A HYBRID FEC CODE FOR MITIGATING JAMMING ATTACKS IN FAULT-MODEL-CLASSIFIED COGNITIVE RADIO NETWORKS

A Dissertation

Presented in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

with a

Major in Computer Science

in the

College of Graduate Studies

University of Idaho

by

Victor Balogun

November 2013

Major Professor: Axel Krings, Ph.D.

AUTHORIZATION TO SUBMIT DISSERTATION

This Dissertation of Victor Balogun, submitted for the degree of Doctor of Philosophy with a major in Computer Science and titled "A Hybrid FEC Code for Mitigating Jamming Attacks in Fault-Model-Classified Cognitive Radio Networks," has been reviewed in final form. Permission, as indicated by the signatures and dates given below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor ________________________________ Date____________________

Dr. Axel Krings

Committee Member ________________________________Date____________________

Dr. Gregory Donohoe

Committee Member ________________________________Date____________________

Dr. Jim Alves-Foss

Committee Member ________________________________ Date___________________

Dr. Ahmed Abdel-Rahim

Department Administrator __________________________________ Date__________________

Dr. Gregory Donohoe

Discipline’s College Dean __________________________________Date__________________

Dr. Larry Stauffer

Final Approval and Acceptance by the College of Graduate Studies:

__________________________________Date__________________

Dr. Jie Chen

ABSTRACT

Opportunistic sharing of spectrum through the concept of Cognitive Radio network (CRN) has been considered the next stage in the effort of the government to alleviate the spectrum shortage problem. The idea is to salvage spectra that are rendered fallowed by the licensed (primary) users during their on and off period. Recent literature reiterated that this next generation wireless network is capable of providing ubiquitous connectivity of high-bandwidth communications to both urban and rural communities. It has diverse application domains ranging from distance education to other government-delivered essential services like remote health care delivery and emergency rescue response. The security requirements of CRN must be specified because of the sensitive nature and variety of application domains of the network. Violation of any of the requirements, either maliciously or inadvertently, by users like jammers constitute security threats that will significantly affect the trust and acceptance of the public and further use of the technology. The task of mitigating malicious attacks, especially jamming in this type of network, is very challenging. This is because potential attackers could be operating as Cognitive Radios themselves, and as a result they are capable of preying on the adaptable and reconfigurable features of CRN with the intent of causing serious denial of service to the users of the network. In addition, it will be shown that these jammers are capable of introducing value faults in pathological cases.

In this research, we first classified CRNs using a hybrid fault-model taxonomy that considers different fault types including omissive and transmissive value faults. Through simulation, we investigated the performance of CRNs operating in the presence of jamming attacks that are capable of introducing value faults. We observed through performance evaluation like Packet Delivery Ratio (PDR) that these jammers are very effective in their operations and they could bring down the entire CRNs, leading to close to zero packet being delivered by the networks to the destination, even when the rate of jamming is just about 30%. We also discovered that none of the existing solutions are able to mitigate these malicious attacks. As a result, a hybrid forward error correction (FEC) code is proposed that incorporates data integrity checking into an efficient forward error correction mechanism.

The new hybrid FEC code is defined as the concatenation of the Raptor code and the Secure Hash Algorithm-2 (SHA-2). We use the Raptor code part of the code to recover data loss, since Raptor code is an FEC code that allows encoded communicated data to be recovered at the destination without the need for a re-transmission, with high probability even when some of the data is lost due to jamming. The SHA-2 part of the code is used to verify the integrity of data received at the destination since CR jammers are capable of manipulating transmitted data. This is possible because SHA-2 can transform any set of data elements into a unique fixed length hash value, which can be verified at both the sender and the receiver for any data manipulation by CR jammers. When data communicated is detected to be lost or manipulated, the approach explores a recovery block based on the application scenario.

The algorithm for our proposed solution is presented and its validity and threshold functions are established. We simulate our proposed solution in NS-2 and evaluate its performance by comparing it with ordinary FEC code implementation. The result of the analysis using suitable performance metrics shows that the proposed solution is very efficient and robust against the different rates of jamming perpetrated by CR jammers in a fault-model classified CRN. In addition, the encoding and decoding algorithm of the proposed hybrid FEC code was found to be very efficient because of the observed high recovery rate of the algorithm capable of providing consistent low packet loss ratio (PLR), even at the jammers worst case scenario of 100% jamming rate.

ACKNOWLEDGMENTS

I would like to thank the following people and groups:

My supervisor, Dr. Axel Krings for his mentoring and invaluable contribution to the success of this work. He was always there to give me direction many times when I was lost at what to do next.

My committee members, Dr. Gregory Donohoe, Dr. Jim Alves-Foss, and Dr. Ahmed Abdel-Rahim for their comments, guidance and patience in completing this dissertation.

My wife, Deborah, son Nicolas and daughter Michelle for their support and encouragement towards the completion of this work.

University of Idaho, Computer Science faculty, staff and students for how they have impacted my life and studies and for providing enabling environment for me to work.

DEDICATION

I would like to dedicate this dissertation to the Giver of life and wisdom: God the Father, God the Son, and God the Holy Spirit, in Him I live and move and have my being.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

The following abbreviations were used in this dissertation:

AODV - Ad hoc On-Demand Distance Vector

CCC - Cognitive Control Channel

CPU - Central Processing Unit

CRCN - Cognitive Radio Cognitive Network

CRN - Cognitive Radio Network

CSS - Cooperative Spectrum Sensing

CTS - Clear to Send

DoS - Denial of Service

DSA - Dynamic Spectrum Access

DSDV - Destination-Sequenced Distance Vector

DSSS - Direct Sequence Spread Spectrum

FCC - Federal Communications Commission

FEC - Forward Error Correction

FH - Frequency Hopping

IDD - Iterative Decoding Delay

IPTV - Internet Protocol Television

ISM - Industrial, Scientific and Medical

ITU - International Telecommunication Union

LDPC - Low Density Parity Check

LT - Luby Transform

NS-2 - Network Simulator 2

OSI - Open System Interconnect

MAC - Medium Access Control

MACAW- Multiple Access with Collision Avoidance for Wireless

MD - Message Digest

MTU - Maximum Transmission Unit

PDR - Packet Delivery Ratio

PLR - Packet Loss Ratio

PU	Primary User
PUE	Primary User Emulation
QoS	Quality of Service
RSD	Robust Solition Distribution
SHA-2	Secure Hash Algorithm 2
SNR	Signal-to-Noise Ratio
SSDF	Spectrum Sensing Data Falsification
SU	Secondary User
Tcl	Tool Command Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VOD	Video on Demand
VoIP	Voice over Internet Protocol
WCETT	Weighted Cumulative Expected Transmission Time
WLAN	Wireless Local Area Network

Chapter 1

INTRODUCTION

1.1 General Introduction and Motivation

Cognitive Radio (CR) [3], a concept which allows incumbent (primary/licensed) users to share their spectrum with unlicensed (secondary) users in a non-interference manner, has been regarded as a way out of the problem of spectrum shortage/underutilization [33]. This problem emanates from the fact that most of the frequency spectra has been allocated to licensed users, even though these users do not necessarily occupy these frequencies all the time. Examples of these licensed users are the radio and television stations. The free spectra that are not licensed are congested because most of the mobile devices operate at these free frequencies. With the CR technology, the unlicensed users can opportunistically make use of the licensed spectrum during the OFF period of the incumbents, or use some portions of the spectrum that are not in use even when the primary users are present. But the unlicensed users must vacate the spectrum immediately when the incumbents become present at those frequencies.

Accurate sensing of spectrum for the presence of the incumbents is a very crucial issue in CR networks. This is because detection could be significantly affected by impairments such as shadowing, resulting from obstacles shielding signals from intended receivers, and multipath fading, which is signal fading resulting from signal components reaching the receivers through multiple paths. Cooperative spectrum sensing (CSS) [4], which allows cooperating CR nodes to share their sensing information about the presence or absence of an incumbent in a CRN environment, has been proposed and found to be effective in mitigating these impairments. In spite of the anticipated success and potentials of CSS, the presence of malicious users like jammers operating in the neighborhood of a CSS Cognitive Radio network, is of great security concern. In order for the CRN to be successful in its mission, its security requirements must be specified. Violation of any of the requirements by a legitimate user or a malicious user like a jammer will pose as a security threat to the technology. This will serve as a deterrent to continued usage of the network by the public. Central to this security concern is that CRs have flexible and reconfigurable features that make them easy prey to security threats like jamming attacks [35], Primary User Emulation (PUE) attacks [52], masquerading of CR [18], Spectrum Sensing Data Falsification (SSDF) attacks [53], and many more that are either specific to CRs or inherited from traditional wireless networks.

Jamming, which is the main focus of this research, has been addressed by different authors, e.g., [28, 35, 42, 49], with respect to traditional wireless networks, and different anti-jamming techniques have been proposed. However, there is a need to investigate the performance of CRs in the presence of jammers, as CR networks differ significantly from traditional wireless networks. The reasons are the fact that CRs operating as jammers are able to take advantage of their reconfigurable features to manipulate spectrum sensing data being shared among the cooperating CRs, and that the jammers are also able to prey on the adaptable features of legitimate CRs with the intent of causing serious DoS. Though several authors [9, 26, 27, 44] have published work on jamming in CR networks, to the best of our knowledge none have studied jamming in the context of fault model classifications (including value faults) and their respective fault handling.

1.2 Objectives and Research Contributions

The main objective of this dissertation is to provide an enhanced security solution that is capable of mitigating jamming attacks in Fault-Model-Classified CRNs, especially to jammers that are capable of introducing value faults in pathological cases. Specifically, this dissertation has the following five objectives:

Objective #1: To formulate a jamming taxonomy that considers transmissive and omissive value faults.

Objective #2: To study (through simulation) the different jamming types that consider hybrid fault models with the intent of measuring and analyzing their impact on CR Networks.

Objective #3: To investigate possible solutions for this jamming adversarial model.

Objective #4: Formulation of an enhanced solution that is capable of mitigating jamming in

Fault-Model-Classified CRNs considering value faults.

Objective #5: Evaluate the performance of the proposed solution in all the jamming scenarios identified in objective #1

1.3 Outline

The next chapter (Chapter 2) will cover the background information. Chapter 3 addresses objective #1 introduces a jamming taxonomy for CSS Cognitive Radio networks that considers transmissive and omissive value faults. Chapter 4 addresses objective #2 by quantifying through simulation the effect of different jamming types on CSS Cognitive Radio networks. Objectives #3 #4 and #5 are addressed in Chapter 5 as the intended solution strategy for jamming that considers hybrid faults and a hybrid code that is capable of mitigating this type of fault is proposed. In addition, an evaluation of the performance of the proposed solution is presented. Finally, Chapter 6 provides a summary of the dissertation and future work. The remaining part of the dissertation is the references of cited work.

Chapter 2

COGNITIVE RADIO NETWORKS

2.1 Evolution and Definition

The beginning of Cognitive Radio network (CRN) can be traced to the concept of Software-Defined Radio (SDR) [2], which was defined by the Federal Communications Commission (FCC) [1] as one in which operating parameters such as frequency range, modulation type, or output power can be altered by software without making any changes to the hardware components. Today's usage of the term Cognitive Radio is normally referred to mean a radio system that has the cognitive ability to sense its radio frequency environment (Spectrum Sensing), analyze the spectrum (Spectrum Analysis), take decision based on the analysis (Spectrum Decision), and adapt itself accordingly to benefit from this cognitive ability (re-configurability or adaptability). CRN was also described as a next generation wireless network that is computationally intelligent in that it is able to explore radio resources available in its neighborhood, able to determine the communication needs of the users of this network, and also able to provide radio resources and wireless services that are most appropriate to meet those needs [2].

The recent burst in multimedia applications' usage and their communications have made the need for increased wireless spectrum unabated. The free unlicensed bands, e.g. Industrial, Scientific, and Medical (ISM) bands at 900 MHz, 2.4 GHz, and 5.8 GHz, have been said to be congested because most mobile devices using WLAN and Bluetooth operate at these free frequencies. These factors have significantly affected the free spectra available for data transmission. Despite this shortage, the report [1] published by the FCC asserts that large portions of the licensed spectrum are underutilized. Examples of services that use frequencies that are licensed by the FCC are radio stations, broadcast television stations, and telecommunications providers. In its report, the FCC proposed a better approach to spectrum management that will allow the unlicensed users to share the licensed spectrum with the incumbents in a non-interference manner. The term "Dynamic Spectrum Access" (DSA), as opposed to static spectrum access, is used to describe this dynamic approach to spectral access and management. This report fuelled an upsurge in research to make this new spectrum management approach a reality. As a result, CRN was proposed in [2] as a technology that will serve as a key player towards achieving the DSA requirements of the proposed technology. The Cognitive Radio technology will allow unlicensed users to share the unused part of the licensed bands of the spectrum with the incumbents when present (Underlay spectrum sharing approach), or during the incumbents' OFF period (Overlay spectrum sharing approach). The cognitive radio network standard does not allow any level of interference to the incumbents, i.e., the CRs must be able to detect the presence of a primary user and immediately vacate the spectrum (Spectrum mobility) whenever a primary user appears in the spectrum occupied by a CR user. The standard also does not permit any level of modification to the incumbents' networks. The benefits of CRNs range from providing low cost Internet access by reducing the substantial cost component associated with the purchase of spectrum to providing access to education or health in underserved and rural regions.

The remaining sections of the chapter present the relevant background information that are pertinent to the successful deployment of CRN that is capable of operating in a hostile environment, especially in the presence of malicious users. The next section describes the security requirements of CRNs, followed by a section that describes the different CSS architecture proposed for CR networks. The remaining sections provide an overview of some of the challenges that pose as great threat to CRNs operating in an adversarial environment.

2.2 Security Requirements of CR Networks

Some security requirements are specifically pertinent to the success of CRNs. The following requirements are adapted from the "Telecommunication Networks Security Requirements" by International Telecommunication Union (ITU) in [6] to suit Cognitive Radios:

- Controlled access to resources - Secondary Users (SUs) should only have access to those resources that they are authorized to access and no more.
- Robustness - CRN should be robust against any threat from malicious users.

- Protection of confidentiality - CRN must ensure the confidentiality of communicated (sensed) data.
- Protection of data integrity - CRN must guarantee the integrity of communicated data.
- Compliance to regulatory framework - CRN must comply to non-interference to incumbent regulatory requirement.
- Accountability/Non-repudiation - CRN must ensure that the SUs cannot deny responsibility of any of their activities.
- Verification of identities - CRN must be able to verify the identity of SUs.

Violation of any of these stated requirements by CRN legitimate or malicious users like jammers will pose as a security threat to the technology. Furthermore, it will significantly affect the trust of the public in the continued usage of the network. The CRN should therefore be designed in such a way that it is prepared to handle any violation of its requirements by any of its users.

2.3 CSS Architecture in CR Networks

Cognitive Radios that sense spectrum in an isolated or non-cooperative manner have been found to be ineffective in attempting to share the spectrum with the incumbent in a non-interference manner [11, 12, 13]. Channel impairments like deep shadowing [62, 60], resulting from obstacles shielding signals from intended receivers, and multipath fading [55], which is signal fading resulting from signal components reaching the receivers through multiple paths, are some of the reasons why a secondary user might not be able to sense the presence of an incumbent without assistance from other CRs operating in its neighborhood. Cooperative spectrum sensing (CSS) [4, 14, 11] has been proposed and considered to be effective in handling these impairments. In [4] three different CSS architectures for CR networks based on the way that cooperating CRs share the sensing information among themselves in the network are identified. These architectural types are briefly discussed below:

2.3.1 Centralized Cooperative Sensing CR Networks

This is an architecture where the cooperating CR nodes share their sensing information via a centralized infrastructure known as the base station or fusion center. The fusion center is responsible for aggregating the sensed data from all the cooperating nodes and making the final decision about the presence of an incumbent. Centralized CSS is illustrated in Figure 2.1.

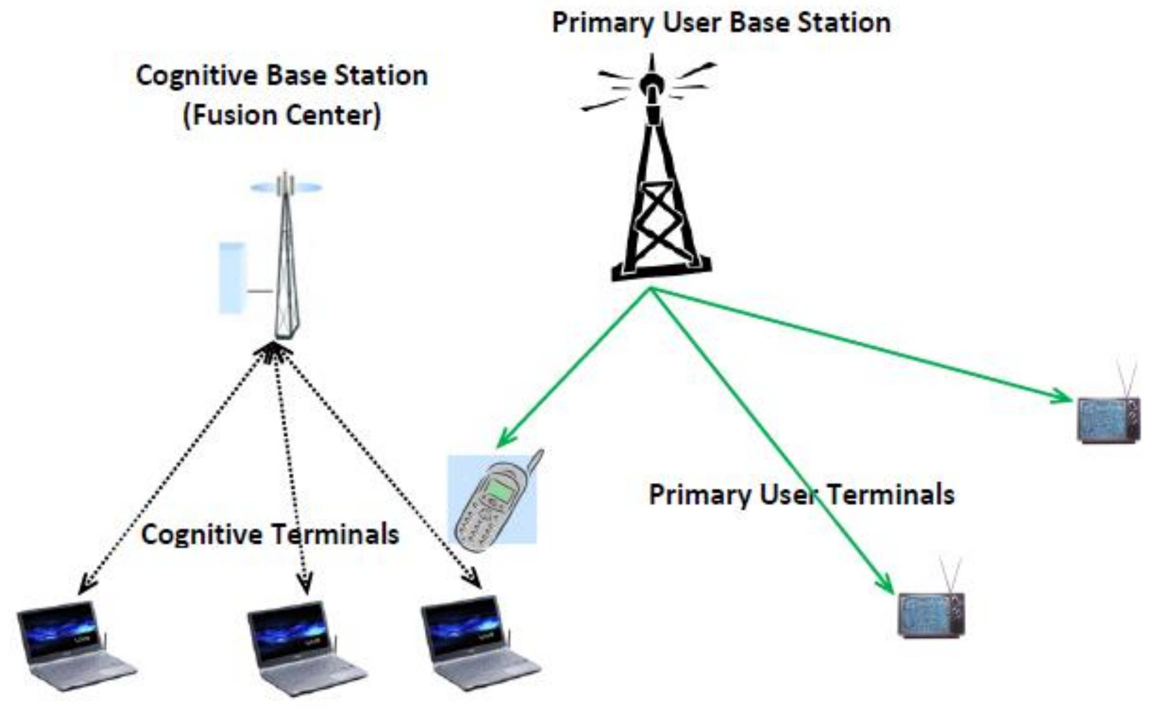

Primary User Base Station
Cognitive Base Station (Fusion Center)
Primary User Terminals
Cognitive Terminals

Figure 2.1: **Centralized Cooperative Sensing CR Networks**

In this figure, the Cognitive Radio terminals (CR1, CR2, CR3, CR4 and CR5) communicate with the cognitive base station about the status of the spectrum whether the primary user (PU) of the spectrum is present or absent. The cognitive base station, which is the fusion center, aggregates the information received from the cognitive terminals and uses some agreement algorithms to make a final decision of the status of the primary user. This final decision will then be communicated to the cognitive terminals, who in-turn responds accordingly either to vacate the spectrum if the PU is present or to continue to use the spectrum if the PU is absent. The PU, when present, uses the spectrum to communicate only with the primary user base station. There is no interaction between the Cognitive Radio users and the primary users.

2.3.2 Distributed Cooperative Sensing CR Networks

The distributed CSS is illustrated in Figure 2.2. As can be seen from the figure, the cooperating CR nodes (CR1, CR2, CR3 and CR4) share their sensing information without a centralized infrastructure or base station like the case of the centralized CSS CR networks. A CR node might be designated as a fusion center where sensing information is collected and the final decision is communicated to the cooperating nodes. Similarly, each of the nodes might take turn in acting as the fusion center and then a distributed agreement algorithm can be used to make a final decision. The latter design removes the workload from a single node, which is of interest especially where power consumption is a factor. There is also no interaction between the CR users and the primary users.

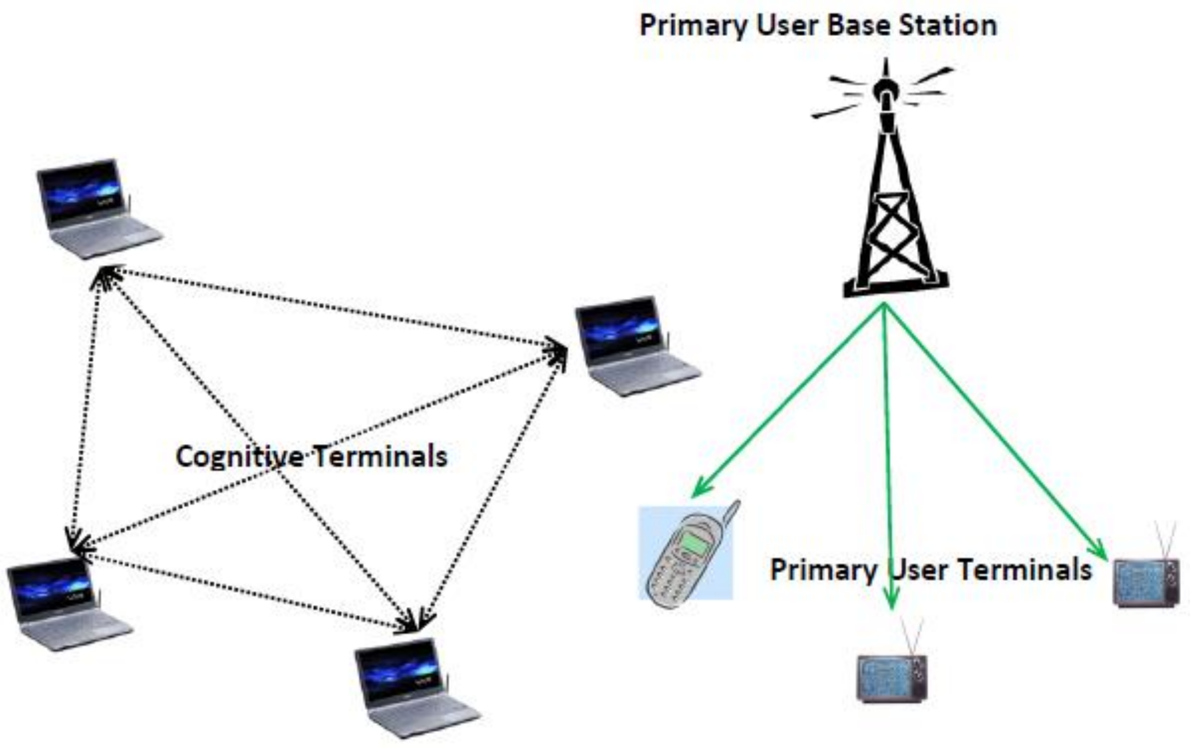


Figure 2.2: **Distributed Cooperative Sensing CR Networks**

2.3.3 Relay-Assisted Cooperative Sensing CRNs

The Relay-Assisted Cooperative Sensing CR network is kind of an extension provided to the Centralized CSS CRNs to accommodate nodes that are more than one-hop from the centralized base station. In this configuration, as illustrated in Figure 2.3, the spectrum sensing information and decisions are communicated to CR nodes (CR1 and CR4) that are more than one hop from the designated fusion center through multi-hop relays. There is also no interaction between the CR users and the primary users.

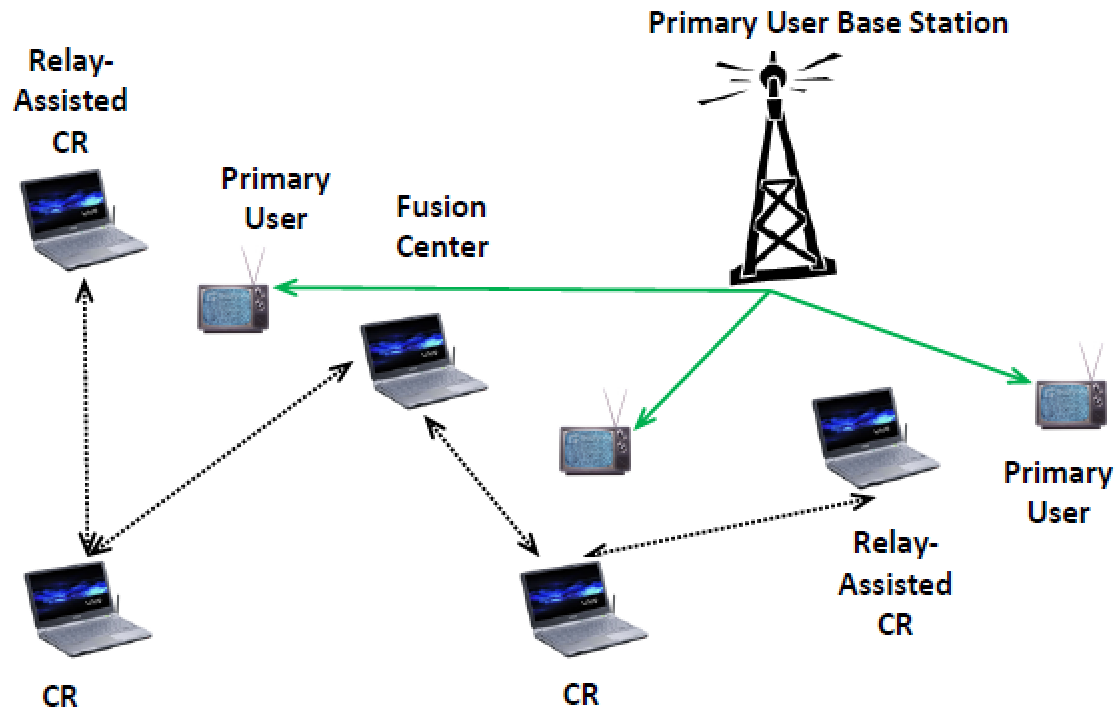


Figure 2.3: **Relay-Assisted Cooperative Sensing CR Networks**

2.3.4 Combined CSS Architectural Model for CRNs

We specify another category of CSS CRN where one considers the combination of all three CSS types earlier mentioned into a single architecture. Figure 2.4 shows such combination into what we formulated as the combined CSS architectural model for CRNs. Our intention is to present the CR users with the flexibility of being part of any of the three architectural types depending on their location in the network. CR nodes (CR6, CR4) far away from the fusion center can send and obtain result of sensing from the fusion center either through other nodes (CR2) who are connected in a distributed architecture, or through a node that is acting as a CR relay node (CR5). As can be seen in the figure, nodes that are one-hop away from the fusion center (CR1, CR2 and CR3) will simply communicate directly with the fusion center. The advantage of this architecture stems from the fact that, as a CR moves around in a CRN environment, it is not constrained by an CSS architecture but has the choice to be part of any architecture that is best suited for it to send and receive sensing information from the fusion center depending on its location.

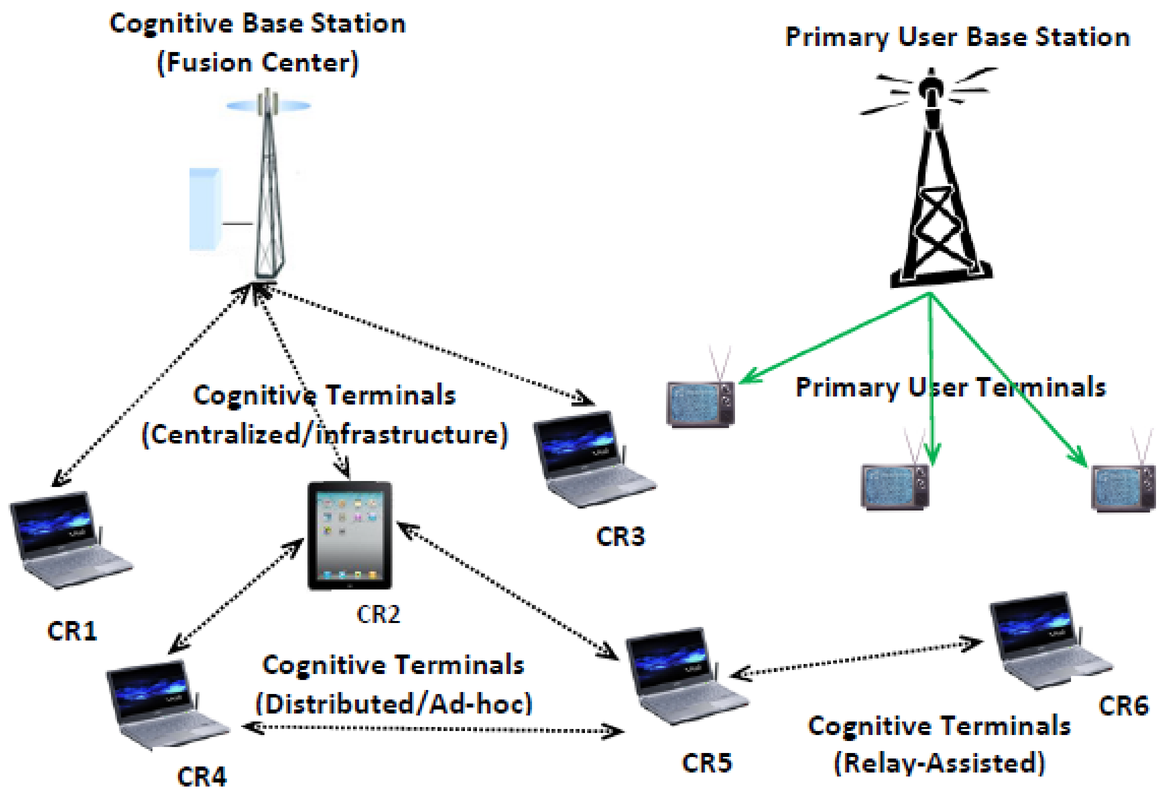


Figure 2.4: **Combined Architectural Model for CSS in CR Networks**

2.4 Challenges of CSS CR Networks

In spite of the anticipated success and potentials of CSS, the presence of malicious users like jammers operating in the neighborhood of a CSS CRN is of great security concern. Central to this concern is that CRs have flexible and adaptable features that make them easy prey to security threats like jamming, primary user emulation (PUE) attacks, Spectrum Sensing Data Falsification (SSDF) attacks and many more that are either specific to CRs or inherited from traditional wireless networks. Some of these attacks are briefly discussed below while jamming, which is the main focus of this research, will be discussed in detail in the next section.

