Adapting Sensing and Transmission Times to Improve Secondary User Throughput in Cognitive Radios Ad Hoc Networks

by

Namrata Arun Bapat

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved March 2012 by the
Graduate Supervisory Committee:

Violet R. Syrotiuk, Chair
Gail-Joon Ahn
Guoliang Xue

ARIZONA STATE UNIVERSITY

May 2012

ABSTRACT

Cognitive Radios (CR) are designed to dynamically reconfigure their transmission and/or reception parameters to utilize the bandwidth efficiently. With a rapidly fluctuating radio environment, spectrum management becomes crucial for cognitive radios. In a Cognitive Radio Ad Hoc Network (CRAHN) setting, the sensing and transmission times of the cognitive radio play a more important role because of the decentralized nature of the network. They have a direct impact on the throughput. Due to the trade-off between throughput and the sensing time, finding optimal values for sensing time and transmission time is difficult. In this thesis, a method is proposed to improve the throughput of a CRAHN by dynamically changing the sensing and transmission times. To simulate the CRAHN setting, *ns-2*, the network simulator with an extension for CRAHN is used. The CRAHN extension module implements the required Primary User (PU) and Secondary User (SU) and other CR functionalities to simulate a realistic CRAHN scenario. First, this work presents a detailed analysis of various CR parameters, their interactions, their individual contributions to the throughput to understand how they affect the transmissions in the network. Based on the results of this analysis, changes to the system model in the CRAHN extension are proposed. Instantaneous throughput of the network is introduced in the new model, which helps to determine how the parameters should adapt based on the current throughput. Along with instantaneous throughput, checks are done for interference with the PUs and their transmission power, before modifying these CR parameters. Simulation results demonstrate that the throughput of the CRAHN with the adaptive sensing and transmission times is significantly higher as compared to that of non-adaptive parameters.

DEDICATION

To my parents

for their support and faith in me

and for inspiring me every day of my life.

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor Dr. Violet R. Syrotiuk for guiding me through this undertaking. Her constant support and patience was vital to me. I thank her for encouraging me to take up this project and directing me towards its successful completion. I truly appreciate her faith in me.

I am very grateful to Dr. Marco DeFelice for sharing the CRAHN code with me and taking the time out to answer my questions elaborately and swiftly.

I am also thankful to Dr. Douglas C. Montgomery for helping me through data analysis. His valuable suggestions and comments facilitated better a understanding of design of experiments.

I would like to sincerely thank my committee members Dr. Gail-Joon Ahn and Dr. Guoliang Xue for their support.

I am also thankful to my friends and family who stood by me through this entire journey here at ASU.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Need for Cognitive Radios

With the recent advances in wireless technology and rapid deployment of wireless networks, the industrial, scientific and medical (ISM) band has faced a considerable amount of saturation. The ISM band is also shared by license free communication devices. Due to all these factors, the band is heavily congested. On the other hand, licensed bands do not face this problem. In fact the bandwidth allocated to licensed users is used intermittently. This creates what are known as *spectrum holes* [10]. The spectrum holes are unused spectrum space at that point in time. This discrepancy brings to light the inefficient spectrum allocation techniques used for both licensed and unlicensed radio frequencies. The need arises for a new technology to overcome this problem. We need to develop new radios that can use the spectrum holes for communication and thus provide some relief to the congested bandwidth. CR technology is envisioned to solve this inefficiency problem by using the available spectrum opportunistically without interfering with the licensed users. It is a new paradigm in the wireless communication networks that promises reliable communication by sharing the spectrum effectively. It introduces a flexible way to optimally use the bandwidth.

1.2 Cognitive Radios Overview

A Cognitive Radio is a radio that changes its transmission and reception parameters based on the radio environment [3]. Therefore, the main function of a Cognitive Radio is to exhibit reconfigurability. Along with dynamic reconfiguration, CR node should also learn about the surrounding radio environment. The cognitive aspect of the radio is to sense the surrounding by monitoring the spectrum as well as the licensed users and make informed decisions based on the information it has collected. In a CR network there are two types of users: Primary Users (PUs) and Secondary Users

(SUs). A PU is a licensed user while an SU is an unlicensed user that has CR capabilities. The SU has to detect the spectrum holes in the licensed band. When it detects a spectrum hole, it is allowed to use the band until the licensed user comes back. The SU can either vacate the spectrum or stop transmission. It should not interfere with the licensed user's transmission. The aim of a CR node is to ultimately use the best available spectrum by reconfiguring itself based on the channel characteristics.

The use of CR technology in an ad hoc wireless network setting is known as a Cognitive Radio Ad Hoc Network (CRAHN).A CRAHN is very different than the traditional wireless ad hoc network. Like ad hoc networks, the nodes in CRAHN are self organising and do not need a backbone infrastructure. However, CR nodes switch continuously between different channels. They also change their parameters over the course of communication, based on the current channel they are using. Unlike multi-channel ad hoc networks where all channels are available for transmission, in a CRAHN the SU has to always for PU activity and PU interference. For these reasons, the problems faced by a CRAHN are different than other ad hoc networks and the protocols used for ad hoc networks do not necessarily apply to CRAHNs. Reliability of data transmission, quality of service, security, resource allocation are some of the issues encountered by CRAHNs.

1.3 Problem Statement

Since a CRAHN is a relatively new technology, the protocols used are not completely standardized. Most implementations of CRAHNs set the CR parameters to fixed values. As a result, the CR nodes in the implementation do no truly exhibit dynamic reconfiguration abilities. Such implementations cannot take advantage of the behaviour of the CR nodes. Some research has been done to enable the CR nodes to adapt their parameters. The key CR parameters that have a direct impact on the SU throughput are sensing time and transmission time [9]. Researchers have proposed

different methods to adapt the sensing time and transmission time of a CR node. Although, the main problem is to devise efficient spectrum sensing techniques, it is also important that transmission time be modified for better SU throughput. Values of sensing time and transmission time have to be selected carefully to maximize the throughput. We address this issue of adapting both, sensing time and transmission time to boost the performance of the CR nodes. We approach the problem by performing extensive data analysis and exploiting an interaction between sensing time and transmission time of the SU throughput.

1.4 Contribution

The proposed technique is a heuristic algorithm to dynamically change the CR node parameters. This technique is implemented at the Link Layer of the protocol stack. Prior to implementing the algorithm, data analysis was conducted. It included designing an experiment, observing various plots that describe the distribution of the sample data, and identifying the dependencies among input factors. The results obtained from this analysis formed the basis of the implementation. A performance evaluation of the results was carried out to verify the implementation and behaviour of the adaptive method. We were able to conclude that the adaptive model complies to the data analysis and delivers better performance.

1.5 Document Organization

The remainder of the document is structured as follows. Chapter 2 details the working of CRAHNs. We look at the related work for existing techniques to implement adaptive sensing and transmission time in Chapter 3. The CRAHN simulator model for *ns-2* is explained in detail in Chapter 4. Design of experiment, generating data model, data analysis and observations of the data model are described in Chapter 5. Chapter 6 explains in detail our proposed adaptive algorithm based on the findings of data analysis. The performance evaluation of the implementation is described in Chapter 7. And finally, Chapter 8 concludes the thesis.

Chapter 2

BACKGROUND

The system model of a CRAHN is vastly different from other wireless communication networks. CR nodes have to employ some specific functions to achieve CR behaviour. Spectrum management is an integral part of the CR implementation. The radio interfaces also have to be such that they support all the CR functions.

2.1 Spectrum Management in CRAHNs

Spectrum Management in CRAHNs is divided into four major functions Spectrum Sensing, Spectrum Decision, Spectrum Sharing and Spectrum Mobility. Figure 2.1 shows the inter-workings of the parts of Spectrum Management.

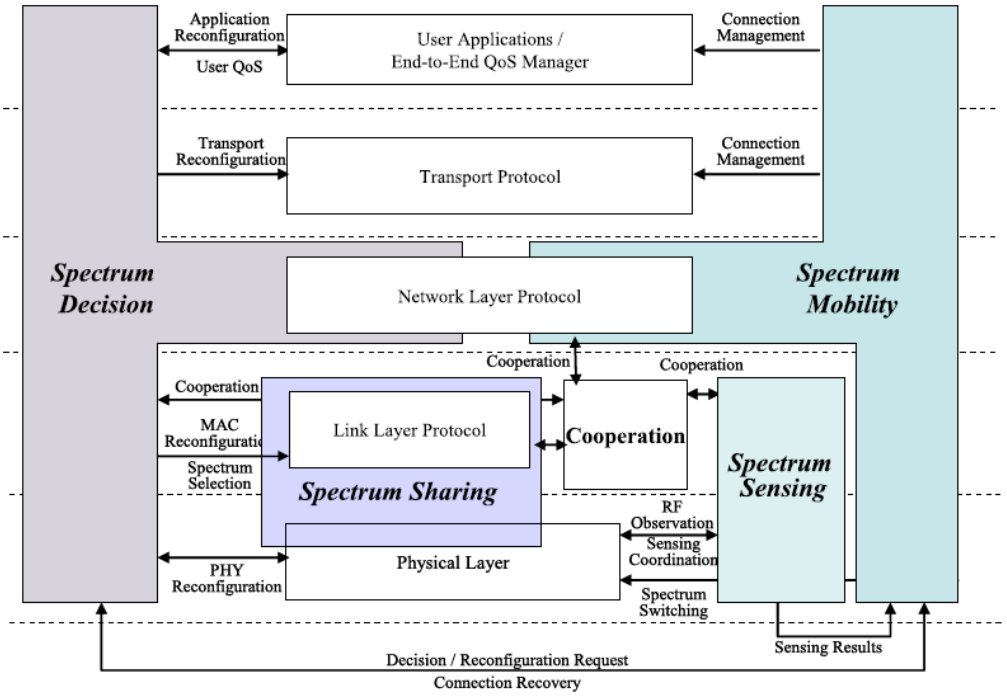

Figure 2.1: CRAHN Spectrum Management [3]

Spectrum Sensing

Spectrum sensing is performed to detect spectrum holes and PU activity. CR nodes should not interfere with the PU transmission and therefore, have to exercise sensing control. Spectrum sensing has to be sensitive to detect the weakest PU signal and

determine any PU activity. PU detection can be performed using techniques such as transmitter detection, receiver detection, and interference temperature management [3]. PU transmitter detection involves sensing for the weakest PU signals. A PU signal can be detected using matched filter detection, energy detection or feature detection [3]. The methods for transmitter detection concentrate on checking for noise and maximizing signal to noise ratio (SNR). PU receiver detection looks for any PU activity by checking if any PU is receiving data within the transmission range of the CR node. Interference temperature management aims at measuring the transmission threshold for CR nodes. It accounts for the collective radio frequency (RF) energy from all transmitters and decides on a maximum limit of energy that the PU will allow without any interference. This threshold is called as the interference temperature limit [3].

Along with various PU detection techniques, CR nodes also have to decide other aspects of spectrum sensing. After a PU detection technique is put in place, the CR node should choose a sensing time interval. Determining how long and how frequently should a CR node sense a spectrum is crucial. It is also important to find out how quickly does the CR node detect PU activity. A low sensing time leads to PU misdetection, thus causing interference, whereas higher sensing time results in less data transmission and thus affects the throughput.

Spectrum Decision

Spectrum Decision deals with deciding the next course of action after detecting PU activity. The main functions of Spectrum Decision are selecting a new spectrum, spectrum characterisation, and reconfiguration [3]. It may happen that when the CR node senses, it finds multiple channels that are available. The characteristics of each channel can be different. Based on the channel, the CR has to reconfigure its parameters. Also, in a multi hop environment, the channel characteristics play an important role in end-to-end delivery. Therefore, spectrum decision is critical.

The CR node has to determine the spectrum characteristics. Based on these characteristics and the available channels, a CR node has to decide the best channel it can switch to for reliable communication. At the same time, it also has to account for reconfiguration of hardware or software depending on the channel switch. Spectrum decision is closely coupled with routing protocols [3]. When selecting a new channel, the CR node has to consider QoS requirements, data rate, bandwidth, packet error rate and other parameters. It should also be able to establish the time required to reconfigure itself.

Spectrum Sharing

Wireless channels are shared between multiple PUs and CRs. A CR node has to share the spectrum with not only the CRs but also PUs. When a CR detects PU activity, it need not always leave the channel. It can either stop transmitting and wait for the PU to complete its communication or it can share the spectrum with the PU, provided its transmission power does not interfere with that of the PU. At the same time, CR nodes have to coordinate among themselves and share the spectrum. CR nodes have to adopt some sharing policies for equal distribution of the spectrum among CR nodes.

Spectrum sharing shares some functionalities with spectrum sensing such as resource allocation and spectrum access [3]. Spectrum access includes many MAC functionalities. Spectrum access can be implemented using time sharing methods, random access or even on-demand access [3]. Since it deals more with the link layer, it has to account for collision detection and mitigation plans. If spectrum sharing is done on the principle of time sharing then time synchronisation is required. Hybrids, that implement multiple methods of spectrum sharing have also been proposed [3]. Even with all these methods, coordinating with multiple CRs in a dynamic and fluctuating environment can be a challenge. To share spectrum, CR nodes have to communicate with their neighbours. Without a common control channel, this task becomes complex.

Spectrum Mobility

When a CR node decides to switch channels, it has to perform a spectrum handoff. A spectrum handoff releases the current connection and transfers it to a new channel. A spectrum handoff occurs when a CR nodes detect PU activity or when the QoS requirements are not met [3]. A spectrum handoff can be proactive or reactive [3]. In a proactive spectrum handoff, the CR has to have the knowledge of PU activity before switching channels. This may be based on a prediction of PU activity; the CR can switch channels before a PU starts transmitting. A simpler technique is a reactive spectrum handoff, where the CR node switches channel only when it detects a PU.

Spectrum handoff is a complex process. The CR node has to first stop the current transmission and save its state. It then has to switch to the next channel selected by the spectrum decision module. The receiving CR node also has to agree to the new channel that has been selected. Once it has been agreed upon, the CR node releases the current channel and re-establishes communication on the new channel. When developing techniques for spectrum handoff, delay in switching channels has to be taken into consideration. Connection management and spectrum handoff delay are important to spectrum mobility. They have to be fast and seamless.

A regular cycle for a CR node starts with sensing for available spectrum. It then switches to a channel in the spectrum and establishes connection with another CR. Once a connection is established it can transmit data. It has to repeat sensing and transmission cycles over the duration of the communication. During a sensing cycle, a CR node cannot transmit any data. On detecting any PU activity, the CR node can either share the channel or it can search for any other available channel and switch to it. During spectrum handoff, the CR node cannot transmit any data.

2.2 Common Control Channel

Given the complex nature of CRAHNs, it is important that CR nodes share information with one another. For this purpose, researchers have proposed a common control channel (CCC) for CRAHNs [3]. A CCC can help streamline CR functions and provide an effective way for the transmission of control messages. A CCC can facilitate neighbour discovery [3]. It can help with transmission coordination and synchronisation. Above all, it provides a convenient way to exchange information among CR nodes. It can aid in cooperative spectrum sensing, where multiple CR nodes coordinate and sense the spectrum.

