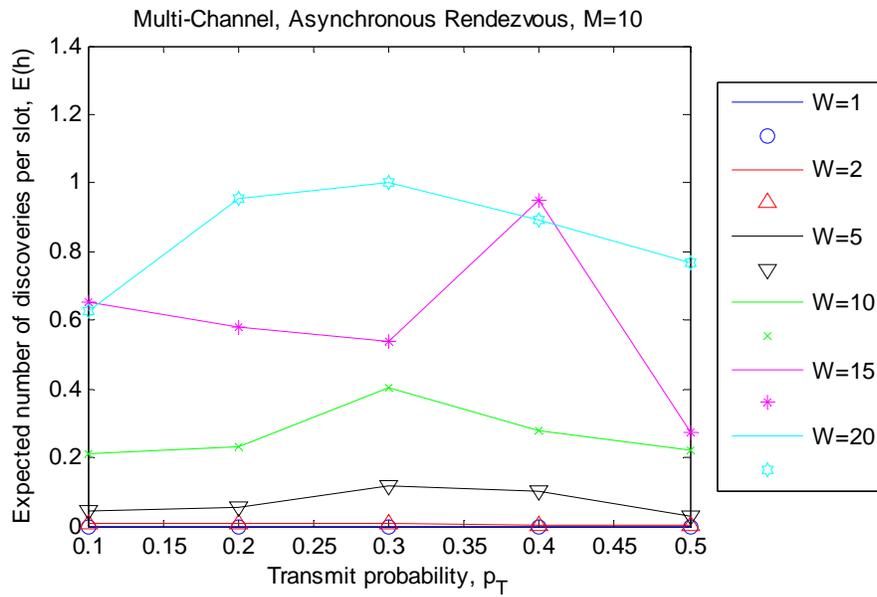


Figure 5.15. Multi-Channel, Asynchronous, Experimental Results, $M=10$.

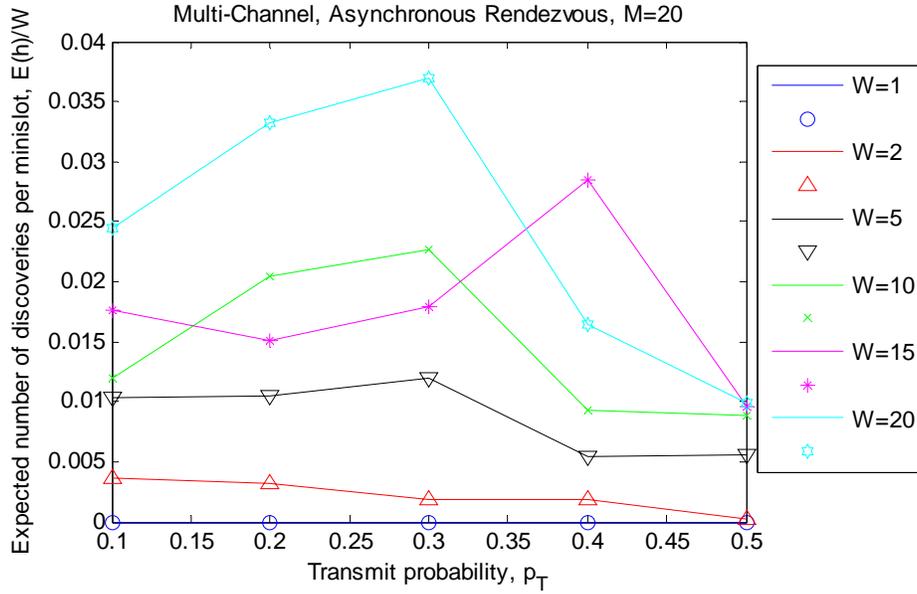


Figure 5.16. Multi-Channel, Asynchronous, Experimental Results, $M=20$.

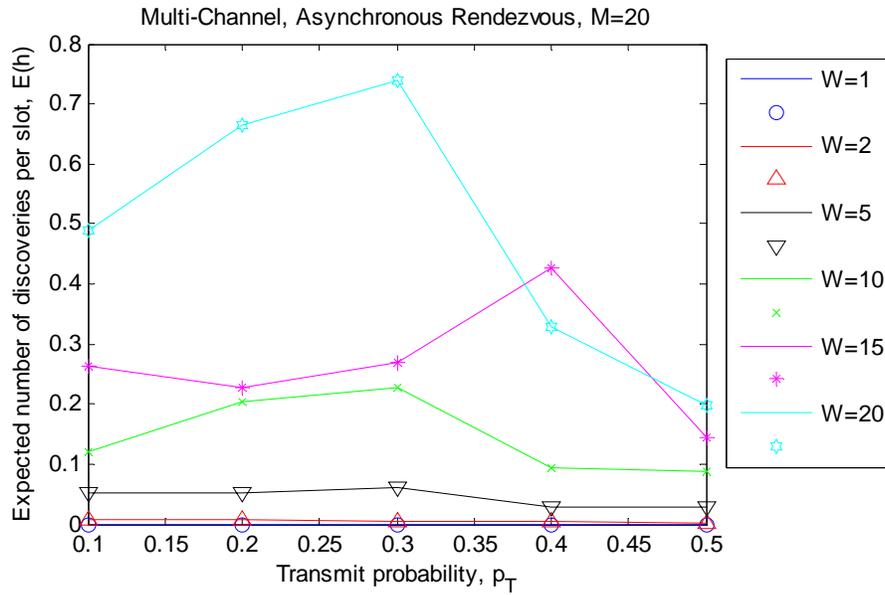


Figure 5.17. Multi-Channel, Asynchronous, Experimental Results, $M=20$.

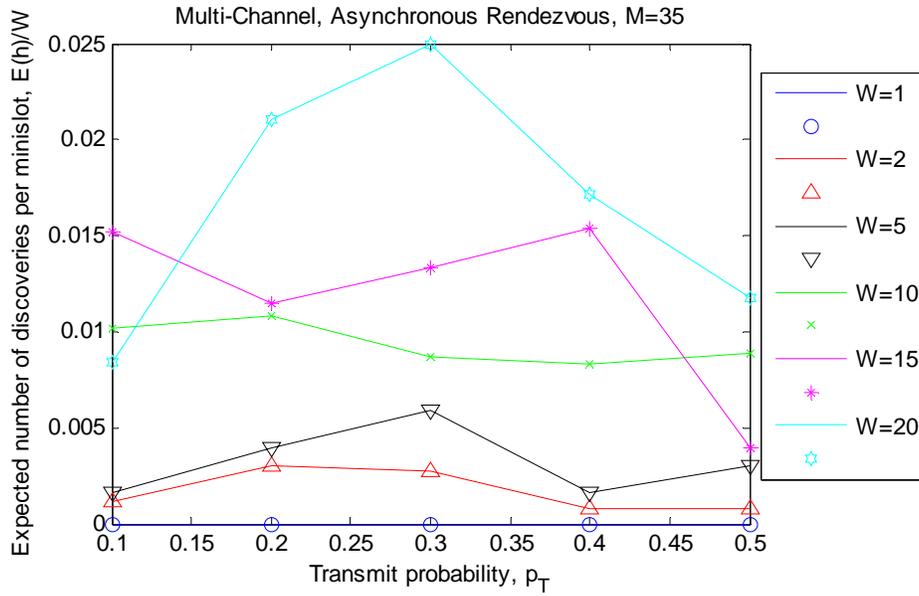


Figure 5.18. Multi-Channel, Asynchronous, Experimental Results, M=35.

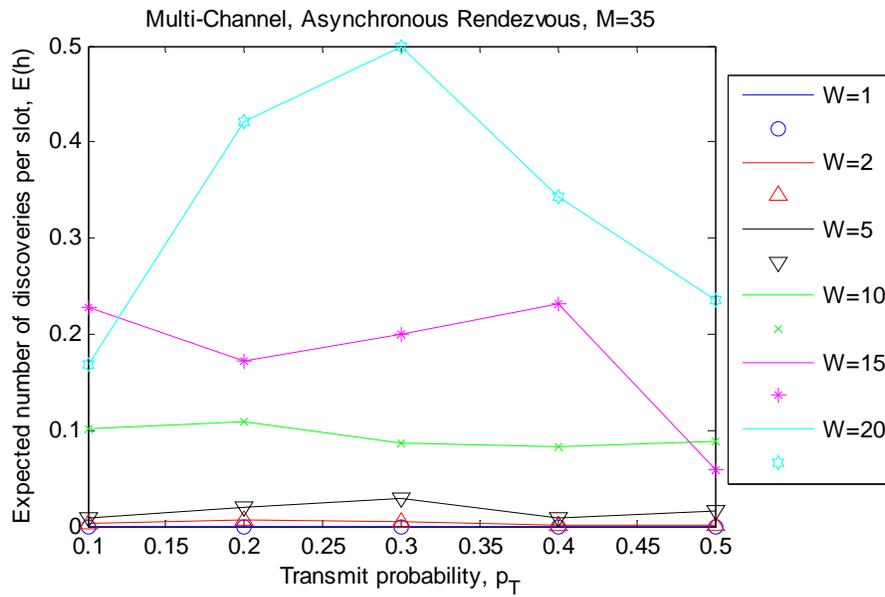


Figure 5.19. Multi-Channel, Asynchronous, Experimental Results, M=35.

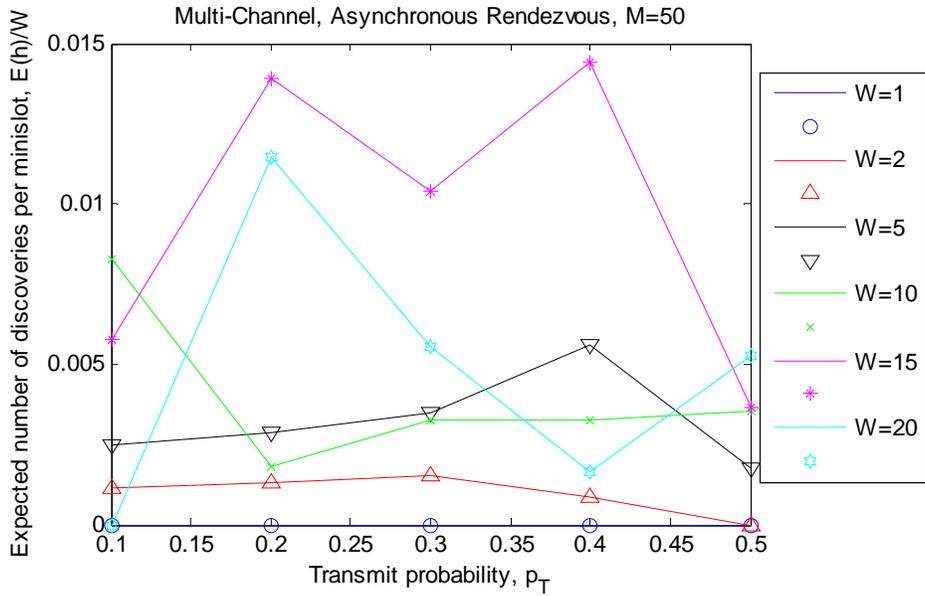


Figure 5.20. Multi-Channel, Asynchronous, Experimental Results, M=50.

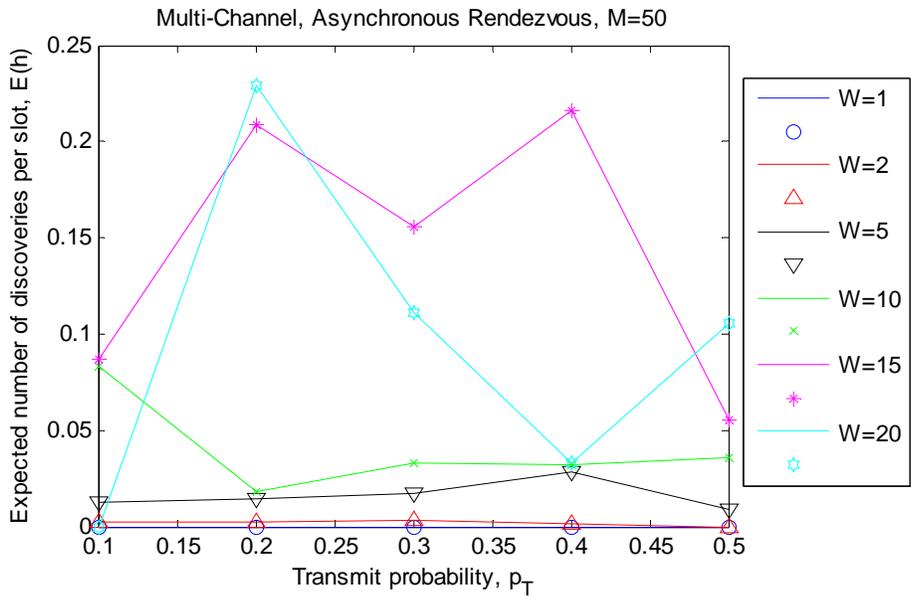


Figure 5.21. Multi-Channel, Asynchronous, Experimental Results, M=50.

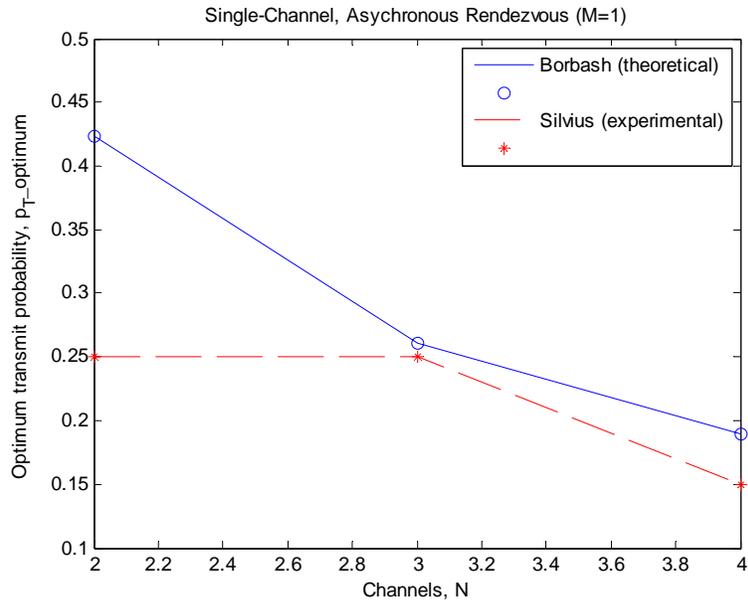


Figure 5.22 Comparison of my optimum transmit probability results, Silvius (experimental) to that of Borbash (theoretical).

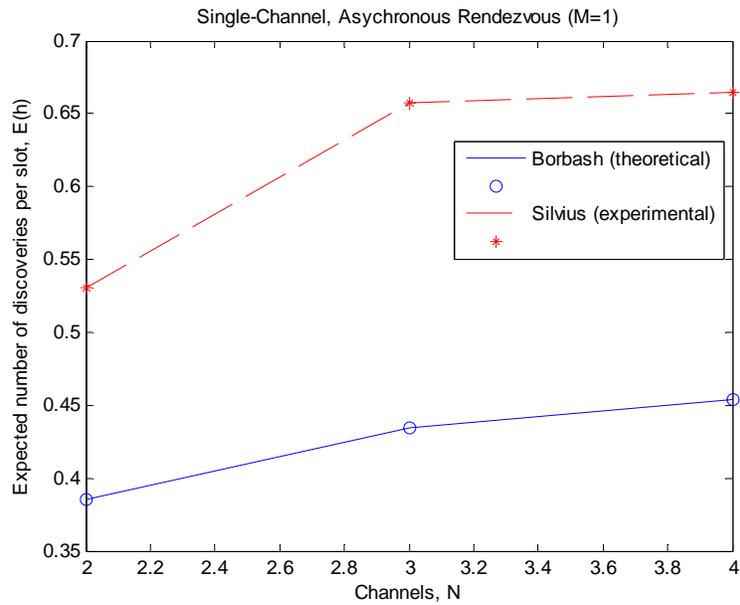


Figure 5.23 Comparison of my expected number of discoveries results, Silvius (experimental) to that of Borbash (theoretical).

5.6 Conclusions

The use of an experimental testbed enabled us to estimate the optimum operating parameters for the multi-channel, asynchronous rendezvous problem for a four node neighborhood. The results demonstrated the feasibility of building and testing a rendezvous protocol in a prototype system and offered a starting point for optimum parameter selection for future simulation and experimentation.