2.4.1 Primary User Emulation (PUE) Attack

This is a type of attack in which a malicious CR tries to deceive other CRs into believing that a primary user is present, while in reality the primary user is absent. The intention of the malicious CR node is to gain an undue advantage of the use of the spectrum. In order to perpetrate this malicious act, the malicious CR transmits signals that emulate the characteristics of that of the primary user. When other CR users sense the spectrum, they will detect signals that resemble that of the primary users and they will come to a conclusion that the primary user is present. If CR users are already transmitting, they will vacate the spectrum as specified in the CRN standard. On the other hand, if the CR nodes are just intending to transmit, they will not transmit as long as the malicious node keeps on transmitting signals that resemble that of the primary user [52, 67, 64]. Figure 2.5 is a representation of this attack in which three CRs in a centralized CSS are prevented from using the spectrum even when the primary users are absent because the malicious node transmits signals with the same characteristics as that of the primary users.

2.4.2 Spectrum Sensing Data Falsification (SSDF) Attack

This is a type of attack in which a malicious CR node tries to inject false spectrum sensing and spectrum sharing information to the CRN's fusion center in a centralized CR CSS architecture, or to other CR nodes in a distributed CR CSS architecture [53, 52]. The intention of such malicious node is to gain an unfair advantage by lying that a primary user is present when it is not present. The malicious node might just inject false information to disrupt the network. This type of attack becomes very effective when the malicious node's sensing information constitutes a higher percentage of the total aggregated sensing information from other CR nodes. This might be due to its close proximity to the primary user compared to the location of other CR nodes in the network. The SSDF attack is also referred to as masquerading CRs [18] because the malicious node is assumed to be in disguise or pretending to be a good node, while in the real sense it is malicious in its activities.

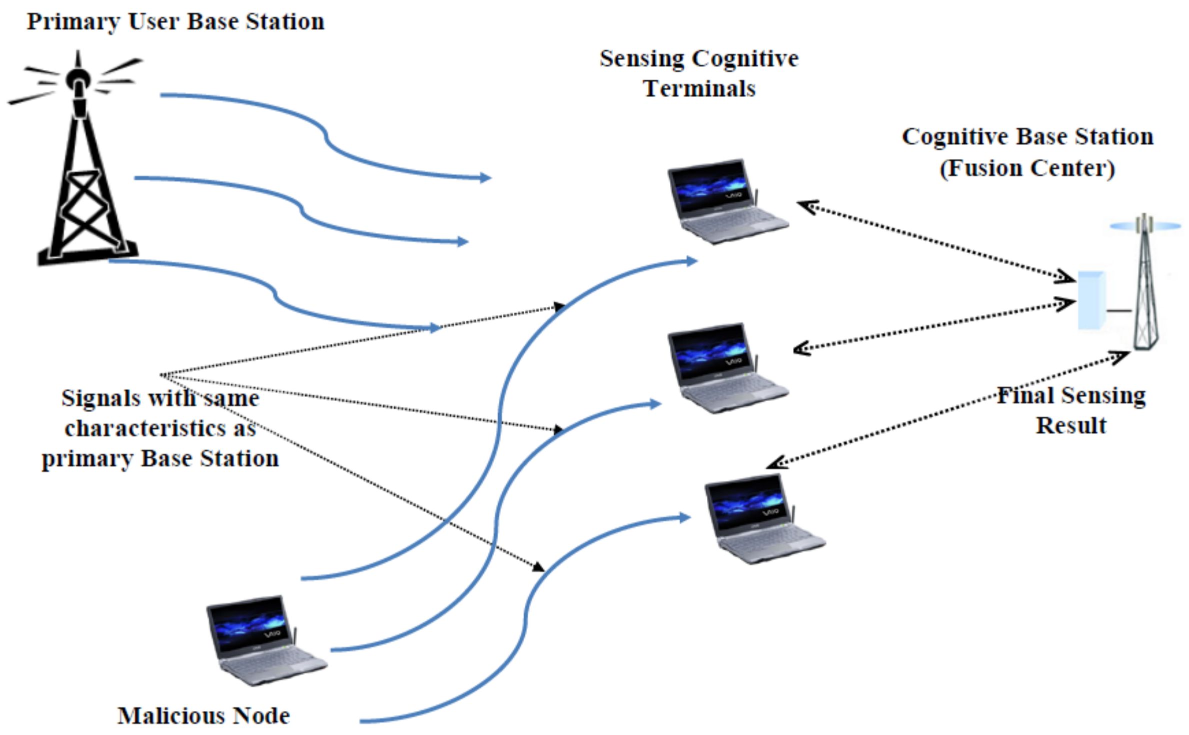

Primary User Base Station
Sensing Cognitive Terminals
Cognitive Base Station (Fusion Center)
Signals with same characteristics as primary Base Station
Final Sensing Result
Malicious Node

Figure 2.5: **Primary User Emulation Attack in CSS in CR Networks**

Figure 2.6 depicts a typical scenario of the SSDF attack. Nodes CR1, CR2, CR3 and CR4 are cognitive radio nodes that share their sensing information through the fusion center. If CR4 becomes malicious, it will be able to negatively influence the aggregated sensing information at the fusion center, because of its closeness to the primary user base station.

2.5 Jamming Attacks

Jamming is a deliberate or unintentional transmission of radio signals that causes disruption to communication either by reducing the ratio of the signal of a legitimate user to the surrounding noise (signal-to-noise ratio (SNR)), or by injecting faults into the signal causing data error or manipulated data [45, 66]. Jamming is unintentional if a legitimate sender transmits signals without the knowledge that there is an on-going transmission. This could be due to the node not being able to perceive signals from other sources because of its location or because it fails to obey the MAC protocol guiding the use of the channel. A jamming is intentional if it is done with a malicious intent to cause denial of service or to gain undue advantage over other users of the channel by introducing value faults. In this research, we only consider the categories of jamming that are done with malicious intent.

The remaining of this section discusses jamming attacks both in traditional wireless networks and the CRNs.

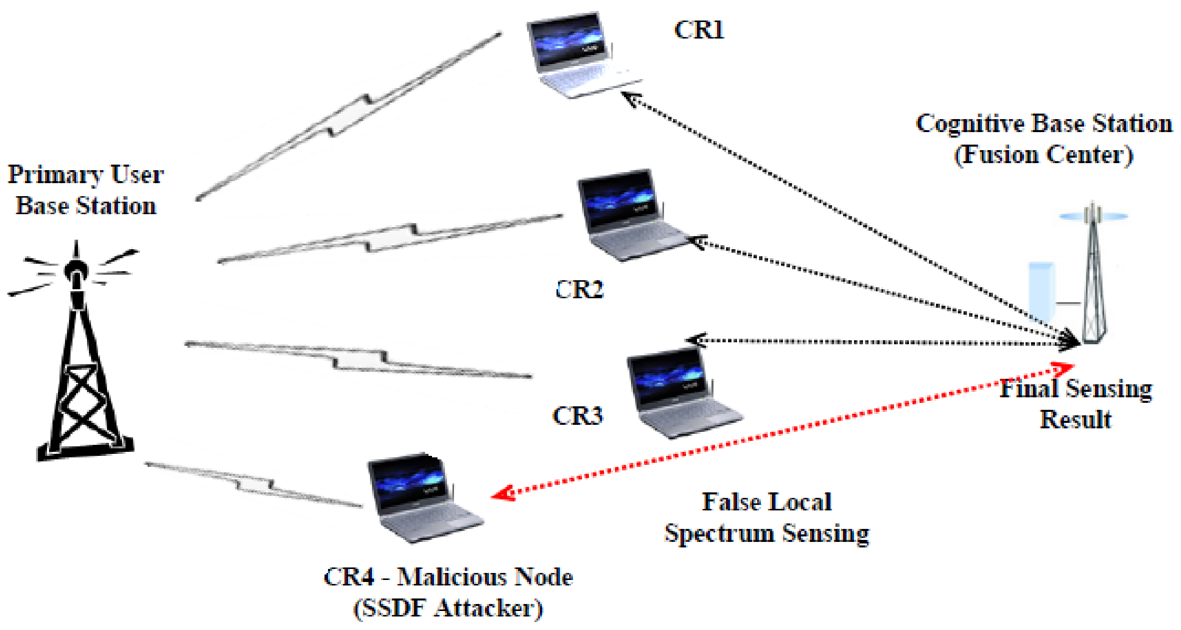


Figure 2.6: **Spectrum Sensing Data Falsification Attack in CSS CR Networks**

2.5.1 Traditional Jamming

Jammers in traditional wireless communications can cause denial of service (DoS) at either the transmitter or the receiver if they adequately inject interfering signals into the same spectral region as that of the legitimate users [28]. Jammers use several techniques to perpetrate their malicious act. They might target a specific frequency (spot jamming), sweep across available frequencies (sweep jamming), or jam a range of frequencies at once (barrage jamming). They could even jam in a coordinated way [41], where jammers collaborate to gain the knowledge of the network with the intent of efficiently reducing the throughput of the network. Different classes of jammers have been identified by [35, 49]. These include:

Constant Jammer

This jammer emits jamming (useless) signals continuously on a specific channel without obeying the MAC protocol, e.g., MACAW [57], thereby preventing a legitimate user who is obeying the protocol from transmitting.

Deceptive Jammer

This jammer, like constant jammer, continuously transmits jamming signals; however, they seem similar to regular data packets from a legitimate user. For this reason, other users of the network are deceived into believing that there is an on-going legitimate transmission. The level of denial of service caused by this type of jammers could be enormous if the jamming is undetected over long periods of time.

Random Jammer

This class of jamming involves a malicious node alternating between jamming and sleeping, rather than jamming continuously like the deceptive or constant jammer. There are two categories of random jammers: (i) Random jammers that behave like constant jammers, alternating between sleeping and waking as they jam, but they send random (useless) signals during their wake time. (ii) Random jammers that behave like deceptive jammers, sending regular packets during their wake period. Another feature of this jamming type is that the sleep and wake period could be specified to be either fixed periods (in seconds) or random values. When power consumption of a jammer during the wake time is considered along with the jamming efficiency of the jammer, the sleep and the wake period can be used as threshold values. This is because a jammer uses more power when awake than when asleep. The jammer is also more efficient in the level of denial of service it causes if it is more awake than asleep. Therefore, through experiments, a threshold value can be reached that minimizes power usage but at the same time maximizes jamming efficiency.

Reactive Jammer

This category of jammer is very efficient in that it is not easily detected. The jammer transmits jamming pulses only when it finds the channel to be busy. This is to cause collision to any on-going transmission. Since the jammer does not transmit when the channel is free, but transmits when the channel is busy, legitimate CR nodes will be unable to detect the activity of such a jammer. This is based on the assumption that a CR node believes that all other nodes are obeying the MAC protocol, and therefore no other node is transmitting at the same time.

2.5.2 Jamming in Cognitive Radio Networks

A cognitive radio operating as a jammer is able to cause DoS to all the CSS architectural types discussed earlier depending on its location and power. Figure 2.7 is an illustration of how jamming could be done in a combined CSS cognitive radio network. The impact of the jamming node is shown in the figure by the communication it impacts. As depicted in the figure, the jamming malicious node was able to transmit jamming signals to all the CRs operating within the reach of its signal depending on the signal power. The jamming node also jams signals from the primary user base station because it has no regard to obey the non-interference to primary user requirement specified in the CRNs standard.

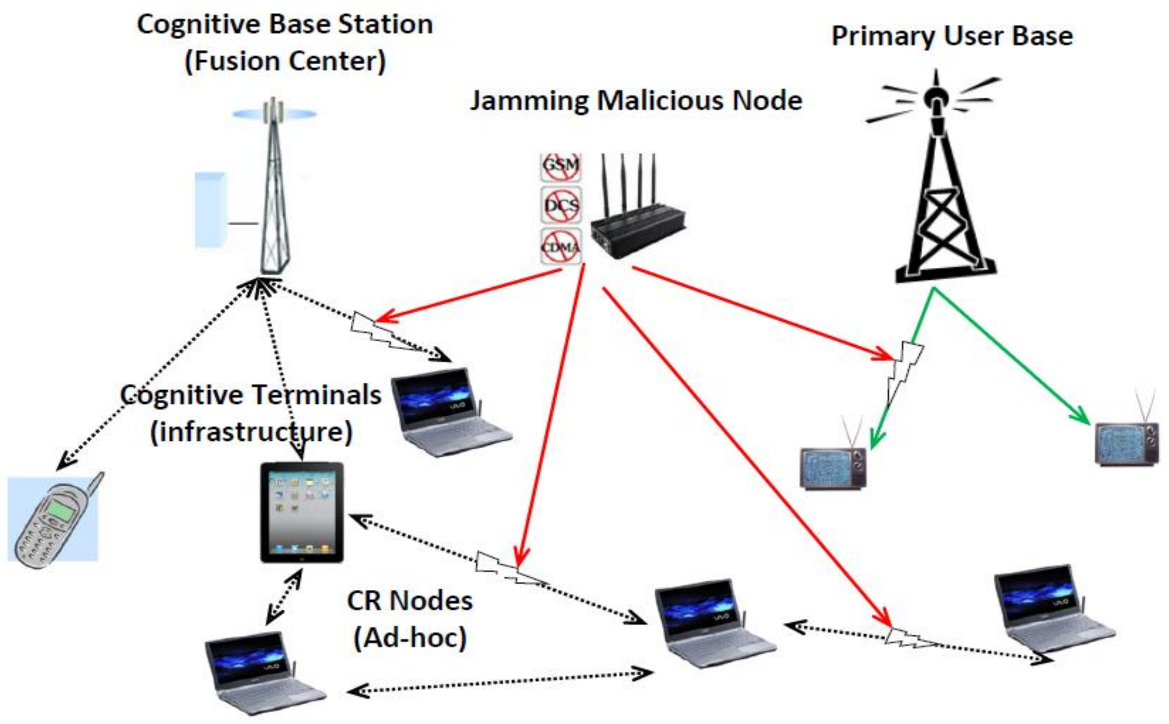


Figure 2.7: **Combined Jamming Scenario in Cooperative Sensing CR Networks**

2.5.3 Discussion

In this chapter, we provide the background of relevant topics to establishing the objectives of this dissertation. We discussed the concept of Cognitive Radio network by highlighting its evolution, motivation, security requirements, architectural taxonomy and challenges. We introduced the subject of our focus for this research, which is jamming, with the intent of preparing the reader for the jamming classification that we presented in the next chapter.

Chapter 3

A FAULT MODEL-BASED TAXONOMY FOR JAMMING ATTACKS IN CSS COGNITIVE RADIO NETWORKS

Cognitive Radio jammers are more difficult to handle than the traditional jammers, not only because of the level of DoS they can cause, but also because of the high level of difficulty of detecting them. Closely aligned to this is the fact that they are capable of manipulating or corrupting data sent and received in a cognitive radio system by introducing value faults [7]. The mechanisms used for handling jamming resulting in data loss do not necessarily work for jamming leading to data manipulation. Therefore, it is desirable to classify the CRNs into categories of faults, considering benign faults and those that consider value faults. In this chapter, jamming and its impact on CR are presented in the context of hybrid fault models and a classification is given that considers transmissive and omissive value faults, which will be defined next.

3.1 The Fault Models: An Overview

Different jamming scenarios can result in different fault types, classified by fault models. A fault model [15] captures the behavior of faults and identifies levels of redundancy needed to tolerate a single fault type or combinations of fault types. The evolution of classifications of faults is shown in Figure 3.1, which describes the different refinements to fault models that were made over the years. The simplest fault model by Lamport [31] considers only one fault type. This model does not make any assumption about the behavior of a fault and considers all faults to be simply malicious or asymmetric. An asymmetric fault, also called Byzantine fault, makes no assumptions on fault behavior. This classification is represented with the first level of Figure 3.1. Combinations of fault types have been addressed using hybrid fault models. Meyer and Pradhan in [16] partitioned all types of faults into two categories: benign faults, which are faults that are self-incriminating as they are obvious to all non-faulty receiving processes, and malicious faults, which are all other faults that do not qualify as benign. This classification is depicted by the second level of Figure 3.1.

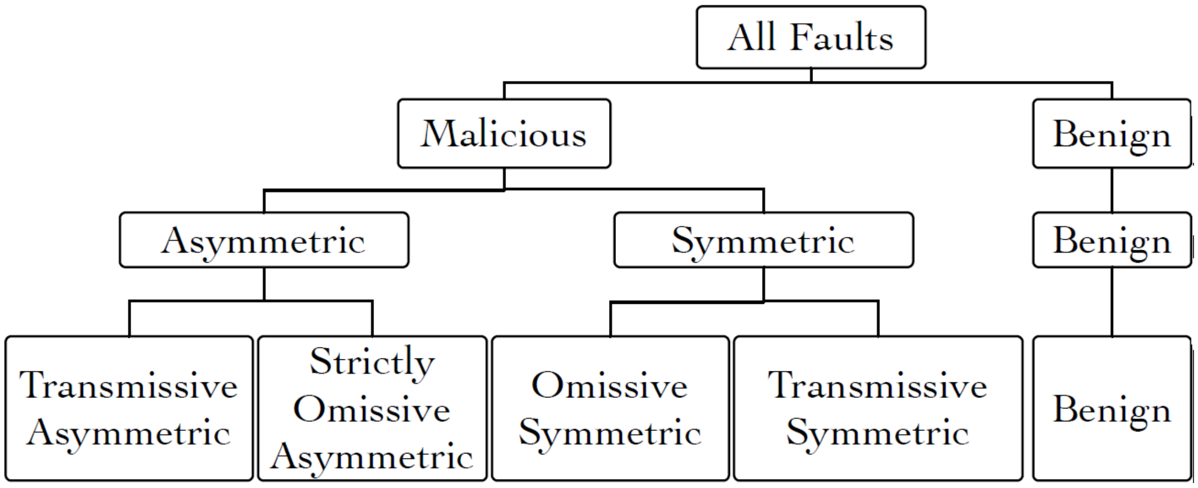


Figure 3.1: **Hybrid Fault Models**

Thambidurai and Park [43] consider the partitioning of faults into three fault types: benign, symmetric and asymmetric faults, where the latter two fault types are refined from the malicious faults in the model in [16]. This classification is represented in level three of Figure 3.1. Symmetric faults assume that faulty values are perceived identically by all non-faulty processes, i.e., the faulty process delivers the same erroneous value to all receiver or fails to deliver any value to any receiver. On the other hand, an asymmetric fault arises when a faulty process is able to send conflicting values to different non-faulty processes. Further refinement of faults was presented in the five-fault hybrid fault model by Azadmanesh et al. [15]. The model extended the three-fault model of Thambidurai, i.e., benign, symmetric, and asymmetric faults, by considering transmissive and omissive versions of symmetric and asymmetric faults. This level of refinement is shown in lowest level of Figure 3.1. This hybrid fault model of [15] is the basis of the classifications presented in this research.

The five-fault hybrid fault model is described as follows:

- **Benign:** A fault that is self-evident to all non-faulty nodes.
- **Transmissive symmetric:** This is a situation in which a single erroneous message is delivered to all receiving nodes. The faulty messages delivered are all identical. This type of fault is synonymous with the symmetric faults of Thambidurai and Park [43].

- **Omissive symmetric:** This is a fault scenario where no message is delivered to any of the receiving nodes. All receiving nodes are affected the same, as was the case for transmissive symmetric faults, however, the omission may result in the receiving nodes adopting a different action as if the message had been received.
- **Transmissive asymmetric:** This fault typifies any form of arbitrary asymmetric behaviour that is capable of delivering different erroneous messages to different receivers. This is synonymous with the asymmetric (Byzantine) fault in the three-fault model defined in [43].
- Strictly omissive asymmetric: In this fault scenario, a correct message is delivered to some nodes, while the other nodes do not receive any message. In this case, the omissive behavior is capable of affecting a system in an asymmetric way. This is because the nodes that did not receive the message might react differently from those that received the message.

3.2 Fault-model-based CRN View

We now classify jamming in CR networks based on hybrid fault models described in [15]. The motivation for this classification is the fact that in real-life systems fault scenarios are not likely to exhibit worst-case behavior. It therefore makes sense to use hybrid fault models that consider faults of different severities in their impacts. This will help us to avoid treating every fault as pathological, which would lead to over-estimating the level of redundancies and reliability requirements needed to handle the faults [15, 43]. The classification will allow faults due to jamming of different degrees to be easily represented in a CSS CRN with the intent of investigating the level of redundancies needed to deal with all the faults coexisting in the network. Using the hybrid fault model, we identify several scenarios for CSS in CRNs under this classification:

3.2.1 Total jamming of the Cognitive Control Channel (CCC)

This is a situation where all the channels used as CCC in an infrastructure-based (centralized) CR networks are jammed. The jammer is in a location in the network that allows it to transmit jamming signals that interfere with the channel(s) used by the CRN for sending control messages. The CRs sharing spectrum sensing information use the control channel to communicate with the fusion center their view of the network with respect to the presence of the primary users (PUs). If a jammer is able to jam or manipulate all the signals going to and coming from the fusion center of a centralized CSS CRN the same way, the type of faults that could occur as a result are:

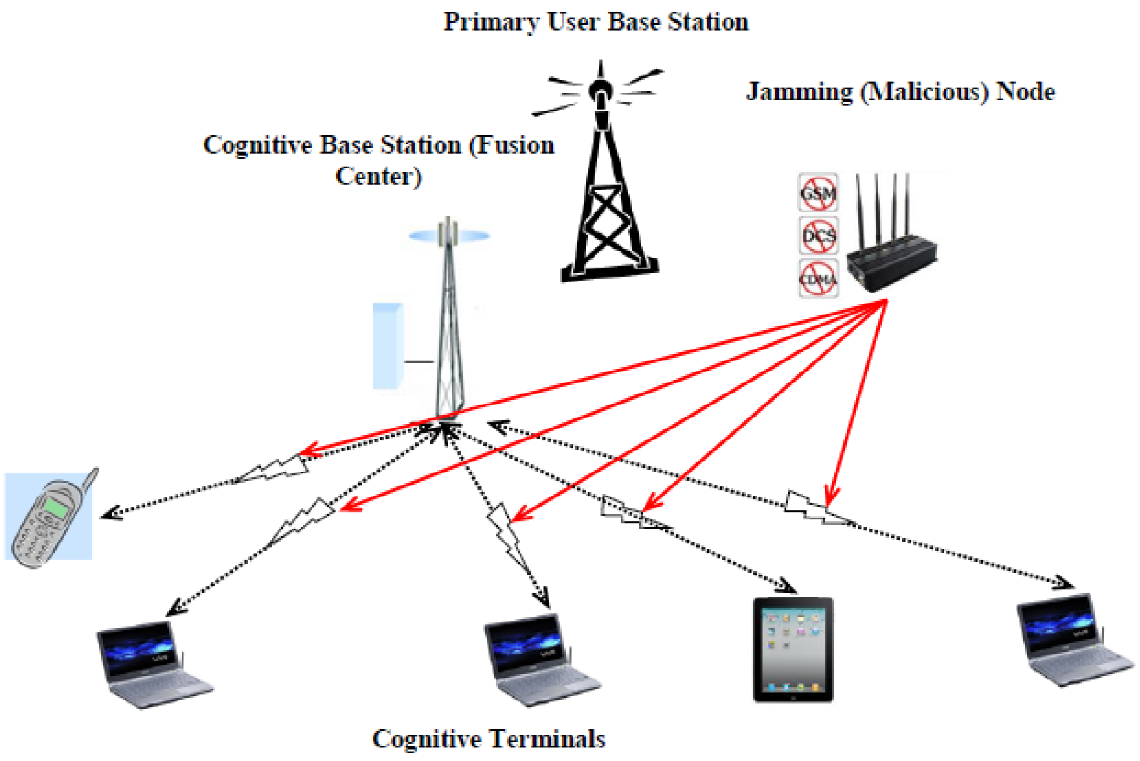

Primary User Base Station
Jamming (Malicious) Node
Cognitive Base Station (Fusion Center)
GSM
DCS
CDMA
Cognitive Terminals

Figure 3.2: **Total Jamming of CCC in Infrastructure-based (Centralized) CR Networks**

(a) **Benign**: This is a scenario where jamming is experienced by all the CR nodes as they can no longer receive any data relating to the presence of the PU the fusion center because the CCC is being jammed. The nodes can now communicate directly with themselves without the use of the CCC in an ad-hoc network manner to verify if other nodes are experiencing jamming too. This will eventually lead to all the nodes coming to an agreement that a jammer is operating in their vicinity.

(b) **Omissive Symmetric:** This scenario is similar to that of the benign case, but in this case it has not been globally diagnosed that a jammer is operating in the neighborhood. However, individual CR nodes are able to detect that they are not able to obtain sensing information from the CR fusion center via the CCC.

(c) **Transmissive Symmetric:** This is a case of data manipulation or value faults being sent to all the CR nodes by the jammer via the CCC. The jammer simply injects faults to the signals transmitted by the fusion center. Since all the CR nodes receives the same message from the fusion center, the manipulated data are received the same by all the CR nodes.

Figure 3.2 is a representation of all the fault types identified under this classification. The links represented by the dotted red arrows from the jammer depict the extent and degree of its jamming. This is total jamming as all the control channels used by the CR nodes to communicate with the fusion center are all jammed.

3.2.2 Partial Jamming of CCC

This is a scenario where only some of the CR nodes involved in spectrum sharing using CCC in infrastructure-based (Centralized) CR networks are jammed or have the data being sent to them manipulated. This could be if the location of the jammer in the network is not close enough to jam signals from some CR nodes or because of its signals strength not being strong enough to jam some nodes further away from the jammer. If the CRN makes use of more than one CCC, a jammer could also choose to jam only some of the channels to conserve power. A jammer will be able to jam for a longer period of time if it only jams a subset of the CR nodes or few of the CCC at a time. The faults identified in this scenario using the hybrid fault model classification are:

(a) **Strictly Omissive Asymmetric:** If the jammer jams sensing information to only some of the CR nodes but does not jam the rest of the CR nodes, i.e., some CR nodes receive the correct message from the fusion center while the rest does not receive any message from the fusion center. It could also be that the jammer jams some of the CCCs and does not jam other CCCs used by the network. This will make all the nodes using the CCCs that are not jammed receive correct information, while the nodes using the jammed CCCs will not receive any message. Figure 3.3 is a general representation of this classification. It can be seen that jamming signals only impact some of the nodes or CCCs represented by the red arrow emanating from the jamming node.

(b) **Transmissive Asymmetric:** This is the situation where the CR jammer is able to jam and inject different erroneous information to the signals being transmitted to the CR nodes from the fusion center. A CR node operating as a jammer has adaptable and reconfigurable radio parameters that enable it to be able to transmit different messages on different CCCs. Legitimate CR nodes communicating with the fusion center using these different CCCs will eventually receive different information that has been manipulated by the CR jammer. In

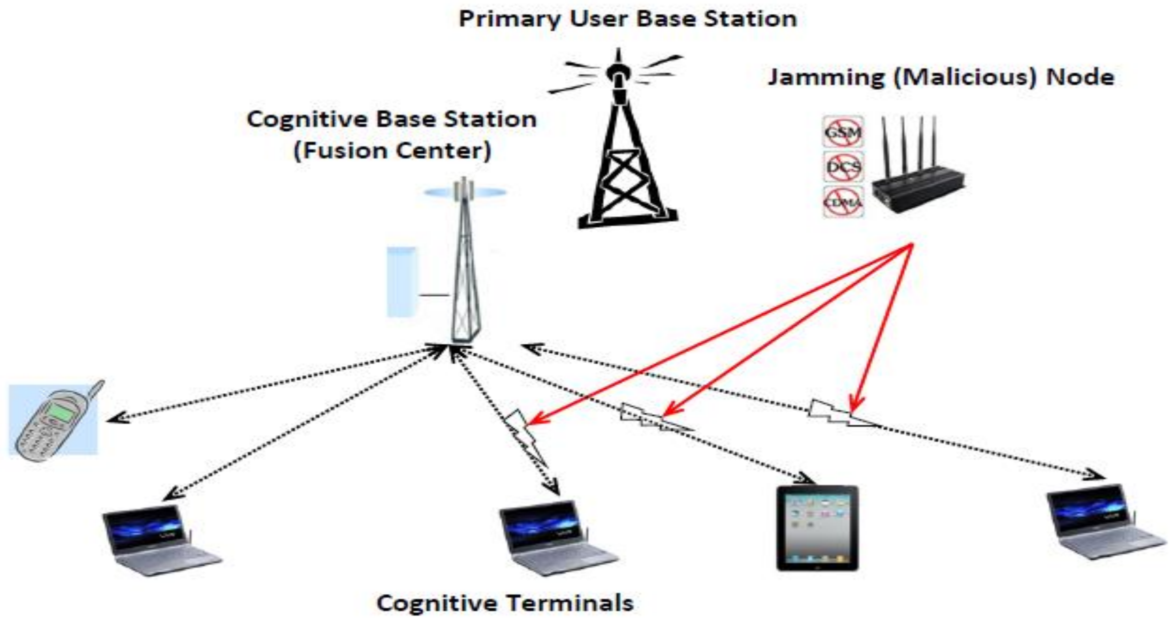

Primary User Base Station
Jamming (Malicious) Node
Cognitive Base Station (Fusion Center)
GSM
DCS
CDMA
Cognitive Terminals

Figure 3.3: **Partial Jamming of CCC in Infrastructure-based (Centralized) CRNs**

Figure 3.4, which is a representation of this scenario, the jamming node manipulates signals either to or from the fusion center selectively. The blue arrows from the jamming node represent the same data manipulation to two CR nodes in the network, while the red arrows represent different but identical manipulation to three nodes in the network. This scenario will lead to two nodes receiving data that are different from the data received by the remaining three nodes.

3.2.3 Total jamming of a Distributed CRN

This is when a jammer causes a DoS to all the nodes in an ad-hoc (distributed) CR cooperative spectrum sensing and sharing network. This classification also includes jammers injecting faults into or manipulating transmitted data in a distributed CSS CR network.