A CCC is divided into an In-band CCC and an Out-of-band CCC [3]. The In-band CCC uses same channel for data transmission and control signalling. This way it does not need another channel to send control messages. But this method can have a substantial impact on the throughput of the CR nodes [3]. Since the same channel is being used for data and signalling, the time allocated for data transmission is reduced. To avoid this problem, the Out-of-band CCC can be used. It uses a dedicated channel to send control messages. This channel can be used by all the CR nodes in the network and it is therefore not affected by PU activity. The Out-of-band CCC boosts the throughput because CR nodes can use data channels for longer durations. However, a separate radio interface needs to be used for the Out-of-band CCC. Depending on the CRAHN implementation, either technique can be used for the CCC.

2.3 Cross-Layer Design

In the traditional layered network architecture, inter layer communication is done using procedure calls [17]. New wireless technologies call for a cross layer design that accommodates their constraints and enables adaptation. A cross layer design exploits the layers of the protocol stack by redefining layer boundaries and creating

adaptability between layers. The layers of the traditional protocol stack are rigid and are too limiting for upcoming wireless technologies such as CRAHNs. A cross layer design can help resolve issues faced by the existing protocol stack such as the need for higher layer protocol to use physical layer details. A protocol stack that responds to the environment and network conditions will be very useful for cognitive networks [17].

Many models have been proposed for cross layer design. They include creating new interfaces, merging adjacent layers and better information sharing between layers [17]. Other proposals include sharing a global database across all layers and developing completely new abstractions as compared to the existing layered design. A global database for all layers sounds promising for CRAHNs.

Given the complexity of a CRAHN, several methods have been proposed to optimize various CR parameters using different techniques. Due to the lack of standardized protocols and trade-off's existing between different CR parameters, optimizing these parameters to improve SU throughput becomes difficult. In the next Chapter, we look at the various methods proposed for optimal spectrum sensing, and adaptive sensing and transmission times. These methods suggest several different ways of CRAHN optimization and modification.

Chapter 3

RELATED WORK

Much research has been conducted in the general field of CRAHNs. It is a new technology and efforts are being made to come up with standardized protocols and network stack for CRAHN. While many CRAHN implementations suggest using static CR parameters for convenience, much work has been done to implement adaptive CR parameters. Before describing the adaptive techniques, we study the methods that have been proposed for an optimal spectrum sensing framework. Some works suggest focussing on the interference avoidance problem [13] to develop new sensing framework. This work discusses optimizing the sensing parameter which leads to an optimal observation time. It helps maximize the sensing efficiency while keeping the constraint on PU interference under control. It also suggests using efficient and fast spectrum selection and scheduling methods. It is important to select the best spectrum bands for sensing such that it maximizes the sensing capacity. This work also focuses on sensing efficiency problem.

In CRAHNs, throughput vs sensing time trade-offs are present. A shorter sensing time implies more time for data transmission. This results in higher SU throughput. But a PU misdetection can occur easily during a shorter sensing time which can lead to PU interference. If the sensing time is longer then the probability of PU misdetection decreases. But it affects the throughput negatively because a CR node now senses for a longer time and transmits for a shorter time. Given the tradeoff, sensing time design has to be integrated with the CR design [3]. Due the intricacies in the sensing time design, much research has focussed on efficient spectrum sensing techniques [4]. Various parameters such as PU detection, cooperation, and sensing control [3], have to be considered to develop new spectrum sensing algorithms. Spectrum sensing becomes very complex because detecting some types of signals is easier than others. Due to this, the RF front-end must have the requirements for

sensitivities of different signals [4]. The digital signal processing techniques that are used have to implement a good sensing function.

Some research suggests techniques that implement energy detection [9] for spectrum sensing. Channel monitoring is performed by detecting the energy of a licensed user. Spectrum sensing and sharing techniques have to be implemented carefully, so as to not interfere with a PU. This method tries to optimize the sensing time, sensing periodicity, channel search and channel monitoring tasks. The underlying mechanism used is an energy detector in the form of a filter. The energy detector takes into account the interference range of the SU which is the maximum distance from a PU at which harmful interference can occur.

Another method also suggests using energy detection for spectrum sensing. But here it uses an optimal fusion rule [18] to calculate the error probability of false PU detection. This paper concludes that the optimal fusion rule is the half-voting rule. This technique calls for cooperation between the cognitive radios (SUs) within the network to perform spectrum sensing. One of the challenges faced by CR networks is a hidden terminal problem [4]. In a hidden terminal problem, the CR node is fails to detect a PU which causes interference. For this reason, a CR node should communicate with neighbouring CR nodes and coordinate spectrum sensing. It suggests that to implement a fast sensing technique for a large CR network requires fewer than the total number of CR nodes present.

Research also shows using inter sensing times [15] to reduce the sensing errors caused by misdetection. This work finds inter sensing times such that the throughput is maximized while keeping a constraint on the maximum interference with the PUs. The SU tries to measure the optimal duration for the next sensing interval. At the same time it transmits data on the available channel. These inter sensing computations performed by the SU help improve the SU throughput. Other works suggests a sensing scheme that adapts itself and decides whether to sense the channel or transmit data

based on the results of the previous sensing interval [5]. This technique uses a star topology with a master node at the centre; the other nodes are slaves. Only the master node performs channel sensing. Based on the previous sensing results, the master node decides the next course of action and orders the slave nodes to perform the required tasks.

Most research concentrates on spectrum sensing and sharing algorithms. Not many ideas have been proposed for adaptive transmission time. Since Spectrum Sensing is seen as an integral function of CR adapting transmission time is secondary. Most studies modify the transmission time based on the optimal spectrum sensing technique implemented. There are proposed techniques that determine the transmission power based on the interference temperature [11]. Though the method depends on spectrum sensing algorithms, it does suggest adaptive transmission power using modulation and coding. The proposed method adaptively selects the modulation order considering the transmission power. This work uses the IT model introduced by the Federal Communications Commission (FCC). It measures the distance between the CR node and the PU and determines the maximum transmission power it can use to transmit data and prevent interference.

In this thesis, we propose a method of dynamically changing the sensing time and transmission time, by calculating sensing time as a function of transmission time. We study the impact of this adaptive model on various combinations of sensing time and transmission time and how it affects the overall average throughput. We, first study the default CRAHN model to understand its implementation. The default CRAHN model helps us locate the implementation of the key CR functionalities. The next Chapter explains in detail the default CRAHN extension used in *ns-2*. It emphasizes on the division of tasks between different modules and how the model simulates a CRAHN setting as described in Chapter 2.

Chapter 4

THE CRAHN SIMULATOR

To perform simulations we used the *ns-2*(network simulator) [2] with an extension to support CRAHNs [8]. The CRAHN extension is implemented as a multi-radio, multi-channel system model. Each CR node has three interfaces: a control interface, a switchable interface and a receiving interface. The control interface is always tuned to the common control channel. It is used to broadcast packets and send control messages. As the names suggests, the switchable interface can switch between the available channels. This interface has CR capabilities. The receiving interface tunes itself to a data channel for communication. The switchable interface and the receiving interface of two CR nodes have to agree on the channel in order to transmit data from one CR node to the other. The system model of the *ns-2* CRAHN extension is shown in Figure 4.1 . The model is divided into sub-modules to outline different CR functionalities.

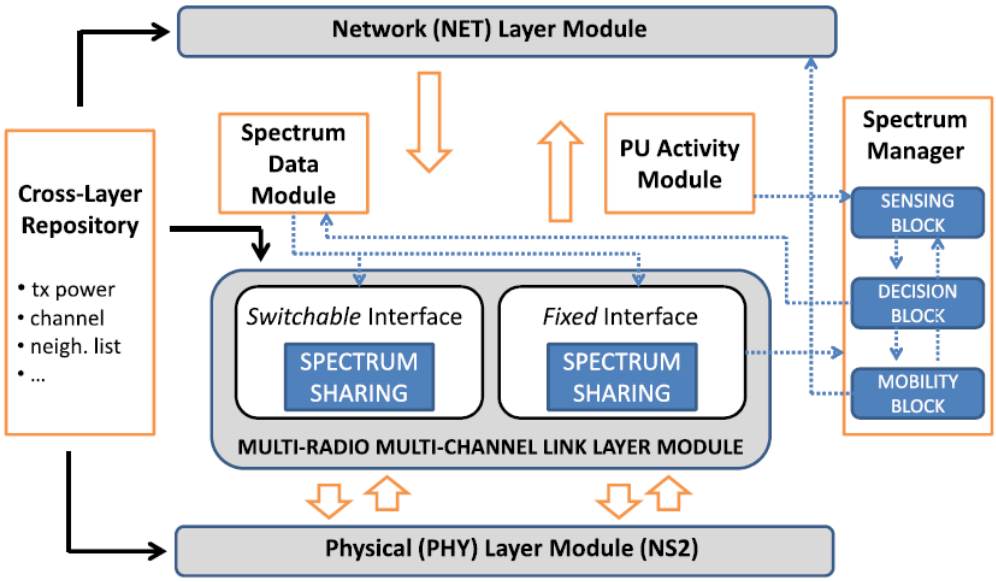

Figure 4.1: *ns-2* CRAHN extension system model [8]

4.1 Primary User Activity Module

The Primary User Activity Module determines the PU activity. Complete PU activity is saved in a map file. Before each simulation begins, the CRAHN extension reads the map file and saves it in a global data structure. The map file contains information about the channel the PU will be using, the location of the PU, the location of the receiving interface, the transmission range, the PU On time (α) and the PU Off time (β). The file also contains information about the number of times a PU arrives and leaves the channel and lists all the arrival and departure times. Each PU is allocated to a channel. The arrival and departure times are calculated using α and β which use an exponential model [13]. The map file saves these times for each CR one after the other in the following format:

<Number of arrival times>

<arrival time>

<departure time>

.....

<arrival time>

<departure time>

The PU map files are static and are created before the actual simulation. A separate script is used to create these files. The script determines the position of each PU using cosine or sine angles and a bounding radius. The script makes use of α and β to determine the arrival and departure times. A map file is created for each combination of α and β. According to the exponential On/Off model [13], an exponential distribution seeded by using a random number generator with a mean of $\frac{1}{\alpha}$ and $\frac{1}{\beta}$ for the on and off time, respectively. Depending on the number of the times a PU leaves a channel, that many pairs of arrival and departure times are calculated. The simulation uses a total of 10 pairs of PUs. Thus, each map file contains a pair of times

for each PU. Using a combination of seeds, values of α and β as given in Table 5.1 21, several map files are created. The simulator accesses information from the appropriate map file depending on which combination of parameters it is currently simulation. The code snippet in Listing 4.1 shows how α and β are used to generate the arrivals and departures of a CR node on a given channel.

```
set alphaV(0) [new RandomVariable/Exponential]
$alphaV(0) set avg_ [expr 1/$val(alpha)]
set betaV(0) [new RandomVariable/Exponential]
$betaV(0) set avg_ [expr 1/$val(beta)]
$alphaV(0) use-rng $rng
$betaV(0) use-rng $rng
for {set k 0} { $k < $val(nnPU)  } {incr $k} {
        set counter 0
        set time 0
        set endTime 100
        for {} { $time < $endTime } {} {
            set on [$alphaV(0) value]
            set off [$betaV(0) value]
            set time [expr $time + $off ]
            set arrival($counter) $time
            set time [expr $time + $on ]
            set departure($counter) $time
            set counter [expr $counter + 1]
        }
}
```

Listing 4.1: Calculate Arrival and Departure time for PU

4.2 Spectrum Manager Module

The Spectrum Manager Module implements the CR capabilities. This module can be further divided into sub-modules depending on the CR functionality. A typical cognitive cycle which comprises spectrum sensing, spectrum decision, and spectrum

mobility are implemented in this module. The Spectrum Manager Module coordinates all the activities of these sub-modules.

Spectrum Sensing

This sub-module implements the *sense* method for a CR node. Channel sensing is performed by looking up the PU map file. It decides whether a PU signal is detected or not depending upon the current channel, the sensing interval used and the simulation time. Along with this, the CR node also has to check for the transmission of the PU. Using both these methods, a CR node can detect a PU. However, it still does not account for misdetection of a PU. The probability of detecting a PU accurately is described in [13]. For simplicity, in this implementation the probability of misdectection is calculated based on th sensing time of the CR node. If the sensing time is less than $0.2s$ then the probability of misdetection increases. Using this probability and the information from PU log file, the module decides if a PU is active on the current channel. When the sensing cycle is running, the CR node cannot communicate with any other node.

Spectrum Decision

This module focuses on the actions to be taken based on the output from the Spectrum Sensing module. The Spectrum Decision sub-module implements two main functions: the decision to switch channels and the next channel to which to switch. Depending on the result of PU detection received from the sensing sub-module, this sub-module determines whether the CR node should stay on or leave the current channel. It provides two different switching policies. The CR node can pick either policy. The two switching policies are: an always switch policy where the CR node always vacates the channel, and a probabilistic switch policy where the CR node either switches with some switch threshold or keeps sensing the current channel until the PU stops its transmission.

The other function of this sub-module is to decide the next channel by using one of the two allocation policies [12]. The CR node can switch to the next channel by using the round robin allocation technique or a random allocation technique. In the round robin technique, the CR node switches to the next available channel. On the other hand, in the random technique the CR node selects any active channel at random and switches to it. Once a channel switch event has occurred, the higher layers are informed. The CR node is also responsible for sharing this change with its neighbouring nodes. We used the probabilistic switching policy and the random allocation policy when running simulations.

Spectrum Mobility

The Spectrum Mobility sub-module has to take care of spectrum handoff. While the Spectrum Decision sub-module decides the next channel where the CR node has to switch, the Spectrum Mobility sub-module actually performs the channel switching operations. Once the decision sub-module decides to switch the CR node to another channel, it invokes this module and also informs it of the new channel selected. Before switching to the next channel selected, the mobility module calls the sensing module to check for PU activity on that channel. If no PU activity is detected then the mobility module performs the handoff. The CR node loads the characteristics of the new channel and reconfigures its parameters accordingly. During spectrum handoff, the CR node cannot transmit any data.

4.3 Spectrum Data Module

The Spectrum Data module provides information about the channel characteristics. All the channel information is stored in a channel log file. It contains the channel id, the bandwidth and the packet error rate for each channel. This module reads the channel log file and saves the information in respective global data structures.

4.4 Cross-Layer Repository Module

The Cross-Layer Repository module implements a cross layer repository [17]. It maintains the global state of all the channels and makes this information available to all nodes, PUs and CRs alike. It is implemented in order to facilitate different layers of the protocol stack to communicate and share information among each other. It maintains two global data structures. The first data structure saves information about all the receiving channels that can be used by a node. The second data structure saves information about which channel is currently being used by which CR node. When a CR node switches channels, the repository is immediately updated to reflect the change in the current simulation scenario. The data structures are shown in Listing 4.2. These global data structures help avoid any synchronization problems.

```
struct repository_recv {
        // receiving channel
        int recv_channel;
};
struct repository_send {
        //Flag indicating wheter the channel is used for transmitting
        bool active;
        //Last time the channel was used
        double time;
};
repository_recv repository_table[MAX_NODES];
repository_send repository_table_sender[MAX_NODES][MAX_CHANNELS];
}
```

Listing 4.2: Repository for channels

4.5 Multi Radio Multi Channel Link Layer Module

This module implements link layer management. Each radio implements the 802.11 MAC DCF protocol [8]. Each of the three interfaces of a CR node is tuned to a

separate channel. The control interface of the CR node broadcasts control messages on the CCC. Every CR node has to update its neighbours regarding the channel it is currently using. Each channel implements its own queue of packets. The MAC layer implementation of *ns-2* was modified to accommodate the CR functionalities of the switchable interface explained in the Spectrum Manager Module (see 6.4).