Chapter 6

Channel Change Protocol

This chapter describes the development of the Channel Change protocol (CCP), a dynamic spectrum access protocol developed for the CWT Smart Radio communication platform. The protocol enables the smart radio to detect the return of legacy analog family radio service (FRS) and smart radio primary users, and to orchestrate the *change-channel* procedures to an alternate, vacant block of spectrum. In this way, the CCP allows primary users and secondary users to co-exist in the same frequency band without interference as shown in Figure 6.1. CWT originally designed the CCP specifically to meet the requirements of the public safety earthquake response scenario for SDR Forum’s Smart Radio Challenge 2007, as described in Chapter 2; however, the CCP was also incorporated into the CWT Smart Radio 2008 architecture, as detailed in Chapter 3. This chapter uses OMNeT++ simulations, as well as experiments on the Smart Radio Network Testbed from Chapter 4 to evaluate the protocol’s performance.



Figure 6.1. Testing two secondary users and a single primary user node.
[M. Silvius’ Image from [6]. Used with permission, © 2008 SPIE.]

The remainder of the chapter is organized as follows. Section 6.1 reviews the problem scenario that motivated the development of the CCP. Section 6.2 summarizes the contributions of this chapter. Section 6.3 reviews previous MAC designs in the literature and explains how CWT incorporated aspects of these features into the design of the CCP. Section 6.4 explains, in depth, the design and conceptual operation of the CCP. Section 6.5 explains the used to evaluate

the CCP. Section 6.6 and Section 6.7 detail the setup and the results of OMeT++ simulations to evaluate the performance of the CCP. Section 6.8 and Section 6.9 summarize the setup and results of laboratory experiments to also evaluate its performance. Section 6.11 compares the results of the simulations and the experiments, and Section 6.11 discusses an additional experiment to evaluate the stability of the experimental results.

6.1 Problem Statement

White space studies highlight the underutilization of the frequency spectrum and emphasize the utility of dynamic spectrum sharing communication systems [70]. Here, secondary users operating in this scenario would need to detect the presence of an actively transmitting analog or digital radio, and immediately vacate the channel, resuming operation on a different unoccupied channel. An existing, fixed wireless data communications system, such as 802.11 WiFi, would be a poor choice for a secondary user system. Currently, this system does not fulfill the necessary sensing and adaptability requirements. This popular wireless LAN technology has little provision to continually sense the channel for primary users, or to adapt and reconfigure itself to use to an alternate channel when the primary user appears. As a result, its data communications would be corrupted from the interference from the primary user, in same way it would corrupt the voice/data communications of other primary users with its interference.

In an experiment, we demonstrated how 802.11 WiFi is degraded by other wireless communications waveforms. In this mock scenario, WiFi had a noticeable decrease in QoS, even with a single burst of data from a primary user transmitting a GMSK waveform, as shown in Figure 6.2. Here, the goodput of the WiFi link drops to zero during the duration of the primary user's transmission. The large spike at $t=40$ represents the rapid transmission of packets that had been buffered during the period that the goodput was zero. Likewise, the primary user's QoS would be degraded by the presence of the WiFi user's signal. To address this issue, we present a straightforward and flexible primary user avoidance technique—the Smart Radio Channel Change Protocol (CCP). This protocol could be retrofitted in an existing wireless LAN technology like 802.11 WiFi, or be readily implemented in a software defined radio (SDR) based cognitive radio. This radio access protocol allows a secondary user to avoid the interference generated by competing legacy analog and digital primary user radios in the environment, so as to improve overall user QoS. We apply our initial prototype to the problem of co-existence

between digital cognitive radios and analog and digital radios in a wireless communication network. This chapter builds on CWT's recent efforts in cognitive radio and dynamic spectrum access systems [5, 6, 24, 71, 72].

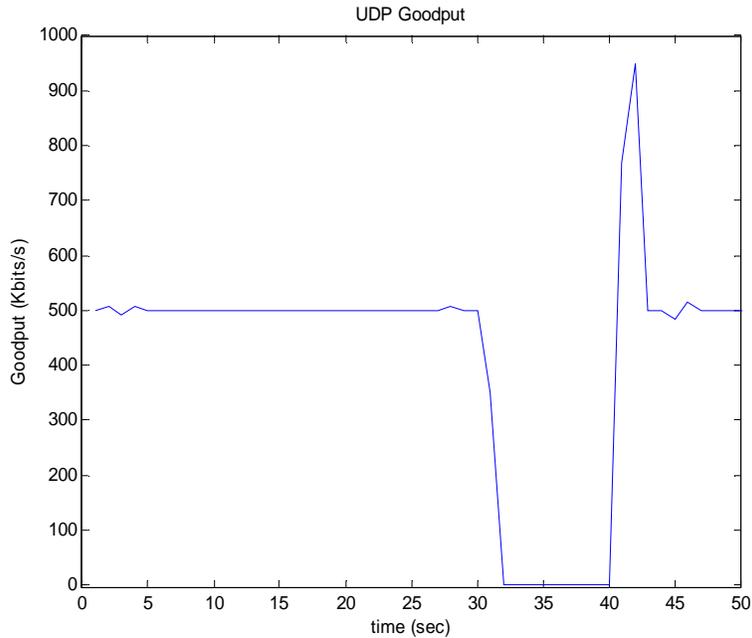


Figure 6.2. A 802.11 WiFi secondary user's UDP goodput is degraded during the transmission of a primary user with a GMSK waveform, between $t=30$ s to $t=40$ s. The large spike at $t=40$ represents the rapid transmission of packets that had been buffered during the period that the goodput was zero. [M. Silvius' Image from [72]. Used with permission, © 2009 IEEE.]

6.2 Contributions

The primary contribution in this chapter is the Channel Change protocol. The CCP is a spectrum sharing, dynamic spectrum access protocol which enables a secondary smart radio to share spectrum with other primary users in the band. It also allows a network of secondary users to co-exist in the same band as a network of primary users, reducing the interference between the users and improving the resulting network QoS. One of the significant advantages of the CCP is that it is a straightforward and flexible primary avoidance technique, which could be retrofitted in an wireless LAN technology like 802.11 WiFi, or be readily implemented in a SDR-based cognitive radio. In this chapter, we also demonstrate the performance our protocol by implementing the CCP in the CWT Smart Radio, built on the GNU Radio and USRP foundation.

This chapter also contributes by presenting QoS measurements on the effectiveness of the CCP using both OMNeT++ simulations and laboratory experiments in the CWT Smart Radio Network Testbed. Both sets of results complement each other and show that the use of the CCP in mixed primary and secondary user scenarios leads to an increased goodput and reduced jitter, packet loss, and latency over scenarios not using the CCP.

6.3 Previous Work

The CCP-equipped smart radio parallels related developments in multi-channel wireless network systems with its ability to rapidly change its operating frequency in response to dynamic interference conditions. However, the CCP provides a significant advantage over these other systems' frequency management schemes in terms of design simplicity and implementation. The CCP could be paired with any smart radio system's MAC layer, given that it is based on the carrier-sense multiple access (CSMA) approach. The CCP provides a quick and inexpensive means of incorporating spectrum sensing dynamic spectrum access capabilities, requiring few modifications to the existing subsystems.

In its operation, the CCP, in combination with the underlying MAC protocol, shares many commonalities with legacy MAC layer designs, as well as recent dynamic spectrum extensions. As a starting point, this chapter reviews the operation of the 802.11 wireless and the hybrid GNU Radio MAC designs, and discuss the modifications required of these to implement CCP. It also reviews similar recent developments in multi-channel MAC schemes from literature. These multi-channel schemes use links of multiple channels in order to maximize network capacity as well as to dynamically avoid interference from other non-cooperative nodes. They are also very similar to frequency-hopped spread-spectrum schemes, which include systems such as Bluetooth and associated Slotted Seeded Channel Hopping (SSCH) [73]. In addition, recent Bluetooth improvements allow for the dynamic adaptation of the pseudo-random hop sequences in response to changing interference conditions in a way that minimizes interferences and maximizes throughput.

6.3.1 GNU Radio

GNU Radio is a free software toolkit for building software radios [8]. It is designed to run on a general purpose processor, i.e. desktop and laptop computers. Combined with minimal

hardware, GNU Radio allows the construction of simple software radios. By introducing signal processing blocks, GNU Radio turns a typical hardware problem into a software problem. Each block functions in a single signal processing task, such as I/O operation, modulation, demodulation, filtering and so on, so that GNU Radio offers the ability to create radios that change on the fly.

GNU radio provides a library that contains many commonly used signal processing blocks. New blocks can be added as needed. A radio is built by connecting the blocks to form a flowgraph, which is a fundamental concept in GNU Radio. Figure 6.3 shows an example of flowgraph. The rectangles are GNU Radio blocks and the arrows show data streams.

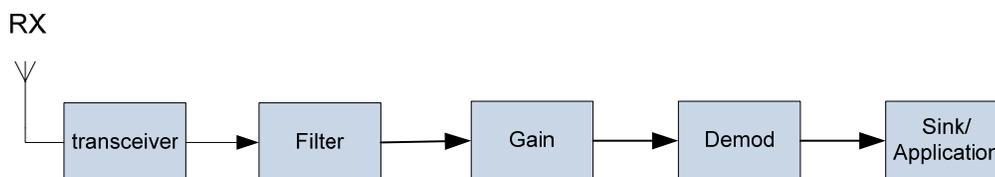


Figure 6.3. An example of a GNU Radio flowgraph.
[Y. Shi's Image. Used with permission of Y. Shi.]

The Universal Software Radio Peripheral (USRP) [9] is an hardware board which incorporates analog-to-digital/digital-to-analog converters (ADC/DACs) and a field-programmable gate array (FPGA) which performs the computationally expensive pre-processing of the input signal [74, 75]. The USRP is a very flexible USB device that is well suited as a RF end for GNU Radio.

6.3.2 GNU Radio's MAC

CWT used GNU Radio in its implementation of its CWT Smart Radio, and we will likewise use the GNU Radio MAC layer as a model for the simulations in this chapter, with the goal of better understanding its current capabilities. However, we recognize that the GNU Radio MAC layer suffers from a number of limitations. We will summarize these limitations in further detail in this section. CWT is actively working to improve the GNU Radio MAC layer for future releases of the smart radio. For now, we have coded our OMNeT++ simulations to accurately mirror the current smart radio implementation with the goal of understanding and improving the overall smart radio system's design.

GNU Radio gives much flexibility to the design, implementation, and reconfiguration of radios. However, due to some of inherent features, implementing a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) MAC protocol has several challenges. GNU Radio was originally designed to support signal processing on a continuous data stream, instead of processing discrete packets or frames. The fundamental concept of flowgraph in GNU Radio is based on this streaming-centric design mechanism. GNU Radio uses a scheduler, which is implemented as a single thread, to execute processing blocks sequentially in the flowgraph. However, a true CSMA/CA MAC requires more complex controls on the scheduling of actions. In GNU Radio, a radio is built by setting-up two independent paths, a transmit path and a receive path. CSMA/CA needs to coordinate between transmit and receive, and unfortunately, GNU Radio cannot build the two paths in one flowgraph. Another concern is the latency introduced by the GNU Radio/USRP platform. For a typical CSMA/CA MAC layer, the required turnaround time is usually on the degree of tens of microseconds, or at most, one hundred microseconds. According to the study of [76], the minimum receive latency through USRP hardware, USB port and GNU Radio processing is 600 microseconds and the minimum transmit latency is 200 microseconds for burst data, given that both use general purpose hardware and simple signal processing.

Future versions of GNU Radio may address these issues better. Nevertheless, how to design an appropriate MAC layer with GNU radio for packet-based network is an open problem. Note that the challenges of implementing an efficient CSMA/CA MAC protocol not only lie in GNU Radio, but also all other software defined radio platforms.

6.3.3 The IEEE 802.11 MAC

The IEEE 802.11 MAC protocol employs CSMA/CA based on an exponentially increasing random backoff strategy. Access to the wireless medium is controlled by coordination functions. It first checks to see that the medium is clear before transmitting. To avoid collisions, network hosts use a random backoff after each frame, with the first transmitter seizing the channel.

Data communication can be preceded by an exchange of control packets, such as request-to-send (RTS) or a clear-to-send (CTS). When a source S wants to transmit to a destination D , it first senses the local channel with physical carrier sensing. If the channel is busy, it backs-off

using an exponentially increasing backoff window algorithm. Otherwise, if the channel is idle for more than a Distributed Coordination Function, Inter-Frame Space (DIFS), the source transmits an RTS control message to the destination. If the local channel around D is free, D replies with a CTS message. After this exchange, data packets can be transmitted from S to D , and an ACK packet transmission from D to S follows. If the channel around D is busy, S times-out waiting for the CTS message and it retransmits the RTS packet.

Both RTS and CTS packets contain the duration of upcoming transmissions. Nodes located in the vicinity of communicating pairs overhear one or both of these control messages and must defer transmission for this proposed duration. This is called virtual carrier sensing which is performed in addition to the physical carrier sensing mentioned earlier. It is implemented by means of the Network Allocation Vector (NAV). A node updates the value of its NAV whenever it hears an RTS or CTS packets. Thus the nodes lying within the transmission range of the transmitter or the receiver have multiple chances to update the channel condition.

The CSMA/CA MAC protocol uses a backoff interval to resolve channel contention. When a node needs to transmit data or control frames, it senses the local medium. If the medium is busy, the node will defer transmission until the medium is free for DIFS.

After DIFS, if the node's current backoff value is zero, the node will choose a random backoff time and then transmit packets. In mathematical terms,

$$\text{Backoff time} = \text{Random}() \times \text{slot time} \quad (28)$$

where $\text{Random}()$ is a random integer in the range $[0, cw]$ and cw represents the contention window which varies between cw_{min} and cw_{max} . Both depend on PHY layer parameters.

After the sender chooses a backoff time, it will decrement its backoff counter by one after every idle slot time. The sender transmits its packet once the counter reaches zero. If collision happens, the sender increases its cw exponentially, and chooses a new random number from the new window range and then tries to retransmit. The contention window size doubles after every collision until it reaches a maximum threshold, cw_{max} , and cw will remain at this value until it is reset. In the backoff stage, if a node senses the channel is busy, it freezes its backoff counter. When the channel becomes idle for DIFS, the counter resumes counting down from its frozen value. The contention window size is always one less than a power of two.

6.3.4 Multi-Channel and Frequency-Hopped MACs

The CCP-equipped smart radio parallels related developments in multi-channel wireless network systems with its ability to rapidly change its operating frequency in response to dynamic interference conditions. The system's design involves the combination of the best aspects of other multi-channel MAC systems.

The overall goal of multi-channel MAC designs is to use multiple-channel links to improve overall network capacity. Multiple channels may be achieved through the use of multiple dedicated radio interfaces operating in parallel, or through a single radio interface, approximating the operation of multiple interfaces through a pre-determined switching schedule. The trade-off in each design is the improved capacity at the expense of the level of complexity required to manage such a network. More specifically, improved capacity may be interpreted as improved goodput and as reduced latency and lower packet loss.

One approach to grouping the various MAC layer designs, is to classify each by the strategy it uses to establish its control channel(s) [77]. A control channel is the logical channel used by the nodes in the radio network to exchange configuration information or control frames. This configuration information could include, but is not limited to channel allocation, routing or topology information. In single channel MAC, all of the control frames must flow over the same link as the convention data frames. This can happen on a random or as-needed basis, or by a specific time schedule. For example, in a Time Division Multiple Access (TDMA) system, there is a very specific frame schedule where control frames, as well as data frames from each user, are sent in pre-coordinated time slots. In multi-channel MAC systems, the radio nodes now have much more flexibility on how to organize their *logical* control channels. They may be on a dedicated frequency-channel on one radio interface, or spread out over multiple frequency-channels over multiple radio interfaces. On any particular link, they may happen in a random or as-needed basis, or in a pre-coordinated scheduled time.

Control channels can also be *Single Rendezvous*, meaning there is only one logical channel being used at any one time to exchange configuration information; or *Parallel Rendezvous*, where there can be multiple logical control channels operating simultaneously at any given time [77]. In [77], Mo also defines four main categories of multi-channel MACs, to include *Dedicated Control Channel* like [78], *Common Hopping*, *Split Phase* like MMAC [79]

and Multi-channel Access Protocol (MAP) [80], and *Parallel Rendezvous* like SSCH [73].

Another type of multi-channel MAC systems is a traditional frequency hopped spread-spectrum system (FH-SS) like Bluetooth. [41, 54, 81]. Here, the system uses one radio interface and all nodes hop among all the available frequencies allotted to the system. A logical channel is determined by the specific hopping sequence used by one or more nodes. Bluetooth nodes in a direct master-slave relationship are in the same personal area network (PAN), and are all synchronized to the same pseudorandom hopping sequence.