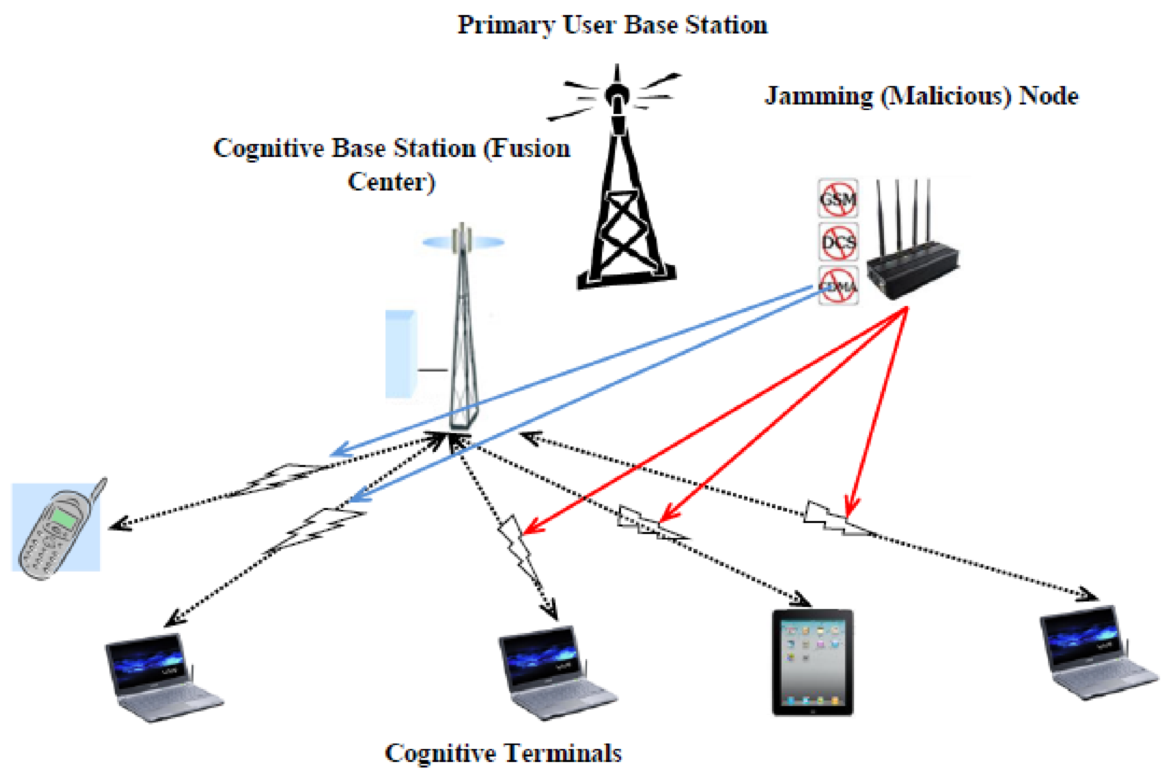

Primary User Base Station
Jamming (Malicious) Node
Cognitive Base Station (Fusion Center)
GSM
DCS
CDMA
Cognitive Terminals

Figure 3.4: **Partial Jamming of CCC in Infrastructure-based CR Networks: The Case of a Transmissive Asymmetric Fault**

The jamming, which causes complete absence of data being sent and received, affects all the CR nodes participating in the CSS. The faults identified here are:

(a) **Benign**: If all the nodes are able to come to an agreement that they are been jammed as a result of their inability to receive or send any data to any of the nodes. This is a scenario where the CR nodes switch to another channel that is not being jammed and communicate with one another that the initial channel being used is under jamming attack, making the fault to be globally diagnosed.

(b) **Omissive Symmetric:** Here it has not been globally diagnosed that a jammer is operating in the neighborhood, but individual CR nodes are able to notice the effect of jamming.

(c) **Transmissive Symmetric:** This is a case of fault injection or data manipulation perpetrated by a malicious CR node operating as a jammer. The jammer interferes with the sensing information shared by the CR nodes in a distributed way. Even if each CR node takes turn as a fusion center and an agreement algorithm is used to come to a decision on the spectrum sensing information, since the same erroneous values are delivered to all the CR nodes, the fault might still go undetected.

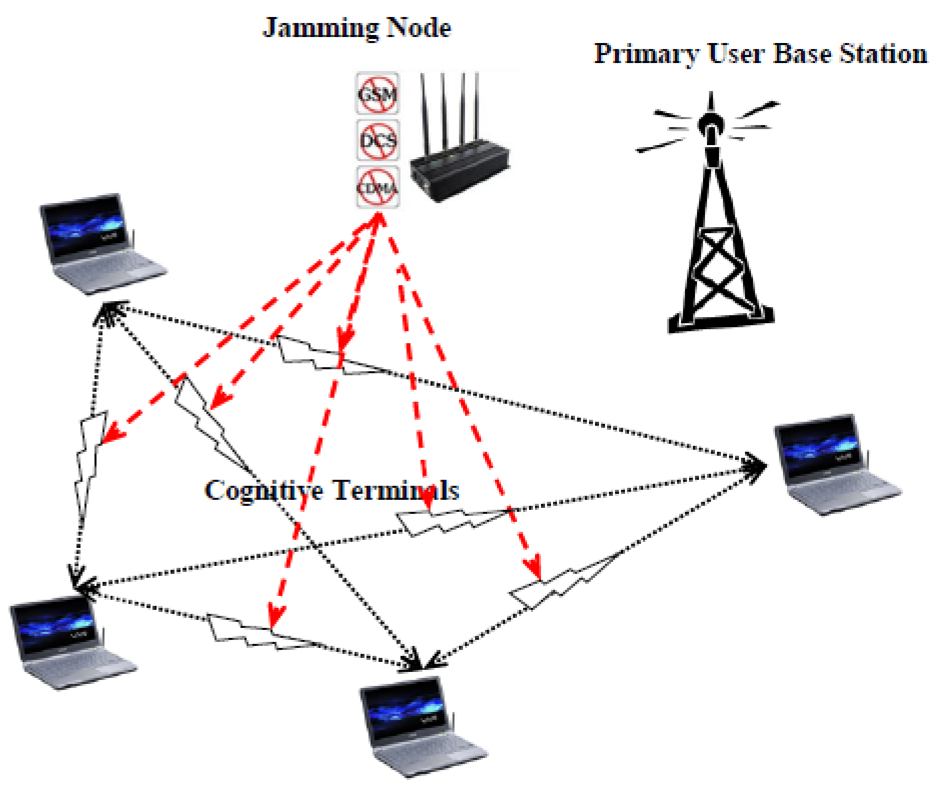


Figure 3.5: **Total Jamming of Distributed CSS CR Networks**

Figure 3.5 shows all the fault scenarios identified in this classification. The impact of jamming is illustrated with the red arrows emanating from the jammer. The figure shows that the jamming experienced at all CR nodes exhibits the same behavior, i.e., either loss of data or manipulation of data.

3.2.4 Partial Jamming of Distributed CRN

This is the type of jamming experienced by some (but not all) of the nodes in an ad-hoc (distributed) CR spectrum sensing and sharing network. A typical scenario will be the presence of a jammer in a CSS CR network environment that transmits jamming signals that affects some CR nodes but not all of them due to the location of the jamming node or its jamming signal strength. It could also be a scenario where the distributed CR nodes use more than one channel to communicate with one another. A jammer might jam some of these channels leading to partial jamming scenario. The faults identified under this classification are:

(a) **Strictly Omissive Asymmetric:** If jamming causes some of the CR nodes not to be able to receive data sent by a CR node, while the rest of the CR nodes are able to receive the correct data (because they are not been jammed). The CR node designated as the fusion center will therefore receive some correct data, while there will be no sensing data for the rest of the nodes that are being jammed. The fusion center can decide to use default values to replace the missing value.

(b) **Transmissive Asymmetric:** If the jammer manipulates sensing data shared by a distributed CSS CR network nodes in such a way that individual nodes receive different sensing data. The CR node designated as the fusion center will also receive conflicting data, as the data have been manipulated differently before getting to the fusion center. This scenario could also arise if the different channels used by the cooperating CR nodes are jammed differently, i.e., the jammer injects different faults into different channels.

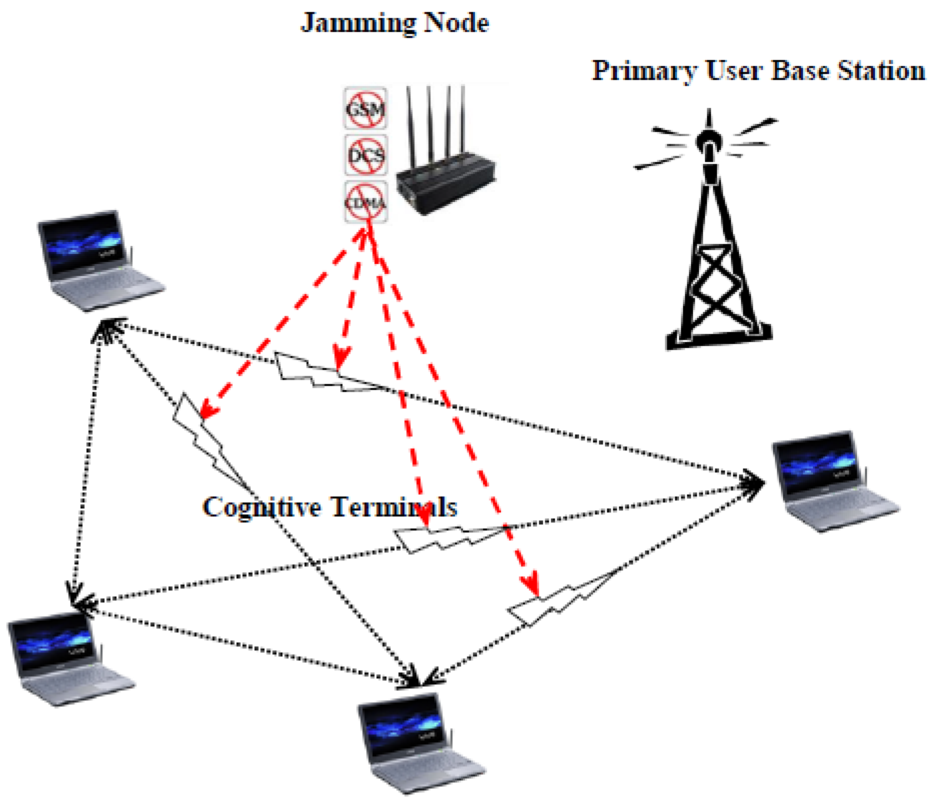


Figure 3.6: **Partial Jamming of Distributed CSS CR Networks**

The classification is shown in a general form in Figure 3.6. The dotted red arrows from the jamming node depict the communication jammed (loss of data) or that have their data manipulated (value fault).

3.2.5 Total Jamming of Relay-Assisted CSS CRN

This is the jamming of all the nodes involved in relaying and sharing sensed spectrum in a relay-assisted CSS cognitive radio network. The faults identified here are:

(a) **Benign**: If all the CR nodes are able to reach an agreement that the network is experiencing jamming as a result of not being able to receive any sent or relayed data.

(b) **Omissive Symmetric:** Unlike in the benign case there is not necessarily agreement that jamming is going on, however, none of the nodes receive the transmitted data.

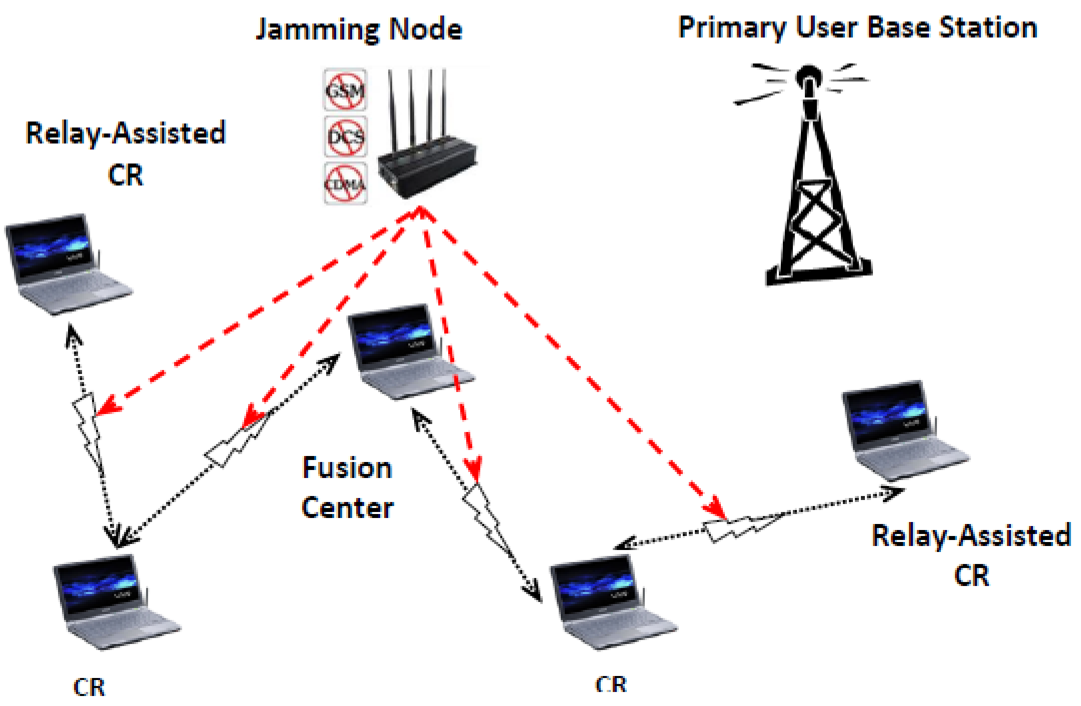


Figure 3.7: **Total Jamming of Relay-Assisted CR Networks**

(c) **Transmissive Symmetric:** If all the participating CR nodes receive the same erroneous values as a result of data manipulation by the jammer. The data is manipulated in the same manner and as such each CR node receives the same erroneous message. Figure

3.7 is a representation of this classification. It shows the impact of jamming with the dotted red lines as it affects all the CR nodes.

3.2.6 Partial Jamming of Relay-Assisted CRN

Here some (but not all) of the nodes involved in relaying and sharing sensed spectrum in a relay-assisted Cognitive Radio CSS architecture are jammed. This could be either as a result of the jamming signal not able to reach some of the CR nodes or a result of the jammer jamming only a segment of the channels available for relaying sensing information. The faults identified are:

(a) **Strictly Omissive Asymmetric:** If the jamming cause loss of data to some of the CR nodes while the rest of the CR nodes are able to receive correct data sent by a designated fusion center or a relay-assisted CR node.

(b) **Transmissive Asymmetric:** If the jammer is able to manipulate sensing data that are relayed by the relay-assisted CSS CR network nodes in such a way that individual nodes receive different sensing data. A representation of this classification is also illustrated in Figure 3.8. The figure shows the CR nodes whose data are manipulated differently at a point in time.

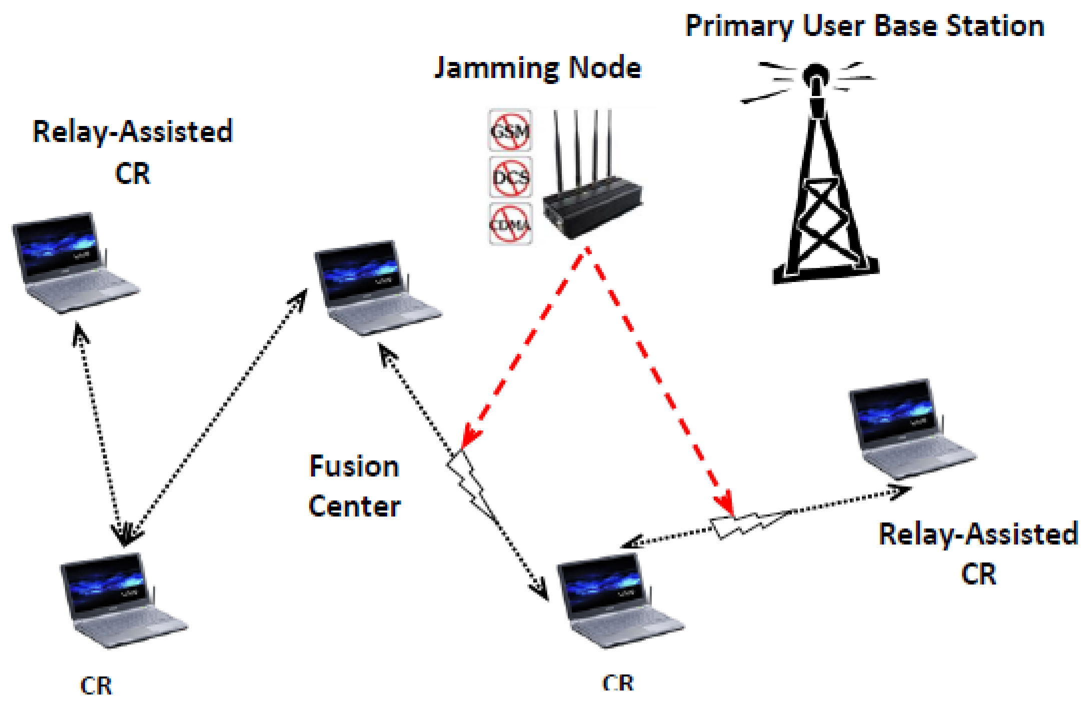


Figure 3.8: **Partial jamming of relay-assisted CSS CR networks**

3.3 Discussion

In this chapter, we identified fault scenarios and fault types with the expectation that this will enable us to accommodate jamming faults of different severities in a CSS CR network. The hybrid fault model approach was used and as a result, we are able to represent these faults in the context of hybrid fault models using a classification that considers transmissive and omissive value faults. This enables us to adequately specify what it will take to handle these faults.

Chapter 4

ON THE IMPACT OF JAMMING ATTACKS IN CSS COGNITIVE RADIO NETWORKS

4.1 Introduction

Since CR networks differ significantly from traditional wireless networks in their mode of operations and capabilities, potential attackers like jammers, which might be CRs themselves that have become malicious, are capable of preying on the adaptable features of CRN with the intent of causing serious denial of service to other legitimate CR users. In pathological cases, these jammers are even capable of introducing value faults. Jamming has not been studied in the context of fault model classifications that include value faults and their respective fault handling. In this chapter, we investigate the performance of CRs operating in the presence of this type of jammer. The purpose of this investigation is to know the extent of denial of service these jammers are able to cause, the factors that make these attacks very effective, and what can be done to mitigate this type of jamming attack.

4.2 Research Objective

In order to investigate the performance of CRs in a fault model classified CRNs, simulation was used to quantify the extent of jamming attacks. The jamming types simulated include random, constant, deceptive and reactive jamming in a CSS CRNs that consider hybrid fault models. We measured and analyzed the results of the simulations to quantify the impact of these jammers on CRNs with the intent of providing a suitable solution for mitigating these attacks.

4.3 NS-2 Simulation Setup for Cognitive Radio Networks Support

The Cognitive Radio Cognitive Network (CRCN) [47] simulator based on open-source NS-2 (network simulator 2) [46] was used for the simulations conducted in this research. This section describes the different configurations that we made in the CRCN simulator in order to successfully set up a simulation environment that possesses all the functionalities of a CRN.

We made use of additional functionalities provided by the simulator which include support for dynamic spectrum resource allocation, configurable CR MAC, and adaptive CR routing protocols in order to setup a functional CRN. Each CR node is configured to either use a single radio or multiple radios with the added options of either transmitting with a single channel or multiple channels, depending on the simulation scenario. A summary of these functionalities and how we implemented them in our simulation is presented in this section.

4.3.1 Simulation Setup for CR Routing Support

In order to configure our simulation for CR routing and channel assignment supports we made use of the network layer routing components, which include support for Multiple-Radio-Multiple-Channel configuration. This involves setting up nodes with more than one interface, with each interface being able to transmit on more than one channel at a time. This allows us to either setup CR nodes of equal number of radios and channels (homogeneous configuration), or nodes with unequal number of radios and channels (heterogeneous configuration). We added the following four components to our simulations to create the required Multiple-Radio-Multiple-Channel scenario:

Component 1: Setting up number of Radios

```
set val(ni) 3
```

Component 2: Creating new channel objects

```
for {set i 0 } {$i < $val(ni)} {incr i} {
        set chan_($i) [new $val(chan)];        #Creating new channel objects
}
```

Component 3: setting up the nodes to use the interface

```
$ns_node-config -adhocRouting $val(rp)\
-ifNum $val(ni) \ ;                # configuring the interface number for node
-channel $chan_(0) ; # configuring the first channel object
```

Component 4: Assigning different channel objects with different spectrum parameters

```
# configuring the ns channel...
```

```
$ns_add-channel-new-phy 0 $chan_(1) 0.6 Propagation/FreeSpace
# updating this information for node 2
set node_(2) [$ns_node] ; # applying the change on node
$ns_add-channel-new-phy 0 $chan_(0) 0.6 default
```

4.3.2 Simulation Setup for CR MAC Support

We set up our simulation to support CR MAC components made available by CRCN. These functionalities include: (i) Support for Multiple-Radio-Multiple-Channel configuration, (ii) Support for Single-Radio-Multiple-Channel configuration, (iii) Interface for Channel Decision, (iv) Interface for Transmission Power Decision, (v) Interface for setting and obtaining both Interference and Traffic information. In order to create Multiple-Radio-Multiple-Channel environment for MAC in our simulation, we added the following components to the script:

Component 5: Defining the number of Radios and Channels in the TCL (Tool Command Language) script

```
set val(ni) 3 ;                # This set number of radios/interface to 3
set val(channum) 2 ;           # This set number of channels to 2
```

Component 6: Creating new channel objects according to the number of radios and channels

```
for {set i 0 } {$i < [expr $val(ni)*$val(channum)]} {incr i} {
        set chan_($i) [new $val(chan)] ; # new channel objects
}
```

Component 7: Configuring the multiple radio and channel options provided

```
$ns_node-config -ifNum $val(ni) -channel $chan_(0)
$ns_node-config -ChannelNum $val(channum)
```

Component 8: Assigning the channel objects to the channel array of the simulator

```
for {set i 0 } {$i < [expr $val(ni)*$val(channum)]} {incr i} {
        $ns_add-channel $i $chan_($i)
}
```

We experimented with the two new categories of MAC protocols that were integrated into CRCN for Cognitive Radio support. The first category is the "Contention-Based" CR MAC. The new MAC in this category is the Maccon [47], in which each node randomly selects a channel from the available channels as specified in the simulation script and contends for the usage of that channel. The second category is the “Collision-Free” CR MAC. This category of CR MAC was used to implement instances of intelligent and schedule-based MAC, in which CR nodes negotiate channel decision and transmission parameters over a common control channel. The new MAC protocols in this category are Macng and Macenhanced [47]. Macenhanced uses several rounds of negotiation to implement Dynamic Spectrum Access (DSA) based MAC. It also implements the detection of the existence of a primary user using the packet-based detection approach. In this approach, the primary user periodically sends out a primary user packet to signal its presence. On the other hand, a legitimate CR node is able to distinguish this packet from the regular packets send by a regular CR node.

4.4 Simulation Setup to Implement Jammers in CRCN

The purpose of carrying out the simulation described in this section is to investigate the performance of CRNs operating in the presence of CRs operating as jammers. These jammers take advantage of all the capabilities they possess as full functional CR nodes in terms of adaptable and reconfigurable radio parameters to cause jamming to legitimate users of the network. The jammers are able to cause serious DoS to other user by either jamming the signals or manipulating the spectrum sensing data of legitimate CR nodes to their advantage. The effect of this jamming could be very damaging to the normal operation of the CRN. Hence, there is a need to investigation the extent of DoS these jammers are able to cause, to determine the factors that make the jamming attack very effective and what solution space should be considered in order to mitigate this type of jamming attack.
This section discusses the different modifications and parameters used in order to set up the jamming types in the NS2 Extension for Cognitive Radio. Since this research requires jamming scenarios that allow the user to capture and measure specific network metrics like the rate of jamming for each jamming type, the NS2 Error model approach for simulating communication error was adopted. This will not only make it possible to specify a

transmission model for the jammers describing their behaviors, but it will also make it possible to specify the different rates of jamming or the probability of data corruption experienced as a result of a typical error rate specified.

4.4.1 NS-2 Error models

Error models in NS-2 are used to simulate packet errors during transmission. Packets in error are either dropped or just marked as being in error depending on the user's requirement and specification. A packet error could be specified according to the underlying stochastic or empirical error model. An error model is stochastic if packets are corrupted according to underlying random variable distributions. The error rate could be specified as any point within the range allowed by its distribution. The unit of error (e.g. bit, byte or packet) along with the error rate are set by the user in the simulation script. For example, setting the error rate to 0.2 and the error unit to packet will in the long run have about 20% of the transmitted packet corrupted and eventually dropped during a simulation. The error models used for implementing jammers in this research are stochastic in nature. This makes it possible for different jammers to be specified by different distribution depending on the behavior of the jammers and also the effect of the jammers can be easily measured based on the error rate.

Communication errors due to jamming can be modeled in a network by inserting a given statistical error model over outgoing or incoming wireless channels. The outgoing error module causes all the receivers to receive the packet suffering the same degree of errors. In this configuration, the error is determined before the wireless channel module at each node copies the packet. The incoming error module lets each receiver get the packets corrupted with different degree of error, since the error is independently computed at each node [46]. The node-config command is inserted into a simulation script with option "IncomingErrrProc" or "OutgoingErrProc", or both. For our implementation of jammers we configure the jammers to use the incoming error module alone, while other CRs in the network use both the incoming and the outgoing error modules. This configuration will ensure that outgoing packets from the jammers do not suffer loss or corruption as they are intended to cause denial of service to packets transmitted by legitimate CR nodes. On the other hand, all outgoing packets from and incoming packets to the legitimate CR nodes suffer from the same degree of data loss/corruption specified by the error rate. An instance of a global procedure that we use to create error model object for the example of constant jamming is presented below:

```
proc UniformErr
set err [new ErrorModel]
$err unit packet
$err rate_ 0.1
return $err
```

We then use the name of this global procedure as argument to the two options to add the error module into the wireless protocol stack by configuring the node-config as follows:

```
$ns node-config -IncomingErrProc UniformErr\
-OutgoingErrProc UniformErr
```

ErrorModel classes with more functionalities such as multi-state (e.g. two-state) error model, which implements time-based error state transitions, was used to implement scenarios like random jamming. In this configuration, transitions to the next error state occur at the end of the duration of the current state. The next error state is then selected using the transition state matrix. An example of how we use a two-state error model to implement random jamming in our simulation scripts is presented below:

```
set tmp [new ErrorModel/Uniform 0 pkt] ; #sleeping state
set tmp1 [new ErrorModel/Uniform 0.1 pkt] ; #wake state
# Array of states (error models)
set m_states [list $tmp1 $tmp2]
# Durations for each of the states, e.g. sleep & wake durations
set m_periods [list 0.05 0.25]
# Transition state model matrix
set m_transmx { {0.95 0.05 }
{0 1} }
#set the unit of error: one of pkt, byte, or bit.
```

```
set m_trunit bit
# Use time-based transition
set m_sttype time
#set number of states
set m_nstates 2
#set the beginning of state to 0
set m_nstart [lindex $m_states 0]
set em [new ErrorModel/MultiState $m_states $m_periods $m_transmx
$m_trunit $m_sttype $m_nstates $m_nstart]
```

The Uniform Error model, which is one of the error models we use to model packet errors due to jamming, is a Bernoulli distribution that generates a random variable *u*, uniformly distributed in the range [0, 1]. When the error rate, rate_ is specified in the simulation, the u will serve as a threshold in the range [0, *rate_*] with probability *rate_*. That means that for us to have the probability of data corruption (due to e.g. jamming) of rate, *rate_*, we need to generate a uniformly distributed random number, u, such that the data corruption occurs if and only if $u < rate\_$. We also specified the unit of error in our simulations as bit so that we can quantify the probability of having a certain amount of bits of a packet getting corrupted. For example, given packet size, size, the probability of having exactly *i* bits of the packet corrupted is indicated as:

$$\binom{size}{i}(rate\_)^{i}(1 - rate\_)^{size-i}$$

where *rate_* is the bit error probability. For a comprehensive description of different error models and how they can be implemented in NS2 simulation, the reader is referred to [46].

4.4.2 Simulation Parameters

Based on the general behavior of each jamming type discussed in Chapter 2, Tables 4.1 to 4.4 present the simulation parameters specified to meet the requirements of each jamming type in our simulations. Here is a brief description of these parameters:

- **Error Model:** The pattern of communication error generated by the jammers, e.g., uniform, selective, multi-state, etc.

- **Unit of Error:** This is used to designate the unit of measurement of the error rate as specified in the error model, i.e. bit, byte or packet.

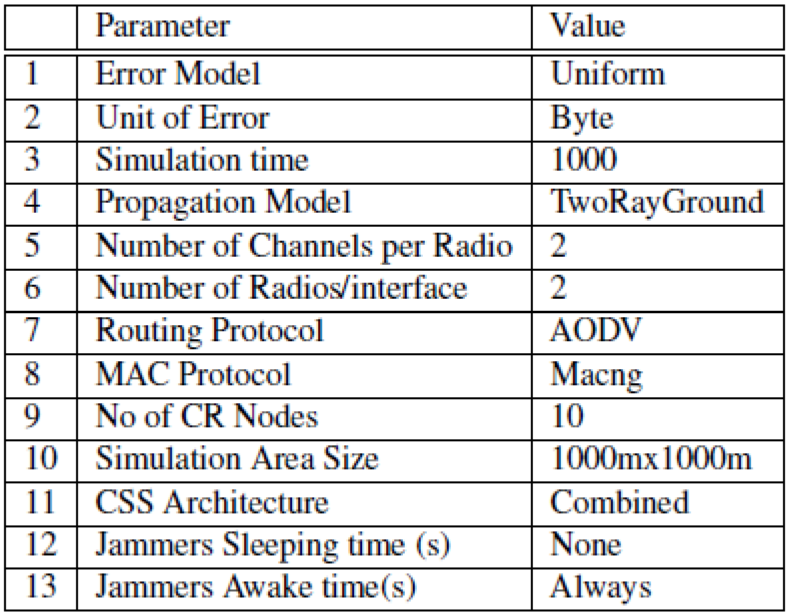
Table 4.1: **Simulation Parameters for Constant Jamming**

	Parameter	Value
1	Error Model	Uniform
2	Unit of Error	Byte
3	Simulation time	1000
4	Propagation Model	TwoRayGround
5	Number of Channels per Radio	2
6	Number of Radios/interface	2
7	Routing Protocol	AODV
8	MAC Protocol	Macng
9	No of CR Nodes	10
10	Simulation Area Size	1000mx1000m
11	CSS Architecture	Combined
12	Sleeping time (s)	None
13	Awake time(s)	Always

Table 4.2: **Simulation Parameters for Deceptive Jamming**

	Parameter	Value
1	Error Model	Uniform
2	Unit of Error	Byte
3	Simulation time	1000
4	Propagation Model	TwoRayGround
5	Number of Channels per Radio	2
6	Number of Radios/interface	2
7	Routing Protocol	AODV
8	MAC Protocol	Macng
9	No of CR Nodes	10
10	Simulation Area Size	1000mx1000m
11	CSS Architecture	Combined
12	Jammers Sleeping time (s)	None
13	Jammers Awake time(s)	Always

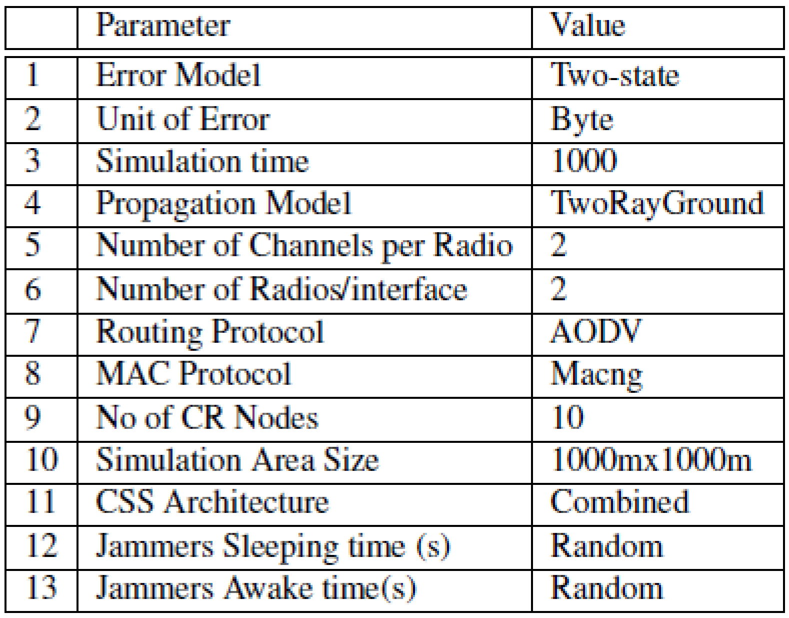
Table 4.3: **Simulation Parameters for Reactive Jamming**



	Parameter	Value
1	Error Model	Uniform
2	Unit of Error	Byte
3	Simulation time	1000
4	Propagation Model	TwoRayGround
5	Number of Channels per Radio	2
6	Number of Radios/interface	2
7	Routing Protocol	AODV
8	MAC Protocol	Macng
9	No of CR Nodes	10
10	Simulation Area Size	1000mx1000m
11	CSS Architecture	Combined
12	Jammers Sleeping time (s)	None
13	Jammers Awake time(s)	Always

- **Simulation Time:** The duration of the simulation in seconds
- **Propagation Model:** Used to predict the received signal strength of each packet. It could be specified as free space, two ray ground reflection or shadowing. A commonly used propagation model for simulation in NS-2 is the two ray propagation model. This is because the model considers both the direct path and ground reflection path propagation in the way that data are sent from source to destination in wireless networks. The advantage is that it gives a more accurate prediction when the distance is significantly long, than the free space model, which propagates signals with the assumption that there is only a single line-of-sight path between the sender and the receiver. The shadowing model is a more realistic representation of real-life signal propagation. In this model, the received power at a certain distance is modeled as a random variable due to multipath propagation or fading effects [24].
- **Number of Channels per Radio:** A radio/interface can be configured to use more than one channel for transmission.
- **Number of Radios:** A CR node can have more than one interface or radio for transmission.
- **Routing Protocol:** The protocol used for routing packets, e.g., DSDV, WCETT, AODV, etc.