4.6 Simulation Set-up

Each simulation scenario consists of ten pairs of PUs and two CR nodes. The goal is for the CR nodes to communicate in the presence of ongoing communication among PUs. The set-up has a total of 11 channels: ten data channels and one control channel. Before the simulation begins the CR nodes are configured. Along with the regular configuration of the wireless channel, propagation model, network interface, MAC type, link layer type, antenna model, interface queue length and others, the CR node also has to configure its CR characteristics. The sensing time, transmission time, α, and β of the CR node are assigned. The repository, PU activity and channel information are then saved in their respective data structures. When the simulation begins, the PUs start communicating. The CRs must find a channel for their communication. So the CR uses the Spectrum Manager module to initiate spectrum sensing and search for an available channel. Once a channel is detected, the CR node informs the other CR of the selection on the CCC, switches to it and starts communication. Depending on the values of sensing time and transmission time, the communication continues. If any PU activity is detected then the CR node stops transmission or switches to the next channel and performs spectrum handoff. In this way communication continues between the CR nodes. A regular cycle performed by CR node is given in Figure 4.2.

Using the simulator described in this Chapter, we performed several simulations. To evaluate the default CRAHN algorithm, and understand and predict its behaviour, we perform a wide range of data analysis. The following Chapter describes

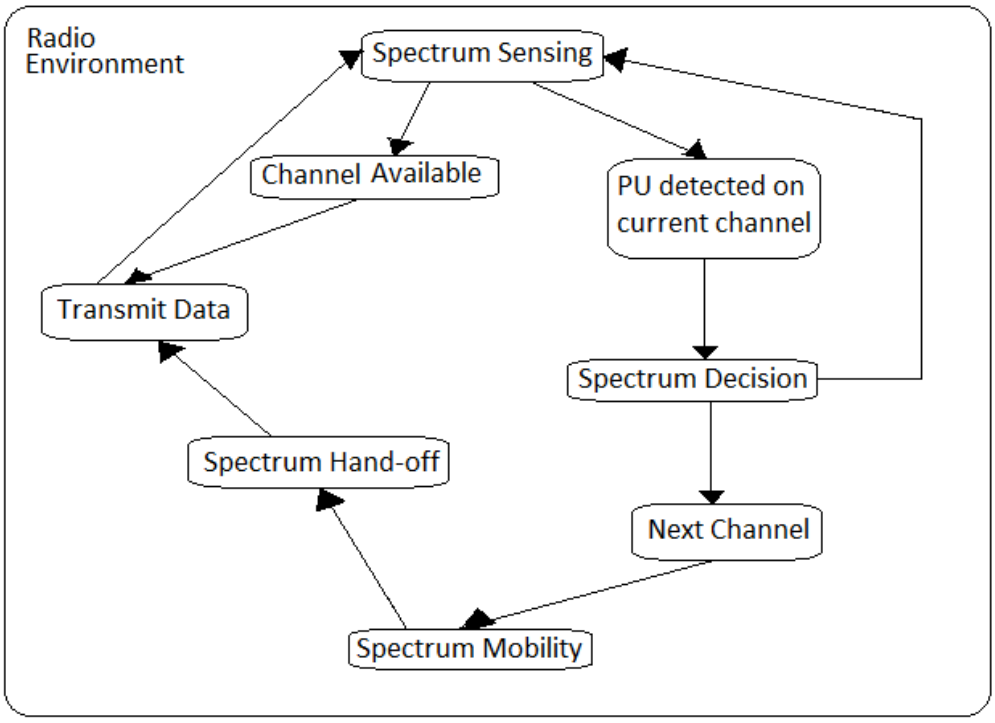

Figure 4.2: Cognitive Radio Cycle [3] [8]

the data analysis performed on a sample simulation data obtained from the default CRAHN algorithm. It explains how an experiment was designed to model the sample data. It also gives details of the resulting diagnostics and interaction plots, and how to interpret the results of the experiment.

Chapter 5

DESIGN OF EXPERIMENT & DATAT ANALYSIS

Experiments are performed to study a process or a system. Tests or series of tests that make purposeful changes to the input parameters of a process or system in order to identify and analyse the reasons for changes in the response variable are referred to as ***experiments*** [16]. Designing and performing experiments assist in interpreting the data and drawing valid conclusions from the resulting patterns. Experiments use techniques of data analysis, a body of methods that process, transform, and model the data and then help suggest conclusions, test hypotheses and predict the behaviour of the data [14]. The goal is to develop a system that is minimally affected by external factors. We can develop ***empirical models*** of a system by designing an effective experiment that models the system performance appropriately [16].

5.1 Initial Data Analysis

To propose an adaptive algorithm, we need to understand the behaviour of the default CRAHN implementation. We observed the SU throughput which was obtained after running several simulations using the default CRAHN extension. The analysis concentrated on five parameters (factors) of Cognitive Radios: Sensing time, transmission time, PU On time (α), PU Off time (β)and number of channels. The SU throughput was the response variable (performance metric). Each parameter takes on the values as shown in Table 5.1. Each simulation consisted of a combination of each of the parameters from the Table 5.1. Using all combinations, the experiment included $11 \times 5 \times 7^2 \times 4 = 10780$ scenarios. Each experiment used a different seed that was given as an input to the simulation. A total of 50 seeds were used; creating 50 replicates of the experiment.

The number of simulations for all the replicates resulted in a large amount of data. The data was observed and we found large variance in the SU throughput. To

Table 5.1: Cognitive Radio parameters

Parameter	**# Levels**	**Values**
Sensing Time	11	0.025, 0.050, 0.075, 0.100, 0.125, 0.150, 0.175, 0.200, 0.225, 0.250, 0.275
Transmission Time	5	0.20, 0.40, 0.60, 0.80, 1.0
PU On Time (α)	7	0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 2.00
PU Off Time (β)	7	0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 2.00
No. of Channels	4	4, 6, 8, 10

simplify the process of analysis, we calculated the average throughput across all 50 replicates. For further clarity, we considered only the minimum and maximum level of each parameter. The experiment was now transformed into a 2^5 screening experiment. The initial data analysis of this experiment focused on creating scatter plots and histograms to get an high level idea of the behaviour of the throughput. We generated scatter plots to observe the effect of each parameter on the performance. The scatter plot for sensing time and transmission time are seen in Figure 5.1 and 5.2, respectively.

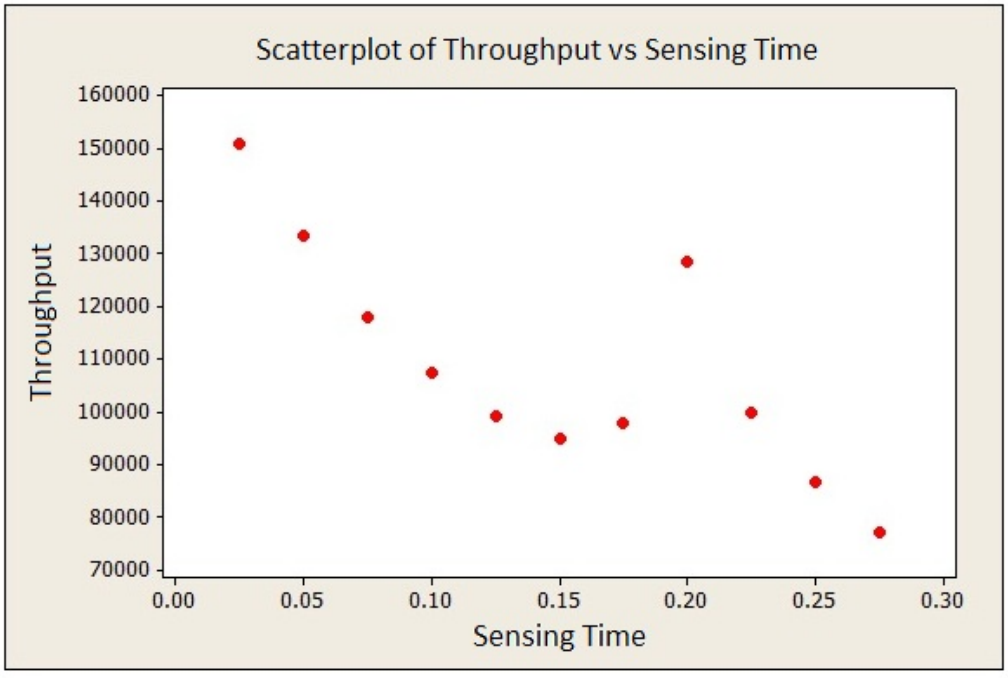

Figure 5.1: Scatter plot: Sensing time on throughput.

The scatter plots give a fair idea of the effect of sensing time and transmission time on the performance. We can see that the throughput falls gradually with an increase in sensing time. On the other hand, the throughput starts increasing as the transmission time increases. This is in accordance to expectations as suggested in Chapter 2. The scatter plot was useful to analyse the relation between individual parameter and SU throughput.

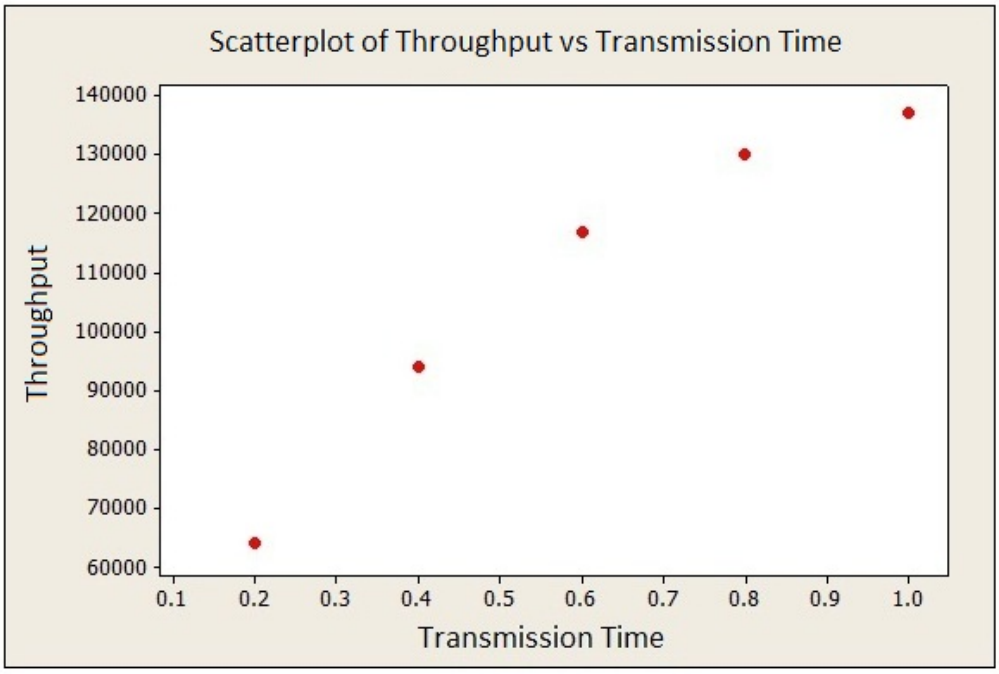

Figure 5.2: Scatter plot: Transmission time on throughput.

To understand the nature of throughput for each simulation which each combination of the parameters, we generated histograms for the screening experiment. In the case of histograms, we did not average the throughput across 50 replicates. This way we were able to see how the throughput varied over all replicates. After generating all the histograms, we observed that sharp spikes were present in the throughput. For the same combination of parameters, throughput was varying over a large range for different replicates. A histogram for one such combination is shown in Figure 5.3. Such behaviour of the throughput, indicates that the system is not stable. This could be explained by the fact the SU throughput depends on the PU activity.

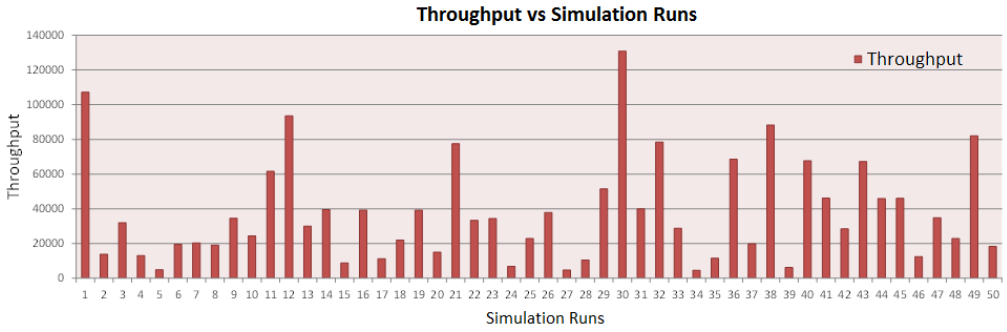

Figure 5.3: Histogram: Sense time - 0.275, Transmit time - 0.2, α - 2, β - 2, No. of channels - 4

5.2 ANOVA

After analysing the scatter plots and histograms, we move our attention to designing an experiment to generate a model for the system. We used ***Design Expert 8*** [1] to perform the experiment. We designed a 2-level factorial experiment using the parameters from Table 5.1 as input factors. On completing the design, we performed a *natural log transformation* as suggested by ***Design Expert***. A transformation is performed to better fit the data in the model. Based on the transformation selected, a mathematical function is applied to the response data. Post transformation, we studied the effect of each parameter on the performance of the model. The effects list, as shown in Table 5.3, illustrates the percentage contribution of each parameter to the response data. Based on their contribution, we can decide the parameters to be included for analysis. For this experiment, we selected all terms that had a contribution of 2% or more to the performance metric. From Table 5.2, we can see that the terms A, B, C, D, AB, AC, AD, BD and ABD have a % contribution of 2 or more. Therefore, only those terms were selected. The effects list of the selected terms is shown in Table 5.3.