6.3.5 Comparisons and Advantages of the CCP

This CCP-equipped smart radio's design incorporates many of the relevant features from previously introduced multi-channel MACs. First, unlike the frequency hopping systems discussed in [77], the CCP-equipped smart radio does not require strict global time synchronization. Primary user detection and channel-switching can occur on a localized basis in response to interference. This provides greater tolerance to timing delays. Despite the channel switching synchronization and delay, the requirements are not as severe as in a TDMA or frequency-hopped system. Second, it does not require a fixed common control channel. Control information is transmitted over the same link as data traffic. This alleviates the control channel bottleneck problem that is found in dedicated control channel systems. This chapter assumes that the initial combined control/data link waveform is either known or determined at the start of the network. For instance, a rendezvous protocol [6, 7] would enable smart radio nodes to find and identify all of their neighbors in the network. Third, unlike many multi-channel systems, a smart radio using the CCP can operate with a single radio interface, avoiding the complex overhead of multiple interfaces. This is possible due to CCP's primary focus on in-band monitoring for incumbent radios, with only occasional pauses to monitor for out-of-band incumbents. Moreover, the out-of-band spectrum scanning can occur on a centralized basis, on a dedicated spectrum scanning node, not located within the data node. This reduces the complexity and delay which can be an issue with the spectrum scanning routines needed for other dynamic spectrum access protocols and systems. Fourth, the protocol can be easily ported to any existing 802.11 hardware, and the CCP could be paired with any smart radio system's MAC layer, given that it is based on the CSMA approach. The CCP provides a quick and inexpensive means of incorporating spectrum sensing dynamic spectrum access capabilities,

requiring few modifications to the existing subsystems. Fifth, except for the channel switch initiated immediately after a primary user is detected, the majority of the time the CCP enabled radio remains on a single channel. Unlike in frequency-hopping systems, the CCP strategy minimizes the switching delay by changing to a new operating channel, which can be a significant limitation in such systems as SSCH. Also, since the fallback channels are pre-determined and pre-negotiated before the occurrence of the interference condition with the establishment and dissemination of a fallback dictionary, the CCP eliminates the delays associated with channel negotiation during switching. This eliminates the mass transmission of new time-critical waveform negotiation messages which can clog up a control or data channel. *A priori* fallback channel negotiation and dissemination can then occur more gradually and with less bandwidth. Sixth, the CCP focuses on the concept of interference mitigation to/from the primary user, and less on cooperation or coordination. This simplifies the problem by not requiring complicated network etiquette protocols as discussed in [82, 83]. When the primary user is detected, the CCP does not expend effort by trying to classify the signal or to negotiate a time-based sharing schedule of the channel. It just vacates the channel so as to preclude interference. This eliminates the processing overhead involved with scheduling or signal classification. Seventh, the CCP is well suited for being integrated into a smart radio system for several reasons. Firstly, the CCP is well suited for implementation on a smart radio's SDR platform. SDR platforms excel at rapidly changing channel settings, reconfigurability, and adaptability, and they can utilize more diverse waveforms than are provided by existing 802.11 adapters. Secondly, CWT's current work in spectrum sensing [5, 24] can be directly applied to the selection of the CCP's fallback dictionary. Thirdly, the CWT cognitive engine [71] can be used to optimize the selection of channel, i.e. waveform parameters. In other words, the CCP could be viewed as a *waveform* change protocol, where waveforms can be selected and designed in an optimal fashion to fill and operate in a band identified by the spectrum scanner.

6.4 Presented Solution

The CCP enables the CWT Smart Radio to detect the return of a primary user and to orchestrate the change-channel procedures to an alternate, vacant block of spectrum. In the smart radio, when a primary user is detected, the protocol will stop current communications, switch to a vacant channel, and then restart communications. The new channel can be selected in

a random fashion, or after consulting a dictionary of pre-arranged *fallback channels*. The CCP relies on sensors that have been embedded in the physical (PHY) and MAC layers of the smart radio's framework.

6.4.2 Conceptual Operation

The first task of the CCP is primary user detection. We exploit the frame length to distinguish between the primary and secondary waveforms. We use a timer, coupled to the PHY and MAC layers of the base system to measure the lengths of the incoming transmissions. The system starts the timer when the PHY and MAC layers detect the presence of the carrier of an incoming frame. If a complete digital frame is received and demodulated within the predicted signal duration, the CCP resets the timer. If the signal persists longer than the expected frame duration, the CCP warns the radio resource manager that a primary user has been detected. At this point, the resource manager—such as a master control module or cognitive engine—can pause the current communication session, and direct the PHY and MAC layers to change to a different operating frequency. See Figure 6.4 for a timing plot showing the CCP's conceptual operation.

Although the preceding discussion explained how the CCP incorporated into a digital radio could detect and avoid an analog radio, the same principles could be used to distinguish between two different digital radios. The technique is equally applicable to digital waveforms such as BPSK, QPSK, 8PSK, and GMSK. In the current design, the primary user would need to have a well-known digital signal structure, and its standard frame length would need to be longer than that used by the secondary user system. However, in future version of the protocol, we plan to allow for a variable secondary frame length and use a cognitive engine to automatically adjust the CCP threshold for multiple secondary user types.

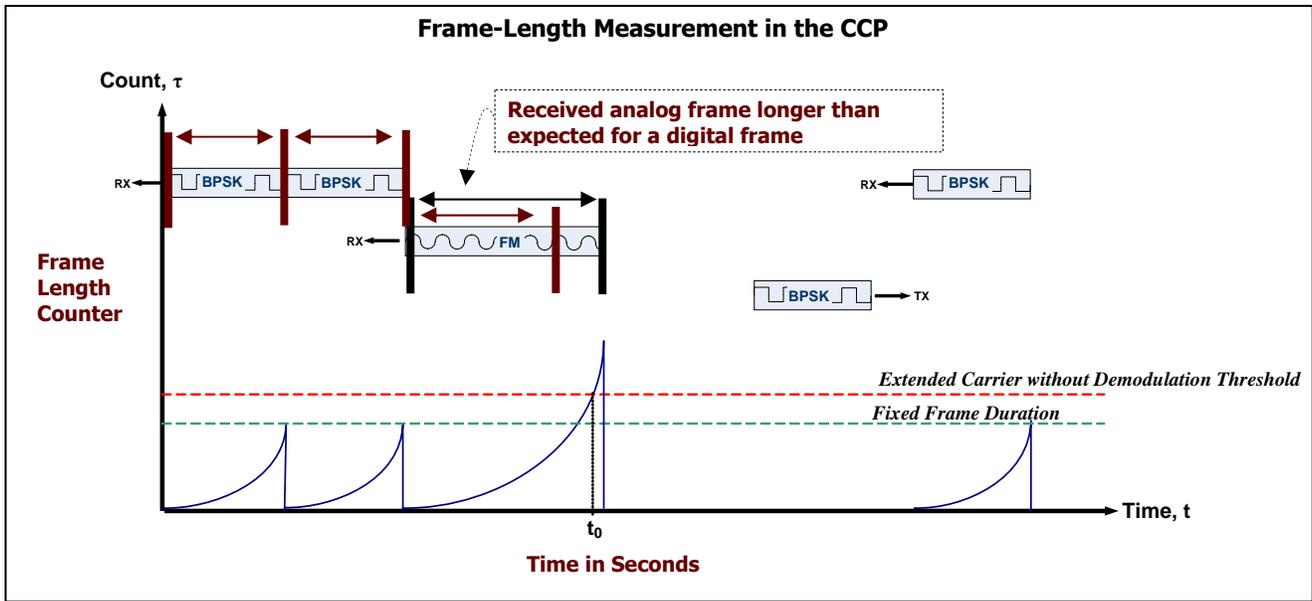


Figure 6.4. Time plot showing the conceptual operation of the CCP.
 [M. Silvius' Image from [72], adapted from [6]. Used with permission, © 2009 IEEE, © 2008 SPIE.]

6.4.2 Implementation

To implement the CCP in a prototype radio for initial testing and evaluation, we selected the GNU Radio [6] and USRP SDR architecture [7]. The pictorial in Figure 6.5 highlights the modifications made to GNU Radio's carrier-sense-multiple access (CSMA) MAC layer. In particular, we added an additional timer to the carrier sensing (CS) module to measure the frame lengths of incoming signals. The bolstered carrier sense module also interacts with the transmit (TX) and receiver (RX) modules. The status of the primary user detection timer is monitored by the radio's master control module (MC). This radio resource manager directs the PHY and MAC layers to switch to a different operating frequency when needed to avoid the primary user.

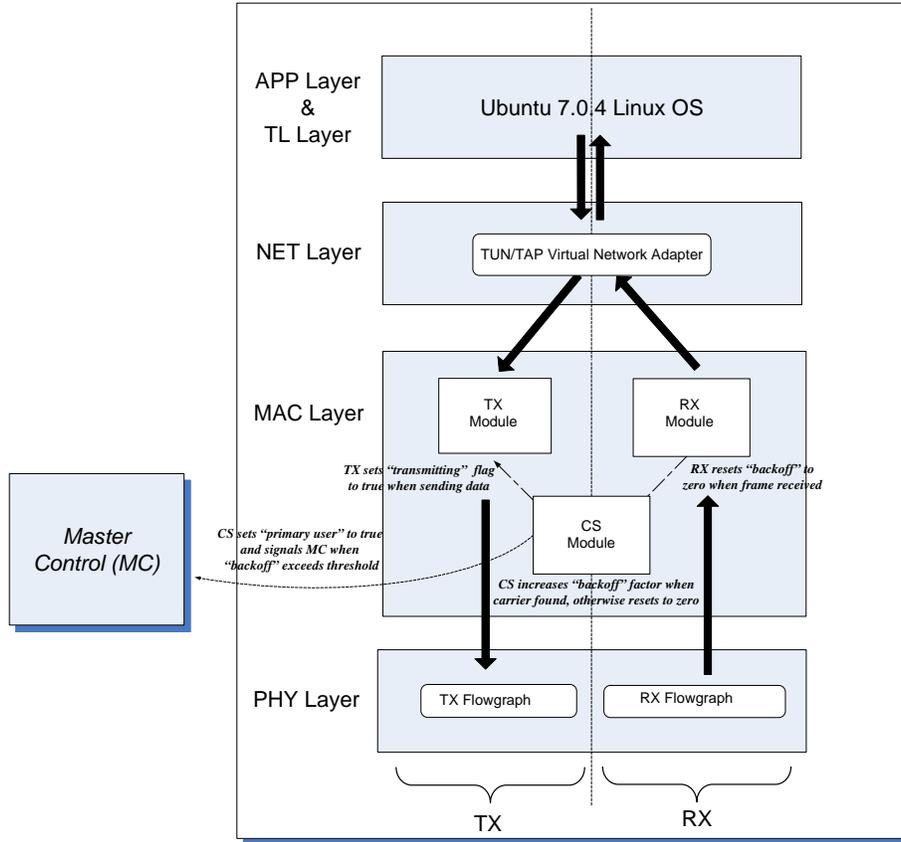


Figure 6.5. Modifications to GNU Radio protocol stack to support the CCP. In particular, this diagram highlights the operation of carrier sense (CS) module and transmission (TX) and receive (RX) modules. [M. Silvius' Image from [72]. Used with permission, © 2009 IEEE.]

6.5 Methodology for Evaluation

We sought to evaluate the CCP using two strategies. In a initial effort, we conducted simulations to model the performance of the CCP using the OMNeT++ software package [84]. In a later effort, we performed laboratory experiments of the CCP when implemented on the Smart Radio Network Testbed, as detailed in Chapter 4. In the remainder of this chapter, we summarize the results of the simulation and experimental work.

To evaluate the performance of the CCP, this work uses OMNeT++ and testbed experiments to collect three primary metrics: goodput, packet loss, and delay. Goodput is a measure of the number of useful bits per unit of time forwarded by the network from a certain source address to a certain destination, which excludes protocol overhead and retransmitted data packets [85]. Goodput should not be confused with throughput, which is a measure of the raw bits per time flowing in link or through a node in a network. When compared to throughput,

goodput gives us an indication of how efficiently data is flowing through the network and how much bandwidth is being wasted as a result of retransmissions, noise, or inefficient management, or sharing of the channel. The definition of packet loss is straightforward. It is a measure of the percentage of transmitted packets that are dropped while being sent to their recipient. When computing packet loss, one can count only the original transmitted packets, as was done when computing goodput, or one can count the loss rate for all packets, both original and retransmitted, as well as acknowledgements. This dissertation only counts original packets toward the packet loss calculation. Lastly, we define latency as the total time it takes to successfully transmit a data packet to a receiver, and for the sender to successfully receive an acknowledgement in return. In this way, we use the round-trip definition of latency, rather than the one-way [85].

6.6 Simulation Setup

Since OMNeT++ is a discrete-event simulator, the software does not model events in the continuous time domain. Rather, important events are scheduled along an event timeline using the simulator master clock. The majority of the protocols used in wireless systems can be thought of as operating as Finite State Machines (FSM). For example, a MAC protocol is typically only performing the tasks for one operation at a time; however, it can switch to executing a different set of tasks in reaction to internal and external events. For the simulations in this dissertation, the author and his colleagues built a custom MAC layer for use with the CCP, as well as simplified application, transport, network, and physical layers broadly modeled after the 802.11 standard.

6.6.1 Medium Access Control Layer (MAC) with CCP Model

Our initial MAC implementation was based on the CSMA algorithm included in GNU Radio and used in the CWT Smart Radio 2007 and 2008 projects. The operation of the MAC layer can be best viewed as a FSM, containing four primary states. First, it is in the idle state while it waits for packets to arrive up from the PHY layer or down from the NET layer. Second, if a packet arrives down from the NET layer, it switches to the carrier-sense state, where it sends a *carrier sense request message* down to the PHY layer. It waits until the PHY layer responds with a measurement of RF power currently in the channels. Third, if there is no carrier power present, it then progresses to the transmit state, where it sends its data frame down to the PHY

layer for transmission. If there is carrier power present in the channel from the other nodes, it enters its backoff state. It remains in this state, delaying for a period of time, whose maximum possible value is determined by the number of previous retransmission attempts. After completing the backoff state, the MAC reenters the carrier-sense state, where it again measures the carrier power present and attempts to retransmit. If the unsuccessful retransmission attempts exceed a predetermined threshold, the MAC simply drops the frame and goes back to the idle state. Figure 6.6 details the FSM representation of the MAC layer.

To implement primary user detection, the CCP keeps track of the number of times the MAC detects the presence of a carrier and enters the backoff state for each data frame. It also keeps track of the current backoff value, as well as the total accumulated backoff delay that the MAC waits for each frame it wishes to transmit. This is equivalent to incrementing the frame length counter discussed in Section 6.4.2 and shown in Figure 6.4.

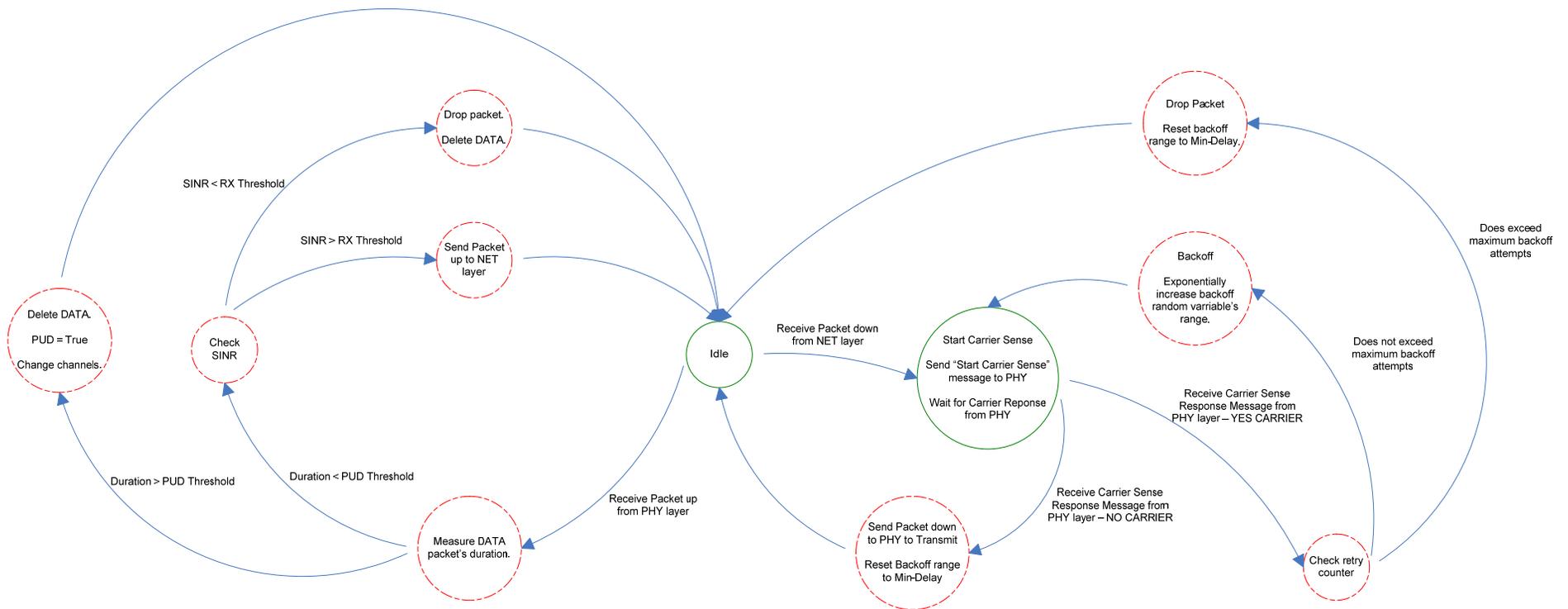


Figure 6.6. The FSM representation of the MAC layer in OMNeT++.