Table 4.4: **Simulation Parameters for Random Jamming**



	Parameter	Value
1	Error Model	Two-state
2	Unit of Error	Byte
3	Simulation time	1000
4	Propagation Model	TwoRayGround
5	Number of Channels per Radio	2
6	Number of Radios/interface	2
7	Routing Protocol	AODV
8	MAC Protocol	Macng
9	No of CR Nodes	10
10	Simulation Area Size	1000mx1000m
11	CSS Architecture	Combined
12	Jammers Sleeping time (s)	Random
13	Jammers Awake time(s)	Random

- **MAC Protocol:** Protocol used for medium access control to gain access to channels e.g. Macng, Maccon, etc.
- **Number of CR nodes:** This is the number of Cognitive Radio nodes participating in spectrum sensing and sharing.
- **CSS Architecture:** The cooperative spectrum sensing architectural model (topology) used.
- **Sleeping Time and Awake Time:** Some of the jammers operate by sleeping and waking in order to conserve power. The sleeping time is the time in seconds when the jammer is not jamming, while the awake time is the time in seconds when the jammer is jamming.

4.4.3 Simulation Assumptions

The following two assumptions were made:

1. The jammers do not interfere with the primary user signals
2. Jamming rates are a measurement of increasing or decreasing degree of data corruption

4.4.4 Results of Simulation

The result of the simulations are shown in Figures 4.1 to 4.4, which depict the impact of constant, random, reactive and deceptive jammers respectively on the underlying MAC protocol when simulated in the NS2 extension for Cognitive Radio. It should be noted that the figures do not simply plot the impact on package delivery from a statistical packet related point of view, but the real impact of the jamming strategy on the underlying MAC. This is seen in the non-uniform distribution of the spurts of packet transmissions once jamming starts. In the absence of jamming, depicted by the "no jamming" plot in Figures 4.1 to 4.4, it can be seen that a throughput of over 100,000 bps is achieved. But as jamming starts, the impact is immediately observable.

The throughput of a network is defined as the average rate of data successfully delivered by the network. In this research, we are specifically interested in the total number of useful data delivered by the network. Therefore we measured throughput in bits per second (bps), which was derived from converting the total number of data packets received per second to bits per second. Starting with constant jamming in Figure 4.1, when the rate of jamming was set to 10%, the number of packets delivered reduced considerably. At a simulation time beyond 70 seconds, constant jamming caused close to zero bits of data to be delivered as the underlining TCP protocol tried to cope with what it perceived as congestion. In order to distinguish between congestion and jamming, we only simulated 10 CR nodes, as can be seen in Table 4.1. At a simulation time beyond 370 seconds the TCP gave in to the jamming, leading to a throughput close to zero up to about 750 seconds. Beyond this point, the MAC protocol was able to cope with the packet loss by assuming that the packet loss was due to congestion. Some bits of data were therefore delivered at the receivers for the rest of the simulation period. Considering 20% constant jamming, shown in Figure 4.1, there was further reduction in the total number of bits of data delivered during the entire simulation period. The behaviour of the underlying MAC is similar to that of 10% jamming. When the jamming rate of the constant jammer was set to 30%, the CR MAC protocol could not handle this level of jamming and as a result there was close to zero bit of data delivery by the network. The same zero bit of data delivery was observed for jamming rates beyond 30%.

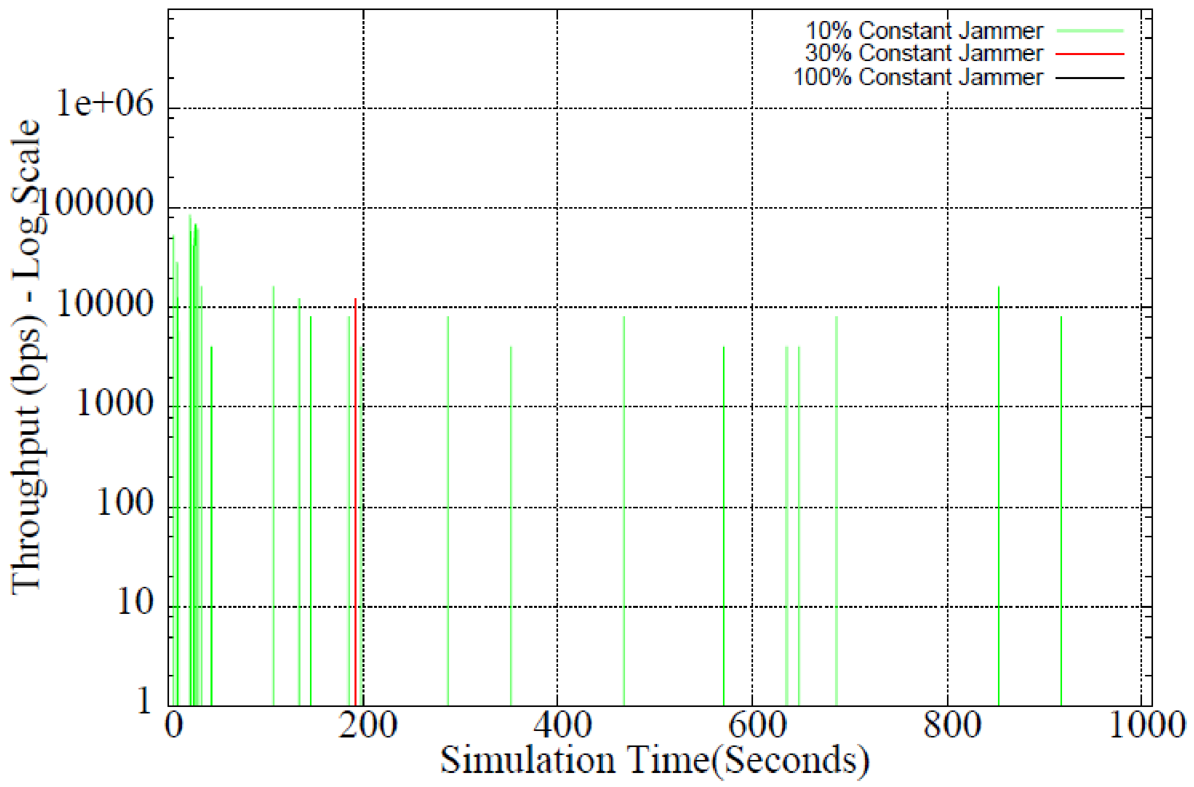


Figure 4.1: Impact of Constant Jammers on Throughput

Since random jammers jam and sleep for some time in an alternating fashion, the two-state error model was used. This is indicated as "Error Model" in Table 4.4. It can also be seen from the table that the jammer's sleeping and wake times are specified to be random. The jamming effects of random jammers when the jamming rate is between 10% - 20%, shown in Figure 4.2, are not as severe as in the case of constant jammers. The underlining protocol tries to handle this jamming by treating it as congestion. But as the jamming rate increases to and beyond 40%, the jamming effect could be clearly seen. Just tiny spurts of data delivery were observed up to a rate of 70%. Beyond 70% jamming rate, the protocol could no longer handle the effect of the random jammers and zero bit of data was delivered at the receiver.

Reactive jammers, that jam only when they perceive that there is an on-going transmission, were able to cause serious DoS almost as high as that of the constant jammers. In order to distinguish the jamming effect from regular packet drop caused by a MAC protocol, we used Macng as the MAC protocol for our simulation as depicted in Table 4.3.

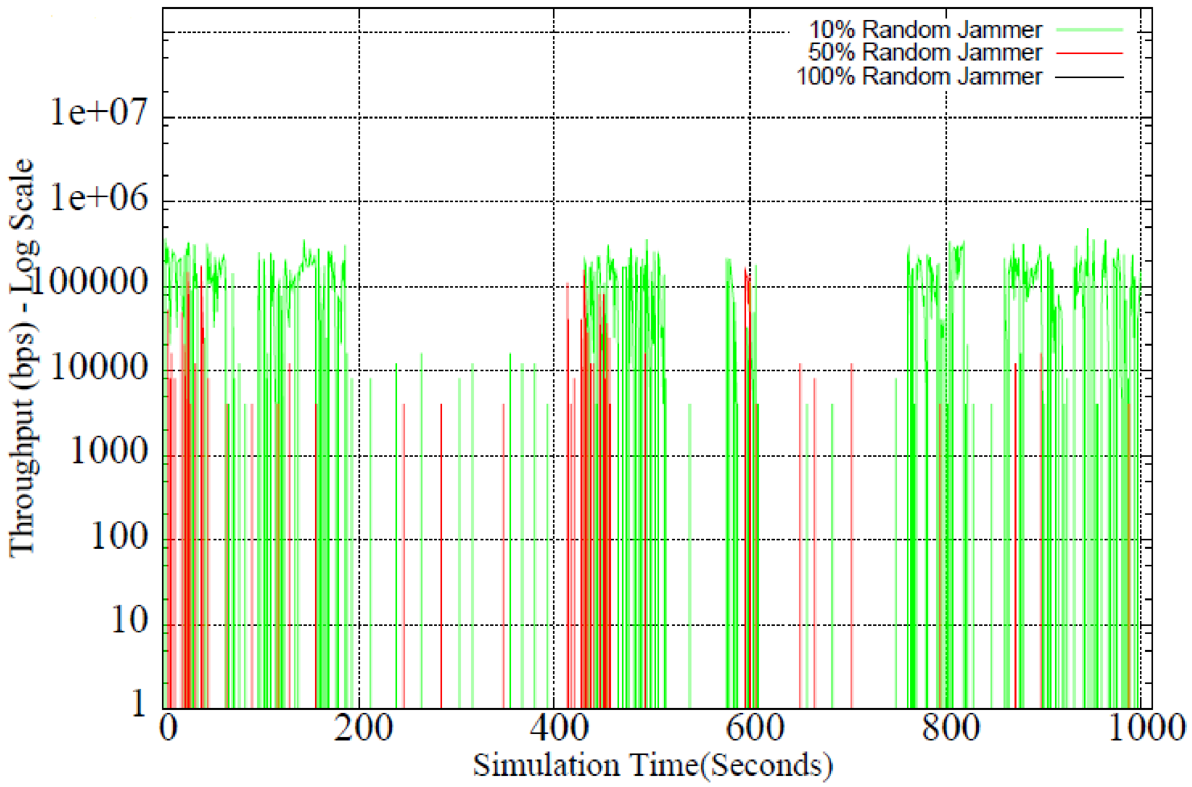


Figure 4.2: Impact of Random Jammers on Throughput

Macng is believed to reduce considerably rate of collision that could be experienced by a CRN due to its adaptable capability [47]. It can be observed from Figure 4.3 that when the jamming rate is 10%, as the jammers react to on-going transmission, the throughput was reduced to almost zero. The 20% reactive jammers behaved the same way as the 10% jamming rate. As the jamming rate increases to 30% and 40%, the CR MAC protocol fell apart considerably and the network could only deliver few bits of data. Jamming rates beyond 40% lead to zero throughput, resulting in complete DoS to the network.

For deceptive jammers, which jam the CRN with regular packets, the DoS they cause is more pronounced than that of the constant jammers. This is because the CR nodes are deceived that a legitimate transmission is always going on. This effect can be seen immediately when the jamming rate is 10% in Figure 4.4. Beyond a simulation time of about 300 seconds, there were intermittent spurts of bits of data delivery. This pattern continues even with a jamming rate of 20%. There was a close to complete breakdown of the network as the jamming rate was set to 30% and 40% respectively, as close to zero through was observed. Beyond the 40% jamming rate, the CRN broke down completely as zero bit of data was delivered by the network.

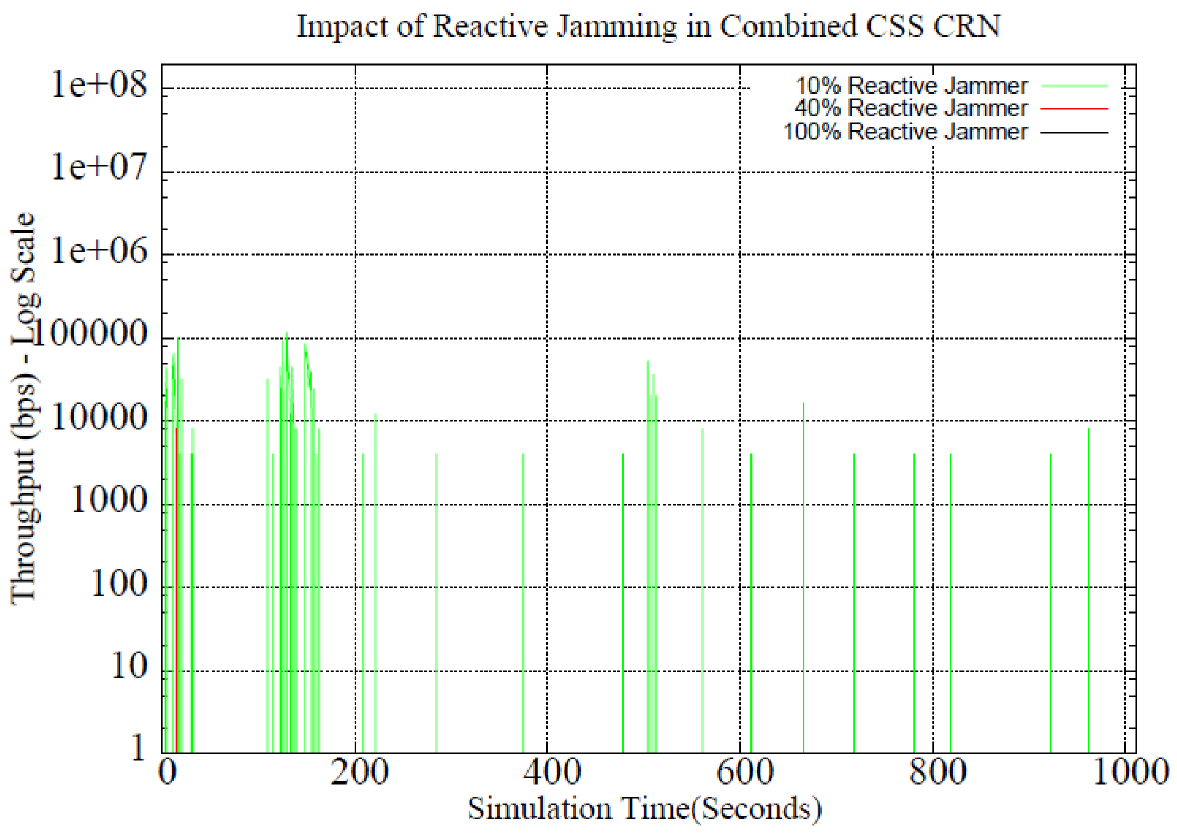


Figure 4.3: Impact of Reactive Jammers on Throughput [in log scale]

The real impact of jamming is more visible if one looks at the average throughput as shown in Figure 4.5. The average throughput is defined as the number of packets received at a layer (e.g. application layer) divided by the total simulation time. The decreasing average throughput can be observed on the log scale in Figure 4.5. At 30% the average throughput with constant jamming is reduced by three orders of magnitude and beyond this point there was zero throughput observed. Similarly, at 40% the average throughput with reactive and deceptive jamming is reduced by three orders of magnitude and beyond 40% the underlining MAC protocol falls apart. The case of random jamming was different from the other three types in the sense that, the CRN continued to deliver bits of data, though at reduced delivery rates. This is because the random jammer alternates between sleeping and waking, making the jamming effects not immediately noticeable at lower jamming rates. But beyond jamming a rate of 70% the network collapsed.

Figure 4.6 compares the impact of these jammers in terms of packet delivery ratio (PDR), which is defined as the ratio of the number of packets received at a layer (e.g., application

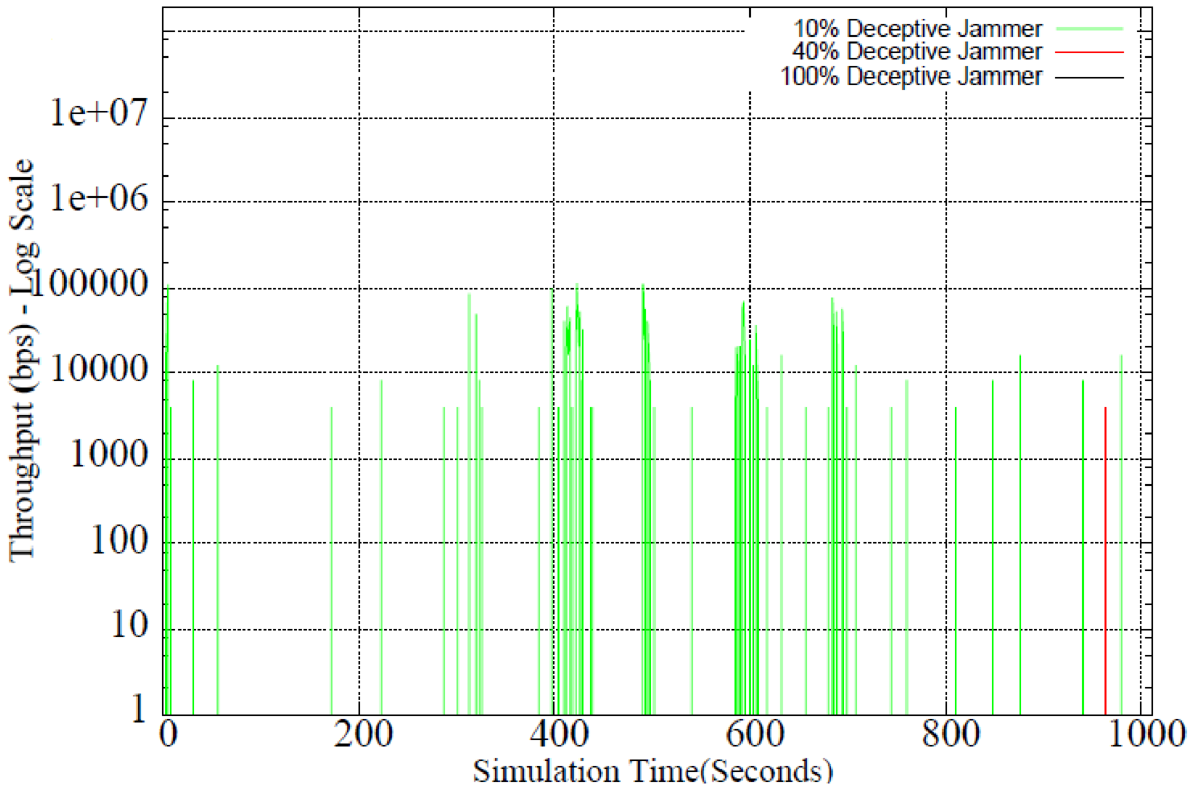


Figure 4.4: Impact of Deceptive Jammers on Throughput [in log scale]

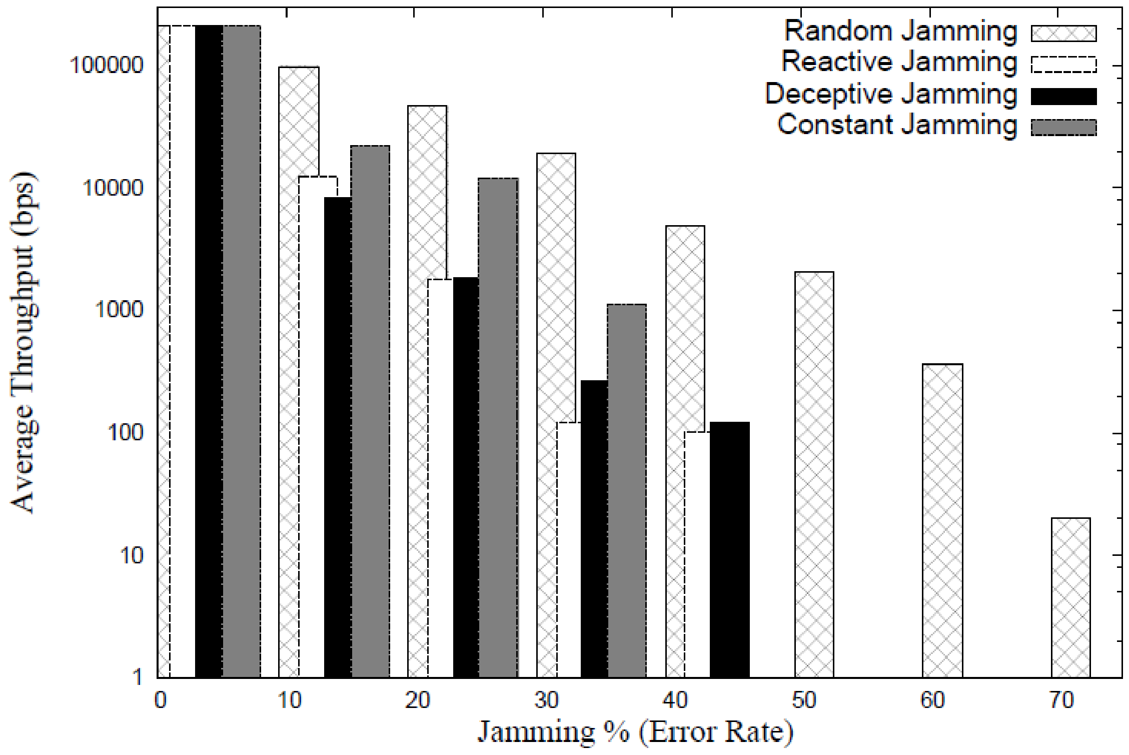


Figure 4.5: Comparing Impact of Jammers on Throughput [in log scale]

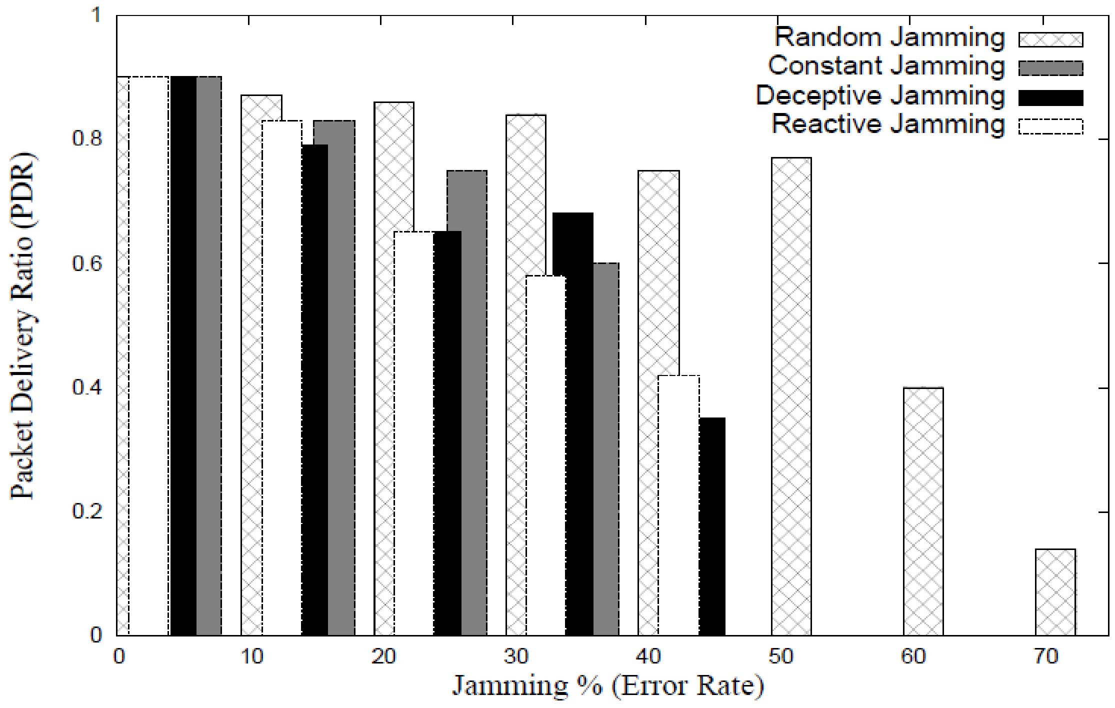


Figure 4.6: Comparing Impact of Jammers on Packet Delivery Ratio (PDR)

layer) to the number of packets sent in the same layer. The authors in [5, 49] demonstrated that the PDR is an effective measurement for distinguishing between a congested network and a network under jamming attack. The authors show that a highly congested network is still able to measure a PDR that is as high as 78%, while the PDR of a network under jamming drops considerably. It can be seen from the figure that the PDR of each of the jamming types using different jamming rates is consistent with our simulation result when we considered the average throughput of the CRN operating in the presence of those jammers. The PDR of each jamming scenario diminishes as the jamming rates increase resulting in the relationship:

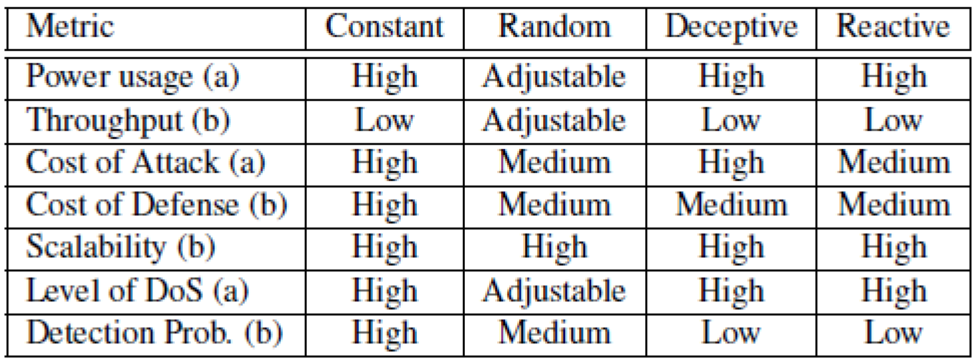
$$\text{PDR} \propto 1/r,$$

where *r* is the jamming rate of jammers. Our simulation results presented with Figures 4.5 and 4.6 support the arguments made in Table 4.5 which is discussed below.

4.4.5 Summary of Jamming Impacts in CRNs

We here identify the impact of the 4 different classes of jammers operating in a CSS CRN environment based on [20]. Table 4.5 presents a summary of these comparisons with a rating of their impact. The table is intended to give the reader a feeling for the impact and what to expect when trying to mitigate specific jamming attacks as found in cited literature and the results of the simulation we described in this chapter.

Table 4.5: **Impact of Jamming Attacks**



Metric	Constant	Random	Deceptive	Reactive
Power usage (a)	High	Adjustable	High	High
Throughput (b)	Low	Adjustable	Low	Low
Cost of Attack (a)	High	Medium	High	Medium
Cost of Defense (b)	High	Medium	Medium	Medium
Scalability (b)	High	High	High	High
Level of DoS (a)	High	Adjustable	High	High
Detection Prob. (b)	High	Medium	Low	Low

The simple values low, medium, high or adjustable are used in the network metrics. A similar approach based on five different values was used in [35]. The impact is considered from two perspectives, i.e., (a) the jammers point of view and (b) the defenders point of view.

The metrics considered here are:

- The power usage of the jammer during the entire jamming duration.
- The throughput, which is the total number of packets delivered to the receiver during the simulation period by the CRN experiencing jamming.
- The cost of attack, which is the total cost of perpetrating the jamming attack. This involves generally cost/complexity of designing the jammer.
- The cost of defense, which is the cost/complexity of designing anti-jamming techniques to handle the jamming.
- Scalability, which refers to the impact of increasing number of the jammers.
- Level of DoS, which is the level of disruption caused by the jammers in the CRN.
- Detection probability, which is the probability of the jammers being detected by the CRN as they carry out jamming.