We then conducted an ***ANOVA*** (Analysis of Variance) on the selected parameters. The model F-value of 93.32 implies that the model is significant. There is only a 0.01% chance that a model F-Value this large could occur due to noise. Values

Table 5.2: Contribution of each factor - All factors

Term	**Effect**	**SumSqr**	**%Contribtn**
A-Sensing Time	-0.68934	3.80157	6.39835
B-Transmit Time	0.54414	2.36877	3.98683
C-Alpha(α)	1.16147	10.79218	18.16410
D-Beta(β)	-1.82089	26.52530	44.64419
E-Number of Channels	-0.15344	0.18837	0.31704
AB	0.72295	4.18125	7.03738
AC	-0.46516	1.731	2.91341
AD	0.69405	3.85371	6.48611
AE	0.05050	0.02040	0.03434
BC	0.14711	0.17315	0.29142
BD	-0.49362	1.94931	3.28084
BE	-0.03888	0.01209	0.02036
CD	0.28811	0.66408	1.1177
CE	0.03971	0.01261	0.02123
DE	-0.08764	0.06144	0.10342
ABC	0.06463	0.03342	0.05625
ABD	-0.58043	2.69525	4.53633
ABE	-0.05422	0.02352	0.03958
ACD	0.12077	0.11669	0.19639
ACE	0.01627	0.00211	0.00356
ADE	0.06891	0.03799	0.06395
BCD	-0.10701	0.09161	0.15419
BCE	-0.01717	0.00235	0.00397
BDE	-0.06105	0.02982	0.05019
CDE	-0.02587	0.00535	0.00901
ABCD	0.04476	0.01602	0.02697
ABCE	0.02009	0.00322	0.00543
ABDE	-0.00389	0.00012	0.00020
ACDE	-0.00045	1.6E-006	2.8E-006
BCDE	0.04573	0.01673	0.02816
ABCDE	0.02582	0.00533	0.00898

of "Prob. $>$ F" less than 0.0500 indicate model terms that are significant [1]. The results of the ANOVA are shown in Table 5.4.

Table 5.5 shows the predicted R-squared and adjusted R-squared values. The "Pred R-Squared" of 0.9460 is in reasonable agreement with the "Adj R-Squared" of

Table 5.3: Contribution of each factor - Selected factors

Term	**Effect**	**SumSqr**	**%Contribtn**
A-Sensing Time	-0.68934	3.80157	6.39835
B-Transmit Time	0.54414	2.36877	3.98683
C-Alpha(α)	1.16147	10.79218	18.16410
D-Beta(β)	-1.82089	26.52530	44.64419
AB	0.72294	4.18125	7.03737
AC	-0.46516	1.73100	2.91341
AD	0.69405	3.85371	6.48610
BD	-0.49362	1.94930	3.28084
ABD	-0.58043	2.69525	4.53632

Table 5.4: ANOVA for selected factorial model

Source	**Sum of Squares**	**df**	**Mean Square**	**F Value**	**p-value Prob F**
Model	57.90	9	6.43	93.32	< 0.0001 significant
A-Sensing Time	3.80	1	3.80	55.15	< 0.0001
B-Transmit Time	2.37	1	2.37	34.36	< 0.0001
C-Alpha (α)	10.79	1	10.79	156.56	< 0.0001
D-Beta (β)	26.53	1	26.53	384.80	< 0.0001
AB	4.18	1	4.18	60.66	< 0.0001
AC	1.73	1	1.73	25.11	< 0.0001
AD	3.85	1	3.85	55.91	< 0.0001
BD	1.95	1	1.95	28.28	< 0.0001
ABD	2.70	1	2.70	39.10	< 0.0001
Residual	1.52	22	0.069		
Cor Total	59.41	31			

0.9640. ”Adeq Precision” measures the signal to noise ratio. A ratio greater than 4 is desirable. A ratio of 29.436 indicates an adequate signal [1].

Table 5.5: R-Sqaured Values

Std. Dev.	0.26	R-Squared	0.9745
Mean	11.06	Adj R-Squared	0.9640
C.V. %	2.37	Pred R-Squared	0.9460
PRESS	3.21	Adeq Precision	29.436

The model for SU throughput (T) generated by the ANOVA is as follows:

$$\begin{aligned} ln(T) &= 12.744 - 14.425A - 0.827B + 1.146C \\ &\quad - 1.972D + 16.903AB - 2.481AC + 8.345AD \\ &\quad + 0.338BD - 7.739ABD \end{aligned} \tag{5.1}$$

5.3 Diagnostics

After designing the experiment and generating the model, we have to analyse various plots to check if the transformation suggested by Design Expert yields a good model. For this reason we take a look at the following plots:

- Normal Plot of Residuals
- Residuals vs Predicted plot
- Residuals vs Run plot
- Predicted vs Actual plot
- Box-Cox plot

Using the Normal Plot of Residuals we can verify if the data conforms to a hypothesized distribution. The normal probability plot indicates whether the residuals follow a normal distribution [1]. In case they do, the sample points follow a straight line. Residuals of a sample data are defined as the difference between the sample itself and the estimated function value. Any data points observed away from the line are outliers. The normal plot of residuals is shown in Figure 5.4. Clearly, the normal plot of the residuals is very good indicating that the data is fit well.

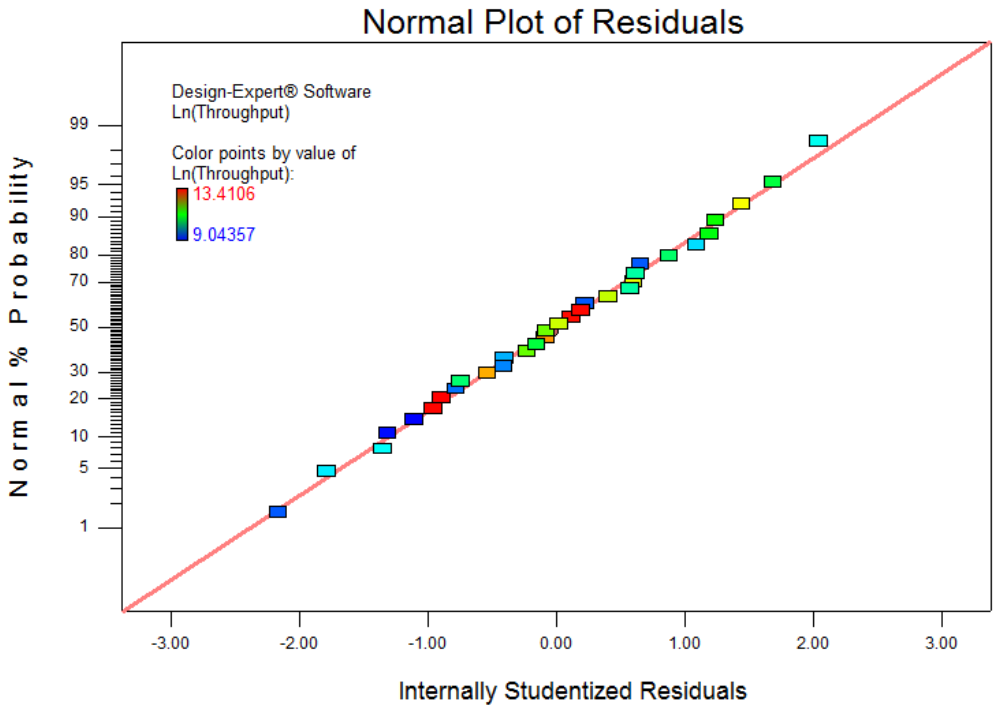

Figure 5.4: Normal Plot of Residuals

Figure 5.5 is a plot of the residuals versus the predicted values of the response data. It is used to test the assumption of constant variance [1]. The method of least squares assumes that each data point is equally reliable. However, in some cases the parameter along the y-axis can be more variable and hence less predictable than the parameter across the x-axis [16]. In such a scenario, the weight has to be assigned to such points that exhibit predictable behaviour. It is assumed that residuals scatter across the graph on either sides of the best fit line. This graph should be random which indicates the data fits well in the model. If any kind of sequence is observed then it needs further transformation. From the Figure 5.5 we can see that there is random scatter and the data fits the model well.

Figure 5.6 is a plot of the residuals versus the run order that was used when running the experiment. After designing the experiment, when it is actually executed the order of the runs is randomized. While less important for experiments conducted in simulation, this is done to check if any variable has any influence on the response

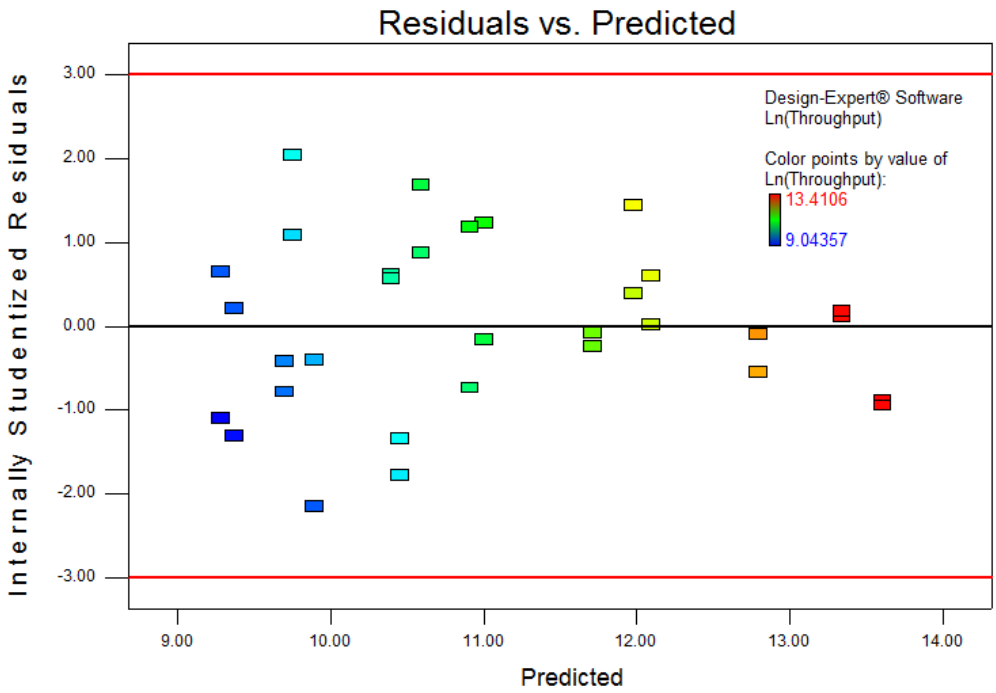

Figure 5.5: Residuals vs Predicted

during the experiment execution [1]. This plot should also show random scatter for a model that fits data well. If any patterns are observed then it indicates some time related dependency of some parameter. Figure 5.6 shows that the data points are spread in a random manner and does not show any correlation among the data points.

Figure 5.7 is a plot of the value of SU throughput predicted by the model in Equation 5.1 versus actual value measured in the simulation. It helps determine the value or group of values of data point(s) which could not be easily predicted by the model [1]. Figure 5.7 shows the graph for Predicted vs Actual. We observe that the Predicted values closely follow the Actual values along the best fit line.

The plot in Figure 5.8 belongs to the family of power transforms. A Box-Cox plot recommends an appropriate power law transformation based on the λ value. The λ value indicates the power to which the sample data in the experiment should be raised. When the confidence interval contains a λ value of 1 then it does not recommend any transformation [1].

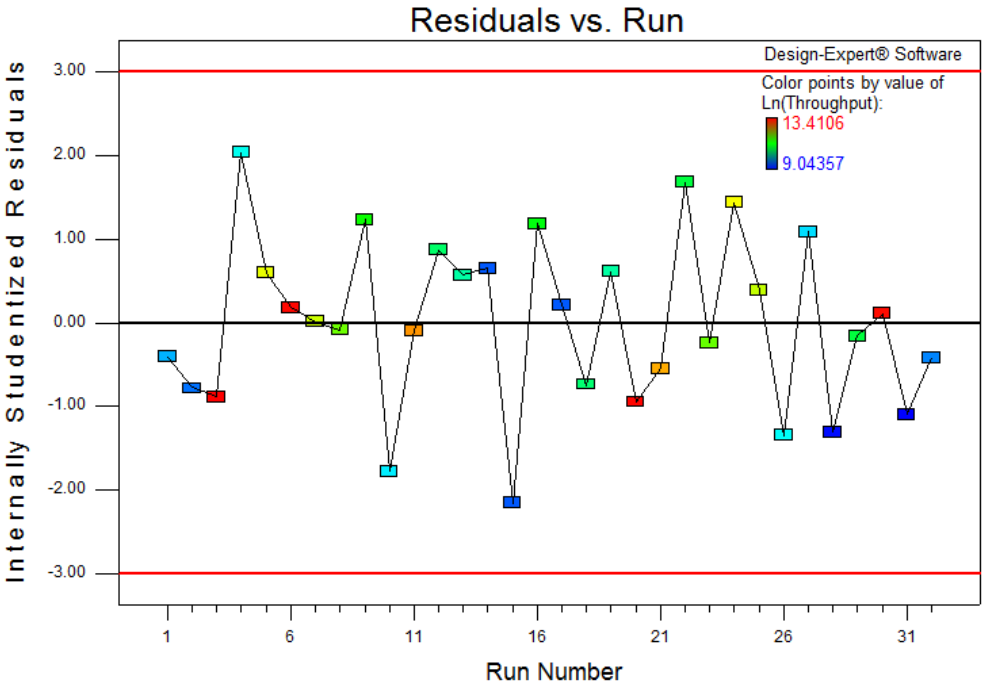

Figure 5.6: Residuals vs Run

Based on the all the diagnostics plots, we can infer that the model generated by the experiment is good and the data fits well. We did not observe any anomalous behaviour in the data. There were no outliers. The model is thus a good predictor.

5.4 Two-way Interaction

We look at the interaction plots to check if there are any dependencies present between any of the parameters. An interaction plot shows how a change in the value of one parameter affects the other parameter with respect to the response variable here, SU throughput. Thus, the value of the performance metric depends upon the settings of those two parameters [16] [1]. Typically, an interaction is represented by two non-parallel lines. The interaction plot in Figure 5.9 shows that there is an interaction between sensing time and transmission time on SU throughput. In the interaction plot, the red line indicates the behaviour of the throughput when the transmission time is $1.0s$. On the other hand, the black line shows the throughput for a transmission time of $0.2s$. On observing the black line, we can see that the throughput falls significantly

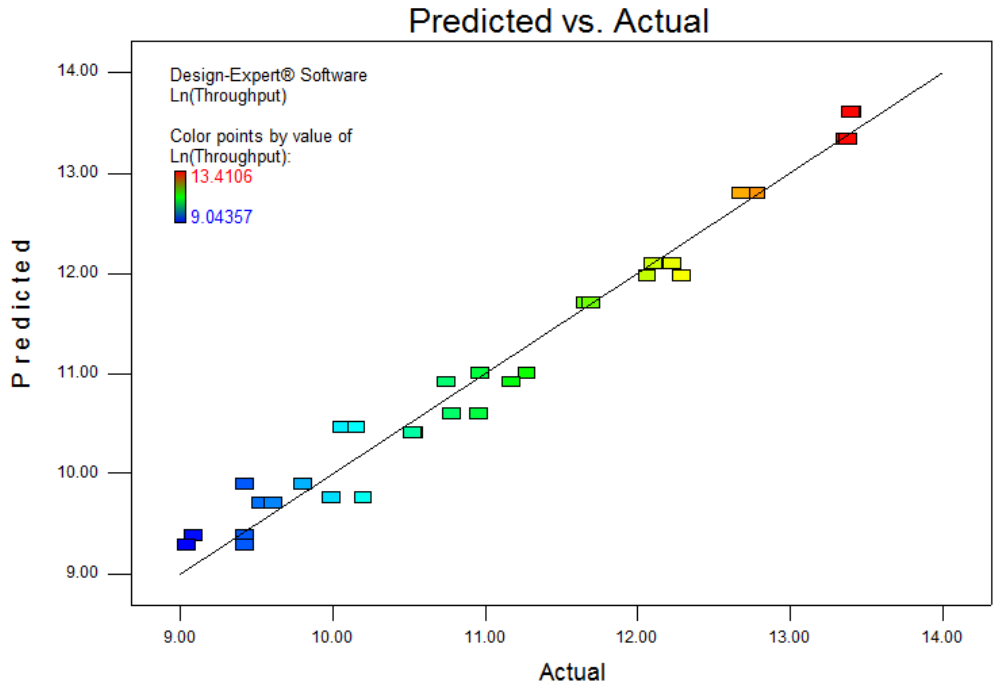

Figure 5.7: Predicted vs Actual

with an increase in sensing time. The red line however, shows consistent throughput for all levels of sensing time. This indicates that for lower values of transmission time, the throughput starts decreasing as the sensing time starts increasing. For higher values of transmission time, the throughput does not get affected much by an increase in sensing time but it does not increase either. As the sensing time starts approaching the value of transmission time, the throughput starts falling. It is clear from the plot that a sensing time higher than the transmission time definitely does not work in the favour of the throughput performance. We can observe a general theme that higher value of sensing time affects the throughput negatively.