6.6.2 Remaining Protocol Stack Layer Models

For our simulation, we modeled our APP/TL as a simple *Stop-and-Wait* protocol [86]. In general terms, the TL would only transmit one data packet over the network at a time. On transmission, it would schedule a timeout clock event, and then begin to wait for an acknowledgement message from the intended recipient. If a positive acknowledgement (ACK) arrived from the recipient, it proceeds with sending out the next new data packet. If a negative acknowledgement (NAK) arrived from the intended recipient, or if the timeout clock event occurs, the TL will re-send out the most recent data packet.

The NET's FSM is much more simplistic than for the APP/TL layer. The NET layer has three main states. First, by default, is the idle state where it is waiting for a packet. Second, if it receives a packet from the TL, it sends the packet down to the MAC layer. Third, if it receives a packet up from the MAC layer, it must analyze the packet's headers. If the packet is addressed for its host node, it sends the packet directly up the TL layer. Otherwise, if the packet is destined for another node, the NET layer updates both the *next-hop* and the *previous hop* fields of the packet, and sends it back down to the MAC layer to be forwarded.

Like the other protocol stack modules, the operation PHY layer can be viewed as very simple FSM with four states. First, it is idle while waiting for packets to arrive up from the channel or down from the MAC. Second, if a packet arrives down from the MAC layer, it switches to the transmit state where it transmits the packet onto the channel. Third, if a packet arrives up from the channel, it switches to the receive state where it forwards the packet up to the MAC layer. Last, PHY is capable of sending the MAC layer measurements of the received RF power currently present in the channel, to be used by the MAC for an exponential backoff routine for CSMA, or for locating the presence of a primary user.

6.6.3 Primary User Model

Figure 6.7 depicts the node topology for the dynamic scenario when one primary user is present in the radio environment. In this scenario, one can also view the primary user as a jammer, interfering with the communication links of a secondary user network. In the simulation, the primary user node closely resembles the operation of the other secondary user nodes, except for two significant differences. First, it transmits only in a broadcast mode, which in terms of the OMNeT++ simulation, means that it sends copies of its frames to all the other

nodes in the network. This is shown by the directly-connected arrows tying the primary user to the other secondary user nodes. Second, the primary user transmits frames that are significantly longer in length than its neighboring secondary user nodes. For most simulations these frames were on the order of five to ten times longer than frames used by the secondary user nodes. The differences in these frames' lengths will be used by the CCP to discriminate between their primary and secondary senders.

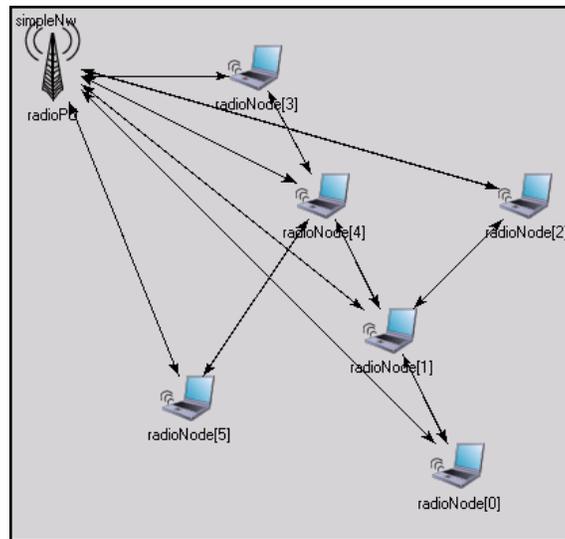


Figure 6.7. Our OMNeT++ simulation scenario with six secondary users and one primary user. The links shown here are logical links, and they operate in parallel with the simulation's wireless propagation model. [M. Silviu's Image from [72]. Used with permission, © 2009 IEEE.]

6.6.4 Configuration Parameters

Table 6.1 lists the parameters used in our OMNeT++ simulations. The transmit powers of the nodes were set to 30 mW, which is the standard transmit power of network interface cards (NIC). The receive sensitivities were set to -95 dBm, which corresponds to the sensitivity of NICs that receive signals at the 1024 Kbit/s linkrate. The CCP measures the duration of the noise carrier. If the carrier's duration is longer than the `PU_DETECTION_THRESHOLD_DURATION`, the CCP will initiate the fallback channel change within the MAC layer. It will then begin to transmit on the new channel. If a node is unable to transmit due to interference exceeding `PhyTxNoiseThreshold`, then it will start an exponential backoff timer and sense the carrier again after the timeout. This procedure will occur with a maximum number of times as defined by `MAX_BACKOFF_RETRIES`. Failure to

transmit will result in a dropped packet at the MAC layer. IFFactor is a correction to the interference which compensates for adjacent channel interference between nodes. AIR_TRANSMIT_DELAY is calculated based on the Msgsize and linkrate, with varying linkrates for multiple scenarios. PhyInterfThreshold, is the maximum noise threshold for a received packet. PhySignalThreshold is the minimum signal threshold for a received packet. PhyTxNoiseThreshold is the maximum threshold for sensed noise for determining whether a channel is busy. The APP/TL layer will timeout and re-transmit a packet if ACK has not been received during the TIMEOUT_DURATION time. The total duration of the simulation was 8 seconds.

TABLE 6.1. PARAMETERS USED IN OMNeT++ SIMULATIONS.

Variable	Value	Units	Comment
pSend (SU Node)	30	mW	Transmit power
alpha (SU Node)	3.5	n/a	Channel attenuation
PUpSend (PU Node)	50	mW	Transmit power
PUalpha (PU Node)	3.5	n/a	Channel attenuation
Msgsize	16-768	Kbits	Data packet length
Linkrate	1024	Kb/s	Link bit rate
fchannel(0-5 Nodes)	1.8-2.4	GHz	SU Tx frequency
Distances between nodes (0-5)	5-30	m	Distances between SUs
DELAY_CARRIER_SENSE_DURATION	0.02	sec	Carrier sense delay
AIR_TRANSMIT_DURATION	n/a	sec	Transmit delay
DELAY_BACKOFF_DURATION	0.02	sec	Min. backoff delay
PU_DETECTION_THRESHOLD_DURATION	0.6	sec	PUD threshold
MAX_BACKOFF_RETRIES	8	n/a	Max backoff retries
MAX_DATA_RETRANSMISSIONS	8	n/a	Max TL data retries
MAX_ACK_RETRANSMISSIONS	8	n/a	Max TL ack retries
TIMEOUT_DURATION	0.6	sec	TL retransmission timer
NOISE_TRANSMIT_DURATION	0.8	sec	PU noise Tx duration
PhyInterfThreshold	-85	dBm	Noise threshold
PhySignalThreshold	-95	dBm	Signal (Rx) Sensitivity
PhyTxNoiseThreshold	-90	dBm	Carrier threshold for Tx
IFFactor	f(freq)	dBm	Interference attn' factor
PU0fchannel	2.4	GHz	PU Tx frequency
Fallback(0-7 channels)	1.8-2.4	GHz	SU fallback channels

These parameters were used in the OMNeT++ simulations.

6.7 Simulation Results

After completing our simulations, we plotted the generated OMNeT++ data to quantify how effectively the CCP-equipped, secondary user smart radios mitigated the interference from the primary user. For each QoS metric, we plotted both the control scenario without the primary user, and the experimental scenario, where the primary user suddenly started transmitting on the secondary users' operating channel at two time intervals.

Figures 6.8-6.13 summarize the results of our simulations where a primary user began transmitting at $t=2.22$ and $t=4.62$. First, Figures 6.8-6.9 record the measured in-channel interference strength (i.e. signal + noise + interference power) received by four of the secondary user radios nodes in the network. In the CCP case, the technique reduced the time that

interference was observed from 1.4s to 0.26s—an interference reduction of approximately 81.4% from the non-CCP case, where there is no primary user detection or avoidance enabled. In both cases, the four curve traces in the figures represent the interference power measured by four of the nodes in the simulation. The noise floor was approximately -85 dBm.

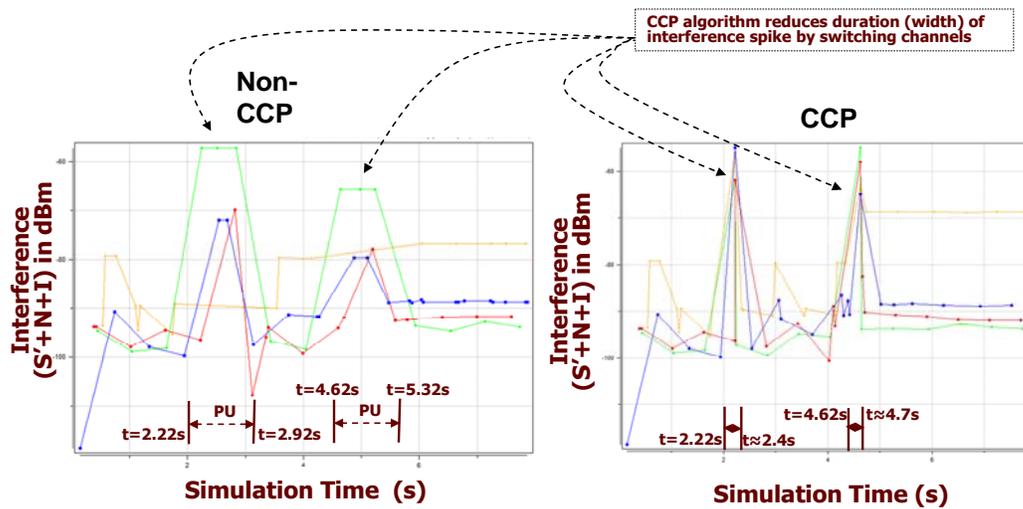


Figure 6.8 and 6.9. In our OMNeT++ simulation, the CCP reduced duration of received interference, as measured in four nodes, shown here by the four curve traces. [M. Silvius' Images from [72]. Used with permission, © 2009 IEEE.]

Second, Figures 6.10-6.11 depict the time plots of the MAC layer; i.e. more specifically, the blue and red dots indicate when Nodes 1 and Node 4 transmitted frames respectively. The large horizontal gaps between the points indicate delays (i.e. latencies), resulting from outgoing frames being queued, rather than transmitted, as result of primary user interference. Mirroring the previous results, these timing plots show that the latencies due to interference decreased from 1.4s to 0.26s, that is, by 81.4%, in the CCP-enabled case

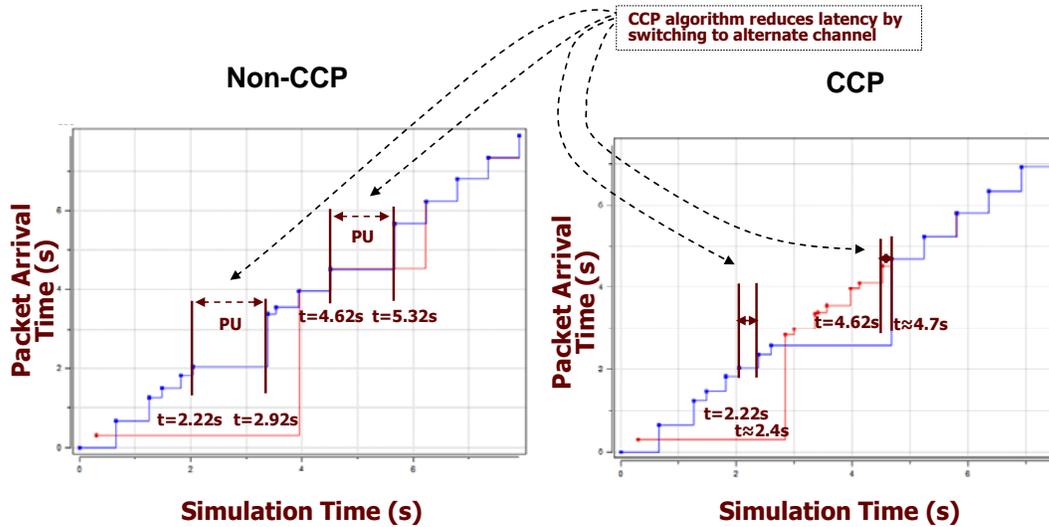


Figure 6.10 and 6.11. In our OMNeT++ simulation, the CCP reduced latency between packet arrival times. (Node 1 represented by blue dots, Node 4 represented by red dots.) [M. Silvius' Images from [72]. Used with permission, © 2009 IEEE.]

Third, we measured average goodput and packet loss incurred during the simulation as a function of the secondary user's frame length, for the both the CCP and non-CCP case, as shown in Figures 6.12-6.13. For the CCP-enabled case, received goodput increased by 21% to 40Kbit/s, and packet loss decreased by 25% to an average loss rate of 22%.

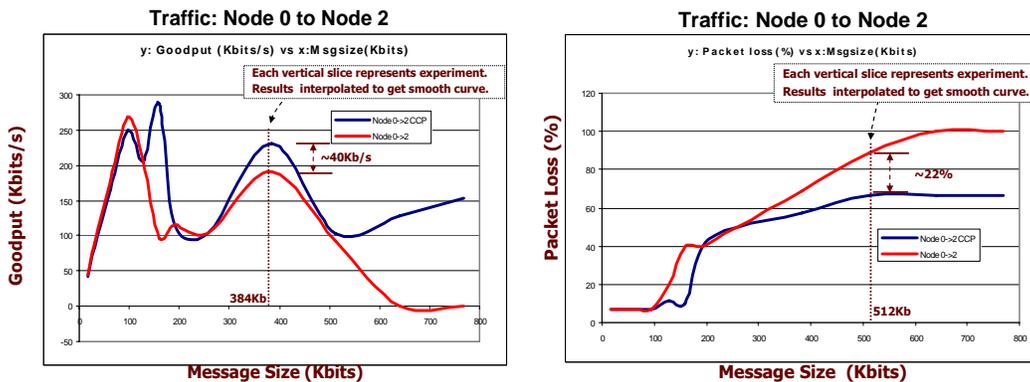


Figure 6.12. In our OMNeT++ simulation, the CCP increased goodput (left). Figure 6.13. Conversely, the CCP decreased packet loss (right). [M. Silvius' Images from [72]. Used with permission, © 2009 IEEE.]

6.8 Testbed Experiment Setup

In addition to software simulations using OMNeT++, we also evaluated the performance of the CCP by using Smart Radio Network Testbed described in Chapter 4. However, for this set of experiments, we expanded the network testbed from four to five nodes, so as to include an extra radio to imitate the action of a primary user. See Figure 6.14 for an updated diagram of this configuration.