4.5 Discussion

In this chapter, we investigated through simulation the performance of CRs operating in the presence of four classes of CR jammers. Using performance evaluation parameters such as PDR and average throughput we quantified the impact of these jammers as they increased their jamming rates. We discovered that these jammers are very effective in their operations as they are able to bring down the entire CRNs even when their rates of jamming are just about 30%. This implies that at about an average of 30% jamming rate for each jamming type, zero packet were delivered by the networks to the destinations. In conclusion, we presented a table that describes the impact of these jammers with respect to power usage, throughput, cost of attack, cost of defense, scalability, level of DoS and the probability of detecting the jammers.

Chapter 5

A HYBRID FEC CODE FOR MITIGATING JAMMING ATTACKS IN CSS COGNITIVE RADIO NETWORKS

5.1 Introduction

In this chapter, based on the hybrid fault model classification presented in Chapter 3 and the perceived impacts of jamming on this model as quantified in Chapter 4, we propose a new hybrid Forward Error Correction (FEC) Code that is capable of mitigating jamming attacks under this classification. In order to determine the suitability of the proposed solution, the new hybrid code is simulated in NS2 and its performance is investigated.

5.2 Research Goals

The goals of the research conducted in this chapter is to specify and evaluate the performance of a hybrid forward error correction code that will serve as solution to jamming in fault-model-classified CRNs, even considering value faults. Based on the outcome of our analysis of the proposed algorithm, we specify factors that are pertinent to its successful deployment.

5.3 Related Work

In this section, we present a brief summary of related work. The selected research categorizes the different approaches that have been proposed. The different anti-jamming mechanisms proposed were identified and classified based on the techniques used. Their effectiveness in handling jamming, especially those that cause value faults were appraised. We also identify their shortcomings and propose a solution that overcomes the shortcomings identified.

5.3.1 Anti-jamming Strategies for Traditional Wireless Networks

Jamming with respect to traditional wireless networks has been addressed by different authors including [28, 8, 66] and different anti-jamming techniques have been proposed. Dong and Liu [28] proposed a jamming-resistant broadcast system that arranges receivers into multiple channel-sharing broadcast groups with the intent of isolating malicious receivers using adaptive re-grouping. A suspicious group with one compromised receiver is re-grouped until the compromised receiver is isolated as a group by itself. The compromised receiver is therefore assumed to stop jamming to prevent it from being caught. Xu et al. [49] used spectral discrimination techniques to enhance the detection of jammers. They introduced what they called consistency checking that uses packet delivery ratios to determine if a radio link with poor utility is due to a poor link quality or due to jamming. Tague [42] proposed a jamming and anti-jamming mechanism based on mobility control, where node mobility is used as a tool to improve network protocol performance. They demonstrated through simulation how local information such as link quality, routing diversity, and resilience to jamming can be reconfigured by determining the node mobility patterns. They concluded that these performance factors can all be incorporated into node mobility control to increase the spatial diversity of the network with the intent of improving its performance against jamming.

Shortcomings: The authors cited consider jamming in traditional wireless networks. There is a need to investigate the performance of CRs in the presence of jammers, since CR networks differ significantly from traditional wireless networks in their capabilities and mode of operations. The solution spaces proposed by these authors are only suitable to deal with benign and omissive faults. However, they are not effective if jamming is done in such a way that produces value faults.

5.3.2 Anti-jamming Strategies for CR Networks

Several authors [9, 25, 26, 27] have investigated jamming in Cognitive Radio Networks. Many of these authors [10, 19, 56, 34] have proposed the use of Spread Spectrum [36, 50] (Frequency Hopping (FH) and Direct Sequence Spread Spectrum (DSSS)) as a solution strategy to alleviate jamming attacks in wireless networks. The popular approach is to spread the signal over a larger bandwidth thereby making it costly for jammers to hinder an on-going transmission. The combination of Spread Spectrum and Orthogonal Frequency Division Multiplexing was also considered by [17, 28, 61] as an efficient means of mitigating jamming attacks in wireless networks. In CRNs based on the Spread Spectrum approach [9, 28] the available spectrum is divided into several pieces of non-overlapping channels in which only a small portion of the channels is used for transmission at a time. The malicious jammer either jams a large number of the channels with negligible jamming effect in each channel or jams few channels, which might not be in use by the cognitive radios. If channels used by a CR are jammed, the CR built on the concept of Software Define Radio has the flexibility to dynamically switch to another channel free of jamming.

Shortcomings: The problem with the related work here is that if a jammer is a CR, it is assumed to have knowledge of the hopping sequence. It could therefore adapt itself to the hopping sequence of the CR nodes, leading to a jamming attack known as Chaser Jamming [25], in which the jammer chases the CR node as it switches from channel to channel. Another issue is that when channels used by a CR are jammed (even when using frequency hopping), any data being transmitted at that instance is either lost or corrupted. As a result, the proposed solutions by these authors are only suitable to deal with benign and omissive faults and are not effective if jamming is done in such a way that produces value faults.

5.3.3 Forward Error Correction (FEC) Codes

Since the cognitive radio's ability to switch from channels to channels does not preclude it from data loss as a result of CR jammers, techniques based on FEC were also considered as anti-jamming strategies in CRNs. FEC schemes like LT-codes (Luby Transform-codes) [32], Raptor code [39] and Low Density Parity-Check Codes (LDPC) [38], can be used to recreate lost data due to jamming in a CRN through information redundancy. The Raptor code is a type of Fountain code [38] with linear time encoding and decoding in which a message made up of a number of k symbols encoded into an infinite series of symbols in such a way that if during transmission some part of the data is lost, e.g., due to jamming, the lost data can be recovered with a non-zero probability that increases as the number of the received symbols increases beyond k [39].

Shortcomings: The problem with the Raptor code is that it has not been investigated for CR Networks. Raptor code by itself can only deal with benign and omissive faults as it is not effective in handling jamming that produces value faults. Jamming in CR that produces value fault will lead to a decoding failure of the raptor code because the use of any of the corrupted packet containing encoding data will lead to the decoding of a message that is completely corrupted or manipulated.

5.4 Hybrid FEC Code Formulation

In order to overcome the limitations of related work, we propose a hybrid FEC code. This is defined by the concatenation of the Raptor code and Secure Hash Algorithm-2 (SHA-2) [40]. SHA-2 is the common name used for three types of computationally efficient hash functions, namely SHA-256, SHA-384, and SHA-512, which transform any set of data elements into 256 bit, 384 bit, and 512 bit fixed length values respectively. The output of SHA-2, known as message digest, can then be used to verify the integrity of the original data sent against the message digest of the data received at the destination. The advantages of using SHA2 are:

1. SHA-2 is collision resistant, meaning that the probability of finding two messages (no matter the similarities) that will hash to the same hash value or message digest will require work equivalent to $2^{n/2}$ hash computations [54, 40]. This property ensures the integrity of any data transmitted using SHA-2 since any manipulation made to the transmitted message will generate a different hash value with a very high probability.
2. SHA-2 computation is a one-way operation. The characteristic of being a one-way operation implies that for a given hash value, it should require 2^n hash computation in order to find any message that hashes to that same value [54]. The one-way operation characteristic of the SHA-2 is the basis of its security.
3. A hash value can be distributed or stored without any need for encryption since it is used for comparative purpose only.

We therefore propose to use the raptor code part of the code to recover any data loss due to omissive fault as a result of jamming. The SHA-2 hash function will be used to handle transmissive (value) fault due to jamming. Any value fault due to malicious jamming or bad channel will be detected with SHA-2 if the message digest generated at the receiver is different from the message digest generated at the sender, since it is computationally infeasible to find two different messages having the same digest or to find a message that hashes to a given message digest. Detection can result in different actions, depending on the CRN application scenario. We therefore identify these different scenarios as the *recovery block* [65] for the hybrid code, which is discussed in Sub-section 5.4.1.

5.4.1 Recovery Block for the Hybrid FEC Code

Whenever the hybrid code detects a discrepancy between the message digest of the data sent and the message digest of the data received, the line of action to be taken could be any of the options listed below depending on the application scenario:

1. The raptor part of the code could be used to iteratively correct suspected corrupted bits of the message as if it was omitted and then SHA-2 is used to regenerate the message digest each time to verify the data received. The iterative process involves extending the number of encoded symbols received at the destination beyond k to reduce the probability of a decoding failure of the Raptor code. After each iteration, a different combination of the encoded symbols (made up of the original message and the redundant symbols) is decoded and a received message is generated for the receiver with the Raptor code. SHA-2 is then used to compute the message digest of the received message, which is again compared with that of the sent message. Since it is believed that the probability of a decoding failure decreases as more encoding symbols are received, any data manipulation due to jamming would have been corrected with high probability using this technique before the entire encoded symbols sent from the source are received at the destination.
2. The erroneous message can be discarded and the system can be suspected to be under jamming attack, requiring further actions. Actions required using this technique include: (i) moving away of the CR nodes from the suspected source(s) of jamming, (ii) re-authenticating all the CR nodes participating in the CRN activities, (iii) re-validating the activities of these CR nodes to see if there is any violation of any of the CR security requirements, or (iv) isolating any suspected jamming nodes from the cognitive radio network.
3. If the channel jammed is a CR control channel, the CRs are capable of switching to another channel that is not jammed. The new channel is made the control channel and control messages are routed through this channel [58].

4. When all the earlier mentioned options fail, the receiver could request re-transmission of the message as a last resort. It is deemed to be a last resort because FEC codes were originally developed to compensate for the need to re-transmit lost messages especially in applications where re-transmission is not allowed because messages are time-sensitive or re-transmission is impossible. Examples of these applications include transmission to space, real-time video streaming and CRNs for emergency rescue where spectrum might just be available for use for a limited time only. In these type of scenarios, the cognitive radios rely on sending "unreliable" UDP packets, which do not require handshake between the sender and receiver, before and after the transmission.

5.4.2 Design of the Hybrid Code

Figure 5.1 describes the design of the new hybrid code. The figure depicts the layers of implementation in the OSI network reference model [51] and also gives an overview of the activities in the system as a message is sent from the sender to a receiver. This procedure is illustrated in more detail with Figure 5.2, which is a flowchart of the complete process. A sender's message is sent from the application layer down the network as a UDP message. The hybrid FEC code operates between the application layer and the transport layer. Therefore, the message digest of the message is generated by the SHA-2 module and the message digest is inserted into the message's packet header while the message is forwarded to the encoder part of the raptor module. The message is then passed through the pre-code stage of the encoder where redundant symbols are added to the message and the LT code is then used to generate encoded output symbols. These encoded output symbols are now sent down to lower layers of the network. The physical layer at the sender's node sends the encoded message to the physical layer of the receiver.

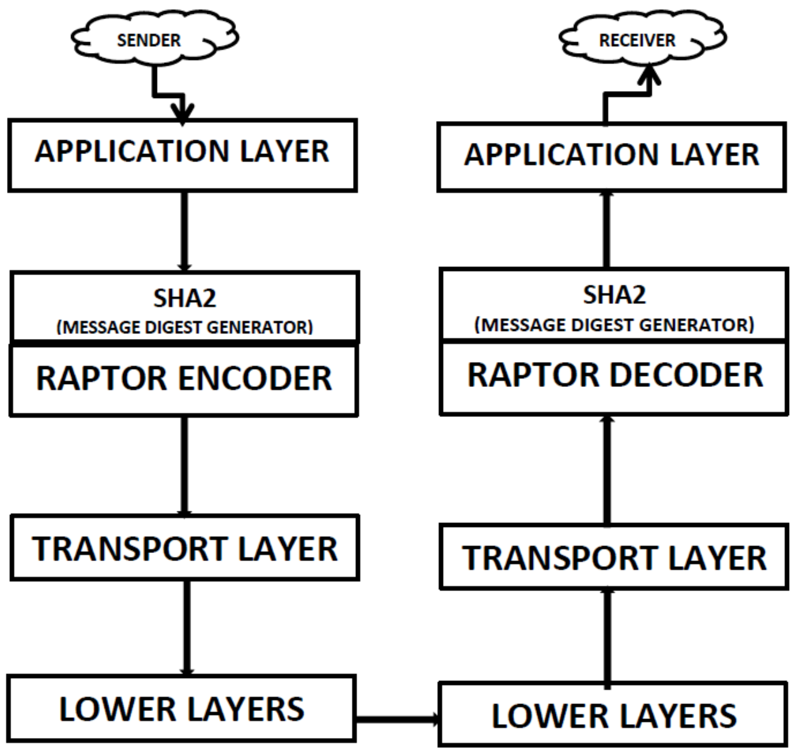


Figure 5.1: **System Design of the Hybrid FEC Code**

At the receiver, the encoded message is sent by the lower layers to the transport layer. At the transport layer, the encoded message is passed through the decoder part of the Raptor code. The decoder starts decoding the message as soon as it receives about $k(1 + \varepsilon)$ of the encoded output symbols. Once the decoding process is successful, the decoded message is sent to the SHA-2 module where the message digest of the decoded message is generated. The message digest generated is then compared with the message digest stored in the packet's header, which was the message digest for the original message. If the two message digests match, then the message is deemed to have been received with no error of any kind and it is forwarded to the receiver through the application layer. If there is a difference between these two message digests, then it means that the message has been manipulated along the transmission path, e.g., by a CR jammer, and the hybrid code will initiate any of the recovery block procedure described in Sub-section 5.4.1.

5.5 The Hybrid Code Algorithm

The algorithm of the proposed hybrid FEC code is described in this section. We first describe the different processes involved in SHA-2 computation of the message digest and later describe the Raptor algorithm part of the code, which is made up of the encoding and decoding processes.

5.5.1 Algorithm for SHA-2

SHA-256, which is the 256-bit version of SHA-2, was chosen for the implementation of the hybrid FEC code. SHA-256 was defined in the NIST (National Institute of Standards and Technology) standard FIPS 180-2 [48]. A message M of length l can be hashed with SHA-256 to produce a 256-bit message digest provided $0 \leq l < 2^{64}$. The lists of parameters and symbols used in computing the hash values for SHA-2 are presented in Tables 5.1 and 5.2 respectively. The SHA-256 algorithm is divided into two stages: (i) Pre-processing stage and (ii) Hash Computation stage. We here present a summary of the different stages of the algorithm as discussed in [48].

Pre-processing

The pre-processing stage involves three essential steps which are discussed as follows:

1. Set the initial hash value, $H^{(0)}$, which consist of eight 32-bit words in hexadecimal. They are obtained from taking the first thirty-two bits of the fractional parts of the square roots of the first eight prime numbers.
2. Pad the message, *M*. Padding is done to ensure that the padded message is a multiple of 512 bits. This could be done before the hash computation stage or during the Hash Computation stage provided it is done before the block(s) that will contain the padding is processed. A message, *M* of length l bits is padded by appending the bit "1" to the end of the message, followed by *k* zero bits, where k is the smallest non-negative solution to the equation $l + 1 + k \equiv 448 \bmod 512$. Then we append the 64-bit block that is equal to the number l expressed using a binary representation.
3. Parse the message into message block. Here, the message and its padding are parsed into *N* 512-bit blocks which can be expressed as sixteen 32-bit words. The first 32 bits of message block *i* are denoted $M^{(i)}_{0}, \ldots, M^{(i)}_{15}$.

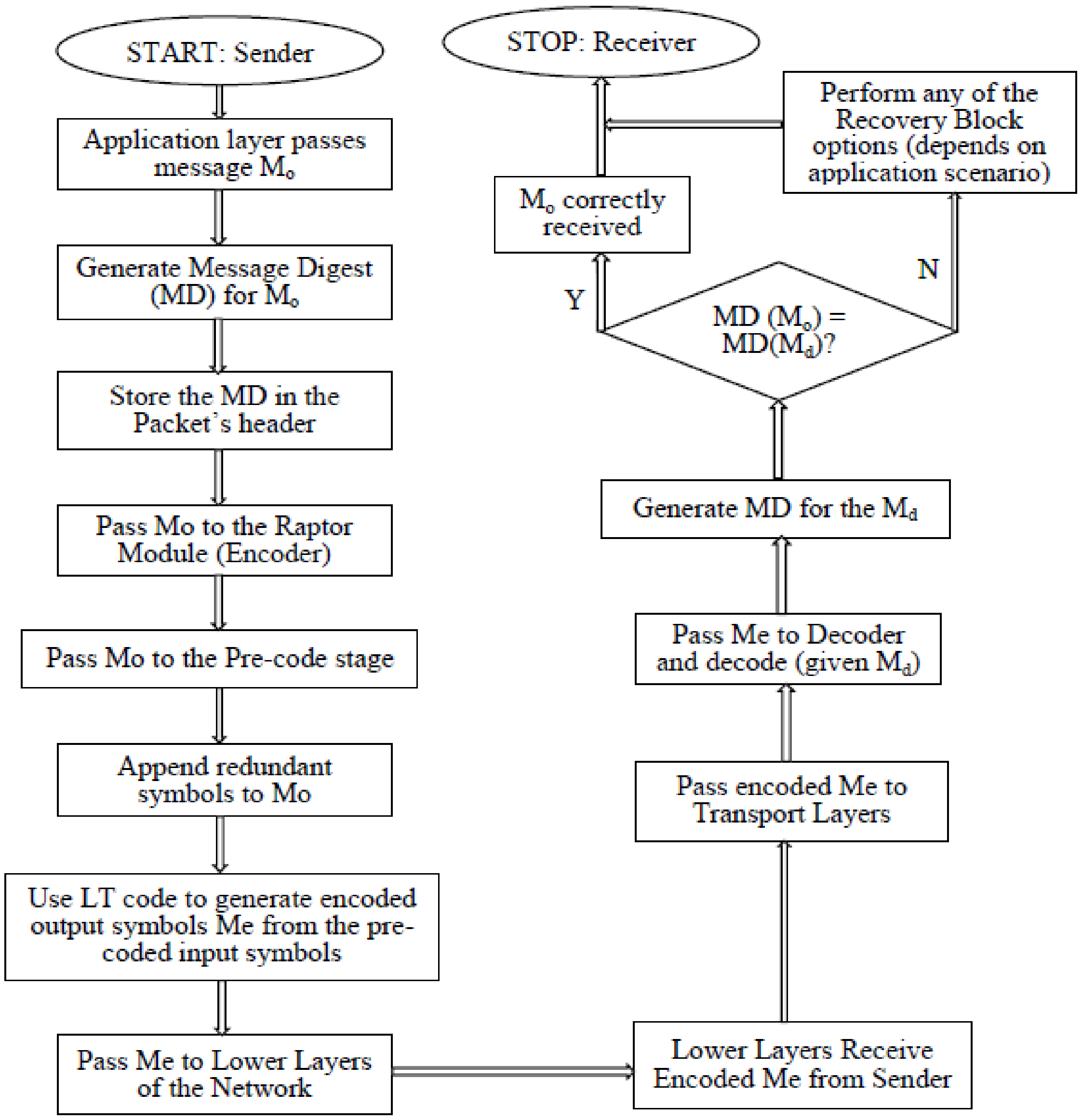

START: Sender
Application layer passes message M_o
Generate Message Digest (MD) for M_o
Store the MD in the Packet's header
Pass Mo to the Raptor Module (Encoder)
Pass Mo to the Pre-code stage
Append redundant symbols to Mo
Use LT code to generate encoded output symbols Me from the pre-coded input symbols
Pass Me to Lower Layers of the Network
Lower Layers Receive Encoded Me from Sender
Pass encoded Me to Transport Layers
Pass Me to Decoder and decode (given M_d)
Generate MD for the M_d
MD (M_o) = MD(M_d)?
Y
N
M_o correctly received
Perform any of the Recovery Block options (depends on application scenario)
STOP: Receiver

Figure 5.2: **Flowchart for the Hybrid FEC Algorithm**

***Table 5.1:* Parameters used in SHA-2 Computation**

	Parameters	Description
1	*a, b, c, ..., h*	Working variables that are the w-bit words used in the computation of the hash values, $H^{(i)}$
2	$H^{(i)}$	The i^{th} hash value. $H^{(0)}$ - is the initial hash value; $H^{(N)}$ - is the final hash value and is used to determine the message digest.
3	$H_{(j)}^{(i)}$	The j^{th} word of the i^{th} hash value, where $H_{(0)}^{(i)}$ is the left-most word of hash value i.
4	K_t	Constant value to be used for the iteration t of the hash computation.
5	k	Number of zeroes appended to a message during the padding step.
6	*l*	Length of the message, M, in bits.
7	m	Number of bits in a message block, $M^{(i)}$.
8	M	Message to be hashed
9	$M^{(i)}$	Message block i, with a size of m bits.
10	$M_j^{(i)}$	The j^{th} word of the i^{th} message block, where $M_0^{(i)}$ is the left-most word of message block *i*.
11	n	Number of bits to be rotated or shifted when a word is operated upon.
12	N (s)	Number of blocks in the padded message.
13	T	Temporary w-bit word used in the hash computation.
14	w	Number of bits in a word.
15	W_t	The t^{th} w-bit word of the message schedule.

***Table 5.2:* Symbols used in SHA-256 Hash Computation**

	Symbol	Definition
1	$\wedge$	bitwise AND operation
2	$\vee$	bitwise OR ("inclusive-OR") operation
3	$\oplus$	Bitwise XOR ("exclusive-OR") operation
4	$\neg$	Bitwise complement operation
5	+	Addition module 2^w
6	«	Left-shift operation, where x«n is obtained by discarding the left-most n bits of the word x and then padding the result with n zeroes on the right.
7	»	Right-shift operation, where $x >> n$ is obtained by discarding the right-most n bits of the word x and then by padding the result with n zeroes on the left.

SHA-256 Hash Computation

Addition (+) in SHA-256 is performed modulo 232. A list of constants and functions used in computing the hash is presented in the standard. Each message block, $M^{(1)}$, $M^{(2)}$, … $M^{(N)}$, is processed in order, using the following procedure:

For i = 1 to N:

{

1. Prepare the message schedule, $\{W_t\}$:

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases}$$, where

$$\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \tag{5.1}$$

$$\sigma_0^{\{256\}}(x) = ROTR^{7}(x) \oplus ROTR^{18}(x) \oplus SHR^{3}(x) \tag{5.2}$$

$SHR^n(x) = (x \gg n)$, is the *right shift* operation,

x is a w-bit word and n is an integer with $0 \le n < w$

2. Initialize the eight working variables, *a, b, c, d, e, f, g,* and *h*, with the $(i-1)^{st}$ hash value:

$a = H_0^{(i-1)}$
$b = H_1^{(i-1)}$
$c = H_2^{(i-1)}$
$d = H_3^{(i-1)}$
$e = H_4^{(i-1)}$
$f = H_5^{(i-1)}$
$g = H_6^{(i-1)}$
$h = H_7^{(i-1)}$

3. For $t = 0$ to 63 :

{

$T_1 = h + \sum{}_1{}^{(256)}(e) + Ch(e,f,g) + K_t{}^{(256)} + W_t$, where

$$\sum{}_1^{\{256\}}(x) = ROTR^{6}(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \tag{5.3}$$

$$Ch(e,f,g) = (e \wedge f) \oplus (\neg e \wedge g) \tag{5.4}$$

$T_2 = \sum_0{}^{(256)}(a) + Maj(a,b,c)$, where

$$Maj(a,b,c) = (a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c) \tag{5.5}$$

$$\sum\nolimits_0^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \tag{5.6}$$

$ROTR^n = (x \gg n) \vee (x \ll w - n)$, is the ***rotate right*** (circular right shift) operation, x is a w-bit word and n is an integer with $0 \leq n < w$

$h = g$

$g = f$

$f = e$

$e = d + T_1$

$d = c$

$c = b$

$b = a$

$a = T_1 + T_2$

}

4. Compute the i^{th} intermediate hash value $H^{(i)}$:

$H_0^{(i)} = a + H_0^{(i-1)}$

$H_1^{(i)} = b + H_1^{(i-1)}$

$H_2^{(i)} = c + H_2^{(i-1)}$

$H_3^{(i)} = d + H_3^{(i-1)}$

$H_4^{(i)} = e + H_4^{(i-1)}$

$H_5^{(i)} = f + H_5^{(i-1)}$

$H_6^{(i)} = g + H_6^{(i-1)}$

$H_7^{(i)} = h + H_7^{(i-1)}$

}

After repeating steps one through four a total of N times (i.e., after processing $M^{(N)}$), the resulting 256-bit message digest of the message, M, is

$H_0^{(N)}||H_1^{(N)}||H_2^{(N)}||H_3^{(N)}||H_4^{(N)}||H_5^{(N)}||H_6^{(N)}||H_7^{(N)}$

5.5.2 Algorithm for Raptor Code

In this section, we present a brief description of the Raptor code algorithm to provide the reader with an overview of what it involves in implementing it as part of our proposed Hybrid code. For theoretical details and proof of this algorithm, the reader is referred to [38]. The Raptor code is defined as follows:

> Let C be a linear code of block-length n and dimension k, and $\Omega(x)$ be a distribution on $(1, \ldots, k)$. A Raptor code with parameters $(k, C, \Omega(x))$ is an LT-code with degree distribution $\Omega(x)$ on n symbols which are the coordinates of code-words in C [29].

The code C is called the pre-code of the Raptor code. The algorithm of the Raptor code could be formulated into a matrix representation in which multiplication operations performed on the matrices yield solutions that are equivalent to executing the Raptor code algorithm [38]. We here first present the formal description of the Raptor code algorithm and then present the matrix interpretation equivalent of the algorithm. The Raptor code is made up of two essential components: (i) Raptor encoder (ii) Raptor decoder.

Algorithm for Raptor Encoder

The Raptor Encoder is made up of two stages:

1. **Pre-code stage:** This stage is usually done with an LDPC code but other codes like Gray code [59] and Hamming code [63] can be used as well. The source or input symbols k are mapped unto l intermediate symbols such that $l < k$.
2. **LT-encoding stage:** At this stage, the intermediate symbols from the pre-code stage is used to generate each encoded symbols by carrying out an exclusive-or (XOR) operation on a subset of symbols randomly selected from the l intermediate symbols in such a way that it agrees with a certain probability distribution. This essential property of being able to generate as many symbols as required during the encoding process attributed to the Raptor code being categorized as a rateless code (meaning no fixed code rate). Using the matrix interpretation, the preceding Raptor encoder algorithm could be represented as operations that involve performing matrices multiplications with vectors with the intent of solving systems of equations that represent the Raptor encoding process. The coefficients of these matrices are either zero or one and therefore are said to be binary. The vectors are vectors of symbols, with each symbol represented as a binary digit. Using

this matrix interpretation, the “Pre-code stage” of a Raptor code is defined as the multiplication:

$$u^T = G.x^T \tag{5.7}$$

where G is an $n \times k$ binary matrix representing the generator matrix of the pre-code and x represents the vector $(x_1, \ldots, x_k)$ made up of the input vectors. For the LT-encoding stage, according to [38], to any given set of N output symbols of the Raptor code, there corresponds a binary $N \times n$-matrix, S. This implies that matrix S contains rows that are independently sampled from the distribution $\Omega(x)$ and they are vectors that corresponds to the output symbols of the Raptor code. The relationship is represented as:

$$S.G.x^T = z^T \tag{5.8}$$

where $z = (z_1, \ldots, z_N)$ represents the encoded output symbols of the Raptor code that is made up of the column vector.

Algorithm for Raptor Decoder

The decoding algorithm for a Raptor code is defined as an algorithm of length m that can regain k input encoded symbols out of any set of m output symbols and errs with a probability that is at most $1/k^c$ for some positive constant c [38]. The decoder can recover the source symbols from any set of $\Theta = k(1 + \varepsilon)$ encoded symbols, with Θ slightly larger than k, where ε is the decoding inefficiency or the overhead. Using the matrix interpretation as in the case of Raptor encoder, the Raptor decoding algorithm corresponds to solving for the vector x in the system of equation given in Equation 5.8 to obtain the input vectors, $x_1, \ldots, x_k$.

5.6 Validity and Threshold Functions for the Hybrid Code

5.6.1 Proof of Validity

The Raptor codes have been described as a potentially strong and efficient extension of LT codes that has both encoding and decoding algorithms that are linear in time and very close to an ideal code operating under any channel situation [23]. It has been proved in [38] that: "if the decoding algorithm for the Raptor code errs with the probability p, then the probability that the Raptor algorithm fails is at most *p*." The summary of the statement of the proof is re-stated as follows:

> Let *S* be the matrix whose rows are the vectors $v_{1}, \ldots, v_{k(1+\varepsilon)}$. The system of Equations 5.8 is solvable if and only if the rank of *S.G* is *k*. The probability of solving the system using the decoding algorithm for the Raptor code is *p*, hence the probability that the rank of *S.G* is smaller than *k* is at most *p*.

This is a proof that the decoding algorithm is valid. For details of the proof that the algorithm for both Raptor encoder and Raptor decoder are valid, the reader is referred to [38].

For SHA-2 the proof of the validity of its algorithm can be inferred from its ability to generate a unique value for any set of data with the condition that given this generated value, it is computationally impossible to get back the original data. A collision is said to have occurred if two given messages produce the same hash value. To the best of our knowledge, no collision has been found to exist for SHA-2, which makes it our preferred hash function for this research.

5.6.2 Threshold Functions of the Hybrid Code

Since the effectiveness of the proposed Hybrid FEC code will be constrained by the thresholds of the Raptor code and SHA-2, we need to formulate expressions for these thresholds. Any manipulation by jamming beyond what these schemes can tolerate will render the hybrid approach ineffective. We here formulate an expression for our solution space (proposed hybrid codes) in terms of threshold function.

Raptor Codes Threshold

The threshold of Raptor code is given by $\Theta = k(1 + \varepsilon)$, which is the minimum amount of the encoded symbols that should be received by the receiver before it can start the decoding process, where k is the number of source symbols and " is the decoding inefficiency or overhead of the Raptor code. The threshold therefore is the minimum amount of symbols that must be received by the receiver to make the decoding successful. Any manipulation by a jammer that reduces this minimum requirement or delays the arrival of these minimum encoded symbols will adversely affect the performance of the hybrid FEC code. The encoding cost of a Fountain code is the number of arithmetic operations that is needed to generate each output symbol. For the Raptor code, it is the sum of the encoding cost of the pre-code divided by k, and the encoding cost of the LT code. The decoding cost of the Raptor code is the number of arithmetic operations needed to recover the k input symbols divided by *k*.

SHA-2 Threshold

Unlike the Raptor code, a hash function having a message digest of length *n*-bits has two properties [40]:

(i) **One Way:** It will require 2^n evaluations in order to find a message that corresponds to the given message digest.

(ii) **Collision Resistant:** Finding two different messages that produce the same message digest, known as a collision, requires an average of $2^{n/2}$ evaluations.

A jammer can only manipulate the proposed Hybrid code if a collision is found for SHA-2 and also if the jammer is able to satisfy the required number of evaluations to find a message that corresponds to a given hash value.