Another interesting interaction is the one between sensing time and α on SU throughput. Though the interaction plot in Figure 5.10 does not show the lines intersecting, we can observe the dependency between those parameters. The red line indicates the throughput for α set to a value of 2 while the black line is for α set to a value of 0.5. As explained in Chapter 4 α indicates the PU On time. For a lower value

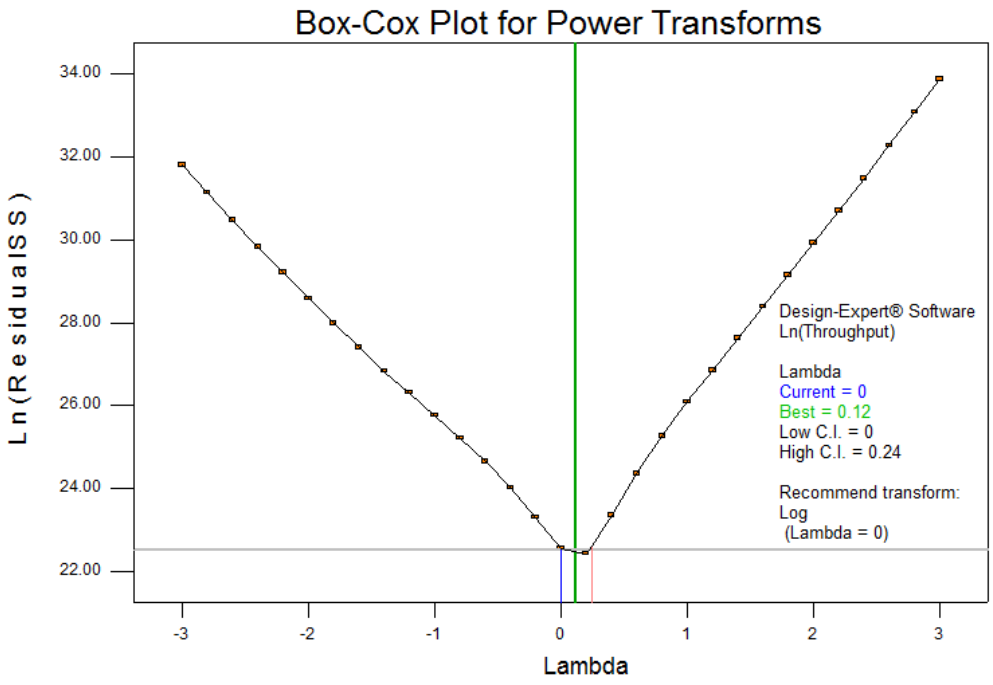

Figure 5.8: Box-Cox Plot for Power Transforms

of α the throughput is consistently low. For a higher value of α, the throughput shows a steady decline as the sensing time increases. This shows that even if PU activity is low, a high sensing time will have a negative impact on SU throughput.

5.5 Three-way Interaction

From the interaction plot in Figure 5.9, we see that the parameters, A(sensing time) and B(transmission time) are involved in a three-way interaction with D(β) as ABD. This is the strongest three-way interaction based on its contribution to the response data. To better understand this three-way interaction we analyse the cube plot generated by the model. A cube is used to interpret the dependency between three parameters at a time. It shows the predicted values of the three parameters selected using the model generated for the combination of levels of the parameters [1] [16]. The cube plot for the ABD interaction is shown in Figure 5.11.

We first observe the plot with respect to the lower value of β. The plot shows that for the maximum transmission time and minimum sensing time the throughput is

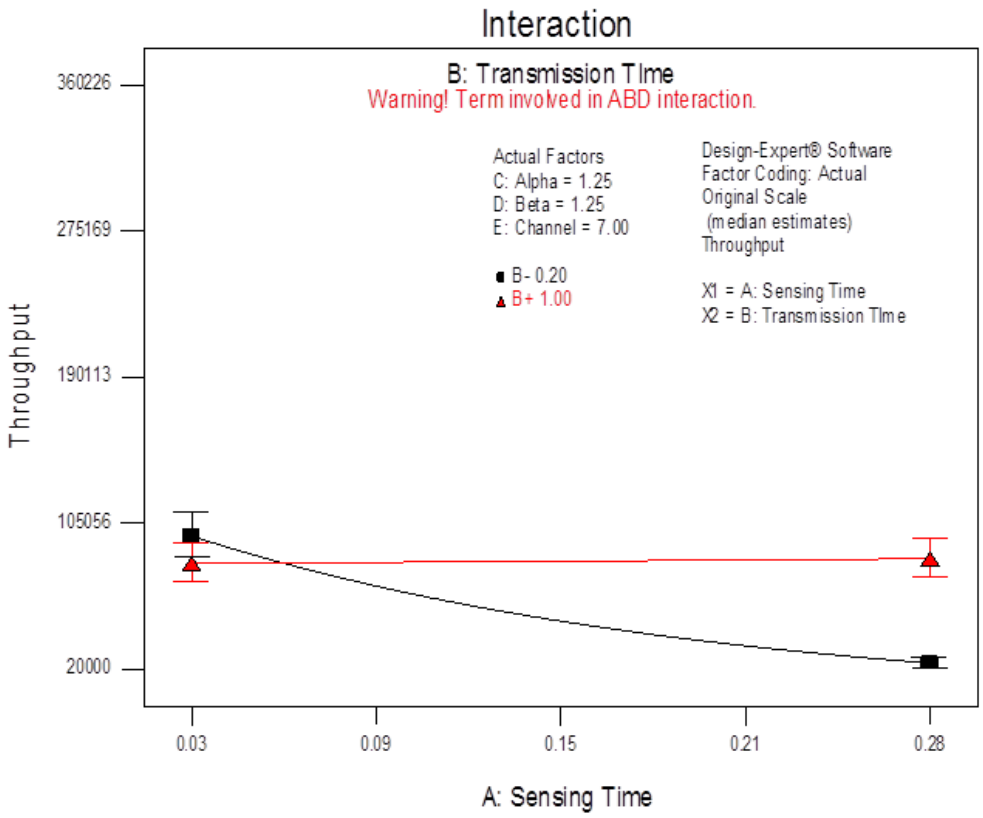

Figure 5.9: Interaction Plot for Sensing time and Transmission time

not very good. This indicates that if the difference between sensing and transmission time is too high then the throughput starts falling. We also observe that for a lower value of transmission the throughput is reasonable. Since the sensing time value is still low, even the smallest value of transmission time can help increase the throughput. Here, more than transmission time, β has larger effect on throughput which results in better performance. As the sensing time starts increasing, we see a steady fall in throughput. Even when the transmission time is highest, the throughput shows only a very small increase in its performance. We also observe that for this model, the highest predicted throughput is seen for the lowest value of β. On the other hand, for the higher value of β the overall throughput is quite low. On analysing the cube plot, we can see that as the value of β increases, throughput starts decreasing. Except for the highest levels of sensing time, transmission time and β. The throughput in such a

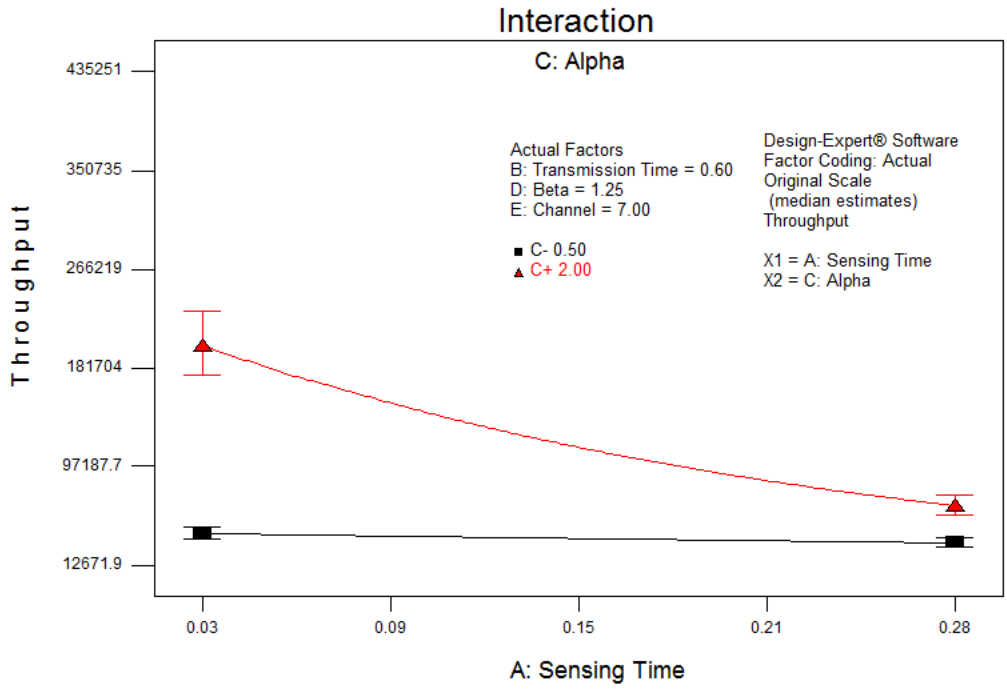

Figure 5.10: Interaction Plot for Sensing time and α

case is quite high but not the highest predicted. The cube plots shows how β impacts the throughput.

Though a three-way interaction among sensing time, transmission time and α did not contribute much to the SU throughput, a quick look at it shows that the highest predicted throughput is seen when α is the highest. Thus α and β have an inverse effect on the throughput. Figure 5.12 shows the cube plot for sensing time, transmission time, and α.

Designing and conducting the experiment, transformed this massive data obtained from the simulations, into a feasible and realistic model. Data analysis helped in isolating the CR parameters which influence the SU throughput. Various patterns observed during analysis, were used to envisage an adaptive algorithm. The next Chapter illustrates the implementation of an adaptive heuristic algorithm. It also explains how the CR nodes gather information from the surrounding radio environment and use it to implement a dynamic behaviour.

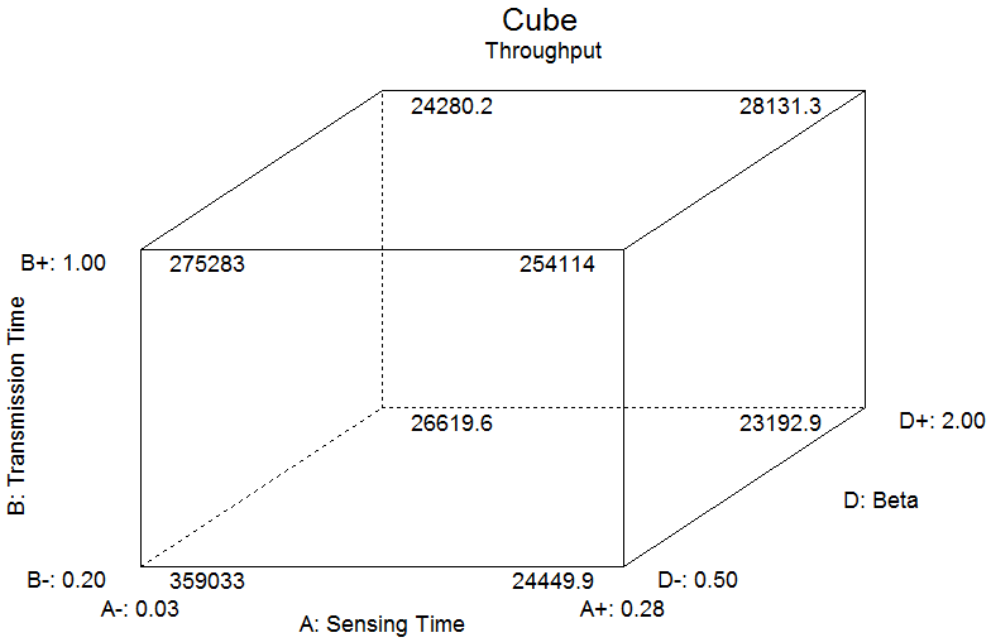

Figure 5.11: Cube plot for throughput as a function of A, B, and β.

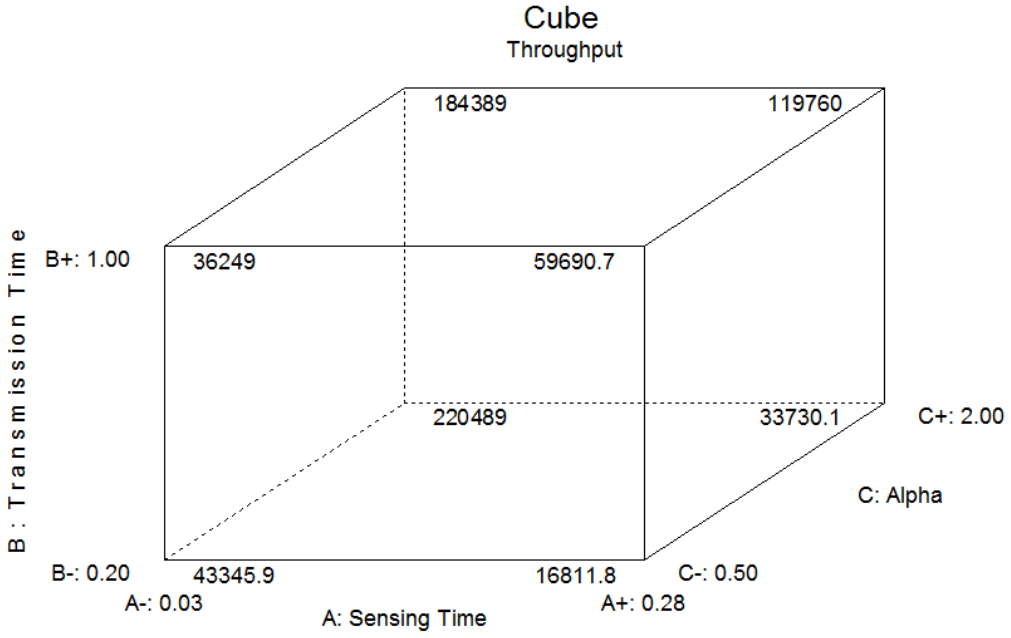

Figure 5.12: Cube plot for throughput as a function of A, B, and α.