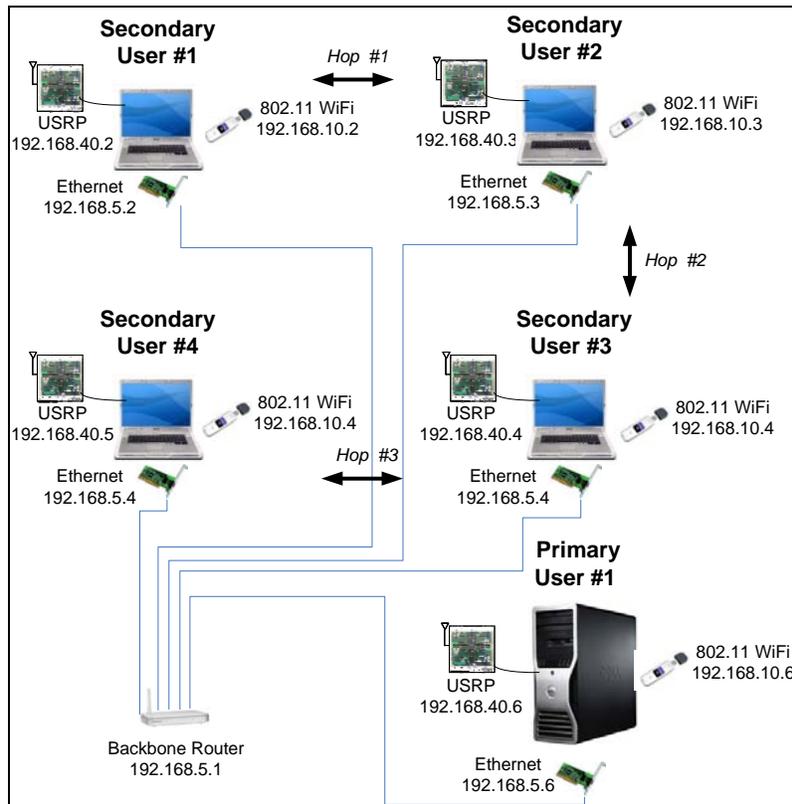


Figure 6.14. Detailed architecture of the network testbed.

Using this updated network testbed design, our physical laboratory setup more closely matched the simulation setup in Figure 6.7. The first four nodes were smart radio nodes, built using the GNU Radio and the USRP, housed on Linux-based laptop computers, and running the CCP. The fifth node acted as a primary user and source of interference. In all our experiments, the primary user transmitted twice, first at $t=10-20s$ using analog FM, and then at $t=30-40s$ using GMSK. All secondary users used GMSK at 500Kbit/s using a maximum frame length of 2200

bytes. The primary user, when in digital mode, also transmitted GMSK at 500 Kbits/s, but with a minimum frame length of 2500 bytes.

First, to measure the QoS of the network formed by the secondary users, we used the IPERF open source software tool [87]. IPERF allowed us to measure the UDP goodput, packet loss, and jitter of the three-hop path from Node 1 to Node 4. When configuring IPERF, we used a stream of 1 Kbyte sized UDP segments at an offered rate of approximately 100 Kbit/s. By transmitting our test stream at less than our link rate of 500 Kbit/s, we ensured that there was space between the outbound transmitted frames, and therefore, our IPERF generated plots were generally smoother and more reproducible. We also used the standard PING command, available in Linux, to measure the round-trip-time along this path.

Second, we evaluated the precision of the CCP's threshold in detecting the primary user. In this experiment, we designated that the maximum secondary user packet size to be 2200 bytes. We set our goal CCP threshold to be slightly higher than this value, at 2300 bytes. Then we had the primary user send a stream of digital frames with frame lengths varying between 100 bytes and 4096 bytes, with step sizes of 100 bytes. For each frame size we ran 100 experimental iterations. We recorded the results for three network conditions: first, when the network was idle; second, when Node 1 streamed packets to Node 3 at an offered rate of 50 Kbit/s; and third, when Node 1 streamed packets to Node 3 at an offered rate of 100 Kbit/s.

6.9 Testbed Experiment Results

Figure 6.15 shows the UDP goodput response of the five node cognitive radio network. In the non-CCP case, the effective goodput of the secondary user network essentially dropped to zero during the two 10s intervals where the primary user was transmitting. Note that at $t=32s$ and $t=36s$, the secondary user network was able to successfully transmit and receive a handful of packets, slipping them past the primary user, during the inter-packet quiet periods of the primary user's transmission cycle. In the CCP case, the technique was able to successfully detect the presence of the primary user and then switch to an alternate operating channel. The periods of network outage due to interference dropped by over 50%, from 10s to 5s, resulting in a 33% increase in goodput. Moreover, the width of the trough in the CCP-case represents a maximum of 4-5s of *switching time*, specific to our SDR platform's implementation. This time could be shortened if alternate, high-performance platform was used. Note that the little upticks in trace

represent the granularity of our measurement system. In additional measurements, we also recorded the jitter, packet loss, and latency response in the UDP stream from Node 1 to Node 4, as shown in Figures 6.16-6.18. The CCP technique, with its ability to avoid the primary user, reduced the periods of instability from 10s to approximately 4-5s, and reflected 65%, 55%, and 42% decreases in these metrics respectively as shown next.

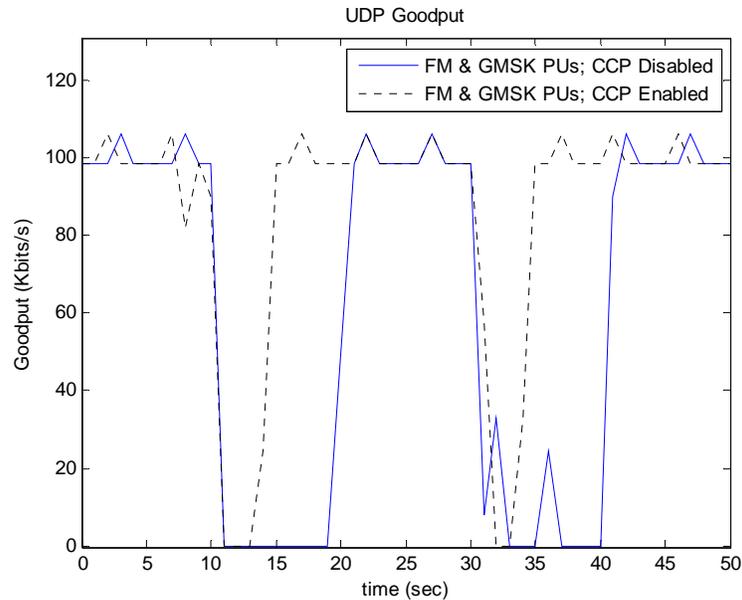


Figure 6.15. UDP Goodput during GNU Radio Experiment. [M. Silvius' Image from [72]. Used with permission, © 2009 IEEE.]

Figure 6.16 shows the jitter response in the UDP stream from Node 1 to Node 4. Large jitter plateaus resulted during both periods of transmission of the FM and GMSK primary user. The values soared from an average of 3-5ms to maximum recordable value 100ms. However, like in Figure 6.15, the CCP technique, with its ability to avoid the primary user, reduces the periods of instability from 10s to approximately 3-4s, for a 65% improvement.

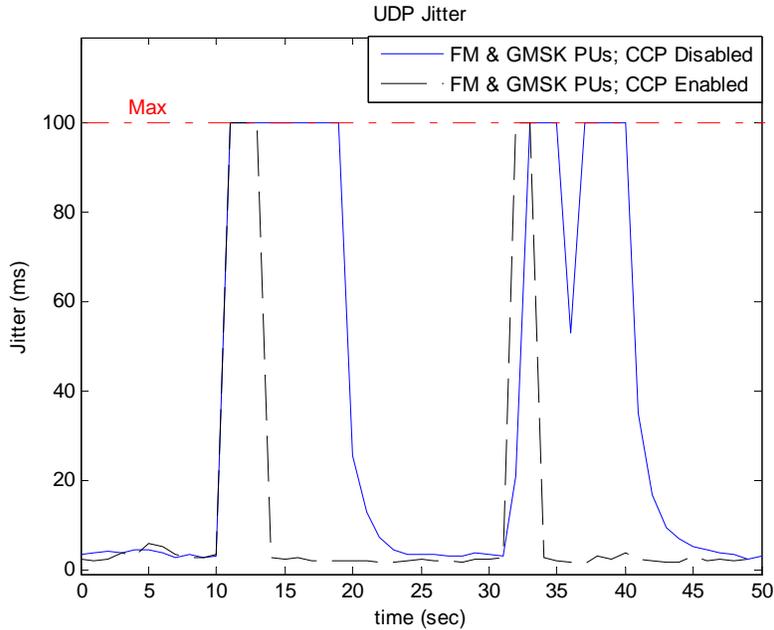


Figure 6.16. UDP Jitter during GNU Radio Experiment.
[M. Silvius' Image from [72]. Used with permission, © 2009 IEEE.]

Figure 6.17 shows the packet loss in the stream from Node 0 to Node 4. The flat plateaus should be viewed not only as periods of packet loss, but also as periods of time when packets were being queued by the transmitter and unable to be sent as a result of primary user interference. As compared to the non-CCP case, the CCP-cased reduced channel outages from approximately 10s to 5s, for a 50% improvement.

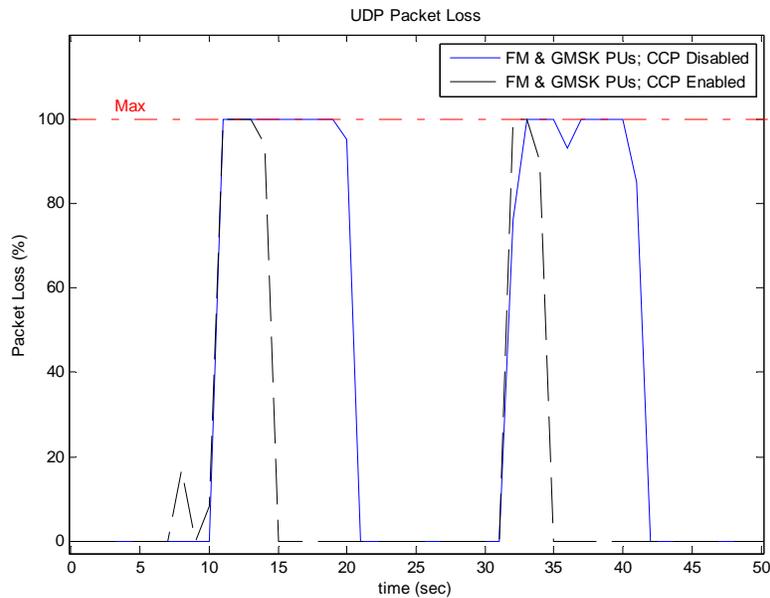


Figure 6.17. UDP Packet Loss Rate during GNU Radio Experiment.
[M. Silvius' Image from [72]. Used with permission, © 2009 IEEE.]

Figure 6.18 shows the round-trip latency of the four node cognitive radio network, as measured with the PING for both the CCP and non-CCP case. The spikes and plateaus in the curves resulted when one or more of PING's ICMP packets successfully returned with a RTT equal to or greater than 500ms, or when the ICMP packet was completely lost during its trip, i.e. with an RTT of infinity. For the non-CCP case, latency soars from an average of 35-40ms to the maximum graph value of 500ms during both the 10s primary user transmission windows. However, for the CCP case, the periods of excessive delay were reduced to approximately 5s and 3s respectively, for 42% improvement.

Interestingly, in the CCP case, the Figure 6.18 also showed a short period of instability following the channel switching operation, as shown at $t=22-24$ and $t=39-43$. This could possibly be an idiosyncrasy of the CCP's interaction with the GNU Radio PHY and MAC layer in the SDR platform. However, this evidence of this phenomenon did not appear in the IPERF generated jitter plot in Figure 6.16, we believe these additional spikes to be an artifact with the way the PING program schedules and transmits its ICMP packets.

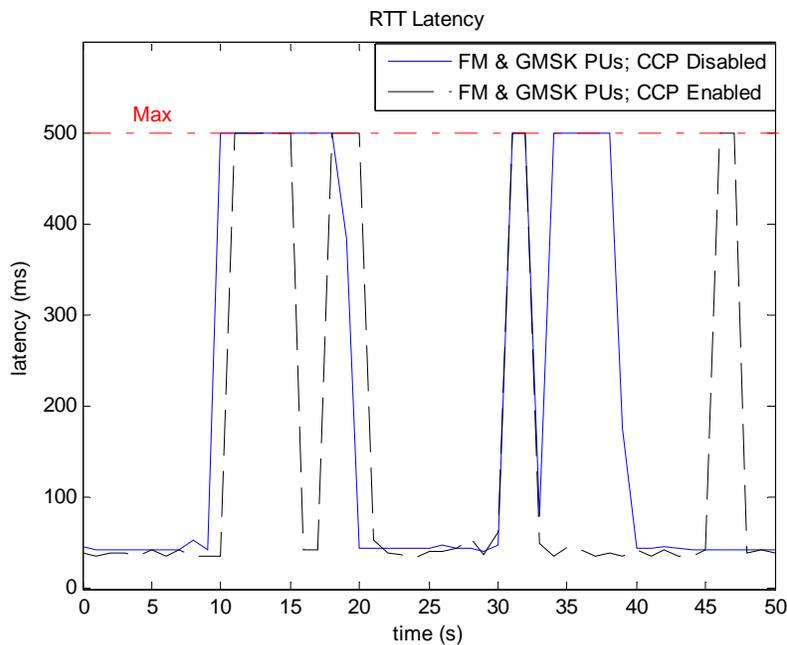


Figure 6.18. Latency during GNU Radio Experiment.
[M. Silvius' Image from [72]. Used with permission, © 2009 IEEE.]

The resulting curve in Figure 6.19 shows the precision of the CCP’s threshold in detecting the primary user. The detection break-point occurred sharply at 2350 bytes, as desired. Detection reliability was also the highest for the idle condition, and gradually decreased as the offered rate increased.

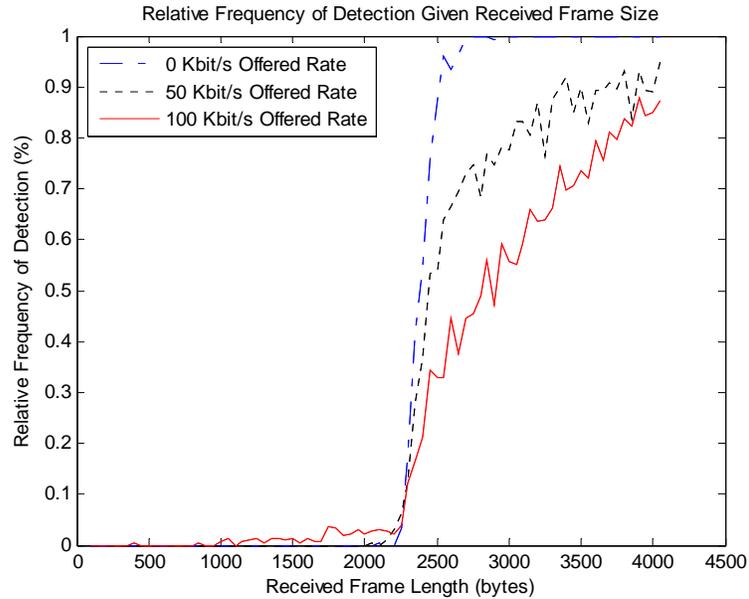


Figure 6.19. Threshold Detection Reliability for a Given Offered Rate into Secondary User Network. [M. Silvius’ Image from [72]. Used with permission, © 2009 IEEE.]

The threshold results in Figure 6.19 highlight a noteworthy trade-off between available network bandwidth and primary user detection reliability. The probability of successfully detecting the primary user is proportional to the length of time the secondary user system spends sensing the carrier. As the secondary user system’s maximum offered rate was reduced, the spectral awareness of the secondary user radio increased and the probability of cross-network interference decreased. In terms of spectrum sharing protocol design, the offered rate determines the spacing between transmitted packets, known as the *quiet period*, in which the node conducts carrier sensing and primary user detection. If packets are transmitted without any spacing—i.e. the quiet period between them—the CCP has no time to sense the channel, and the probability of detecting the primary user drops. However, if there is ample inter-packet spacing, the CCP probability of success greatly improves. This sets a limitation on the maximum duty-cycle for transmission of packets. Initial experiments suggest that a duty cycle of no more than one-half

produces the best results, with a reasonable tradeoff between transmission rate and probability of primary user detection.

6.10 Validation of Simulation Results with Experimental Results

Although we first introduced the conceptual operation of the CCP in [6], at that time, our evaluation consisted of a simple scenario consisting of two digital cognitive radio secondary users and one analog FM primary user. Initial observations established that the CCP-enabled cognitive radios did successfully detect the FM radio and switch to a vacant channel, when the primary user keyed the audio microphone. However, this dissertation quantified to what degree the CCP improved the network QoS of a multi-hop configuration of cognitive radios, and it evaluated the reliability of the detection scheme under varying offered network loads. It also verified that the CCP performed as initially conceived in [6], through the use of OMNeT++ simulations and five node laboratory experiments.