5.7 Creating the Hybrid Code as an Application in NS-2

We here specify the methodology for creating the hybrid FEC code application as an NS-2 agent. The application is intended to run over a UDP connection by simulating a UDP agent that generates a data packet that incorporates in its header specific information like Block ID, symbol ID, size, etc. The data is encoded by adding redundant symbols (using the Raptor module) and transmitted using the send agent of the application over the CRNs to the sender. The receiver on the other hand receives this encoded data using the receive agent, which decodes the message as soon as it receives enough symbols necessary for the decoding process. After successful decoding, the hash value of the decoded data is generated and compared with the hash value stored in the application layer packet's header, which was initially generated at the sender. If these hash values match, then the message is assumed to be correctly received and forwarded up to the receiver. If the hash values do not match, then the network is deemed to be under jamming attack and a recovery block depending on the application scenario is explored for recovery from the jamming attack.

It is worthy of note that the UDP agent allocates and sends network packets as a stream of data and all the information about the data needed at the application layer are sent using the packet's header. Hence, we need to modify the current UDP agent's implementation in NS-2 to be able to incorporate all the necessary information required by the hybrid FEC code into the packet's header.

5.7.1 Design and Implementation

We studied the C++ class hierarchy and specify the name of the class for the hybrid code application as "SharapApp" (representing our concatenated code of SHA-2 and Raptor code) and implement it as a child class of "Application". The OTcl (object Tcl) hierarchy name that matches this is "Application/SharapApp". The way that both the sender and receiver behave in the application is implemented in "SharapApp". We first describe the implementation of the main components of the hybrid code application in NS2 before a detailed description of other sub-components.

Message Digest Generator

The message digest (MD) generator is the SHA-2 component of the hybrid FEC code. The MD generator is defined in the "MessDigest.cc" and "MessDigest.h" of our hybrid FEC code implementation in NS2. The MD generator is implemented at both sender's and receiver's application layers. A sender's message is received at the application layer and represented as "SharapDATA". The MD generator is applied on the message i.e. "SharapDATA" in order to generate the message digest "messDigest" of the sender's message. The hash value in "messDigest" is then stored in the application layer header of the hybrid FEC code implementation. We specify a packet header for the application level communication after the NS2 C++ header structure as " hdr_sharapApp". This application layer header structure is the format used whenever the application has a message to transmit. After storing the message digest of the message using this header structure, the message is then handed over to the transport layer agent "UdpSharapAgent" in the "hdr_sharapApp" format.

Raptor Encoder

The Raptor Encoder is defined in "RaptorEncod.cc" and "RaptorEncod.h" of our NS2 hybrid FEC code implementation. The design of the Raptor Encoder in NS2 is illustrated with Figure 5.3. An application data (sender's message + header) sent from the application layer and received at the transport layer of the sender's network by the "UdpSharapAgent" is first broken down into T data blocks. Each block with a unique ID "BID" is then divided into k input symbols $\Omega = (x_1, \ldots, x_k)$. A symbol is simply a unit of data, meaning that a block comprises of k number of data units. The size of each symbol in bytes is denoted by l(bytes). Every symbol in a block has unique ID "SID". A symbol is represented in our application as "SharapSymb". To reduce the complexities involved in fragmenting and re-assembling of "SharapSymb" packets at the sender and receiver's networks respectively, constrained the size of a symbol to a maximum size l of 1450 bytes. This is because the default maximum transmission unit (MTU) of the Ethernet technology is 1500 bytes. We derived the symbol size of 1450 bytes using the formula:

EthernetMTU - IP_header - UDP_header - Ethernet - Application_header = Symbol_size

Applying the formula we have:
1500 (bytes) - 20 (bytes) - 8 (bytes) - 14 (bytes) - 8 (bytes) = 1450 bytes

The Raptor encoder is made up of two functional modules: (i) Pre-code (ii) LT encoder. The pre-code involves the application of an $n \times k$ binary matrix A, representing the generator matrix of the pre-code, on the k source symbols to obtain intermediate symbols $\rho = (c_1, \ldots, c_\Theta)$. Matrix A is defined in "AMatrix.cc" and "AMatrix.h" in our NS2 implementation. Matrix A is made up of four essential components. The components are (i) LDPC generator matrix, (ii) Gray code generator matrix, (iii) LT code generator matrix, and (iv) the degree distribution matrix representing the Robust Soliton Distribution (RSD) [32] which will be discussed shortly.

The LT encoder stage involves the random selection of the intermediate symbols $\rho = (c_1, \ldots, c_m)$ with $m < k$ by the LT encoder to produce the encoded output symbols

$\tau = (z_1, \ldots, z_\Theta)$. A pseudo random number generator is used to generate the random values used in the LT encoder. Each encoded output symbol is generated by the LT encoder randomly selecting a degree *d* such that d lies between *1* and k, from a suitable degree distribution. The degree distribution used for the hybrid FEC code implementation is the Robust Solution Distribution, which we defined its implementation in matrix *A*. The LT encoder also randomly selects a neighborhood of connected d intermediate symbols. The set of chosen intermediate symbols is then XORed by the encoder to obtain an output symbol z_x. The header of each encoded symbol contains information that is necessary for a successful decoding of the encoded symbols. This information includes, the seed used by the pseudo random number generator "SPRNG", the block identifier "BID" which uniquely identifies the block from which the symbol is obtained from, the "SID" representing the symbol ID, the symbol size *l*, the degree *d* of the degree distribution, and the list of neighbor indices "LNI".

According to [32] a suitable degree distribution that can be used by the LT encoder must satisfy the following design requirements:

(i) The degree distribution must ensure that on average, few encoding symbols are required to ensure the success of the LT encoding process and subsequently the decoding process.

(ii) The degree distribution must also ensure that the average degree of the encoding symbols is as low as possible.

The average degree $\ddot{d}$ of an encoding symbol is defined as the number of symbol operations required on the average to generate that encoding symbol. By implication, the number of symbol operations required on the average to recover the entire data is $\ddot{d}.K$, where K is the number of encoded symbols. We chose RSD as the degree distribution for the LT encoder because it ensures that most of the encoded symbols possess low degree (degree one). The advantages of a degree one encoded symbol are (i) it allows the decoding process to be started as soon as the symbol is received (ii) The number of addition operations required during the encoding and decoding process is significantly reduced.

The RSD is defined by [32] as follows: *Let* $W = c\ ln(k/\delta)\sqrt{k}$ *for some suitable constant c > 0, the Robust Soliton Distribution is defined as*

$$\lambda(d) = \begin{cases} W/dk & \text{for } d = 1, \ldots, k/W - 1 \\ W ln(W/\delta)/k & \text{for } d = k/W \\ 0 & \text{for } d = k/W + 1, \ldots, k \end{cases}$$

where δ is the allowable failure probability of the decoder to recover the data for a given number K of encoded symbols i.e. the decoding failure probability after $\Theta = k(1 + \varepsilon)$ encoded symbols have been received. Note that k is the number of source symbols, d is the degree of the distribution, and W is the expected number of degree one encoded symbols.

Raptor Decoder

The Raptor decoder, which operates at the transport layer of the receiver's network, is defined in "RaptorDecod.cc" and "RaptorDecod.h" components of the hybrid FEC code implementation in NS2. The Raptor decoder begins the decoding of the encoded output symbols sent from the sender's network as soon as it receives $\Theta = k(1 + \varepsilon)$ encoded symbols. This represents the threshold that makes the decoding process of a Raptor code to be successful. In order for the Raptor decoder to be able to decode the received encoded symbols, it will retrieve the degree distribution d used during the encoding process from the headers of the encoded symbols. It will also retrieve the indices of the set of chosen intermediate symbols which are also stored in the encoded symbols headers. After retrieving the necessary information, the decoder applies the pre-code on the set of symbols that have been received and then applies the LT code.

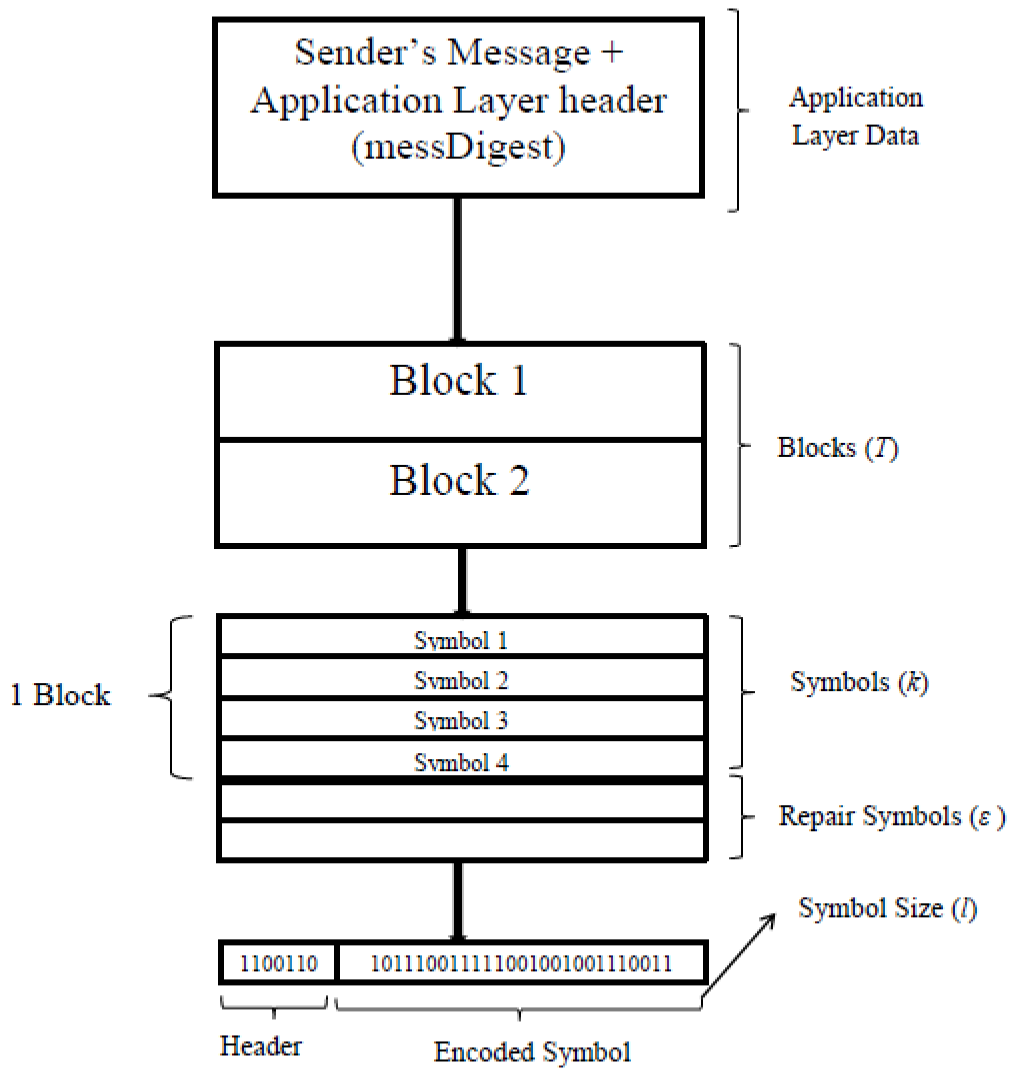


Figure 5.3: The Design of the Raptor Encoder

The equation of the matrix that is solved by the Raptor decoder representing the process of restoring the intermediate symbols ρ from the vector of encoded symbols τ being applied on the invertible generator matrix A of the pre-code is given by:

$$\rho^T = \tau^T .A^{-1} \tag{5.9}$$

Finally, the equation of the matrix that is solved, which represents the process of obtaining the source symbols Ω by applying the generator matrix G_{LT} for the LT code (G_{LT} , is defined in matrix *A* on the intermediate symbols ρ is given by:

$$\Omega = G_{LT} \, . \, \rho \qquad (5.10)$$

The decoding process is complete if all the input symbols in a block have been recovered to generate back the original block. Where the decoding process fails, the Raptor encoder waits for more encoded symbols to arrive and tries again the decoding process. If no more additional encoded symbol is received, the CRN can be suspected to be experiencing jamming and the hybrid FEC code recovery block comes into play here to help the system to recover from such attack.

For our implementation of hybrid FEC code in NS2, the CR network is suspected to be under jamming when there is a decoding failure of the Raptor code or when the message digest of the sent data does not match with that of the received data. If any of these happens, we implement the following components of the recovery block: (i) the CR nodes move away from suspected source(s) of jamming (ii) the CR nodes switch from the use of a jammed channel as control channel to another channel free of jamming.

5.7.2 Implementation Procedure

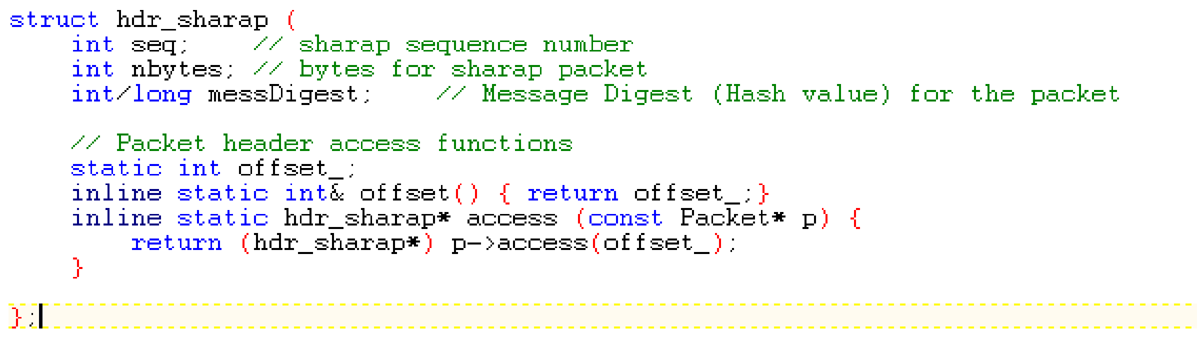
Since the present UDP agent format does not support our implementation (SharapApp), we specify our modified UDP agent as "UdpSharapAgent" and implement it as a child class of "UdpAgent". The OTcl hierarchy names that match these are "Agent/UDP/UDPsharapSender" and "Agent/UDP/UDPsharapReceiver" for both sender and receiver agent respectively. The following is a detail description of this implementation:

- **SharapApp header:** The transport layer agent "UdpSharapAgent" receives the application layer data and breaks the data into blocks using the block size specified in the Raptor encoder. Each block is then divided into k input symbols. The size of each symbol represented in our application as "SharapSymb", corresponds with the maximum UDP packet size specified in our simulation. The "UdpSharapAgent" writes all the necessary data to the "Sharap" header of the packet. We derived the "SharapHeaderClass" class from the NS2 "PacketHeaderClass" class, which is a definition of our new header’s class object. The structure of "hdr_sharap" is presented in Figure 5.4 while the definition for the "Sharap" header class is presented in Figure 5.5. In order to add our new "Sharap" header to the header stack, we added some lines of codes to NS2 "packet.h" and "ns-packet.tcl" files as depicted in Figures

5.6 and 5.7. The newly created header can now be accessed by "UdpSharapAgent" via "hdr_sharap::access()" member function. The detail of the hybrid code application and the modified UDP agent are implemented in "sharap-app.h", "sharap-app.cc", "udp-sharap.h", and "udp-sharap.cc".

- **SharapApp Sender:** The sender uses a timer for scheduling the next encoded data packet transmission. We defined the "SendTimer" class derived from the "TimerHandler" class, and wrote its "expire()" member function to call "send_sharap_pkt()" member function of the "SharapApp" object. Then, we included an instance of this object as a private member of "SharapApp" object referred to as "snd_timer_".
- **SharapApp Receiver:** The receiver uses a timer, named "ack_timer_", to schedule the next data packet transmission. When receiving an encoded symbol as data packet from the sender, the receiver counts the number of received encoded symbols. When this number gets to the threshold required for successful decoding of the encoded symbols, the decoding process commences.
- **UdpSharapAgent**: The NS2 "UdpAgent" is modified to "UdpSharapAgent" with the enhanced features of (i) ability to write into the modified UDP data packet, "Sharap" header, the necessary information received from "SharapApp", (ii) ability to read necessary information from the received packet header so as to forward it to "SharapApp", (iii) ability to carry out both segmentation and re-assembly, and (iv) setting the priority bit (IPv6) to 15 (max priority) for the "SharapApp" packets.
- **Modification to "agent.h"**: Two methods are added to "Agent" class as public in order to make the new application and agent running in NS2. An "attach-agent" OTcl command is also defined in the "command" member function of the "SharapApp" class, so that when this command is received from OTcl, "SharapApp" attempts to attach itself to the underlying agent. First of all, "SharapApp" calls the "supportSharap()" member function of the underlying agent to determine if the agent supports the type of transmission specified by "Sharap". If this is true, "enableSharap()" is invoked. Figure 5.8 depicts how these two methods are added to "Agent" class.

- **Modification to "app.h":** A member function "recv_msg(int nbytes, const char *msg) is added to "Application" class. This is illustrated in Figure 5.9.
- **Modification to "ns-default.tcl":** This modification is made in order to set default values for the newly created parameters for "SharapApp" as an application as illustrated in Figure 5.10 .
- **Modification to "Makefile":** The line of codes added to the object file list in the "Makefile" are "MessDigest.o", "RaptorEncod.o", "RaptorDecod.o", "AMatrix.o" "sharap-app.o" and "udp-sharap.o".



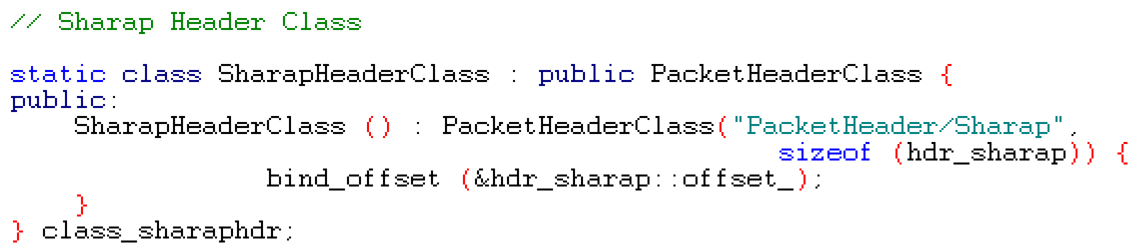
```
struct hdr_sharap {
    int seq;     // sharap sequence number
    int nbytes; // bytes for sharap packet
    int/long messDigest;    // Message Digest (Hash value) for the packet

    // Packet header access functions
    static int offset_;
    inline static int& offset() { return offset_;}
    inline static hdr_sharap* access (const Packet* p) {
        return (hdr_sharap*) p->access(offset_);
    }

};
```

Figure 5.4: **The Sharap Header Structure in “udp-sharap.h”**

5.7.3 Configuring the Hybrid FEC Code Agents in a TCL Script

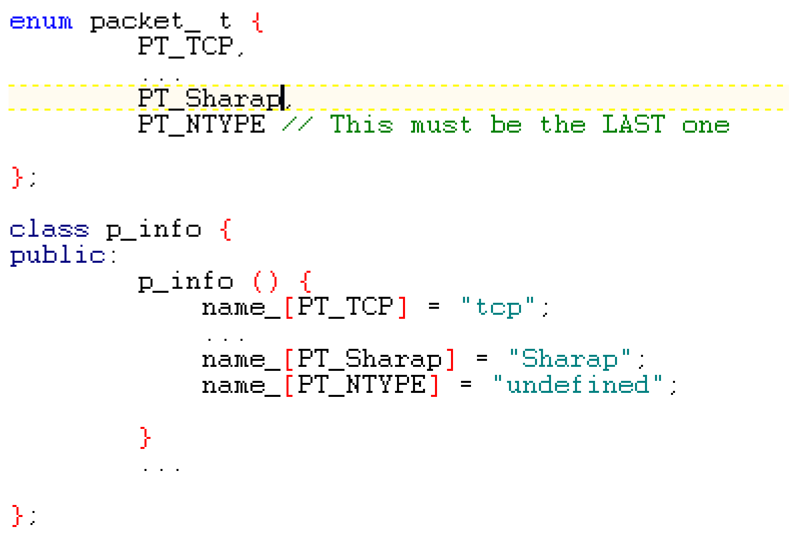
Configuring the Sender Agent Below is a typical example of how to configure the hybrid FEC code in an NS2 TCL script:

```
set SenderAgent0 [new Agent/UDP/UDPsharapSender]
$ns attach-agent $sender_node0 $SenderAgent0
```



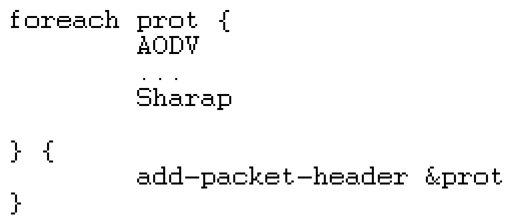
```
// Sharap Header Class

static class SharapHeaderClass : public PacketHeaderClass {
public:
    SharapHeaderClass () : PacketHeaderClass("PacketHeader/Sharap",
                                              sizeof (hdr_sharap)) {
                bind_offset (&hdr_sharap::offset_);
    }
} class_sharaphdr;
```

Figure 5.5: **The Sharap Header Class in "udp-sharap.cc"**



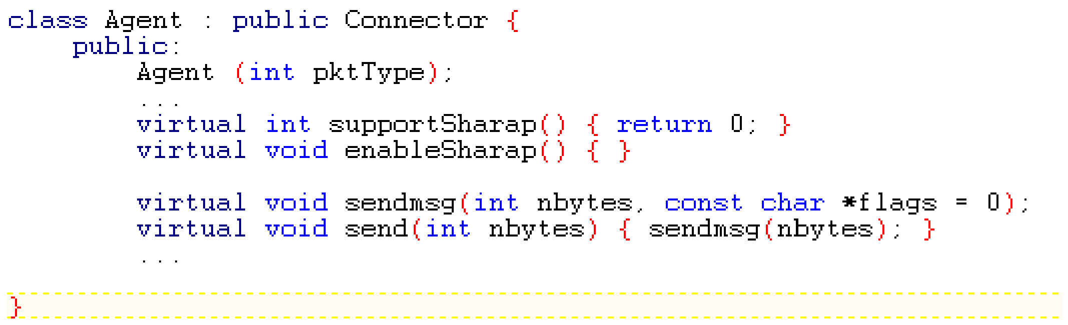
```
enum packet_ t {
        PT_TCP,
        ...
        PT_Sharap,
        PT_NTYPE // This must be the LAST one

};

class p_info {
public:
        p_info () {
            name_[PT_TCP] = "tcp";
            ...
            name_[PT_Sharap] = "Sharap";
            name_[PT_NTYPE] = "undefined";

        }
        ...

};
```

Figure 5.6: **Lines of codes added to "packet.h" file (C++)**



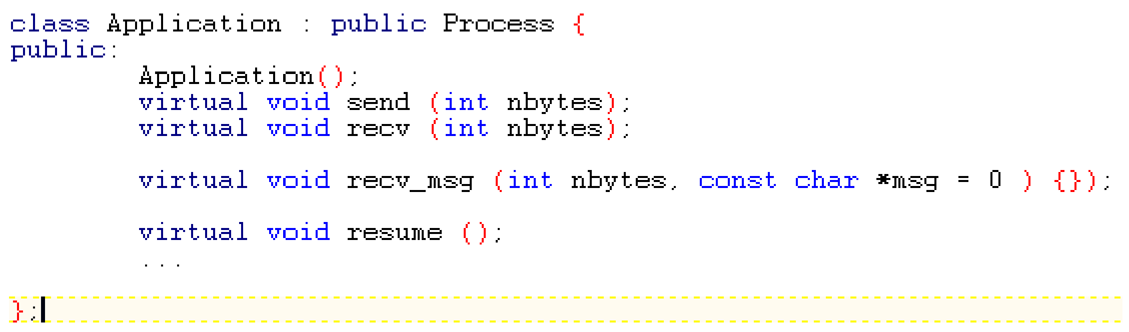
```
foreach prot {
        AODV
        ...
        Sharap

} {
        add-packet-header &prot
}
```

Figure 5.7: **Lines of codes added to "ns-packet.tcl" file (OTcl)**



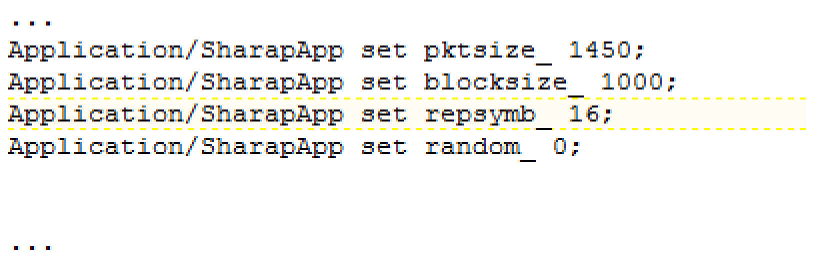
```
class Agent : public Connector {
    public:
        Agent (int pktType);
        ...
        virtual int supportSharap() { return 0; }
        virtual void enableSharap() { }

        virtual void sendmsg(int nbytes, const char *flags = 0);
        virtual void send(int nbytes) { sendmsg(nbytes); }
        ...
}
```

Figure 5.8: **Two member functions added to "Agent" class in agent.h**



```
class Application : public Process {
public:
        Application();
        virtual void send (int nbytes);
        virtual void recv (int nbytes);

        virtual void recv_msg (int nbytes, const char *msg = 0 ) {});

        virtual void resume ();
        ...
};
```

Figure 5.9: **Adding a member function to "Application" class in app.h**



```
...
Application/SharapApp set pktsize_ 1450;
Application/SharapApp set blocksize_ 1000;
Application/SharapApp set repsymb_ 16;
Application/SharapApp set random_ 0;


...
```

Figure 5.10**: Setting new default parameters in "ns-default.tcl"**

Configuring the Receiver Agent

Below is a typical example of how to configure the hybrid FEC code in an NS2 TCL script:

set sink1 [new Agent//UDP/UDPsharapReceiver]

$ns attach-agent $receiver_node1 $sink1

5.7.4 Justification for Hybrid FEC Code Efficient Design

The following factors attributed to the efficient design of the hybrid FEC code that makes it suitable for delay-sensitive CR networks application domains that use technology like Voice over Internet Protocol (VoIP), Video on Demand (VOD) and Internet Protocol Television (IPTV):

- The hybrid FEC code does not send the generated hash value (message digest) from the sender to the receiver using a separate packet. Instead, it uses the packet's header of the message to transmit its message digest to the destination. The advantages of this design include (i) faster encoding and decoding of messages, as the sender and the receiver start the encoding and decoding of messages respectively without first waiting to generate a separate packet for the message digest. (ii) the message digest on transit cannot be lost as long as the original message gets to the destination.

- An encoded symbol sent cannot be more than the specified UDP packet size. This implementation removes the complexity involved in and saves time of fragmenting and re-assembling of data fragments at the sender's and receiver's networks respectively.
- The message digest is generated at the application layer for the entire message. This ensures that a message does not require more than one message digest. Having a message broken down into fragments will require the generation of multiple hash values for all the data fragments.

5.8 Simulation

The following assumptions we made during the course of carrying out the simulations:

- Each jamming rate is derived as a cumulative effect of jamming activities from one or more jammers. This means that the jamming rate is a threshold function whose impact do not increase beyond the threshold even if more than one jammer is present in the network.
- Fusion center or base station for the CRN is trusted and therefore cannot become a jamming CR node.

5.8.1 Simulation Parameters

Table 5.3 presents the different simulation parameters that were used in simulating the hybrid FEC code. It is important to note that we only consider UDP packets for all the simulations carried out in this chapter because of the following reasons: (i) The primary user's spectrum is only available for a limited time, therefore, retransmission of lost packets is not a requirement as the case with TCP packets. (ii) The primary user might suddenly become present as a CR node is receiving a message and hence needs to hand off the occupied channel. This will make TCP operations to break down if this happens. (iii) A jammer might target TCP control messages, e.g., ack, CTS, which makes TCP inefficient for this implementation. (iv) The need for the receiver to respond urgently to a message sent to the fusion center, e.g., whether a primary user is present or not, renders TCP unsuitable as

TCP will introduce delay. (v) Since the applications for CRNs include transmission of multimedia data that requires real time data delivery, for which TCP is unsuitable, the objective of this research is to use UDP, as it does not request retransmission when there is a data loss.

5.8.2 Other Parameters

The following are other parameters that are non-NS2 parameters that we employed in our simulation. Since these parameters are very crucial to successful and correct implementation of the components of our hybrid code, i.e. SHA-2 and Raptor code, we provide the justification for the different values of the parameters used. In [29], the authors analyzed the different parameters that are necessary for the successful implementation of Raptor codes. The objective of their investigation was to determine the parameters that maximize the speed of transmission of data while at the same time decrease the time of decoding of encoded symbols. Essentially, the authors investigated the best trade-off between performance, computational complexity and resilience of the Raptor code. These parameters include:

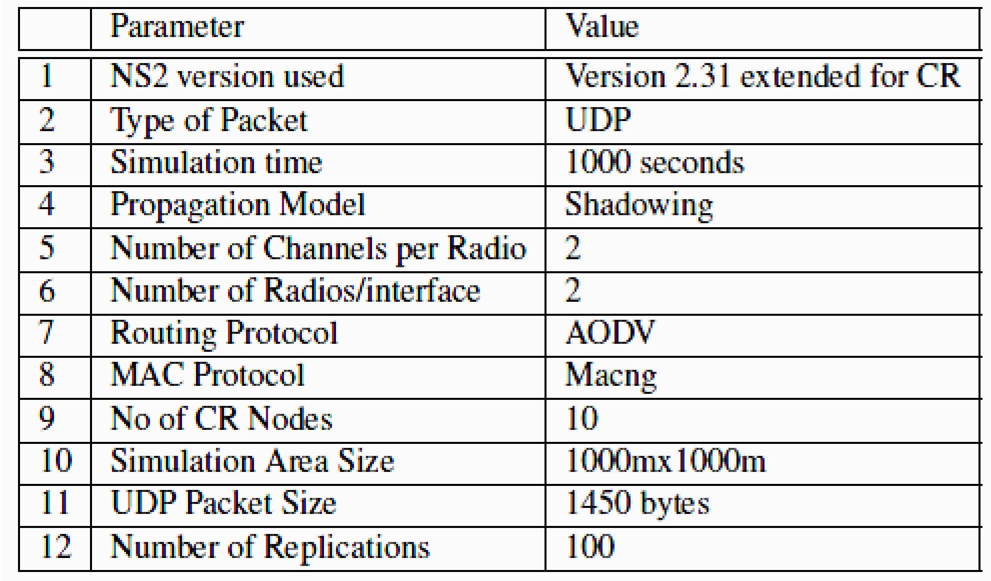
Number of input symbols *k* in a block: In their experiment, the authors in [29] discovered that if the block size is chosen to be very large, the receiver will encounter an excessive start up delay, and that if smaller block sizes are used at the beginning of the transmission, the start-up delay is reduced. They asserted that small block sizes are beneficial in realizing high encoding and decoding speed but that block sizes that are too small, i.e., with $k \leq 32$, will cause content switching overhead for the CPU. As a result of this analysis we use block size $k = 1000$ symbols for our simulation, representing a not too large or too small block size.