Chapter 6

ADAPTING PARAMETER IMPLEMENTATION

The adaptive algorithm is based on the model generated from the analysis of the designed experiment. Using the dependencies and patterns observed among the CR parameters, we develop the adaptive algorithm. Along with adapting sensing and transmission time, the implementation has to stabilize the network and show a significant improvement in the SU throughput.

6.1 Problem Statement

As seen in Chapter 3, many solutions have been proposed to improve the overall throughput of the SU and in turn the CRAHN itself. Some of them suggest adapting the sensing time using techniques that detect transmission power of the PU, while other propose cooperative sensing techniques. But most implementations of CR networks assume fixed sensing and transmission times. That may not yield the highest SU throughput. The SU might spend too much time detecting PUs and less time transmitting data based on the values of sensing time and transmission time. In such cases, the SU throughput starts declining over time.

6.2 Parameter Selection

The analysis of the data helped identify the parameters that most affected the SU throughput in CRAHNs. Among the parameters that were analysed, we found that β has the highest impact on SU throughput. It was followed by α, sensing time and transmission time. To implement the adaptive algorithm, we had to modify the behaviour of these four parameters. But α and β are PU characteristics. CR nodes do not have any control over PU activity. Thus, SUs cannot changes any parameters that are associated with the PUs. But SUs can always change their own CR parameters based on the current network state to improve throughput. Therefore, we focus on changing the sensing time and transmission time of SU. The Channel parameter did

not largely contribute towards the SU throughput. We therefore, completely ignored this parameter for the adaptive algorithm. The CRAHN extension in *ns-2* is modified to exploit the findings of the data analysis.

6.3 Proposed Solution

The default CRAHN extension uses fixed values of CR parameters for the complete duration of the simulation. An adaptive CR node should have capabilities to modify its sensing time and transmission time dynamically based on the fluctuating radio environment. Such an adaptive behaviour may assist in increasing the SU throughput. To make changes to its own parameters, the CR node has to maintain some information about the past and current states of itself and the network. One way to confirm any improvement in the CR performance is to gather the SU throughput. However, calculating throughput over an entire simulation does not serve the purpose. The CR has to learn to change its parameters based on the throughput value at a point in time. We propose a solution to adjust the sensing time and transmission time based on measuring the instantaneous throughput of the SU. We suggest a simple heuristic algorithm that adapts its parameters over time and learns from the previous throughput values. The patterns and the interaction between sensing time and transmission show that they are related. Therefore, the aim is to change the sensing time proportional to the transmission time.

6.4 Instantaneous Throughput

A regular *ns-2* simulation calculates the throughput of the network after the simulation is completed. For the proposed adaptive solution, we need to calculate instantaneous throughput. To measure the instantaneous throughput, we divide the simulation into time intervals. On the completion of a time interval, a timer interrupt is generated and we calculate the throughput in the interval. We used signals to generate the timer interrupt. A signal informs the system of an event. *Alarm Signals* are typically used to indicate the expiration of some timer. The default behaviour is to cause process

termination. But this can be modified by calling a function. To implement the timer we used the structure *struct itimerval*. This structure is used to specify when the timer should expire. The structure contains two member functions. The *it_interval* method is used to implement successive interrupts. The *it_value* method sets the interval to be used for the timer. The *it_value* method can set time in two units: seconds and microseconds. From Listing 6.1 we can see the timer interval is set using a window increment, in units of microseconds. The signal alarm calls the function *signal_to_calc_throughput* which calculates the throughput for that interval. The timer runs in the background without affecting or blocking the simulation.

```
void Mac802_11::SetTimer()
{
        struct itimerval timer_val;
        timer_val.it_interval.tv_sec = 0;
        timer_val.it_interval.tv_usec = 0;
        timer_val.it_value.tv_sec = 0;
        timer_val.it_value.tv_usec = windowIncrement;
        setitimer(ITIMER_REAL, &timer_val,0);
        signal(SIGALRM, signal_to_calc_throughput);
}
```

Listing 6.1: Timer Interrupt

We calculate the throughput for the interval by keeping a count of successful transmission of data packets in that interval. The MAC layer implementation in *ns-2* contains the code that checks for data transmission. We modified the *void Mac802_11::recvACK(Packet *p)* method of *ns-2* to maintain the packet count for each time interval. The interrupt was therefore implemented at the MAC layer. After the time limit elapses, the count is reset. We evaluated time intervals of durations ranging from $10000\mu s$ to $100000\mu s$ in increments of $10000\mu s$. The SU throughput of the current time interval *i* is compared to that in interval *i-1*. After calculating the

difference between the two intervals, the sensing time and the transmission time are modified accordingly. We used the observations from the cube plot in Figure 5.11 to determine a combination of sensing time and transmission time [7].

6.5 Adapting Sensing Time

During data analysis, we observed certain patterns in the behaviour of the SU throughput. Based on the patterns, we propose a novel method of modifying the sensing time relative to the transmission time. We can infer from the analysis that a high sensing time does not yield high throughput. At the same time, we need to check that the sensing time does not fall too low. If the sensing time is very low then PU detection becomes difficult. Among the other observations we made, we saw that a large difference between the value of sensing time and transmission time affects the SU throughput negatively. Also, when sensing time is greater than, equal to or very close to the value of transmission time, the throughput starts falling. For certain combinations of sensing and transmission time, the throughput showed a significant gain in performance.

We therefore implemented the algorithm such that the sensing time varies from 5% to 30% of the transmission time. We denote this percentage setting as P_s. The value of P_s starts at 5% and is incremented by 5% for successive simulations. There has to be a sufficient difference between the values of sensing time and transmission time so that the SU can transmit. But we also have to keep a check that the sensing time does not fall or rise to its extreme levels. Every time the transmission time changes, the sensing time is modified accordingly.

6.6 Adapting Transmission Time

From the analysis performed in Chapter 5, we observed that high transmission time helps increase the SU throughput. The adaptive algorithm therefore, has to let the transmission time increase over the duration of the simulation. But we cannot increase the transmission time by a large value. A sudden change in transmission time results in

a loss of performance. The changes to transmission time have to be small and gradual. The modification is done small steps of $0.05s$ or $0.10s$. We denote this time step as T_x.

The adaptive algorithm compares the throughput in the current interval to the throughput measured in the previous time interval. If the current throughput is less than that in the previous interval, it indicates that the PU activity has increased, spectrum handoff has occurred, the sense time is very low, or congestion was detected incorrectly due to spectrum handoff. It may also indicate that the transmission time was too high to detect any of these mentioned scenarios. As we can control transmission time of the SU, we assume that high transmission time resulted in less throughput. We now modify the transmission time by decrementing it by T_x.

Alternatively, if the current throughput is higher than that measured in the previous interval then the transmission time is increased. An improvement in the SU throughput suggests that the model is working well and a small increment in transmission time should improve the throughput further. But increasing the transmission time beyond a certain level may be harmful to the throughput. In such case, if the throughput falls it means that the transmission time was set too high. The algorithm then falls back and starts decreasing the transmission time value in steps of T_x. As Table 5.1 shows, transmission time values are manipulated by a value of $0.2s$. We needed T_x such that it does not cross over to the next level of transmission time but at the same time allows us to increase the transmission time. Therefore, we chose two values of T_x: $0.05s$ and $0.10s$. The algorithm does not allow the transmit time to fall below $0.2s$ or exceed $1.0s$. The pseudo-code of the adaptive algorithm is shown in 1.

The function call to method *void Mac802_11::signal_to_calc_throughput (int i)* is shown in Listing 6.1. After calculating instantaneous throughput, it makes a function call to the method *void Mac802_11::computeSenseAndTransmitTime()*. Using the instantaneous throughput, *computeSenseAndTransmitTime()* computes the modified sensing time and transmission time. It then resets the packet count and

Algorithm 1 Adaptive Algorithm for Sensing Time and Transmission Time

```
if currentWindowPacketCount ≥ previousWindowPacketCount then
    transmit_time ← transmit_time + T_x
    if transmit_time ≥ 1 then
        transmit_time ← 1
    end if
else
    transmit_time ← transmit_time − T_x
    if transmit_time ≤ 0.2 then
        transmit_time ← 0.2
    end if
end if
sensing_time ← P_s * transmit_time
```

triggers the start of a new time interval. In such a manner, the adaptive algorithm runs the entire length of a simulation. While changing sensing time and transmission time, we also need to account for the transmission power of the PU and the SU. The default CRAHN implementation does use transmission power as a part of sensing cycle. Therefore, the adaptive algorithm does not take into account the transmission power of the PUs.

The efficiency of the adaptive algorithm described in this Chapter, can be verified by evaluating the results obtained from the simulations. The next Chapter presents the performance evaluation of the new algorithm and compares it SU throughput with the default CRAHN algorithm. It was also demonstrates the effect of T_x and P_s on the performance gain. Furthermore, Chapter 7 helps find suitable values or range of values for the CR parameters being studied.

Chapter 7

RESULTS & INTERPRETATION

In this chapter we evaluate the performance of the adaptive algorithm discussed in Chapter 6. We ran several simulations using the adaptive algorithm and recorded the results. The results are divided into performance gain with respect to different sensing times, transmission times for all combinations of P_s and T_x. We also collated the results for the overall throughput of the adaptive CR network and compared it with that of the default CRAHN.

7.1 Experimental Set-Up

Before presenting the results of the adaptive algorithm, we describe the set-up used for simulations. We used the modified CRAHN extension for *ns-2* that includes the adaptive algorithm. We used the same number of mobile nodes, PUs and SUs as described in Chapter 4. The number of channels were also the same. The simulation used TCP-FTP [6] as the transport protocol. The transmission range was set to $500m$. Channel bandwidth was set to $2Mbps$ and the packet size was $1000bytes$. We used five different time intervals starting at $10000\mu s$ going up to $50000\mu s$, in increments of $10000\mu s$, in each simulation. This generated five scenarios of the simulations. The simulation used five different seeds giving five replicates of the simulation. Sensing time, transmission time, α and β took values as given in Table 5.1. A single simulation consisted of all combinations of sensing time, transmission time, α, β, seeds and one time interval. A simulation was executed for all values of timer intervals, and all combinations of T_x and P_s.

7.2 Sensing Time Analysis

We performed simulations for 11 different levels of sensing time. The simulation begins with a fixed value of sensing time. After an interval the sensing time changes depending on the value of P_s. We tested with six different values of P_s, from 5% to

30% increasing by 5%. We divide the simulation results based on sensing time levels and P_s levels.

We calculated the average throughput of the CR nodes for each level of the sensing time. Figure 7.1 shows the throughput for all sensing levels while Figure 7.2 shows the throughput for sensing levels starting from $0.05s$ to $0.175s$. From Figure 7.1 we can see that the adaptive algorithm shows a performance gain for a sensing time of $0.05s$ and more. The adaptive algorithm does not show any increase in throughput only for the sensing level of $0.025s$. For all other levels, the adaptive algorithm outperforms the default algorithm of the CRAHN. On taking a closer look, we also see that the overall throughput of the system starts decreasing as the value of sensing time starts increasing. This result confirms our observations in Chapter 5 that the SU throughput will reduce with higher sensing time. In spite of an overall decrease in throughput, the adaptive algorithm manages to exceed using static values of sensing time and transmission time. For this reason, we infer that to improve the SU throughput of a CRAHN, the best range of sensing time, according to our results is between $0.05s$ and $0.175s$.

Figure 7.2 shows the range of sensing time levels in more detail. We consider $0.175s$ sensing time as a threshold. From the throughput numbers obtained for all levels of sensing time we saw the average throughput rise by 14.8%, with the highest percentage increase of 32% recorded for a sensing time $0.225s$. For the levels of sensing time that fall within the threshold, the gain in throughput was 9.5%. The almost 10% rise in throughput shows that over time the adaptive algorithm indeed improves the SU throughput. It also helped us verify the maximum value of sensing times to be used.

After evaluating the throughput across sensing time levels, we collated the data for sensing time levels for different values of T_x and P_s. Figure 7.3 shows the average throughput for all combinations of T_x and P_s while Figure 7.4 shows the throughput

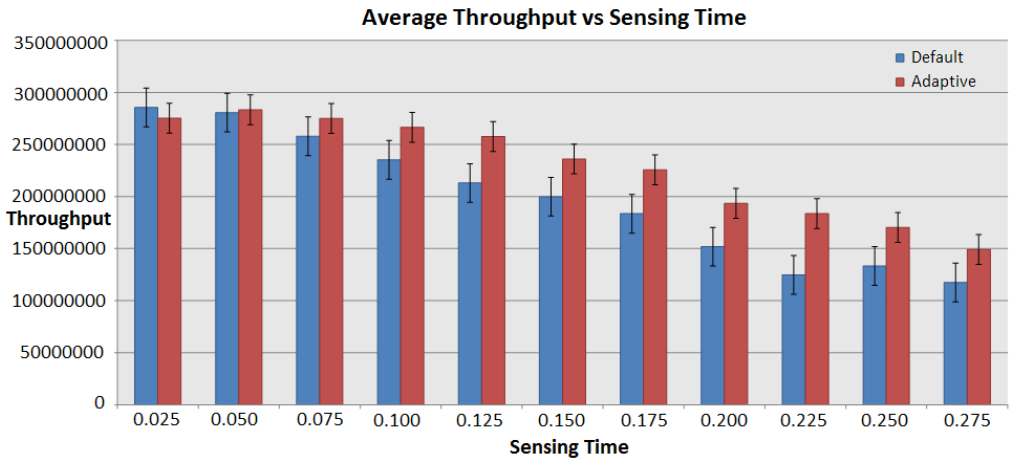

Figure 7.1: SU throughput as a function of Sensing Time: All Levels

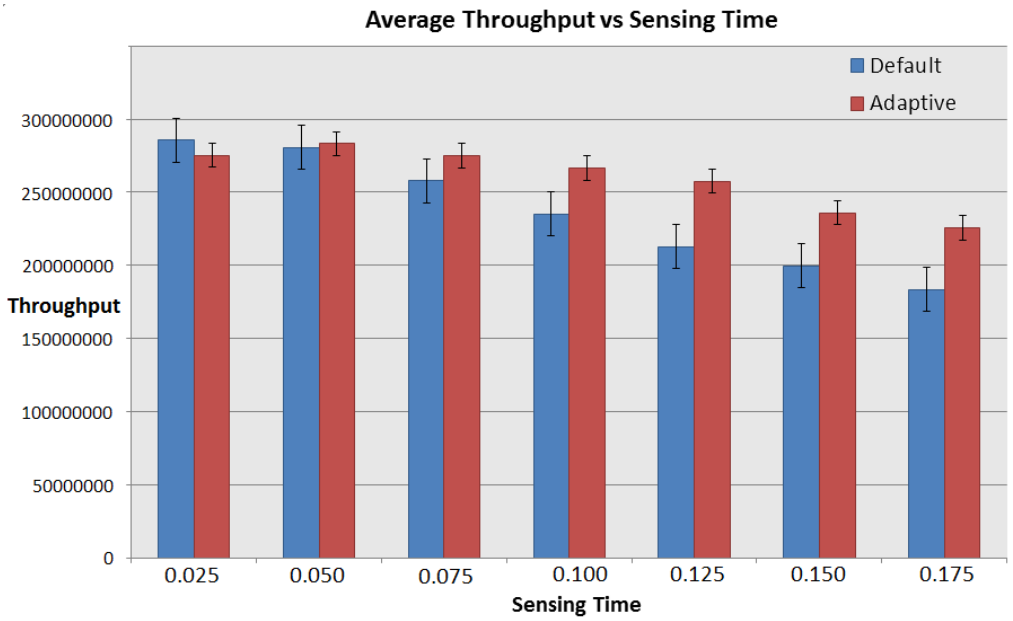

Figure 7.2: SU throughput as a function of Sensing Time: 7 Levels

for P_s ranging from 5% to 15% and for T_x of 0.05s and 0.10s. Figure 7.3 shows that the idea of sensing time changing relative to the transmission time does improve the throughput. We observe that up until a P_s value of 15%, the SU throughput of the adaptive algorithm is higher than the throughput of the default algorithm. For P_s of 20% and higher the adaptive algorithm suffers and its throughput starts decreasing. This again confirms our findings in Chapter 5 that when sensing time starts approaching the value of transmission time the SU throughput is negatively affected.