Table 6.2 summarizes the results of both the OMNeT++ simulations and the five node experiments. Both sets confirmed that the CCP did indeed improve the network QoS metrics. Although implementation differences in the protocol stacks of OMNeT++ and GNU Radio prevent us from doing an exact comparison, the performance trends coincide and verify that the CCP does operate as conceived, enabling wireless co-existence between primary and secondary users.

TABLE 6.2. QoS IMPROVEMENTS WITH CCP TECHNIQUE OVER BASELINE
[M. Silvius' Table from [72]. Used with permission, © 2009 IEEE.]

QoS Metric	Simulation Result	Experimental Result
Inteference Duration	-1.14s (-81%)	≤ -5s (≤ -50%)
Goodput	+40Kbit/s (+21%)	+24Kbit/s (+37%)
Jitter	n/a	-25ms (-67%)
Packet Loss	-22% loss (-25%)	-25% loss (-64%)
Latency	-1.14s (-81%)	-59ms (-29%)

6.11 Stability of Experimental Results

In order to verify the stability of the experimental results, we ran one further set of laboratory experiments. Here, we used a simplified version of the Smart Radio Network Testbed of Chapter 4 consisting of three nodes, as shown in Figure 6.20. This simplified setup was required due to configuration time and hardware usage constraints. Two nodes were digital smart radio nodes, and the third imitated a primary user. Again, we used USRPs with 700 MHz daughterboards. The secondary users used the GMSK modulation with a link rate of 500 Kbits/s and an offered rate of 100 Kbits/s. As in Section 6.7 and Section 6.8, we again had the primary user transmit twice, first at $t=10-20s$ using analog FM, and then at $t=30-40s$ using GMSK. Unlike in the previous experiments, we averaged twenty iterations of experiments together to plot each mean QoS curve. We also summarized the mean QoS metrics in Table 6.3 and Table 6.4.

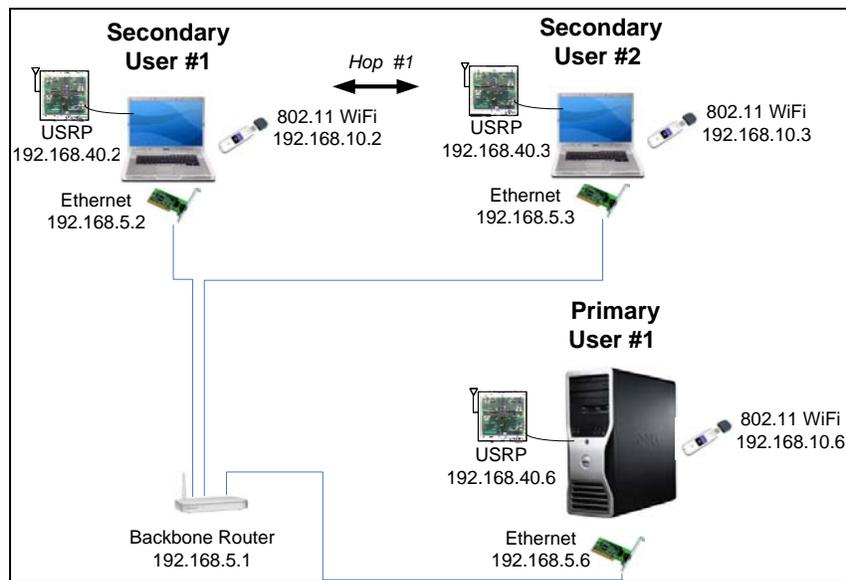


Figure 6.20. Simplified smart radio network testbed.

With the exception of the first row, each cell in Table 6.3 contains two pieces of information. The first number is the mean of the metric for all experiments. The second is the standard deviation of the metric across the experiments. Take for example, the experimental results for goodput. There were 20 experiments, each lasting 50 seconds. The mean goodput for all the 50 second runs was 97.26 Kbits/s. This is an excellent result, considering that the offered to the link rate was set to 100 Kbits/s. From one experiment to the next, the mean goodput had a standard deviation of 2.35 Kbit/s. This reflects that the average goodput varied between experiments, most likely as result of the precision of the Channel Change protocol threshold, as is discussed next.

The mean interference duration was computed by inspecting the resulting goodput curves from Figure 6.21. From the baseline goodput metric results, we set a threshold of the three deviations below and above its mean. This corresponded to 90.51 Kbits/s and 109.17 Kbits/s. Anytime one of the traces in Figure 6.21 went below or above these limits, we flagged the interval as having interference and recorded the time duration. We used to these results to populate row one of Table 6.3. Table 6.4 summarizes the change in metrics between the experiment and the control data.

TABLE 6.3. AVERAGED QOS METRICS
(MEAN OF METRIC, STD OF METRIC ACROSS EXPERIMENTS)

QoS Metric	Baseline (No PU, No CCP)	Control (Yes PU, No CCP)	Experiment (Yes PU, Yes CCP)
Inteference Duration (sec)	0.00, n/a	20.48, n/a	8.36, n/a
Goodput (Kbits/s)	99.84, 0.00	92.38, 0.43	97.26, 2.35
Jitter (sec)	0.82, 0.07	20.00, 1.17	7.84, 1.24
Packet Loss (%)	0.00, 0.00	18.31, 0.94	5.96, 1.97
Latency (sec)	40.07, 19.84	104.01, 25.36	148.49, 74.78

TABLE 6.4. AVERAGED QoS METRIC CHANGE
(ABSOLUTE CHANGE, PERCENT CHANGE)

QoS Metric	Experiment WRT Control
Inteference Duration (sec)	-12.12 (-59.18%)
Goodput (Kbits/s)	+4.88 (+5.28%)
Jitter (sec)	-12.16 (-60.80%)
Packet Loss (%)	-12.35 (-67.45%)
Latency (sec)	+44.48 (+42.77%)

Inspecting these results, one can make a number of conclusions. First, in looking at the control data with respect to the baseline data, we can see that the presence of the primary user significantly degraded each of the metrics. However, the primary user did not significant change the variability of these results from one run to the next—the process was stable. Second, in looking at the experimental data with respect to the control data, we can see that the Channel Change protocol significantly improved each of the metrics. However, the Channel Change protocol increased the variability of these results from one run to the next—it made the process less stable. This was most likely due to the variability and precision of the Channel Change protocol’s threshold. As we showed in Figure 6.19 in Section 6.9, occasionally, the Channel Change protocol fails to detect the primary user. A failed detection would adversely degrade the metrics’ means and increase their standard deviations. Note, it should be stated that the large spike in all the traces, appearing after the primary user stops transmitting, or the after the node changes channel, is the result the rapid transmission of packets that were queued during the interference period.

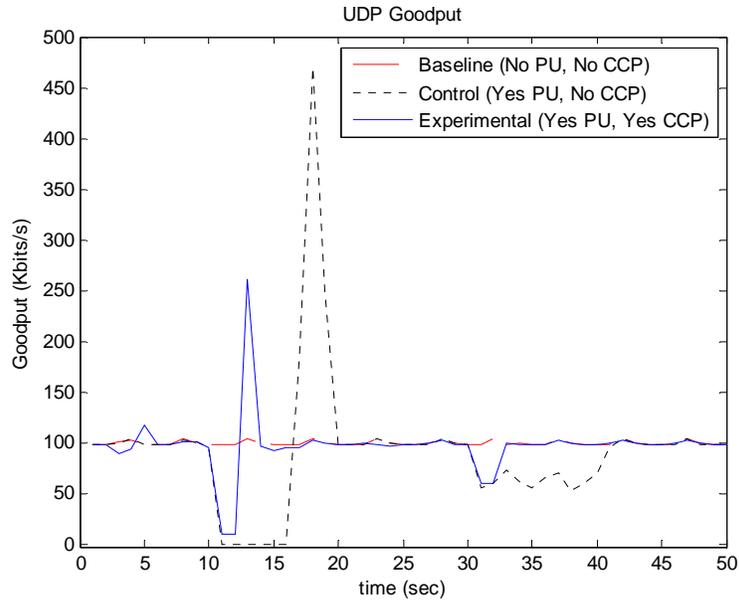


Figure 6.21. Averaged goodput traces.

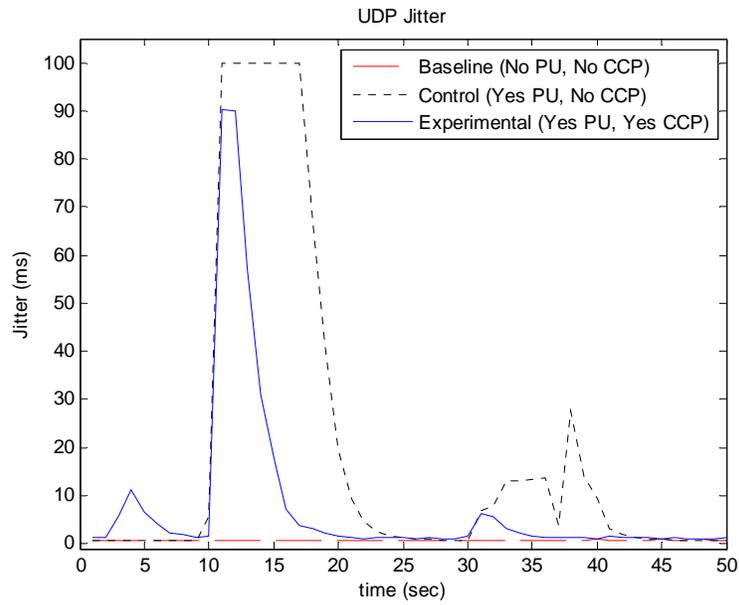


Figure 6.22. Averaged jitter traces.

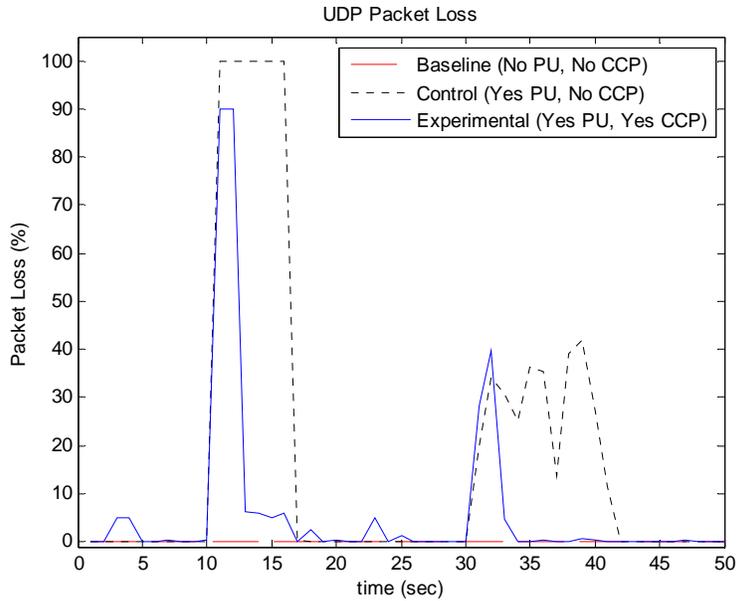


Figure 6.23. Averaged packet loss curves.

The latency traces in Figure 6.24 show more variation, and they deserve additional discussion. First, it is important to note that the latency experiments were run separately, at a later time than the goodput, jitter, and packet loss experiments in Figures 6.21-23. Therefore, the individual features in the latency curve are *not* correlated with the features in the other curves. Second, in Figure 6.24, the *tails* after the spikes, i.e. during $t=20-30s$ and $t=40-50$, represent times when the CCP should have switched channels, but did not, due to a failure to detect the primary user. Stated in another way, the two secondary radio nodes in the testbed became out of synchronization during these times, because one of the nodes failed to detect the primary user and switch channels. Third, the latency metric was measured with the PING program, while the other metrics were measured with the IPERF program. The traces demonstrate that the PING tool's results are unstable, produce numerous spike artifacts, and that the PING tool is inferior with respect to the IPERF tool.

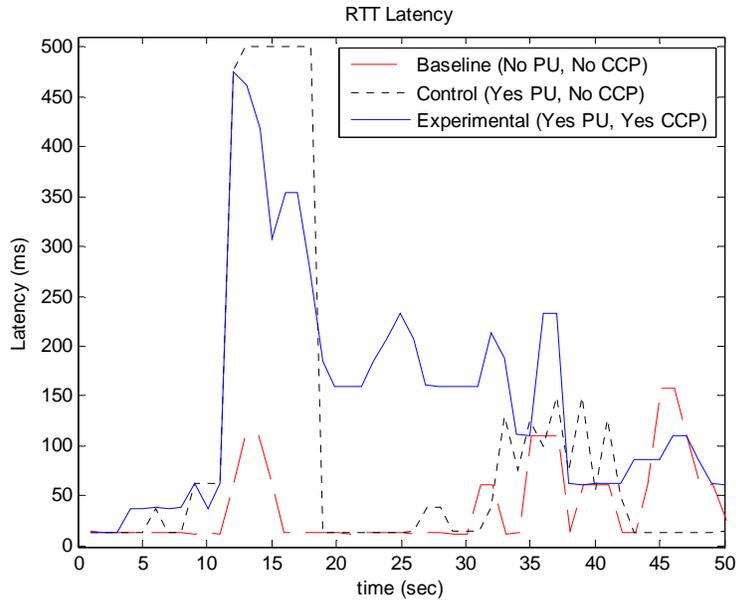


Figure 6.24. Averaged latency curves.

6.12 Summary

The Channel Change protocol offers secondary users a straightforward technique for the detection and avoidance of primary users. This technique allows for reduced interference and improved co-existence between competing communications networks. It can be applied to existing wireless LAN technologies or readily implemented in SDR-based cognitive radios. The result is improved the network quality of service for both primary and secondary users in the band.

Chapter 7

Conclusions

This chapter summarizes the contents of this dissertation, highlights the research contributions made, discusses applications of the developed technologies, and proposes areas of continued and future research.

7.1 Summary

This dissertation presented the results of research to develop technologies and protocols which could provide solutions to communication problems faced by public safety first responders. It demonstrated that smart radios, assembled on top of reconfigurable software-defined radio architectures and coupled with dynamic spectrum access protocols such as neighbor rendezvous and spectrum sharing capabilities, can provide a solution to one of the radio discovery and spectrum scarcity problems faced by first responders in public safety disaster response scenarios.

Chapter 2 summarized the design and development of the CWT Smart Radio 2007. This smart radio was used to demonstrate communications capabilities needed for public safety first responders in an earthquake disaster scenario, and it formed the foundation for later evaluation of the dynamic spectrum access protocols researched and designed in this dissertation. Chapter 3 describes the improvements of this design and the subsequent development of the CWT Smart Radio 2008. This improved radio was used to demonstrate more advanced networking capabilities for the public safety first responders in a subway bombing response scenario. Chapter 4 described how to build the CWT Smart Radio Network Testbed using commercial off-the-shelf equipment and open-source software suitable for conducting network experiments by researchers in a laboratory environment. The testbed also provided an architecture to conduct tests and measurements on the dynamic spectrum access protocols described in Chapters 5 and 6. This work built on the core technologies used in the construction of smart radios for public safety

disaster response scenarios and from both smart radio challenges. Chapter 5 detailed our research in the development of rendezvous protocols. This chapter reviewed previous work by other authors in the design of these protocols, and it presented a taxonomy which enabled us to classify these previously proposed rendezvous protocols based on their design characteristics. In this chapter we further investigated the multi-channel, asynchronous problem, by implementing candidate rendezvous protocols on the CWT Smart Radio Network Testbed of Chapter 4, and we collected and analyzed discovery statistics to determine its performance as a function of the number of channels, nodes, and rendezvous beacon repetitions. Chapter 6 detailed the research and development of the Channel Change protocol. This dynamic spectrum access enabled our smart radio to detect the return of designated primary users and to orchestrate the change-channel procedures to an alternate, vacant block of spectrum. The Channel Change protocol allowed primary users and secondary users to co-exist in the same frequency band without interference.