Length of input symbol l in bytes: The Raptor code has been defined as a code with a linear decoding time of $O(k \log(1/\varepsilon))$, it theoretically follows that this decoding time of the Raptor code could be decreased significantly if a large symbol size l and a large block size is chosen. In their analysis, the authors in [29] asserted that the symbol size l has little or no influence on the encoding and decoding speed. But they observed that when the symbol sizes are significantly large, a slight increase in the speed is noticed. Since the encoding and decoding speed is slightly favored by larger symbol sizes, putting into consideration the

Ethernet maximum MTU, we use a length of 1450 bytes which is the maximum UDP packet size we specified for our simulation.

Number of Repair symbol ε: The number of repair symbols that minimizes the overhead for our chosen block size $k = 1000$ was discovered to be about 16 repair symbols. The experiment in [29] shows that 16 repair symbols used for a block size of about 1000 symbols reduces the overhead to about 1.56%. We therefore use 16 as the number of repair symbols for our simulation.

Table 5.3: **Simulation Parameters for the Hybrid Code**



	Parameter	Value
1	NS2 version used	Version 2.31 extended for CR
2	Type of Packet	UDP
3	Simulation time	1000 seconds
4	Propagation Model	Shadowing
5	Number of Channels per Radio	2
6	Number of Radios/interface	2
7	Routing Protocol	AODV
8	MAC Protocol	Macng
9	No of CR Nodes	10
10	Simulation Area Size	1000mx1000m
11	UDP Packet Size	1450 bytes
12	Number of Replications	100

5.8.3 Definition of Simulation Performance Measures

Throughput

The throughput of a network is defined as the average rate of data successfully delivered by the network. The average throughput is defined as the number of packets received at a layer (e.g. application layer) divided by the total simulation time. The throughput is a measurement of the effective work done by the network per second. In this research, we are specifically interested in the total number of useful data delivered by the network, therefore we measured throughput in bits per second (bps), which we derived from converting the total number of data packets received per second to bits per second. For our implementation of the hybrid FEC code, a high average throughput will mean that the hybrid FEC code is able to enhance the CRN performance inspite of the DoS actions of the CR jammers. A low average throughput will mean that jamming effect moderately affect the performance of the hybrid FEC code, while a close to zero average throughput means that the jammers adversely affect the performance of the hybrid FEC code.

Packet Delivery Ratio (PDR)

The PDR is defined as the ratio of the number of packets received at a layer (e.g., application layer) to the number of packets sent in the same layer. The PDR distinguishes between a congested network and a network under jamming attack as a highly congested network can still produce a high PDR. Since the PDR estimates the recovery action of a FEC code in terms of its recovery rate with respect to the total packet sent by the network and the ability of the FEC code to cope with the effect of jamming, the data recovery rate of the hybrid FEC code can therefore be effectively measured using the PDR metric. The recovery rate of an FEC code is a ratio of the number of received encoded symbols to the total number of source symbols. A low recovery rate arises if the number of received encoded symbols is less than the number of source symbols. Therefore, the efficiency of a FEC code is characterized by its recovery rate. A high PDR will mean that the hybrid code is able to recover a high percentage of data that was successfully sent from the sender despite the DoS activities of the jammers. A low PDR will indicate that the data recovery rate of the hybrid code is low, as a small percentage of the data successfully sent was recovered at the receiver.

Packet Loss Ratio (PLR)

The PLR is defined as: Number of lost packet / (Number of lost packet + Number of packets received successfully). The PLR is closely related with the quality of service (QoS) of the hybrid FEC code. It also gives a more accurate estimation of the recovery rate of the hybrid code, because it does not put into consideration the number of data packets sent from the sender, but calculates the rate at which the hybrid code is able to recover successfully encoded input symbols at the receiver. A high recovery rate implies that the algorithm of encoding, decoding and SHA-2 hash computation and the action of the recovery block for the hybrid FEC code are computationally efficient in the sense that most of the encoded input symbols were successfully recovered without error at the receiver. A low PLR indicates that a high percentage of the packets were received either uncorrupted or that the corrupted packets were recovered by the recovery action of the hybrid code's recovery block. The relationship between the recovery rate of the hybrid FEC code and the PLR can thus be stated as:

$$Recovery\ Rate = 1 – PLR.$$

5.9 Analysis of Simulation Results

The parameters of measurement used for the simulation of the hybrid FEC code in NS2 extension for Cognitive Radio are throughput, PDR, and PLR. We therefore describe the results of our simulation using these metrics as follows.

5.9.1 Constant Jamming

Throughput

Figures 5.11 and 5.12 present snapshots of the plots of throughput against the simulation time for the fair case of constant jamming scenario, i.e., when the rate of jamming was 10%. Since we generated a sequence of replications of the same experiment in our simulation, the cases presented in the figures represent the replications that are closer to the computed averages of the total replications of the same experiment. In the case of ordinary FEC code, depicted in Figure 5.11, the code fails at every instance of a transmissive value fault. This is because any decoded data made up of corrupted encoded symbols is dropped by the CRN. The ordinary FEC code also fails whenever there is a decoding failure, i.e., when the number of encoded symbols recovered at the destination is less than *k*.

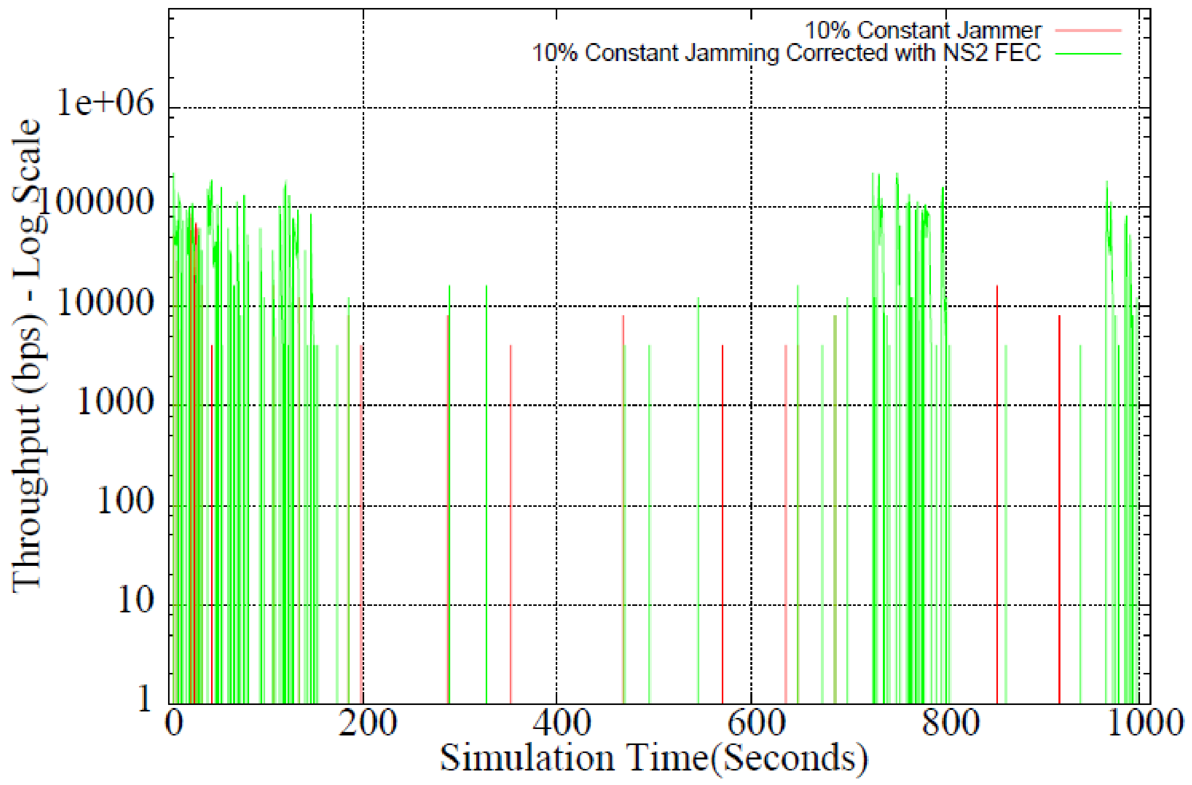


Figure 5.11: **NS2 FEC Code Performance against 10% Constant Jamming**

In the case of hybrid FEC code depicted by Figure 5.12, it is noticeable that the hybrid FEC code was able to mitigate most jamming caused by the constant jammer. The instances where the hybrid FEC code fails, represented by zero throughput in the figure, were results of the action of the recovery block for the hybrid code. The reasons why the hybrid FEC fails in the interval identified by zero throughput is either one or combinations of the following factors resulting from the implementation of the recovery block for the hybrid FEC code:

1. The minimum encoded symbols $k(1 + \varepsilon)$ received for a successful decoding by the hybrid FEC code were corrupted or manipulated (value fault). Therefore, the hybrid code waited for more encoded symbols, i.e., increasing the value of ", thereby introducing a type of delay we called "Iterative Decoding Delay" (IDD). This accounts for the failure of the network to deliver packets to the receiver whenever such delay is experienced by the CR Network.
2. Delay resulting from the time taken by CR nodes to re-locate from a suspected or potential source of jamming.

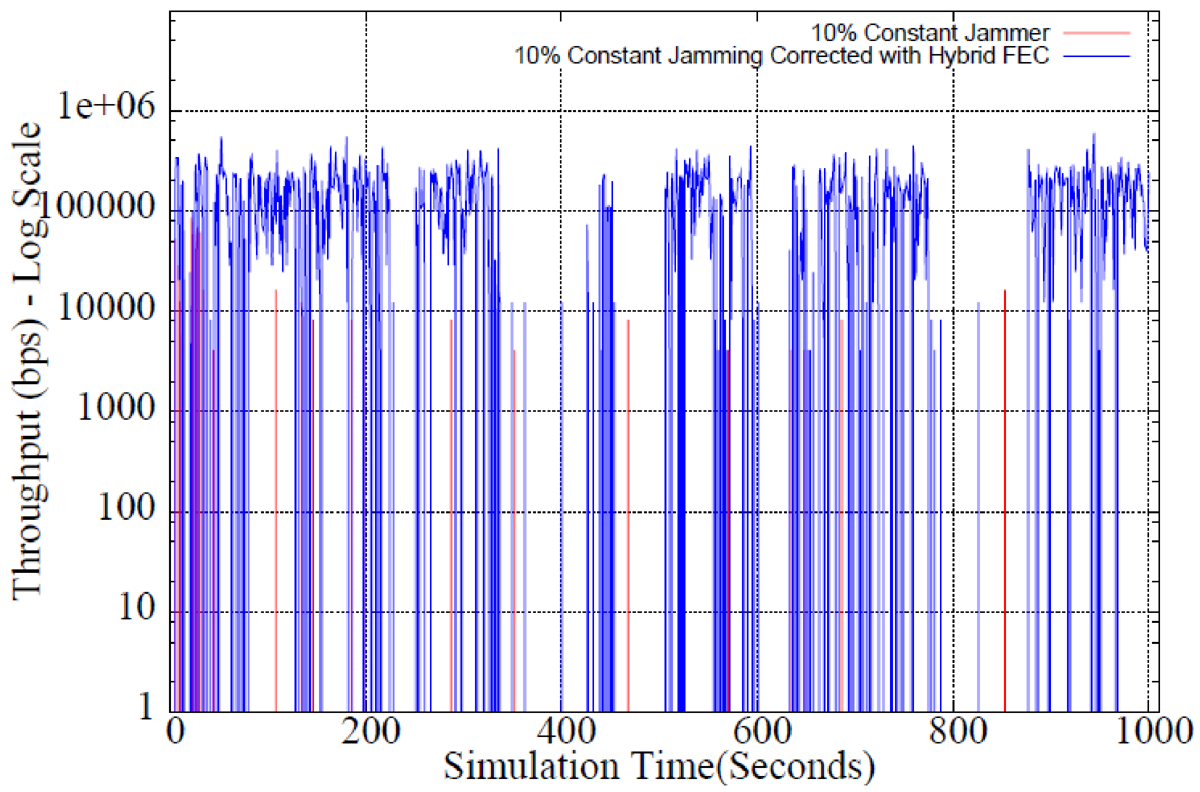


Figure 5.12: Hybrid FEC Code Performance against 10% Constant Jamming

3. Delay resulting from the time to use by the CRN to re-authenticate participating CR nodes in order to isolate a potential jammer.
4. Delay resulting from the time to switch from a jammed CCC to another channel free of jamming by CR nodes
5. The failure of the recovery block due to non-implementation of the retransmission procedure when a UDP data is corrupted. This retransmission procedure will only be implemented in applications that are not delay sensitive.

Figures 5.13 and 5.14 depict the snapshots for the moderate case of constant jamming, i.e., when the rate of jamming was 30% characterized by increasing instances of value faults introduced by the constant jammers. It is noticeable from the Figure 5.13 that the throughput of the ordinary FEC code dropped significantly as the code fails at every instance of transmissive value fault and whenever there is a decoding failure of the FEC code. The hybrid FEC code on the other hand in Figure 5.14 significantly outperforms the ordinary FEC code as it was able to deliver high throughput inspite of the jamming effect of constant jammers. Nonetheless, the hybrid FEC code fails in the intervals represented by zero throughput due to the actions of the recovery block earlier identified in this section.

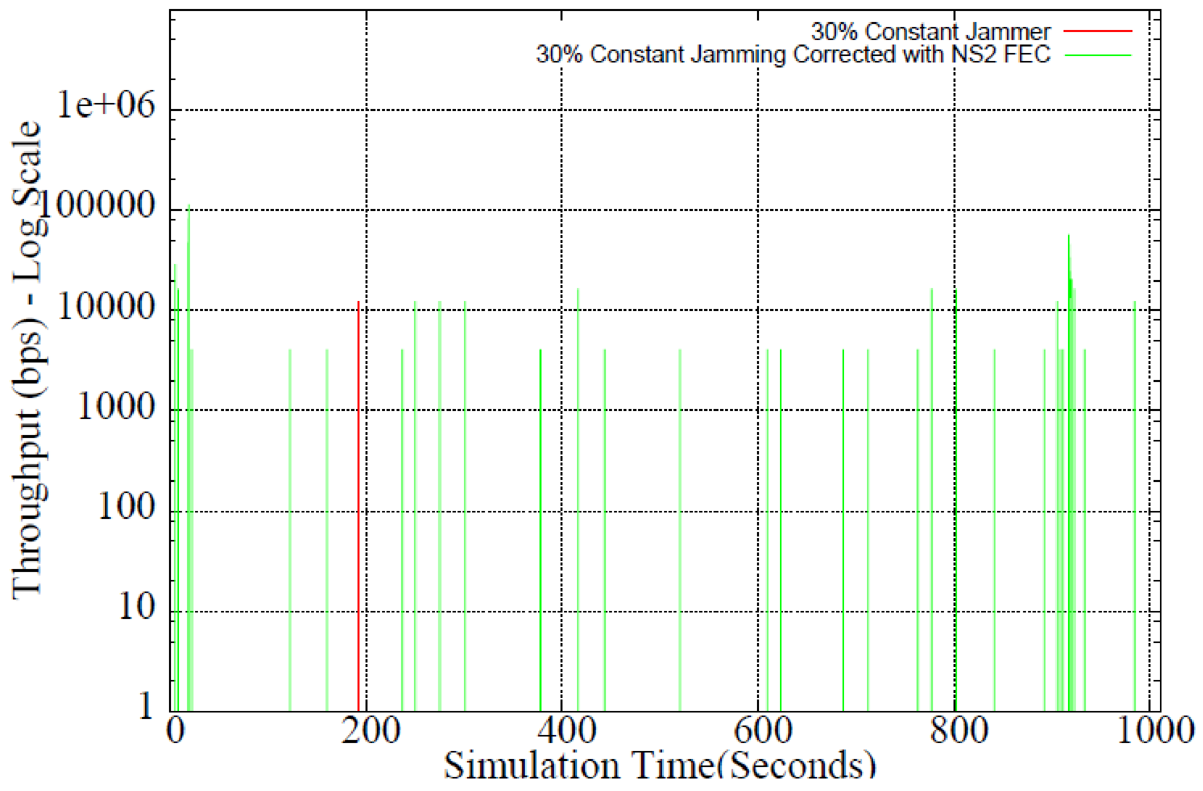


Figure 5.13: **NS2 FEC Code Performance against 30% Constant Jamming**

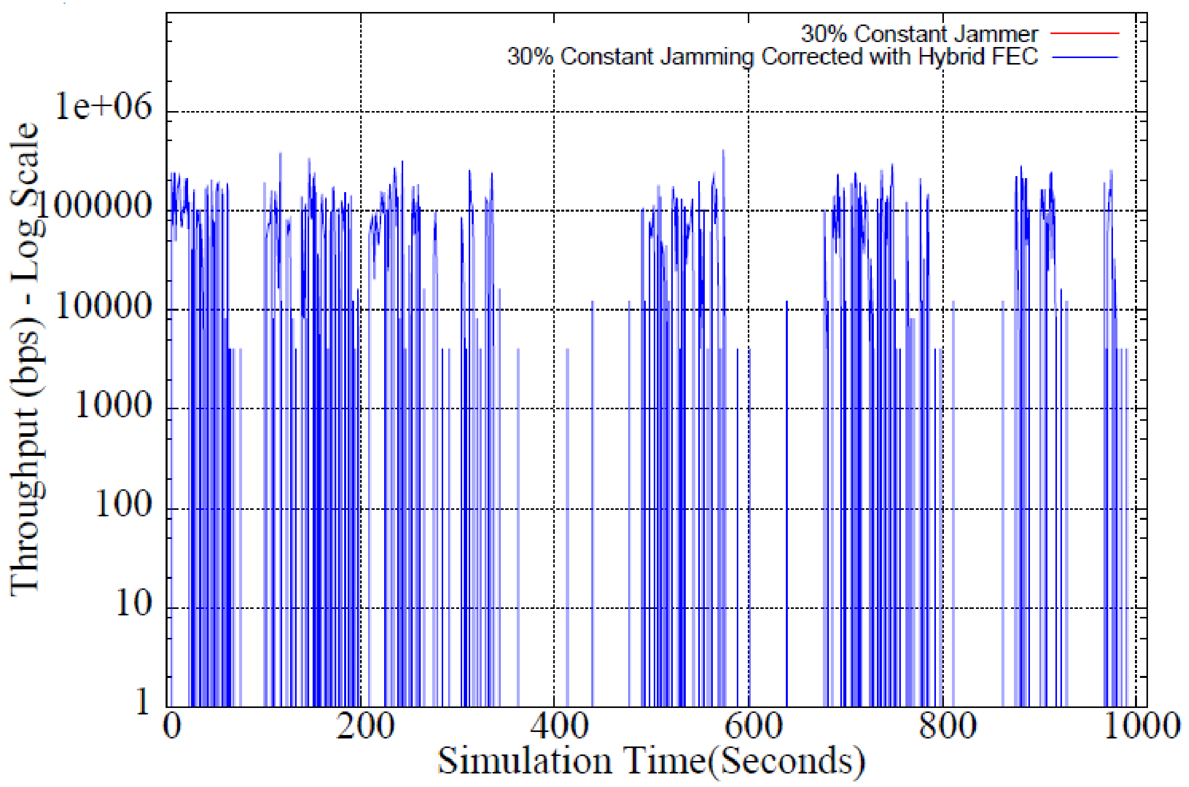


Figure 5.14: Hybrid FEC Code Performance against 30% Constant Jamming

Figure 5.15 presents the worst case scenario of the constant jammer, i.e., when the rate of jamming is 100%. This is a total jamming scenario that resulted into a zero throughput of the CRN without the implementation of the hybrid FEC code. The figure shows that the ordinary FEC code was not able to mitigate any instance of the 100% constant jamming and as a result, no bit of data was delivered at the receivers by the CRN. In the case of our hybrid FEC code, it is noticeable from the figure that the hybrid FEC code was able to deliver significant spurts of bits of data during the simulation time but fails intermittently as it tries to implement the recovery block procedure to cope with the high rate of data corruption or manipulation of the constant jammers.

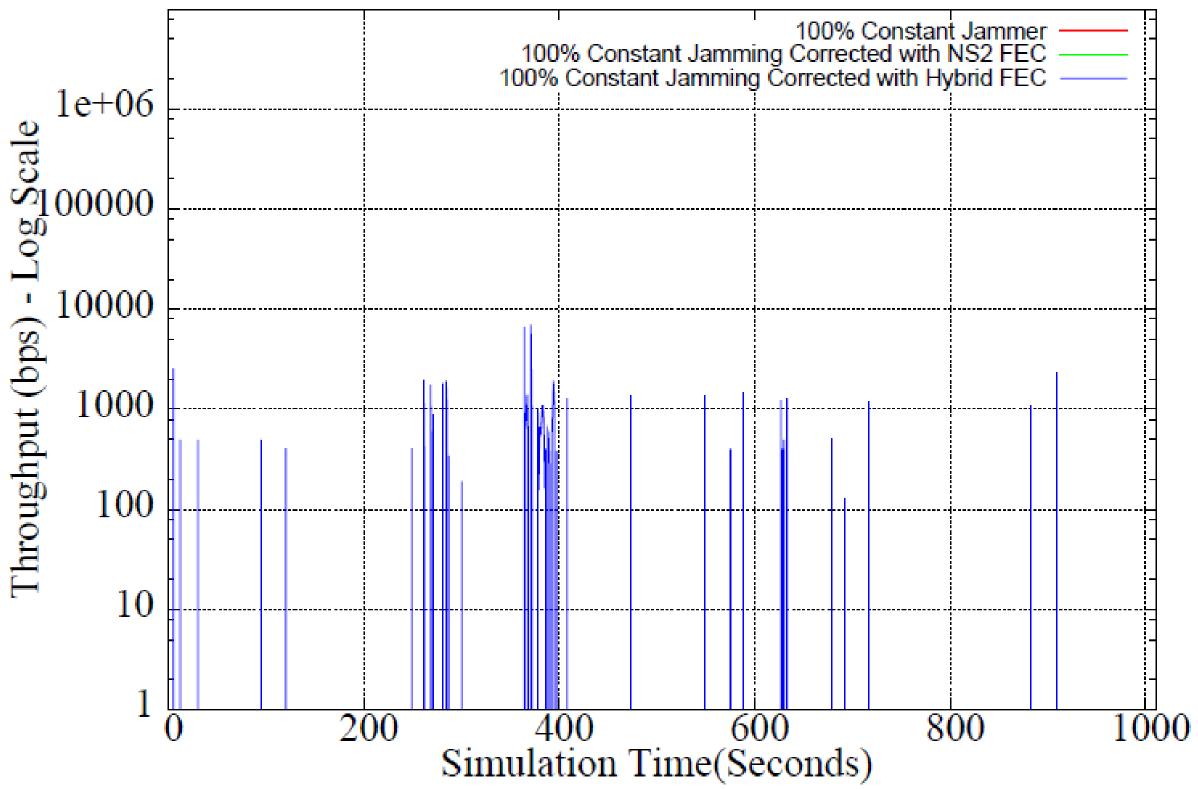


Figure 5.15: **Hybrid FEC Code Performance against 100% Constant Jamming**

In summary, we observed that the hybrid FEC code largely out-performs the ordinary FEC code as it successfully handles most incidences of omissive and transmissive faults. It is also noticeable that even when the rate of jamming was 100%, characterized by increased rate of data corruption and manipulation, the hybrid FEC code was still able to recover significant amount of data packets from the jamming effect as they were successfully delivered to the destinations. The hybrid FEC code out-performs the NS-2 FEC because of the following reasons:

1. The efficient design of the hybrid FEC code algorithm, e.g., the recovery block implementation, the concatenation of the SHA2 components with Raptor code.
2. The careful selection of simulation parameters that lead to efficient performance of the hybrid FEC code.
3. The careful choice and implementation of non-NS2 parameters like the length of input symbol *l*, number of repair symbol ε, and the number of input symbols *k* in a block.

Packet Delivery Ratio

Figure 5.16 shows the average PDR for the total number of replications for each experiment (jamming rate) comparing the ordinary FEC code with the hybrid FEC code. The superior performance of the hybrid FEC code over the ordinary FEC code is immediately noticeable. The PDR of the ordinary FEC code drops significantly as the jamming rate increases. At 50% jamming, the PDR of the ordinary FEC is about 45%, i.e., below average. At the jamming rate of 100%, the PDR of ordinary FEC drops to zero, meaning that the ordinary FEC code breaks down completely. Unlike the ordinary FEC, the hybrid FEC maintains very high PDR of about 70%, even when the jamming rate increases to 90%. It is also noticeable from the figure that at a worst case jamming rate of 100%, i.e. total jamming, the hybrid FEC code still maintains a high PDR of about 65%. These high PDRs recorded by the hybrid FEC code mean that the hybrid code is robust against DoS activities of constant jammers as the hybrid FEC code was able to recover and deliver a high percentage of data that was successfully sent from the sender.

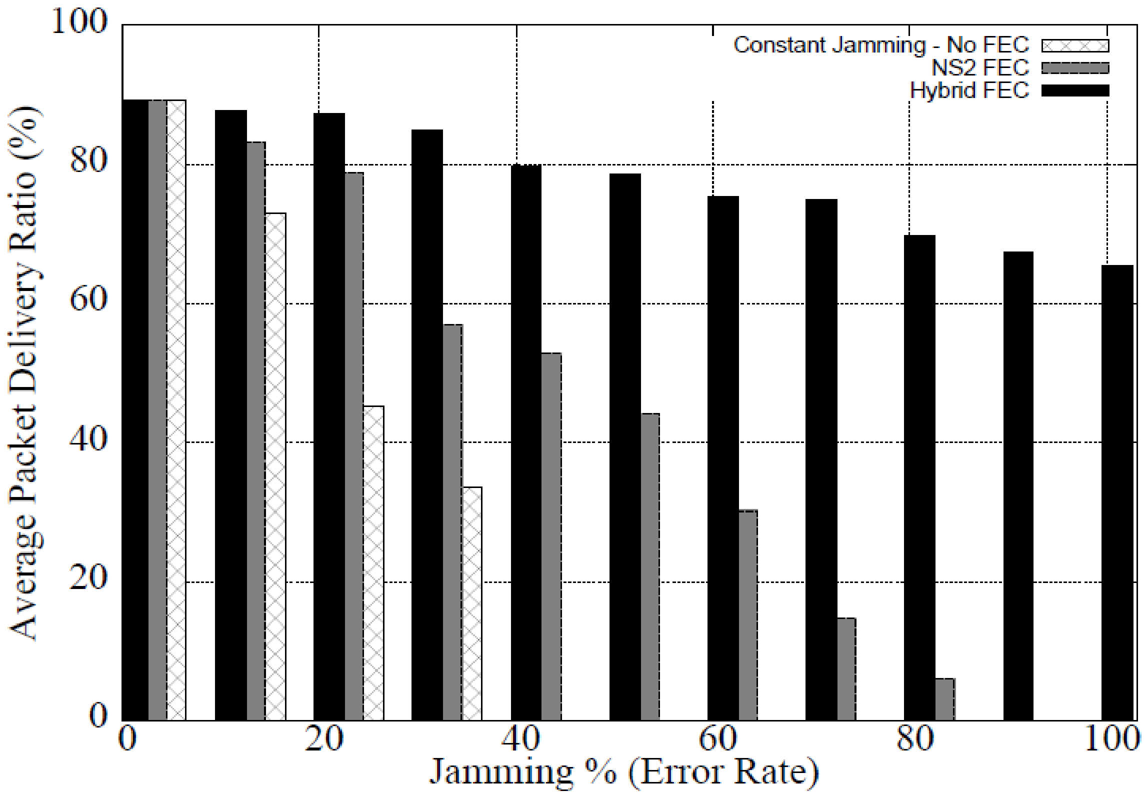


Figure 5.16: **PDR measurement of Hybrid FEC Code Performance against Constant Jamming**

Packet Loss Ratio

Figure 5.17 is the plot of average packet loss ratio (PLR) against jamming percentage. The average represents the average PLR for the total number of replications for the same experiment for each rate of jamming. It is observable from the figure that the hybrid FEC code maintains a very low PLR for the different jamming rate of the constant jammers. On the other hand, the ordinary FEC code though has low PLR up to a jamming rate of 30%, but beyond this the code breaks down significantly as the PLR increases to close to 70%, even when the jamming rate is just about 60%. Since low PLR implies a high recovery rate, it follows that the hybrid FEC code maintains high recovery rate even when the rate of jamming is 100%.

The high recovery rate of the hybrid FEC code recorded against that of the ordinary FEC code implies that the algorithms of the encoding, decoding, and SHA-2 hash computation processes through the action of the recovery block for hybrid FEC code are computationally more efficient and robust against the different rate of jamming than that of the ordinary FEC code.