Figure 7.4 shows the average throughput only for those combinations of T_x and P_s that work the best for the adaptive algorithm. We can infer that when sensing time is between 5% and 15% of transmission time then the SU throughput increases.

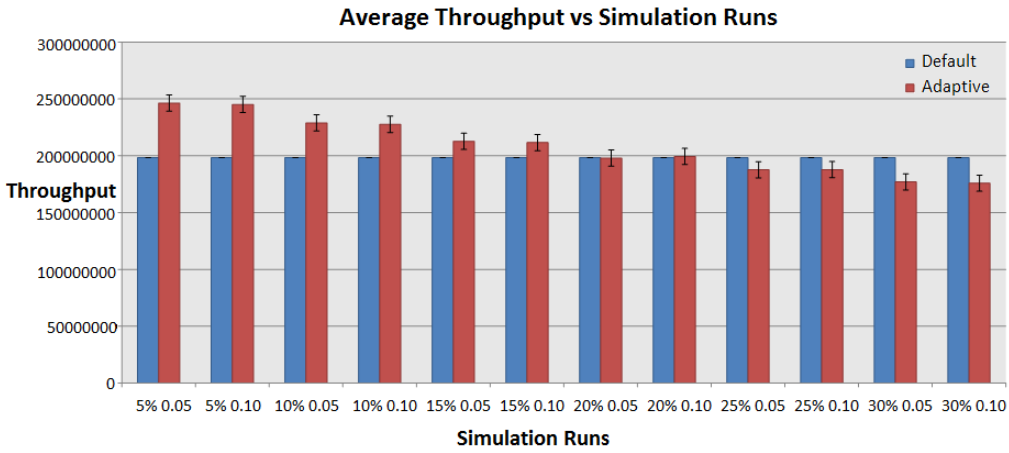

Figure 7.3: SU throughput as a function of sensing time relative to transmit time: All Steps

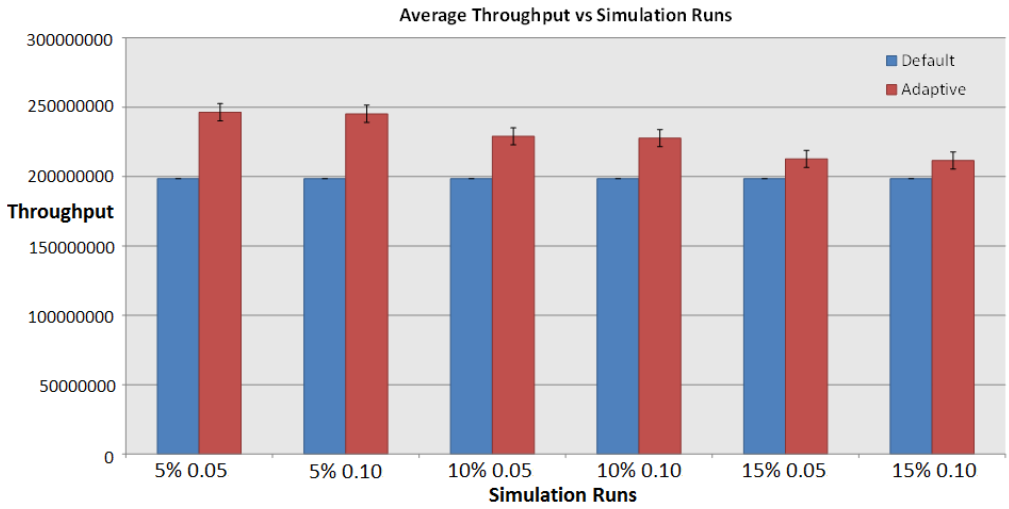

Figure 7.4: SU throughput as a function of sensing time relative to transmit time: 6 Steps

7.3 Transmission Time Analysis

We performed simulations for five different levels of transmission time as discussed in Table 5.1. The simulation begins with a fixed value of transmission time and T_x. As the adaptive algorithm starts working, the transmission time is modified by T_x.

Throughout the simulations the value of T_x remains the fixed. We divide the results based on transmission time levels and T_x levels.

We measured the average SU throughput for five transmission time levels. Figure 7.5 shows the average SU throughput as a function of transmission time levels. We observe that as the transmission time increases, the SU throughput also increases. This is in accordance with the analysis done in Chapter 5. We see a significant percentage increase in throughput for transmission time of $0.2s$, $0.4s$ and $0.6s$.

For transmission time of $1.0s$, the adaptive algorithm shows only a small percent gain. It is not as high as expected but it is still better than the default algorithm. When the transmission time is $1.0s$ even the default algorithm performs well because more transmission time means more throughput. Since the adaptive algorithm does allow the transmission time to go beyond $1.0s$, the percent increase is a little less than others. The average percentage increase in SU throughput for all levels of transmission time is 14.9%. The highest recorded percentage rise in throughput is 30.8% for transmission time of $0.2s$.

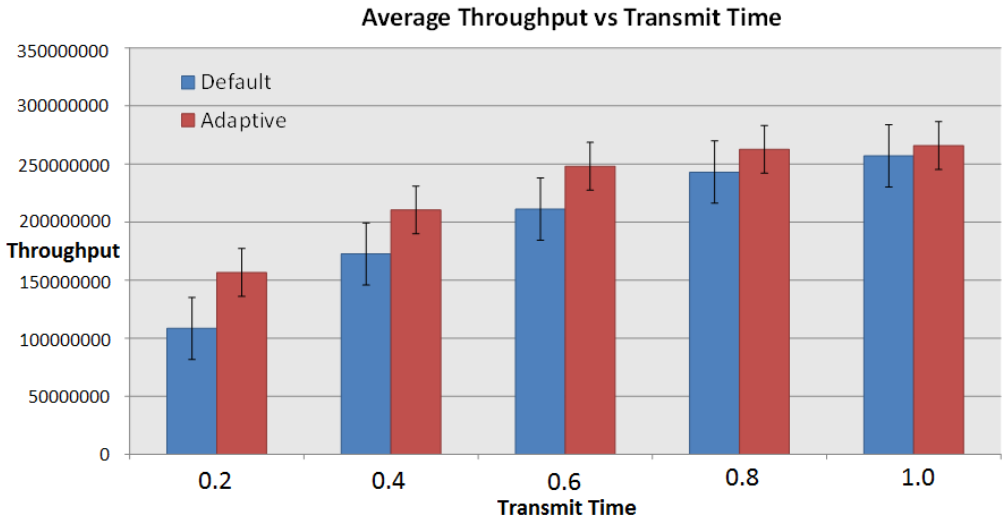

Figure 7.5: SU throughput as a function of Transmission Time: All Levels

Similar to the sensing time evaluation, we generated plots for average throughput over all simulation runs for different combinations of T_x and P_s with

respect to transmission time levels. Figure 7.6 shows the average throughput as a function of simulation runs. Figure 7.7 shows the average throughput for some simulation runs with P_s of 15% and less. Figure 7.7 clearly shows that the adaptive algorithm works best for P_s values of 5%, 10% and 15%. For P_s more than 20% the average SU throughput starts declining, again confirming the observations made in Chapter 5 and validating the data model. We also put together results for values of T_x. For the duration of a complete simulation, the adaptive algorithm used a fixed value of T_x. Figure 7.8 shows the average throughput versus the T_x levels. On comparing these levels, we observe that for both values of T_x the adaptive algorithm performs better than the default algorithm. It is clear from the graph that even though both values of T_x are acceptable, T_x value of $0.05s$ shows a better improvement in the SU throughput. Thus, we infer that T_x of $0.05s$ works best for the adaptive algorithm.

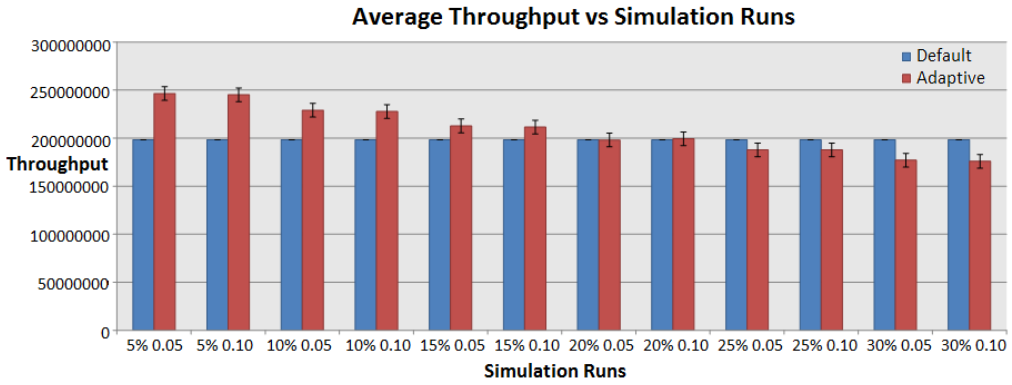

Figure 7.6: SU throughput as a function of sensing time relative to transmit time and T_x: All Steps

7.4 Overall SU Throughput

We collated the results for all the simulation runs for the default and adaptive algorithm. In order to check if the average SU throughput complies with the findings made in sensing time and transmission time evaluation in Chapter 5, we put together the overall average throughput of each simulation (not taking into account sensing time or transmission time levels) and divided the results based on the combinations of T_x and P_s. Figure 7.9 shows the average throughput versus simulation runs. It confirms

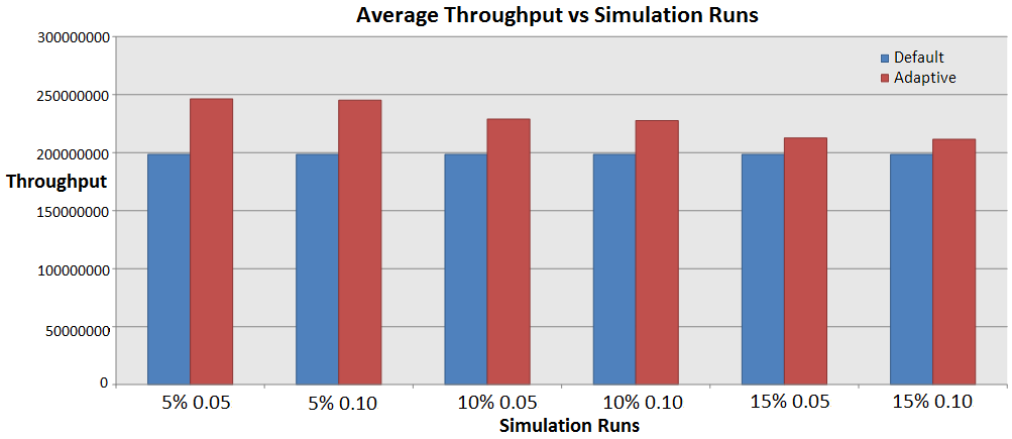

Figure 7.7: SU throughput as a function of sensing time relative to transmit time and T_x: 6 Steps

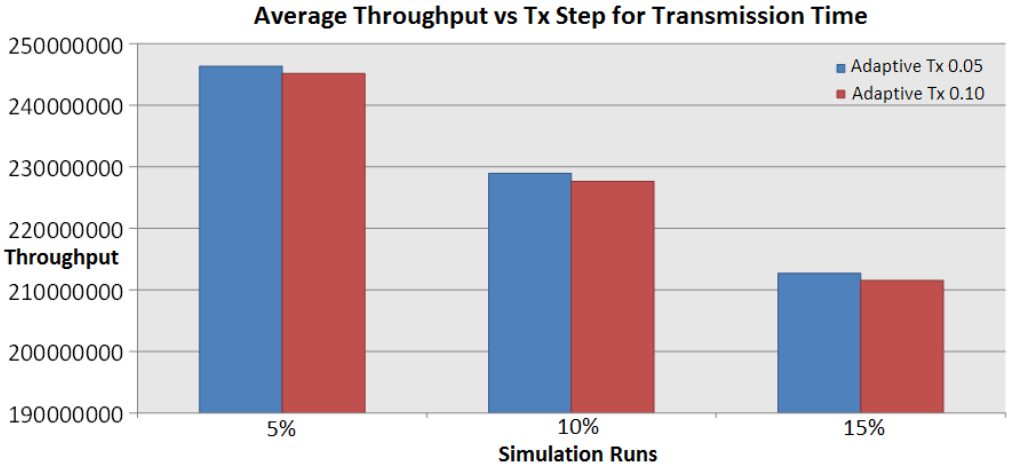

Figure 7.8: SU throughput as a function of T_x step for Transmission Time

the results drawn from sensing time and transmission time evaluation. The SU throughput increases when P_s is between 5% and 15%. It also confirms that T_x value of $0.05s$ works better than $0.10s$.

The adaptive algorithm also aimed at stabilizing the CRAHN SU throughput. When we consolidated the results of the default algorithm, we saw that there were major spikes in the throughput during the simulation runs. Figure 7.10 shows the throughput of the default and adaptive algorithm for a single simulation run. From the figure, we can see that the throughput of the default algorithm shows sudden rises and falls. As a result, we implemented the adaptive algorithm such that it prevents the

throughput from sudden large increase or decrease. Since the changes in sensing and transmission time are done in small steps, the throughput does not see any major spikes. The overall results of the adaptive algorithm are significantly better than that of the default algorithm which used static values for sensing and transmission times. The average rise in throughput for all simulation runs was 4.6% which was negatively affected by simulation runs with P_s greater than 15%. For simulations runs with P_s between 5% and 15%, the average throughput percentage increase was 15.97%. The average across all time intervals had a percent gain of 13.35%. Highest recorded percentage increase in throughput was for time interval of $10000\mu s$. Figure 7.11 shows the adaptive throughput for the five time intervals that were used in the simulations. It shows that for all time intervals the throughput is considerably higher for smaller intervals. Thus, we can infer that a smaller time interval gives the system more opportunities to adapt their sensing time and transmission time. As the time interval increases, the simulation is divided into smaller blocks and hence the adaptive algorithm can do only so much to improve the throughput.