7.2 Contributions

The principal contributions of this dissertation are as follows:

1. Developed smart radio prototypes which met the interoperability, spectrum sharing, and networking requirements of public safety first responders at the scene of an earthquake nature disaster scenario, as well as at the scene of a subway bombing response scenario, as enumerated by SDR Forum during the Smart Radio Challenge 2007 and 2008 competitions respectively. The prototypes were built on the GNU Radio and USRP software defined radio architecture, and they were able to identify available spectrum within a pre-defined band, rendezvous with an intended receiver, and transmit voice and data using a selected quality of service. The systems were also able to share spectrum between analog primary users and digital secondary users using our newly developed Channel Change dynamic spectrum access protocol. For the 2008 competition, the smart radio could also determine when it had entered a communication shaded region, and automatically form an ah-hoc extension to reach back and connect to a fixed infrastructure point.

2. Developed a smart radio network testbed that allowed us to evaluate the performance of the smart radio architecture and dynamic spectrum access protocols. Specifically, the system measured the QoS metrics of goodput, jitter, packet loss, and latency, using a three-hop, ad-hoc network of four smart radios. The smart radios were capable of rapid protocol stack reconfiguration and could transmit and receive with multiple waveforms using a combination of Bluetooth, WiFi, and USRP air interfaces.
3. Extended current research knowledge in the design and development of rendezvous radio discovery protocols for smart radios. Developed a taxonomy to classify the characteristics and discovery strategies of these protocols. Determined and enumerated the expected rendezvous probabilities for the multi-channel, asynchronous discovery problem through experimental tests with a candidate rendezvous protocol, using our four node smart radio testbed. Measured expected rendezvous probabilities as a function of the nodes, channels, and beacon repetitions.
4. Developed a spectrum sharing, dynamic spectrum access protocol called the Channel Change protocol. The protocol allowed primary users and secondary users to co-exist in the same frequency band without interference. The performance of the design was tested using both OMNeT++ simulations and experiments in our newly developed smart radio network testbed. Results demonstrated that the Channel Change protocol decreased packet loss and jitter and increased goodput with an acceptable increase in latency. Additional experiments validated the accuracy of the primary user detection threshold and plotted this accuracy as a function of the offered data rate to the network. Furthermore, the experimental results from the testbed validated and paralleled results from the OMNeT++ simulations.

7.3 Publications

This research in this dissertation has resulted in a number of publications as well as important project reports and whitepapers. The papers published or accepted to be published include:

- M. D. Silvius, A. B. MacKenzie, and C. W. Bostian, “A Survey of Neighbor Discovery Protocols in Modern Wireless Systems and their Application to the Design and Implementation of ‘Rendezvous’ Algorithms in Smart and Cognitive Radios,” in *Virginia Tech Symposium on Wireless Personal Communications*, Blacksburg, VA, June 6-8, 2007.
- M. D. Silvius, F. Ge, A. Young, A. B. MacKenzie, and C. W. Bostian, “Smart Radio: Spectrum Access for First Responders,” in *Society of Optical Engineering (SPIE): Wireless Sensing and Processing III*, Orlando, FL, March 17-18, 2008.
- M. D. Silvius, R. Rangnekar, A. B. MacKenzie, and C. W. Bostian, “The Smart Radio Channel Change Protocol: A Primary User Avoidance Technique for Dynamic Spectrum Sharing Cognitive Radios to Facilitate Co-Existence in Wireless Communication Networks,” in *4th International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM)*, Hannover, Germany, June 22-24, 2009.
- R. Rangnekar, F. Ge, A. Young, M. D. Silvius, A. Fayez, and C. W. Bostian, “A Remote Control and Service Access Scheme for a Vehicular Public Safety Cognitive Radio,” in *70th IEEE Vehicular Technology Conference (VTC)*, Anchorage, AK, September 20–23, 2009.
- M. D. Silvius, A. B. MacKenzie, and C. W. Bostian, “Rendezvous MAC Protocols for use in Cognitive Radio Networks (Accepted),” in *IEEE Military Communications Conference (MILCOM)*, Boston, MA, October 18-21, 2009.

The important project reports and whitepapers include:

- M. D. Silvius, T. Brisebois, C. Chen, Q. Chen, F. Ge, G. Marballie, Y. Wang, A. Young, and C. W. Bostian, “Spectrum Access for First Responders: Smart Radio Challenge 2007 Final Report,” Project Report: Wireless@Virginia Tech--CWT, September 30, 2007.

- M. D. Silvius, T. Brisebois, Q. Chen, A. Fayez, F. Ge, B. Li, G. Marballie, S. Nair, R. Rangnekar, Y. Shi, Y. Wang, A. Young, and C. W. Bostian, “Communications from an Infrastructure Damaged Area: Smart Radio Challenge 2008 Final Report,” Project Report: Wireless@Virginia Tech--CWT, September 30, 2008.
- M. D. Silvius, R. Rangnekar, A. B. MacKenzie, and C. W. Bostian, “An Educational Testbed Illustrating Ad-Hoc Networking and Software Defined Radio,” Unpublished Whitepaper, November 28, 2008.

7.4 Application: The 700 MHz Public Safety Band

A significant application of dynamic spectrum access and spectrum sharing is FCC’s proposed *700 MHz Public Safety System* [11]. In this scenario, a *National Carrier* will have licensed primary access to a 12 MHz portion of the 700 MHz band allocated for broadband usage. The National Carrier will also have secondary, unlicensed access to the remaining portions of the band allocated for narrowband usage, which will be owned and licensed to local public safety jurisdictions. In this system, dynamic spectrum access protocols and smart radio technology could be utilized as an effective and efficient way for the National Carrier system to detect narrowband FM public safety radios and coordinate transfers to alternate, unoccupied portions of the 700 MHz band. [12].

One approach to devising a solution to this scenario would to combine all of the technologies developed for the Smart Radio Challenge 2007 and 2008 and this dissertation’s research with the Rendezvous and Channel Change protocols. Together with other foundational software defined radio technologies, these could form the basis of a prototype solution to the 700 MHz Public Safety Band scenario. More specifically, first, each smart radio used will support all of the features involved in the 2007 competition. Each will be able to scan the operational spectral band, rendezvous with both analog FRS radios and digital smart radio nodes, and communicate using FM, BPSK, QPSK or 8PSK modulations. The smart radio nodes, acting as secondary users, will be able to share spectrum with FRS or legacy public safety radios, acting as primary users. Second, the smart radio will support interface types such as 802.11, Bluetooth, and will be able to route data seamlessly between them. Third, the smart radios will incorporate ad-hoc routing protocols that automatically allow it to update their network routing tables, as the repeater nodes move and change topology. Fourth, the master control module will be able to

analyze the smart radio's surroundings and automatically reconfigure its role as a command, repeater, or hidden node as the situation requires. A scanning module will enable the nodes to detect the loss of network connectivity from the infrastructure point and automatically reconfigure each node to ad-hoc mode. As in the 2008 competition, these capabilities will enable the nodes to provide a temporary fixed infrastructure or act as a *network extension* bridging the ad-hoc network in the shaded region to an existing public safety network.

7.5 Future Work

My future research goals are to adapt and to expand this dissertation's research to similarly-focused military communications research. Just as for public safety first responders, I see smart radios as effective tools for solving existing discovery, interoperability, and spectrum sharing and utilization problems faced by military warfighters. However, instead of needing temporary communication infrastructures at the scene of a natural disaster, warfighters need communication connectivity at forward deployed and austere locations. In these situations, I foresee a solution involving inexpensive, ground-based smart radio nodes, deployed via traditional aircraft or newly-developed unmanned aerial vehicles (UAV). Similar to the Smart Radio Challenge 2008, these smart radio nodes will continue to monitor for the presence of allied troops, aircraft, vehicles, and fixed communication networks. When connectivity or fixed infrastructure is not present, the nodes will automatically form a temporary ad-hoc network infrastructure. As emphasized in this dissertation, the principal formation tool will be the nodal discovery rendezvous protocols. Inter-nodal communications will also similarly adhere to the spectrum policy of the theater of operations, avoiding designated primary users and minimizing interference through the use of frequency spectrum scanners and the Channel Change protocol or similar dynamic spectrum access protocols. Individual smart radio nodes will act as smart gateways and repeaters, bridging existing or legacy military radio systems within a localized region. However, the smart network of nodes can extend the range of these legacy systems by providing a *digital extension* between individual and groups of legacy radios. In addition, the smart radio nodes can provide an *ad-hoc network extension* from the legacy system out of a connectivity-shaded region back to a fixed infrastructure for wide-area or global connectivity.

For my own research, I would like to develop a laboratory testbed in a secure facility, which would be capable of housing and operating actual military radio systems and secure military waveforms. The goal would be for the testbed to be developed by military masters level students, so as to provide a hands-on system design and software coding educational experience. In this way, the system testbed would not only be viable as a military operational testbed, but also as an educational testbed for military communications researchers and students.

A subsequent goal for the individual smart radios, as well as for the entire network testbed, would be the packaging and minimization process, to facilitate the mobility of the testbed to outdoor and field locations for further testing. Further minimization would be needed prior to real experiments involving the deployment of smart radio using aircraft and UAVs.

I firmly believe that field testing and operational evaluation by real users is key to the successful research and design process. In future research, I would strive to continue to build smart radio network prototypes which could be tested in military exercises and in near-real world scenarios. I envision seeking partnerships with military laboratories and test range facilities to test the suitability of the researched smart radio designs for meeting the military warfighters' communications requirements.

Bibliography

- [1] SDR Forum, "Smart Radio Challenge," in <http://www.radiochallenge.org>, 2007-present.
- [2] J. B. Bernthal, T. X. Brown, D. N. Hatfield, D. C. Sicker, P. A. Tenhula, and P. J. Weiser, "Trends and Precedents Favoring a Regulatory Embrace of Smart Radio Technologies," in *2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)* Dublin, Ireland, 2007, pp. 633-648.
- [3] R. Roy, "Radios Get Smart," *IEEE Spectrum Magazine*, vol. 44, pp. 46-50, 2007.
- [4] B. Alfonsi, "Fine-Tuning 'Smart' Radios," *IEEE Intelligent Systems*, vol. 21, pp. 6-7, 2006.
- [5] F. Ge, Q. Chen, Y. Wang, T. W. Rondeau, B. Le, and C. W. Bostian, "Cognitive Radio: From Spectrum Sharing to Adaptive Learning and Reconfiguration," in *IEEE Aerospace Conference*, Big Sky Montana, MT, 2008.
- [6] M. D. Silvius, F. Ge, A. Young, A. B. MacKenzie, and C. W. Bostian, "Smart Radio: Spectrum Access for First Responders," in *Society of Optical Engineering (SPIE): Wireless Sensing and Processing III*, Orlando, FL, March 17-18, 2008.
- [7] M. D. Silvius, A. B. MacKenzie, and C. W. Bostian, "A Survey of Neighbor Discovery Protocols in Modern Wireless Systems and their Application to the Design and Implementation of 'Rendezvous' Algorithms in Smart and Cognitive Radios," in *Virginia Tech Symposium on Wireless Personal Communications*, Blacksburg, VA, June 6-7, 2007.
- [8] GNU Radio, "<http://www.gnu.org/software/gnuradio/>," 2008.
- [9] Ettus Research LLC, "<http://www.ettus.com/>," 2008.
- [10] FCC 04-167, "Second Report and Order: Promoting Efficient Use of Spectrum through Elimination of Barriers to the Development of Secondary Markets," September 2, 2004.
- [11] FCC 06-181, "Implementing a Nationwide, Broadband, Interoperable Public Safety Network in the 700 MHz Band," <http://www.fcc.gov/pshs/spectrum/700mhz/>, December 20, 2006.
- [12] M. D. Silvius, Y. Shi, A. B. MacKenzie, and C. W. Bostian, "Channel Change: A Dynamic Spectrum Access Protocol for Spectrum Sharing in Smart Radios, Virginia Tech Intellectual Property (VTIP) No. 08-017," January 2008.
- [13] SDR Forum, "<http://www.sdrforum.org/>," 2009.

- [14] B. A. Fette, *Cognitive Radio Technology*. Boston: Newnes/Elsevier, 2006.
- [15] J. H. Reed, *Software Radio: A Modern Approach to Radio Engineering*. Upper Saddle River, NJ: Prentice Hall, 2002.
- [16] J. Mitola, "Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio," in *Ph.D. Dissertation* Royal Institute of Technology (KTH), Kista, Sweden, 2000.
- [17] J. Mitola, III and G. Q. Maguire, Jr., "Cognitive Radio: Making Software Radios More Personal," *IEEE Personal Communications*, vol. 6, pp. 13-18, 1999.
- [18] J. Mitola, III, "Cognitive Radio for Flexible Mobile Multimedia Communications," in *IEEE International Workshop on Mobile Multimedia Communications (MoMuC)*, San Diego, CA, 1999, pp. 3-10.
- [19] S. Haykin, "Cognitive Radio: Brain-Empowered Wireless Communications," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 201-220, 2005.
- [20] C. J. Rieser, "Biologically Inspired Cognitive Radio Engine Model Utilizing Distributed Genetic Algorithms for Secure and Robust Wireless Communications and Networks," in *Ph.D. Dissertation, Electrical And Computer Engineering* Virginia Polytechnic Institute and State University, Blacksburg, VA, 2004.
- [21] T. W. Rondeau, "Application of Artificial Intelligence to Wireless Communications," in *Ph.D. Dissertation, Electrical And Computer Engineering* Virginia Polytechnic Institute and State University, Blacksburg, VA, 2007.
- [22] E. Blossom, "Exploring GNU Radio," <http://www.gnu.org/software/gnuradio/doc/exploring-gnuradio.html>, November 2004.
- [23] B. Le, "Building a Cognitive Radio - From Architecture Definition to Prototype Implementation," in *Ph.D. Dissertation, Electrical And Computer Engineering* Virginia Polytechnic Institute and State University, Blacksburg, VA, 2007.
- [24] B. Le, P. Garcia, Q. Chen, B. Li, F. Ge, M. Y. ElNainay, T. W. Rondeau, and C. W. Bostian, "A Public Safety Cognitive Radio Node System," in *Software Defined Radio Forum Technical Conference*, Denver, CO, 2007.
- [25] M. McHenry, E. Livsics, N. Thao, and N. Majumdar, "XG Dynamic Spectrum Sharing Field Test Results," in *2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, Dublin, Ireland, 2007, pp. 676-684.
- [26] DARPA, "Wireless Networks after Next (WNaN) Program," <http://www.darpa.mil/sto/solicitations/WNaN/>, 2006.

- [27] BBN Technologies, "Policy-based Information-centric Reliable Ad hoc Network (PIRANA), Wireless Network After Next Program (WNaN), sponsored by DARPA-STO," 2008.
- [28] D. Scaperoth, B. Le, T. W. Rondeau, D. Maldonado, C. W. Bostian, and S. Harrison, "Cognitive Radio Platform Development for Interoperability," in *Military Communications (MILCOMM)* Washington, DC, 2006.
- [29] B. Le, T. W. Rondeau, and C. W. Bostian, "General Radio Interface Between Cognitive Algorithms and Reconfigurable Radio Platforms," in *Software Defined Radio Forum Technical Conference* Denver, CO, 2007.
- [30] SDR Forum Public Safety Special Interest Group, "Use Cases for Cognitive Applications in Public Safety Communications Systems Volume 1: Review of the 7 July Bombing of the London Underground," <http://www.radiochallenge.org>, vol. Working Document SDRF-07-P-0019-V0.0.3, Version 1.0.0, September 18, 2007.
- [31] MITRE, "Mobile Mesh: Providing Solutions for Mobile Ad-hoc Networking," http://www.mitre.org/work/tech_transfer/mobilemesh/, 2008.
- [32] Ubuntu 7.10, "<http://releases.ubuntu.com/7.10/>," 2008.
- [33] JPERF 2.0.0, "Graphical Frontend for IPERF Written in Java," <http://code.google.com/p/xjperf/downloads/list>, 2008.
- [34] GNU Radio Build Guide, "<http://gnuradio.org/trac/wiki/BuildGuide>," 2008.
- [35] Netgear, "WGT624v4 Wireless Firewall Router Setup Manual," ftp://downloads.netgear.com/files/WGT624v4_SM_30Apr07.pdf, 2008.
- [36] J. Heaps, "Software Defined Radios Help Agencies Communicate (<http://www.policeone.com/police-products/communications/radios/articles/1809008-Software-defined-radios-help-agencies-communicate>)," PoliceOne.com, April 10, 2009.
- [37] Q. Chen and C. W. Bostian, "Cognitive Gateway Design to Promote Universal Interoperability," in *Software Defined Radio Forum Technical Conference*, Washington, DC, October 26-30, 2008.
- [38] Y. Wang and C. W. Bostian, "Demonstration: Dynamic Cellular Cognitive System (DCCS)," in *IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, Chicago, IL, October 14-17, 2008.
- [39] Y. Wang, Q. Chen, B. Le, and C. W. Bostian, "Universal Synchronization Design for Cognitive Radios," in *Software Defined Radio Forum Technical Conference*, Denver, CO, November 5-9, 2007.