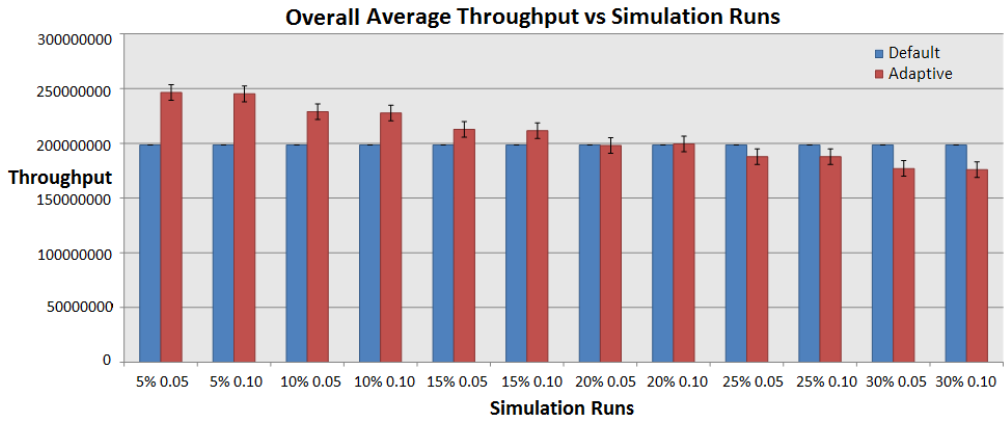

Figure 7.9: Overall Average SU Throughput

From all the graphs, we observe that the evaluations of the adaptive algorithm based on sensing time, transmission time, T_x, P_s and timer interval are consistent with each other and the analysis made in Chapter 5.

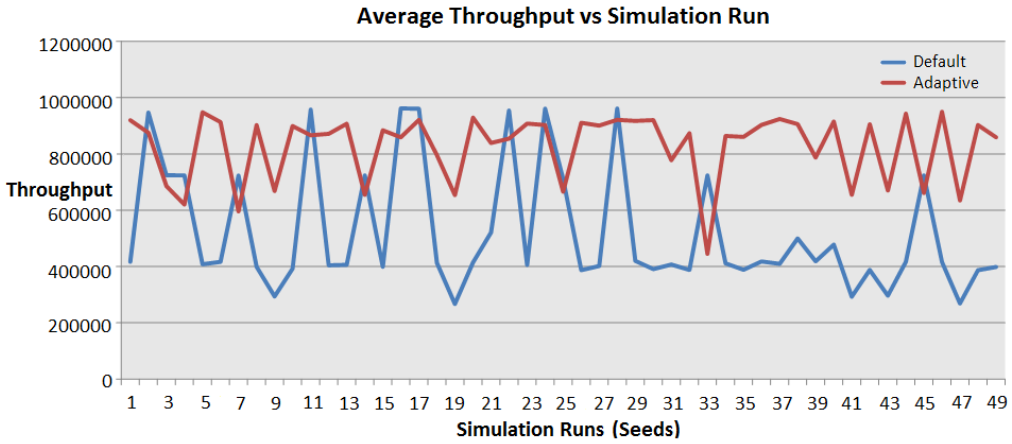

Figure 7.10: Comparison of SU throughput: Default vs Adaptive

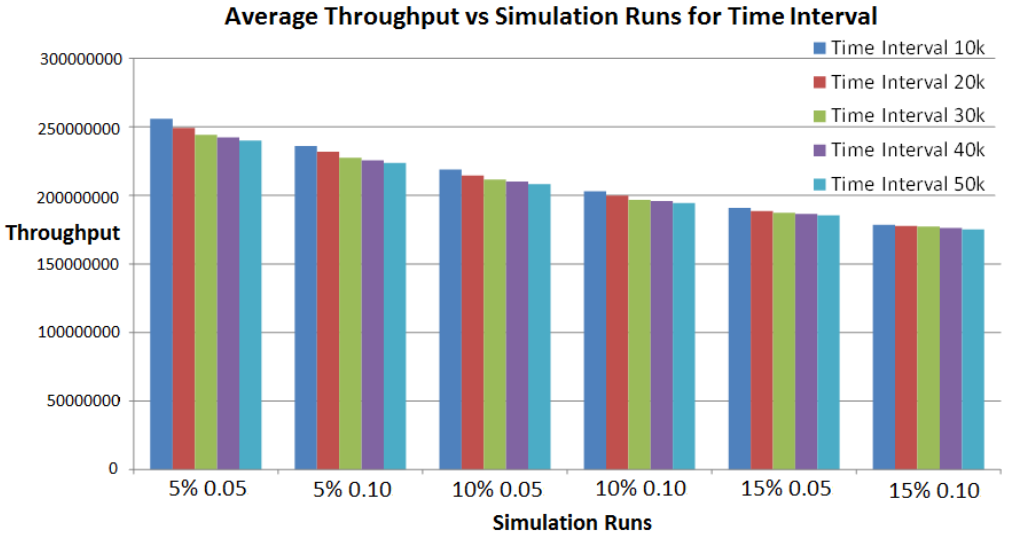

Figure 7.11: SU throughput as a function of Time Interval

We also collated some data to check if the data model given in Equation 5.1 is linear and predicts the SU throughput correctly. Using the intermittent values of the all the parameters, we evaluated the Equation. We calculated the SU throughput as well as the natural log of the SU throughput. Both results suggest that the model is linear and is in accordance with the analysis. Figure 7.12 shows the average SU throughput and the natural log of SU throughput versus the sensing time levels. Figure 7.13 shows the average SU throughput and the natural log of SU throughput versus transmission time levels. Both graphs are linear. Thus the data model works well even for the

intermittent values of the CR parameters.

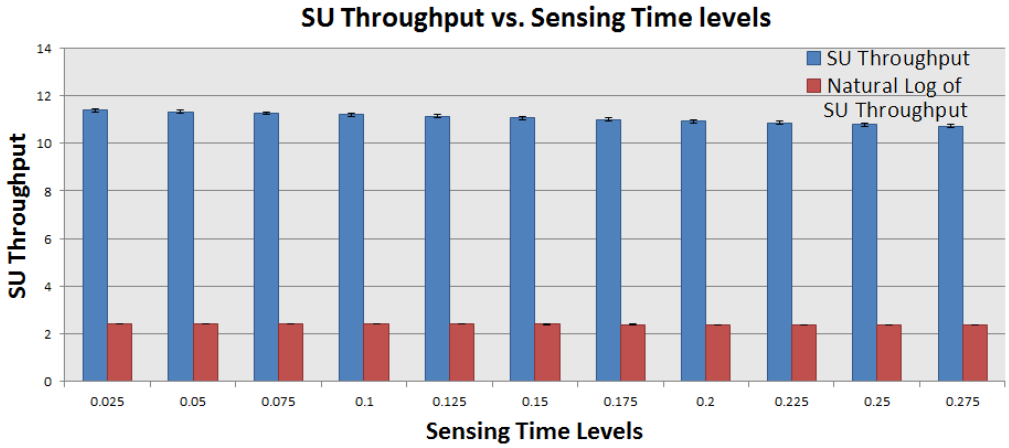

Figure 7.12: SU throughput as a function of Sensing Time using the Data Model

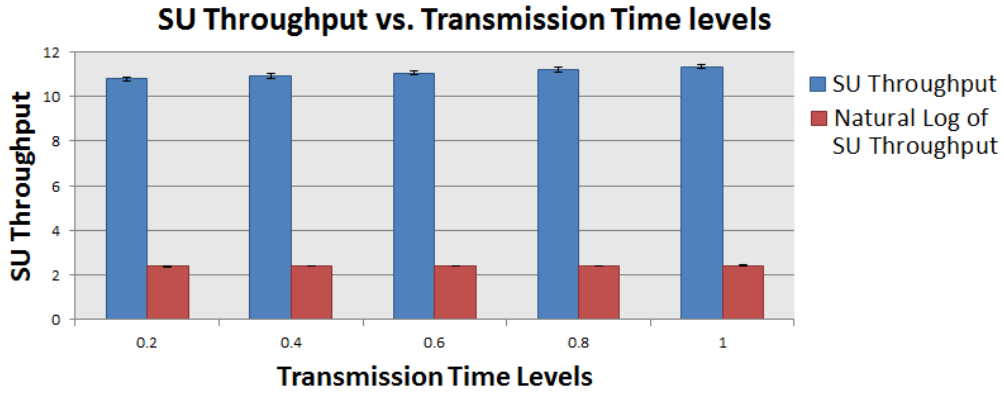

Figure 7.13: SU throughput as a function of Transmission Time using the Data Model

The next Chapter collates the results of the evaluation and concludes the thesis. It also describes a few areas of research that can be explored to improve the efficiency of CR nodes as well the CRAHN throughput.

Chapter 8

CONCLUSION & FUTURE WORK

The aim of this work was to implement the CR nodes such that they can dynamically change their parameters and adapt to the radio environment.

First, we performed data analysis on the default CRAHN implementation. It revealed the interactions among different CR parameters. Based on the observations made during analysis, we suggested the adaptive algorithm. We proposed a novel method to improve the SU throughput by calculating the sensing time proportional to the transmission time. The adaptive CRAHN model dynamically changes the values of sensing time and transmission time depending on the instantaneous throughput measured by the SUs. The adaptive algorithm exploited the interactions of the CR parameters. We then conducted performance analysis of the results obtained after running the simulations using the adaptive model.

The SU throughput of the adaptive algorithm shows a significant improvement over that of the default algorithm. The overall SU throughput for acceptable values of P_s and T_x showed a percentage increase of 15.97, which is considerably high. We were able to show the best range of values for sensing time, transmission time, P_s, T_x, and timer interval. The analysis also helped identify the relation between sensing time and transmission time and how they should be modified to obtain higher SU throughput. The adaptive algorithm is a simple heuristic algorithm that can be used for other CRAHN implementations and modified based on the parameters being observed.

The default CRAHN implementation did not account for the transmission power of PUs. In order to implement accurate sensing techniques, the CRAHN implementation can be changed to sense the PU energy. Based on the changes made to the underlying implementation, the adaptive algorithm can also be modified to accommodate the PU energy. Different techniques can be used for detecting PU

signal. Also, this work focussed on only two CR parameter: sensing time and transmission time. They are many more CR parameters that can be included in the CRAHN implementation. In such a case the algorithm will change based on the significant parameters selected by the data model.

The default model implements a multi-radio, multi-channel ad hoc network. The routing and link layer implementation is done accordingly. To ensure end-to-end delivery of data packets, areas of transport protocol have to be explored. A CRAHN exhibits decentralized network behaviour. In such a case, ensuring end-to-end delivery becomes challenging. Transport protocols for CR networks have been suggested [6]. Also studies are being done to examine if the existing transport protocols work for CRAHNs. Changes in transport protocol will bring in new parameters to be observed which will effectively change any adapting algorithms.

Also, the adaptive algorithm focused only on the CR parameters. CR parameters can be configured in a better and efficient manner if the PU activity can be predicted. Predicting PU activity can help determine how the CR nodes in the network should adapt. One of the future works could involve predicting PU arrival and departure times. A good prediction algorithm can help optimize the CR parameters and improve SU throughput. Prediction algorithms can also help reduce PU interference.

Overall, the adaptive algorithm is a heuristic algorithm. It provides an acceptable range of values for some CR parameters. The algorithm can be tuned and tweaked to establish accurate values of CR parameters that will yield maximum throughput. This implies modifying the algorithm from an heuristic algorithm to an optimizing algorithm. Optimization includes determining combinations of optimal sensing and transmission times and standardizing them, and optimizing the Equation 5.1 described in Chapter 5.

REFERENCES

[1] "Design expert 8." [Online]. Available: http://www.statease.com/dx8descr.html

[2] "Network Simulator Version 2." [Online]. Available: http://www.isi.edu/nsnam/ns/

[3] I. F. Akyildiz, W.-Y. Lee, and K. R. Chowdhury, "CRAHNs: Cognitive Radio Ad Hoc Networks," *Ad Hoc Networks*, vol. 7, no. 5, pp. 810–836, January 2009.

[4] D. Cabric, S. Mishra, and R. Brodersen, "Implementation Issues in Spectrum Sensing for Cognitive Radios," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, vol. 1, November 2004, pp. 772 – 776 Vol.1.

[5] K. W. Choi, "Adaptive Sensing Technique to Maximize Spectrum Utilization in Cognitive Radio," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 2, pp. 992 –998, February 2010.

[6] K. R. Chowdhury, M. Di Felice, and I. F. Akyildiz, "TP-CRAHN: A Transport Protocol for Cognitive Radio Ad-Hoc Networks," in *IEEE International Conference on Computer Communications*, April 2009, pp. 2482–2490.

[7] M. Di Felice, K. R. Chowdhury, and L. Bononi, "Modeling and Performance Evaluation of Transmission Control Protocol over Cognitive Radio Ad Hoc Networks," in *ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, October 2009, pp. 4–12.

[8] M. Di Felice, K. R. Chowdhury, W. Kim, A. Kassler, and L. Bononi, "End-to-end Protocols for Cognitive Radio Ad Hoc Networks: An Evaluation Study," *Performance Evaluation*, vol. 68, pp. 859–875, September 2011.

[9] A. Ghasemi and E. S. Sousa, "Optimization of Spectrum Sensing for Opportunistic Spectrum Access in Cognitive Radio Networks," in *IEEE Consumer Communications & Networking Conference*, January 2007, pp. 1022–1026.

[10] S. Haykin, "Cognitive Radio: Brain-empowered Wireless Communications," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 2, pp. 201 – 220, February 2005.

[11] M. Hong, J. Kim, H. Kim, and Y. Shin, "An Adaptive Transmission Scheme for Cognitive Radio Systems Based on Interference Temperature Model," in *IEEE Consumer Communications & Networking Conference*, January 2008, pp. 69–73.

[12] P. Kyasanur and N. H. Vaidya, "Routing and Link-layer Protocols for Multi-channel Multi-interface Ad Hoc Wireless Networks," *SIGMOBILE Mobile Computing Communications Review*, vol. 10, pp. 31–43, January 2006.

[13] W.-Y. Lee and I. Akyildiz, "Optimal Spectrum Sensing Framework for Cognitive Radio Networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 10, pp. 3845–3857, October 2008.

[14] J. H. Levine, *Introduction to Data analysis: The Rules of Evidence*. Dartmouth College Web Book, 1997. [Online]. Available: http://www.dartmouth.edu/~mss/data%20analysis/002%20table%20of%20contents.html

[15] O. Mehanna and A. Sultan, "Inter-Sensing Time Optimization in Cognitive Radio Networks," *Computing Research Repository*, 2010.

[16] D. C. Montgomery, *Design and Analysis of Experiments*, 6th ed. John Wiley & Sons, Inc., 2005.

[17] V. Srivastava and M. Motani, "Cross-Layer Design and Optimization in Wireless Networks," in *Cognitive Networks*. John Wiley & Sons, Ltd., 2007, pp. 121–146.

[18] W. Zhang, R. Mallik, and K. Ben Letaief, "Cooperative Spectrum Sensing Optimization in Cognitive Radio Networks," in *IEEE International Conference on Communications*, May 2008, pp. 3411–3415.