- [40] Q. Chen, Y. Wang, and C. W. Bostian, "Universal Classifier Synchronizer Demodulator," in *IEEE International Performance, Computing and Communications Conference (IPCCC)*, Austin, TX, December 7-9, 2008, pp. 366-371.
- [41] J. H. Schiller, *Mobile Communications*, 2nd ed. London ; Boston: Addison-Wesley, 2003.
- [42] J. Geier, "802.11 MAC Layer Defined, <http://www.wi-fiplanet.com/tutorials/article.php/1216351>," Wi-Fi Planet, June 4, 2002.
- [43] ANSI/IEEE Std 802.11, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, <http://standards.ieee.org/getieee802/802.11.html>," June 12, 2003.
- [44] P. Houze, S. Ben Jemaa, and P. Cordier, "Common Pilot Channel for Network Selection," in *IEEE 63rd Vehicular Technology Conference (VTC)*, Melbourne, Victoria, Australia, 2006, pp. 67-71.
- [45] E. Mohyeldin, J. Luo, and R. Falk, "Architecture and Certification Methods for Common Pilot Channel," in *SDR Forum Technical Conference Orlando, FL: SDR Forum*, 2006.
- [46] S. Kuppa, M. Thoppian, S. Krishnamurthy, S. Venkatesan, R. Chandrasekaran, N. Mittal, and R. Prakash, "Time-Efficient Layer-2 Auto-Configuration for Cognitive Radios," in *The International Association of Science and Technology for Development (IASTED) Conference on Parallel and Distributed Computing and Systems (PDCS)*, Phoenix, AZ, USA, 2005.
- [47] S. Krishnamurthy, M. Thoppian, S. Venkatesan, and R. Prakash, "Control Channel Based MAC-layer Configuration, Routing and Situation Awareness for Cognitive Radio Networks," in *Military Communications Conference (MILCOM)*, Atlantic City, NJ, 2005, pp. 455-460 Vol. 1.
- [48] M. J. McGlynn and S. A. Borbash, "Birthday Protocols for Low Energy Deployment and Flexible Neighbor Discovery in Ad Hoc Wireless Networks," in *Association for Computing Machinery (ACM), International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Long Beach, CA, 2001, pp. 137-145.
- [49] G. Alonso, E. Kranakis, R. Wattenhofer, and P. Widmayer, "Probabilistic Protocols for Node Discovery in Ad-Hoc, Single Broadcast Channel Networks," in *Proceedings of International Parallel and Distributed Processing Symposium*, Nice, France, April 2003.
- [50] G. Alonso, E. Kranakis, C. Sawchuk, R. Wattenhofer, and P. Widmayer, "Probabilistic Protocols for Node Discovery in Ad Hoc Multi-Channel Broadcast Networks," in *Proceedings of Second International Conference Ad-Hoc, Mobile, and Wireless Networks (ADHOC-NOW)*, Berlin, Germany, 2003, pp. 104-115.
- [51] D. Angelosante, E. Biglieri, and M. Lops, "A Simple Algorithm for Neighbor Discovery

- in Wireless Networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007, pp. III-169-III-172.
- [52] L. Galluccio, A. Leonardi, G. Morabito, and S. Palazzo, "Tradeoff between Energy-Efficiency and Timeliness of Neighbor Discovery in Self-Organizing Ad Hoc and Sensor Networks," in *IEEE Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS)*, Waikoloa, HI, 2005, pp. 286a-286a.
- [53] L. Galluccio, G. Morabito, and S. Palazzo, "Analytical Evaluation of a Tradeoff between Energy Efficiency and Responsiveness of Neighbor Discovery in Self-Organizing Ad hoc Networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 1167-1182, Sept. 2004 2004.
- [54] Bluetooth Core Specification v2.0, "<http://www.bluetooth.com>," November 4, 2004.
- [55] B. S. Peterson, R. O. Baldwin, and J. P. Kharoufeh, "Bluetooth Inquiry Time Characterization and Selection," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 1173-1187, 2006.
- [56] F. Siegemund and M. Rohs, "Rendezvous Layer Protocols for Bluetooth-Enabled Smart Devices," *Personal and Ubiquitous Computing (PUC)*, vol. 7, pp. 91-101, 2003.
- [57] T. Salonidis, "Distributed Topology Organization and Transmission Scheduling in Wireless Ad Hoc Networks," in *Ph.D. Dissertation, Electrical And Computer Engineering* University of Maryland, College Park, MD, 2004.
- [58] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. LaMaire, "Distributed Topology Construction of Bluetooth Wireless Personal Area Networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 633-643, 2005.
- [59] C. Law, A. K. Metha, and K.-Y. Siu, "A Bluetooth Scatternet Formation Algorithm," in *IEEE Global Telecommunications Conference (GLOBECOM)*, 2001, pp. 2864-2869 vol.5.
- [60] K. Balachandran and J. H. Kang, "Neighbor Discovery with Dynamic Spectrum Access in Adhoc Networks," in *IEEE 63d Vehicular Technology Conference (VTC)*. vol. 2, 2006, pp. 512-517.
- [61] E. B. Hamida, G. Chelius, and E. Fleury, "Revisiting Neighbor Discovery with Interferences Consideration," in *Proceedings of the 3rd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous networks (PE-WASUN)*, Terromolinos, Spain, 2006, pp. 74-81.
- [62] S. Fang, S. M. Berber, and A. K. Swain, "Analysis of Neighbor Discovery Protocols for Energy Distribution Estimations in Wireless Sensor Networks," in *IEEE International Conference on Communications (ICC)*, 2008, pp. 4386-4390.

- [63] D. Yang, J. Shin, J. Kim, and C. Kim, "An Energy-Optimal Scheme for Neighbor Discovery in Opportunistic Networking," in *6th IEEE Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, 2009, pp. 1-2.
- [64] S. A. Borbash, A. Ephremides, and M. J. McGlynn, "An Asynchronous Neighbor Discovery Algorithm for Wireless Sensor Networks," *Ad Hoc Networks*, vol. 5, pp. 998-1016, 2007.
- [65] L. A. DaSilva and I. Guerreiro, "Sequence-Based Rendezvous for Dynamic Spectrum Access," in *3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, Chicago, IL, 2008, pp. 1-7.
- [66] C. J. L. Arachchige, S. Venkatesan, and N. Mittal, "An Asynchronous Neighbor Discovery Algorithm for Cognitive Radio Networks," in *3rd IEEE Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, Chicago, IL, 2008, pp. 1-5.
- [67] B. Horine and D. Turgut, "Link Rendezvous Protocol for Cognitive Radio Networks," in *2nd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, Dublin, Ireland, 2007, pp. 444-447.
- [68] B. Horine and D. Turgut, "Performance Analysis of Link Rendezvous Protocol for Cognitive Radio Networks," in *2nd International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM)*, Orlando, FL, 2007, pp. 503-507.
- [69] M. D. Silvius, A. B. MacKenzie, and C. W. Bostian, "Rendezvous MAC Protocols for use in Cognitive Radio Networks (Accepted)," in *IEEE Military Communications Conference (MILCOM)*, Boston, MA, October 18-21, 2009.
- [70] D. A. Roberson, C. S. Hood, J. L. LoCicero, and J. T. MacDonald, "Spectral Occupancy and Interference Studies in support of Cognitive Radio Technology Deployment," in *IEEE Workshop on Networking Technologies for Software Defined Radio Networks*, 2006, pp. 26-35.
- [71] B. Le, T. W. Rondeau, and C. W. Bostian, "Cognitive Radio Realities," in *Wireless Communications and Mobile Computing*, vol. 7: John Wiley and Sons Ltd, Chichester, West Sussex, UK, 2007, pp. 1037-1048.
- [72] M. D. Silvius, R. Rangnekar, A. B. MacKenzie, and C. W. Bostian, "The Smart Radio Channel Change Protocol: A Primary User Avoidance Technique for Dynamic Spectrum Sharing Cognitive Radios to Facilitate Co-Existence in Wireless Communication Networks," in *4th International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM)*, Hannover, Germany, June 22-24, 2009.
- [73] P. Bahl, R. Chandra, and J. Dunagan, "SSCH: Slotted Seeded Channel Hopping for

- Capacity Improvement in IEEE 802.11 Ad-Hoc Wireless Networks," in *Proceedings of the Annual International Conference on Mobile Computing and Networking (MOBICOM)*, Philadelphia, PA, 2004, pp. 216-230.
- [74] D. Shen, "GNU Radio Tutorial, <http://www.nd.edu/~jnl/sdr/docs>," 2005.
- [75] M. Ettus, "USRP User's and Developer's Guide, <http://home.ettus.com>," 2005.
- [76] T. Schmid, O. Sekkat, and M. B. Srivastava, "An Experimental Study of Network Performance Impact of Increased Latency in Software Defined Radios," in *International Conference on Mobile Computing and Networking (MobiCom), Proceedings of the 2nd ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH)* Montreal, QC, Canada, September 10, 2007, pp. 59-66.
- [77] J. Mo, H. S. W. So, and J. Walrand, "Comparison of Multichannel MAC Protocols," *IEEE Transactions on Mobile Computing (TMC)*, vol. 7, pp. 50-65, 2008.
- [78] R. Maheshwari, H. Gupta, and S. R. Das, "Multichannel MAC Protocols for Wireless Networks," in *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON)*, Reston, VA, 2006, pp. 393-401.
- [79] J. So and N. Vaidya, "Multi-Channel MAC for Ad Hoc Networks: Handling Multi-Channel Hidden Terminals Using a Single Transceiver," in *Proceedings of the International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Tokyo, Japan, 2004, pp. 222-233.
- [80] J. Chen, S.-T. Sheu, and C.-A. Yang, "A New Multichannel Access Protocol for IEEE 802.11 Ad Hoc Wireless LANs," in *14th IEEE Proceedings on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Beijing, China, 2003, pp. 2291-2296 vol.3.
- [81] J. Bray and C. F. Sturman, *Bluetooth: Connect without Cables*. Upper Saddle River, N.J.: Prentice Hall PTR, 2001.
- [82] J. Xiangpeng and D. Raychaudhuri, "A Spectrum Etiquette Protocol for Efficient Coordination of Radio Devices in Unlicensed Bands," in *14th IEEE Personal, Indoor and Mobile Radio Communications (PIMRC)*, Beijing, China, 2003, pp. 172-176 Vol.1.
- [83] X. Jing and D. Raychaudhuri, "Spectrum Co-existence of IEEE 802.11b and 802.16a Networks Using the CSCC Etiquette Protocol," in *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN)*, Baltimore, MD, 2005, pp. 243-250.
- [84] OMNeT++ Discrete Event Simulation System, "<http://www.omnetpp.org>," 2008.
- [85] Wikipedia, "<http://en.wikipedia.org>," 2008.

- [86] J. F. Kurose and K. W. Ross, *Computer networking: A Top-down Approach Featuring the Internet*, 3rd ed. Boston: Pearson/Addison Wesley, 2005.
- [87] IPERF 2.0.2, "Internet Protocol Bandwidth Measuring Tool, <http://iperf.sourceforge.net/>," 2008.

Appendix

Scripts

```
sudo /etc/dbus-1/event.d/25NetworkManager stop # Turn off GUI Network Manager
sudo ifconfig eth0 down # Bring wired interface down
sudo ifconfig eth0 192.168.5.x # Set IP address; set last digit to 2, 3, or 4
sudo ifconfig eth0 up # Bring wired interface down
sudo route add default gw 192.168.5.1 dev eth0 # Set the default gateway
sudo "nmserv 192.168.5.1" > /etc/resolv.conf # Set the default DNS server
```

Script 1. Setting up the 802.3 wired network.

```
sudo ifconfig wlan2 down # Bring wireless interface down
sudo iwconfig wlan2 mode ad-hoc essid "testbed" channel 3 key off # Setup wireless parameters
sudo ifconfig wlan2 192.168.10.5 # Set IP Address
sudo ifconfig wlan2 up # Bring interface up

# Routing setup example for Node 1 (192.168.10.2)
sudo route add -host 192.168.10.5 gw 192.168.10.3 # Route packets for .10.5 to .10.3
sudo route add -host 192.168.10.4 gw 192.168.10.3 # Route packets for .10.4 to .10.3
sudo sysctl -w net.ipv4.ip_forward=1 # Enable IP Forwarding on the machine
more /proc/sys/net/ipv4/ip_forward # To check if IP Forwarding is enabled
```

Script 2. Setting up a 802.11 WiFi ad-hoc network.

```
$ route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use I face
192.168.10.5 192.168.10.3 255.255.255.255 UGH 0 0 0 wlan2
192.168.10.4 192.168.10.3 255.255.255.255 UGH 0 0 0 wlan2
192.168.5.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0
192.168.10.0 0.0.0.0 255.255.255.0 U 0 0 0 wlan2
0.0.0.0 192.168.5.1 0.0.0.0 UG 100 0 0 eth0
```

Figure A.1. Static routing table in Linux; example from Node #1.

```
sudo modprobe bnep # Load the bnep module
sudo brctl addbr pan # Create a bridge called 'pan'
sudo brctl setfd pan 0 # Set bridge forward delay to 0
sudo brctl stp pan off # Set STP off for this bridge
sudo brctl show # Show all bridges

sudo ifconfig pan 192.168.20.2 netmask 255.255.255.0 # Assign IP address to the bridge
sudo ifconfig pan up # Bring the bridge interface up
sudo hcitool scan -flush # Scan for new Bluetooth devices
XX:XX:XX:XX:C9:A4 cwt-sdr-1 # Node 1. 192.168.20.2
XX:XX:XX:XX:85:61 cwt-sdr-2 # Node 2. 192.168.20.3
XX:XX:XX:XX:BC:D7 cwt-sdr-3 # Node 3. 192.168.20.4
XX:XX:XX:XX:9A:7E cwt-sdr-5 # Node 4. 192.168.20.5

sudo pand -listen # Listen for Bluetooth connections
sudo pand --connect XX:XX:XX:XX:85:61 # Connect to a MAC address
pand -show # Show current list of Bluetooth connections
sudo ifconfig bnep0 0.0.0.0 # Set bnep IP address to 0.0.0.0
sudo ifconfig bnep0 up # Bring bnep up
sudo brctl addif pan bnep0 # Add bnep to the bridge
```

Script 3. Setting up a Bluetooth network.

```
cd <gnuradio_home>/gnuradio-examples/python/digital

#Start tunnel.py script with parameters
sudo /work/gnuradio/gnuradio-examples/python/digital/tunnel.py -m dbpsk -f 462.6125e6 -v -r 100e3

# Sample configuration for Node 1
sudo ifconfig gr0 down                # Bring down GNU Radio interface gr0
sudo ifconfig gr0 192.168.40.2        # Assign IP to the interface
sudo ifconfig gr0 up                  # Bring the interface up
sudo sysctl -w net.ipv4.ip_forward=1  # Enable ip forwarding
sudo route add -host 192.168.40.4 gw 192.168.40.3 # Route packets for .10.4 to .10.3
sudo route add -host 192.168.40.5 gw 192.168.40.3 # Route packets for .10.5 to .10.3
```

Script 4. Setting up a GNU Radio/USRP ad-hoc network.

```
iperf -s -P 0 -i 1 -p 5001 -f k      # Start the iperf server (-s)
                                     # Accept connections -P, 0 = Unlimited connections
                                     # Report interval -i, 1 second
                                     # Port number -p, default iperf port 5001
                                     # Data format -k, display in Kilobytes (k)
                                     # Accept TCP (default mode) data

iperf -s -u -P 0 -i 1 -p 5001 -f k   # Server, accept UDP (-u) data

iperf -c 192.168.5.2 -P 1 -i 1 -p 5001 -f k -t 120 # Start the iperf client and begin sending data (-c)
                                     # Send data to server at 192.168.5.2
                                     # Number of parallel streams (-P), set to 1
                                     # Report interval -i, 1 second
                                     # Port number -p, default iperf port 5001
                                     # Data format -k, display in Kilobytes (k)
                                     # Time to run test -t, 120 seconds
                                     # Send TCP (default mode) data

iperf -c 192.168.5.2 -u -P 1 -i 1 -p 5001 -f k -b 1M -t 30 -T 1 # Time to Live -T, 1 hop
                                                                    # Client, sends UDP (-u) data
```

Script 5. IPerf commands example