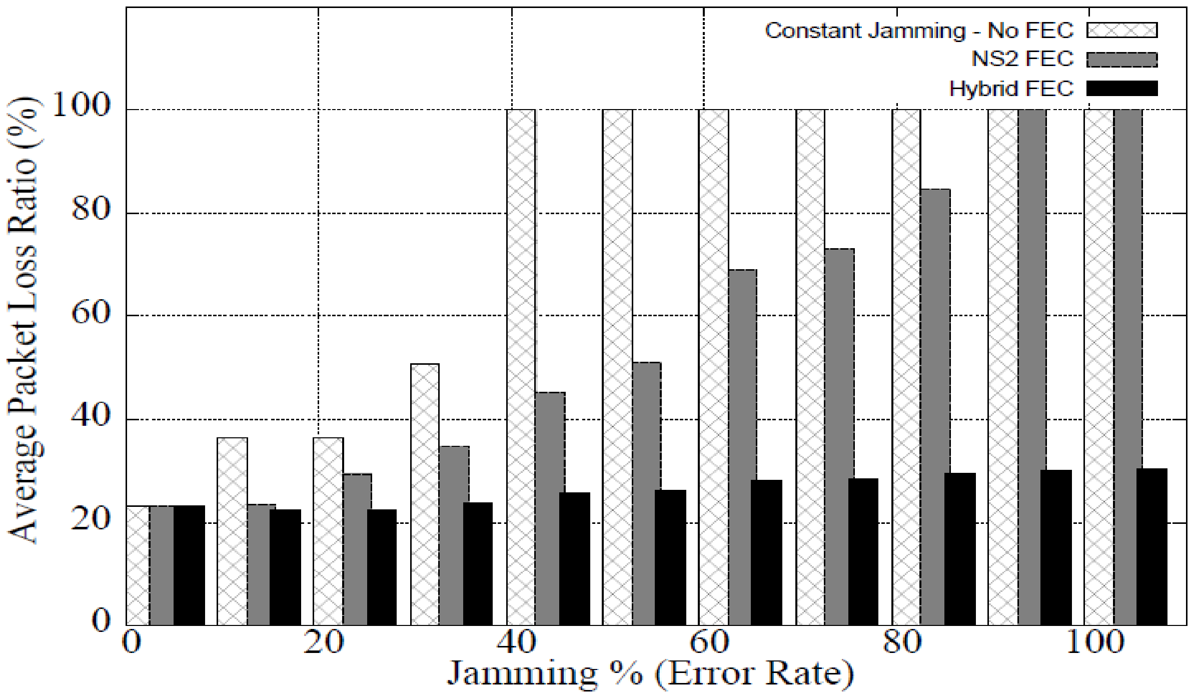


Figure 5.17: **PLR measurement of Hybrid FEC Code Performance against Constant Jamming**

Summary of Statistical Analysis for Constant Jamming in a Combined CSS CRN

Tables 5.4 to 5.6 present the summary of the statistical analysis of the result of our simulation for constant jamming in a combined CSS CR network. We only present the 10% jamming representing the fair case of constant jamming, the 30% jamming representing the moderate case of constant jamming and the 100% representing the worst case of constant jamming.

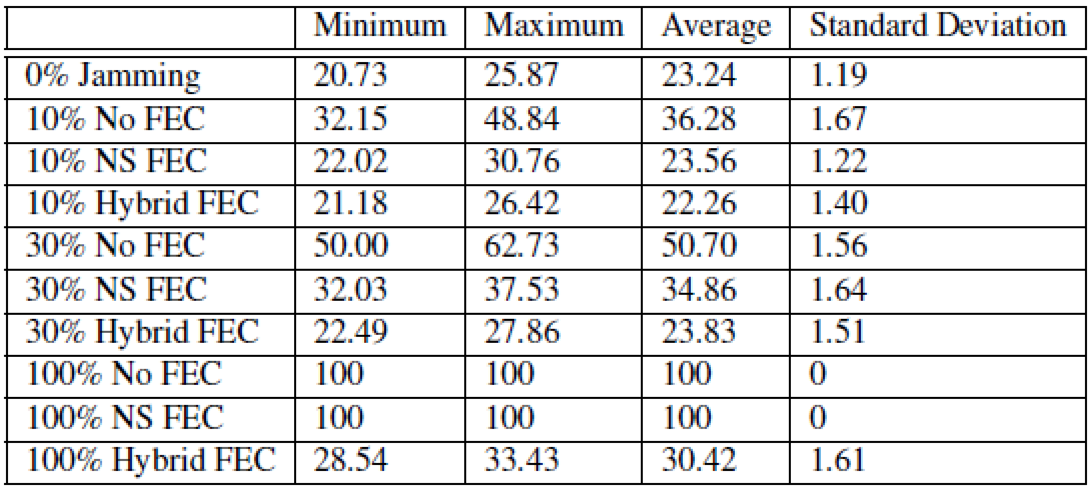
Table 5.4: **Summary of Statistical Analysis for Throughput – Constant Jamming**

	Minimum	Maximum	Average	Standard Deviation
0% Jamming	196027.58	196947.33	196461.38	308.36
10% No FEC	1602.53	1694.34	1639.74	28.13
10% NS FEC	6708.1	6970.43	6837	83.09
10% Hybrid FEC	66444.29	66654.85	66521.05	64.98
30% No FEC	129.74	135.65	131.94	1.77
30% NS FEC	750. 61	789.66	764.53	10.7
30% Hybrid FEC	8908.8	9090.88	8997.49	66.13
100% No FEC	0	0	0	0
100% NS FEC	0	0	0	0
100% Hybrid FEC	4895.12	4999.28	4974.69	30.94

Table 5.5: **Summary of Statistical Analysis for Average PDR – Constant Jamming**

	Minimum	Maximum	Average	Standard Deviation
0% Jamming	88.45	89.76	89.12	0.34
10% No FEC	51.28	81.84	72.96	7.39
10% NS FEC	58.15	85.75	83.11	3.57
10% Hybrid FEC	84.36	88.29	87.62	1.11
30% No FEC	23.33	48.39	33.63	6.76
30% NS FEC	51.65	62.12	57.03	3.42
30% Hybrid FEC	83.74	85.84	84.85	0.67
100% No FEC	0	0	0	0
100% NS FEC	0	0	0	0
100% Hybrid FEC	61.54	71.63	65.30	2.73

***Table 5.6:* Summary of Statistical Analysis for Average PLR – Constant Jamming**



	Minimum	Maximum	Average	Standard Deviation
0% Jamming	20.73	25.87	23.24	1.19
10% No FEC	32.15	48.84	36.28	1.67
10% NS FEC	22.02	30.76	23.56	1.22
10% Hybrid FEC	21.18	26.42	22.26	1.40
30% No FEC	50.00	62.73	50.70	1.56
30% NS FEC	32.03	37.53	34.86	1.64
30% Hybrid FEC	22.49	27.86	23.83	1.51
100% No FEC	100	100	100	0
100% NS FEC	100	100	100	0
100% Hybrid FEC	28.54	33.43	30.42	1.61

5.9.2 Random Jamming

Throughput

Figures 5.18 and 5.19 depict the random jamming scenario of a CRN experiencing a jamming rate of 10%. It is noticeable from Figure 5.19 that the hybrid FEC code was able to mitigate every jamming effect caused by the random jammer, but the ordinary FEC code in Figure 5.18 fails in some intervals characterized by value faults caused by the random jammers. The throughputs of both the ordinary FEC and the hybrid FEC codes were very high because the random jammer alternates between sleeping and waking, and as such the impact of the jamming was contained by both ordinary and hybrid FEC code. Figures 5.20 and 5.21 depict the random jamming scenario when the rate of jamming was 50%. It is noticeable from the figure that the hybrid FEC code was able to mitigate most jamming effect caused by the random jammer but fails in some intervals due to the actions of the recovery block discussed in Sub-subsection 5.9.1. In the case of ordinary FEC code in Figures 5.20, the throughput of the network dropped significantly as the code fails at every instance of transmissive value fault and whenever there is a decoding failure of the FEC code. This trend is noticed at jamming rate between 50% - 90%. Beyond 90% jamming rate, the throughput of ordinary FEC drops to zero while that of hybrid FEC remains moderate even when the rate of jamming is 100%, as seen in Figure 5.22. These figures show that the hybrid FEC code out-performs the ordinary FEC code as moderately high throughput was realized by the hybrid FEC code while ordinary FEC code's throughput drops to zero when a CRN is subjected to a jamming rate of 100%.

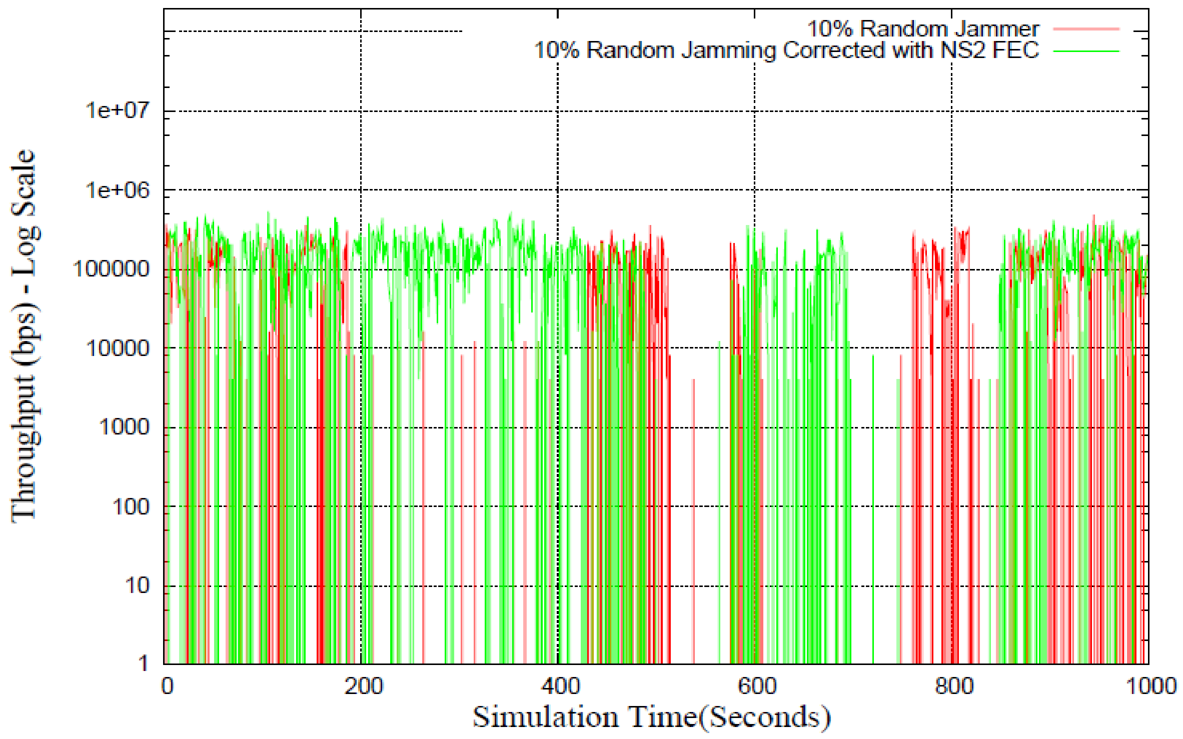


Figure 5.18: **FEC Code Performance against 10% Random Jamming**

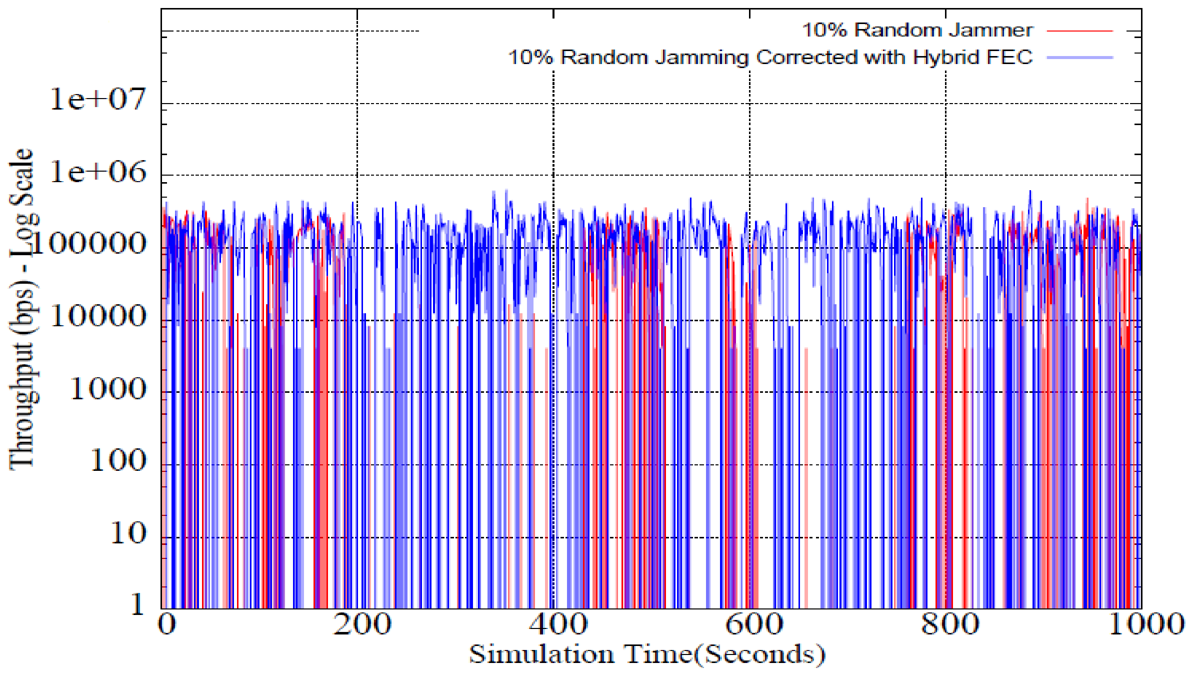


Figure 5.19: **Hybrid FEC Code Performance against 10% Random Jamming**

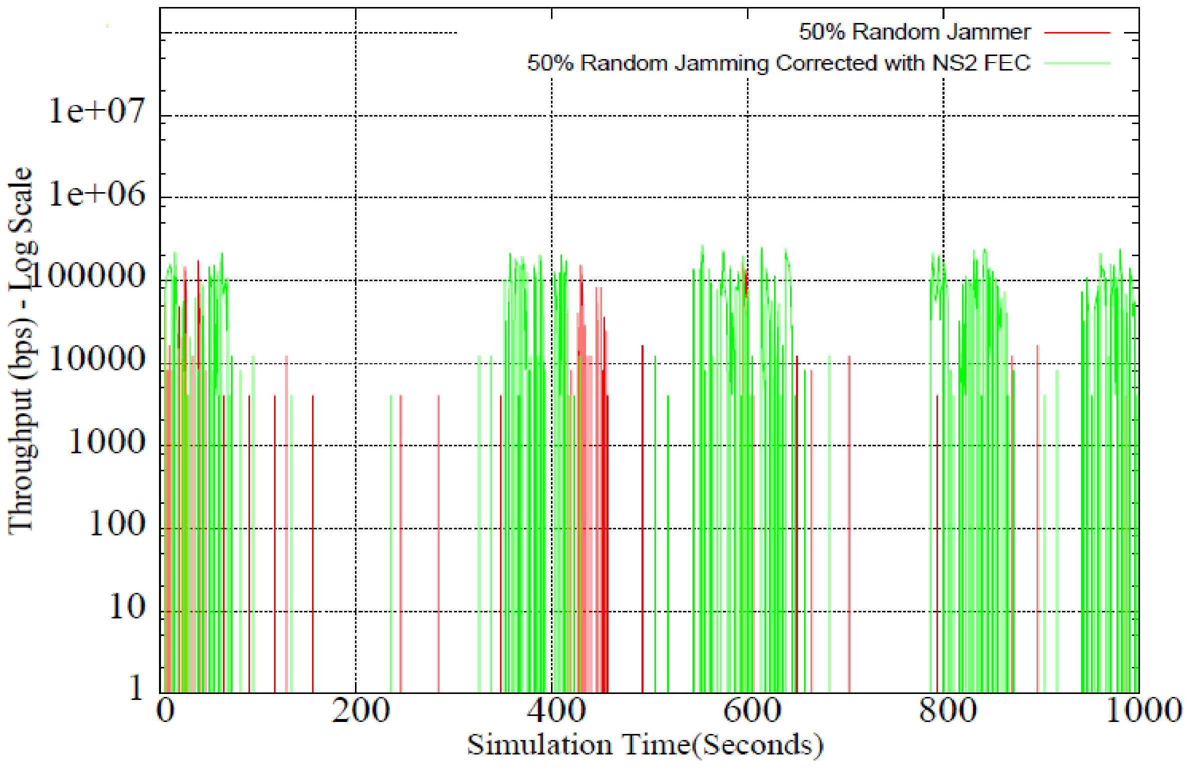


Figure 5.20: **FEC Code Performance against 50% Random Jamming**

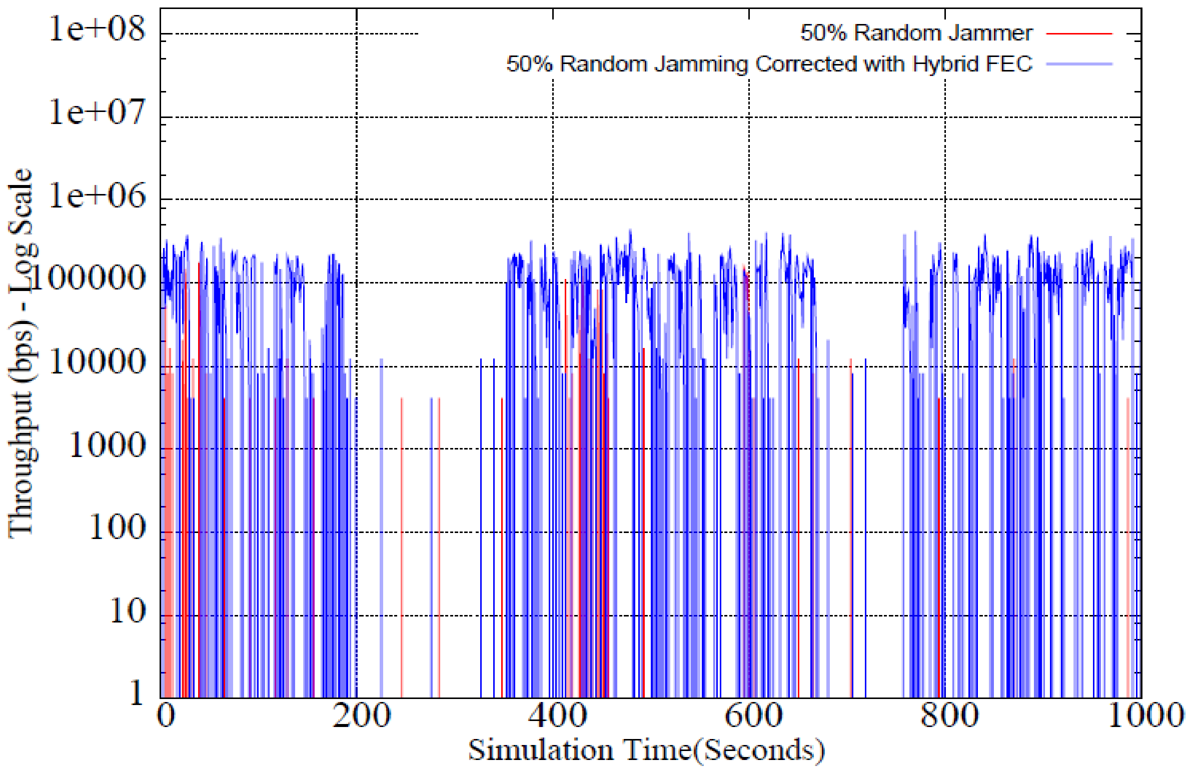


Figure 5.21: **Hybrid FEC Code Performance against 50% Random Jamming**

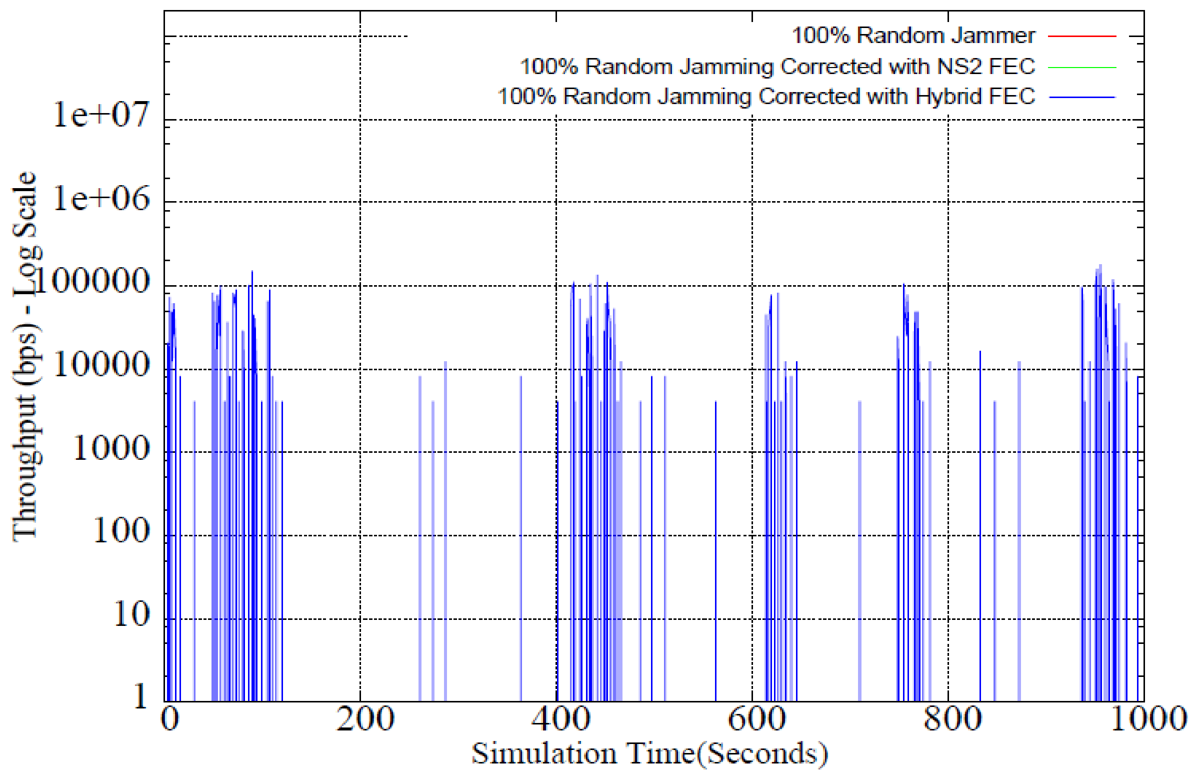


Figure 5.22: **Hybrid FEC Code Performance against 100% Random Jamming**

Packet Delivery Ratio

Figure 5.23 shows the PDR of a CRN implementing ordinary FEC compared with that of the hybrid FEC code. It is noticeable that the PDR of the ordinary FEC code continues to drop gradually as the rate of jamming increases because of the sleeping and waking schedule of the random jammer. At 90% jamming, the PDR of the ordinary FEC is about 15%. Beyond the jamming rate of 90%, the PDR of ordinary FEC drops to zero. The hybrid FEC maintains very high PDR of more than 70% even when the jamming rate increases to 100%.

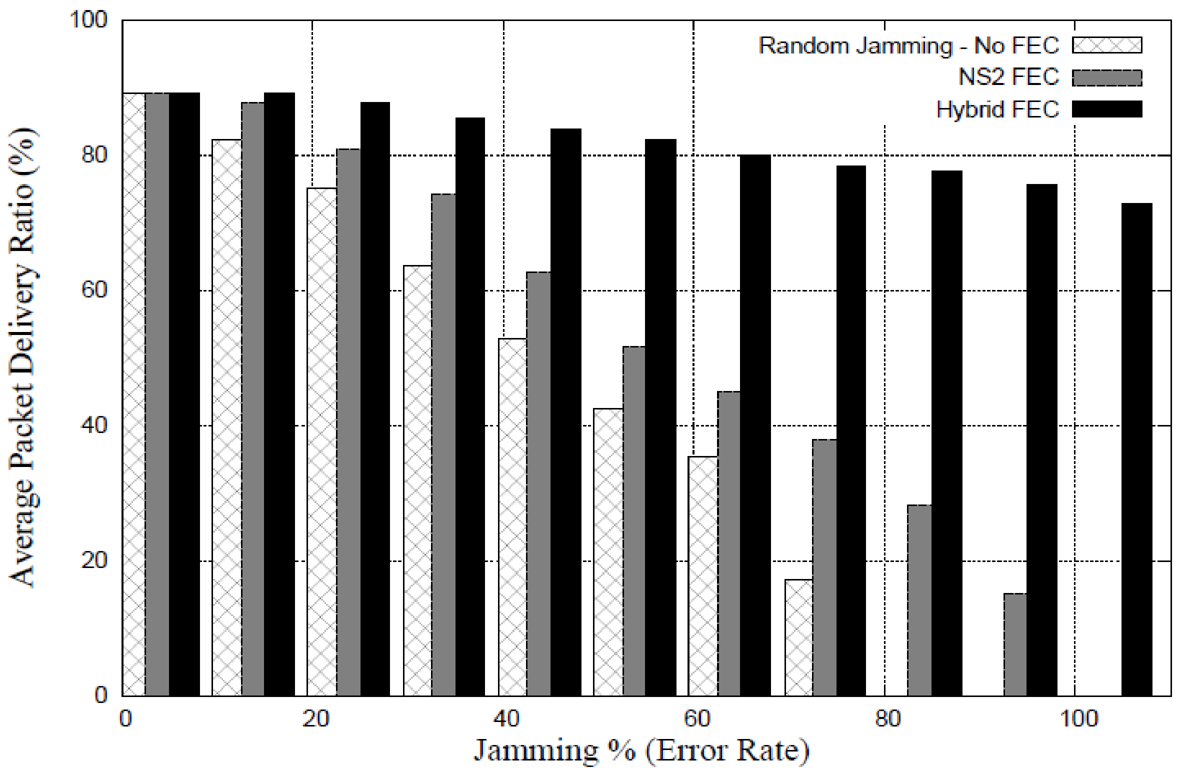


Figure 5.23: **PDR measurement of Hybrid FEC Code Performance against Random Jamming**

Packet Loss Ratio

Figure 5.24 shows average packet loss ratio (PLR) of a CRN under different random jamming attack rates. It is observable from the figure that the hybrid FEC code consistently maintains very low PLR for all the jamming rate of the random jammers. Similarly, the ordinary FEC code has low PLR up to a jamming rate of about 50%, but beyond this point the code breaks down significantly as the PLR increases to about to 65%, even when the jamming rate is just about 60%. This demonstrates that the hybrid FEC code maintains high recovery rates even when the rate of jamming is as high as 100%, showing that the algorithms of the hybrid FEC code are computationally efficient and robust against any level of random jamming and out-performed the ordinary FEC code.

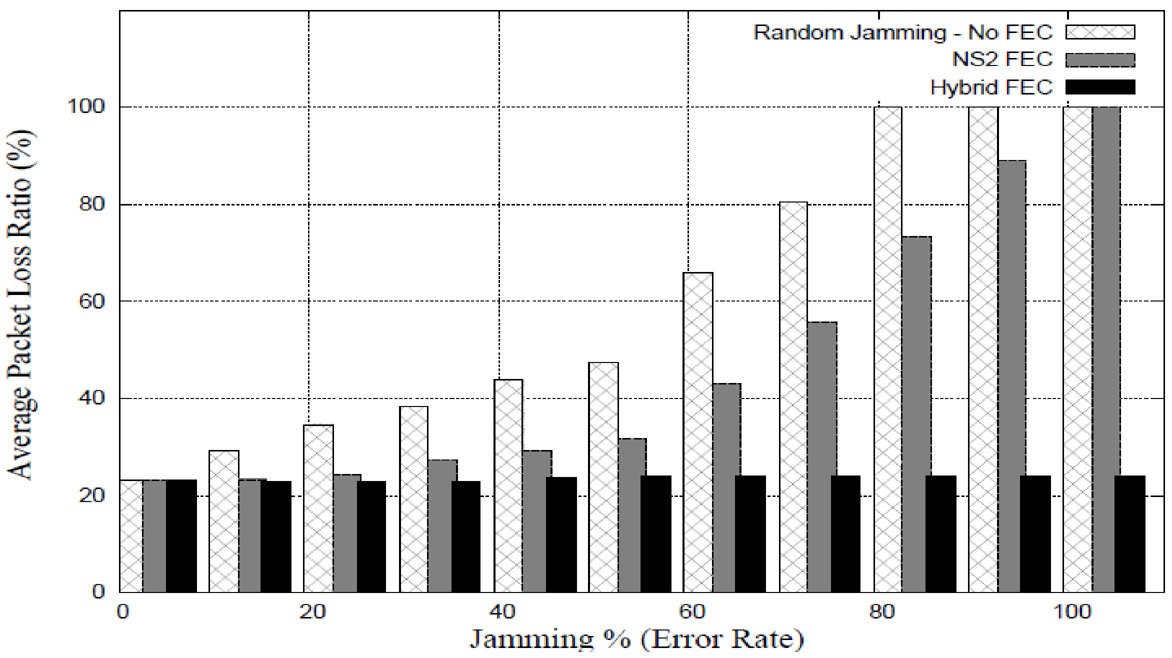


Figure 5.24: **PLR measurement of Hybrid FEC Code Performance against Random Jamming**

Summary of Statistical Analysis for Random Jamming in a Combined CSS CRN

Tables 5.7 to 5.9 present the summary of the statistical analysis of the result of our simulation for random jamming in a CR network. We only present the 10% jamming representing the fair case of random jamming, the 50% jamming representing the moderate case of random jamming, and the 100% representing the worst case of random jamming.

Table 5.7: **Summary of Statistical Analysis for Average Throughput – Random Jamming**

	Minimum	Maximum	Average	Standard Deviation
0% Jamming	196027.58	196947.33	196461.38	308.36
10% No FEC	42792.45	47935.49	45859.84	46.26
10% NS FEC	90241.73	94458.43	93338.61	78.36
10% Hybrid FEC	179726.72	199837.89	189782.86	40.56
50% No FEC	2075.63	2416.38	2297.13	5.63
50% NS FEC	17403.84	23899.39	21072.27	51.44
50% Hybrid FEC	61677.7	64524.48	63027.3	93.65
100% No FEC	0	0	0	0
100% NS FEC	0	0	0	0
100% Hybrid FEC	4989.63	5619.58	5403.36	9.94

Table 5.8: **Summary of Statistical Analysis for Average PDR – Random Jamming**

	Minimum	Maximum	Average	Standard Deviation
0% Jamming	88.45	89.76	89.12	0.34
10% No FEC	85.6	86.63	86.3	0.37
10% NS FEC	87.47	88.02	87.67	0.2
10% Hybrid FEC	88.78	89.6	89.19	0.3
50% No FEC	58.36	66.85	63.47	1.79
50% NS FEC	78.2	81.11	79.72	0.95
50% Hybrid FEC	83.31	84.96	84.33	0.51
100% No FEC	0	0	0	0
100% NS FEC	0	0	0	0
100% Hybrid FEC	69.04	73.91	72.69	1.51

Table 5.9: **Summary of Statistical Analysis for Average PLR – Random Jamming**

	Minimum	Maximum	Average	Standard Deviation
0% Jamming	20.73	25.87	23.24	1.19
10% No FEC	27.91	32.29.84	29.14	0.87
10% NS FEC	20.59	24.01	23.29	1.14
10% Hybrid FEC	19.74	23.44	22.92	1.33
50% No FEC	45.06	49.76	47.46	0.24
50% NS FEC	29.56	33.34	31.64	1.08
50% Hybrid FEC	22.48	24.73	23.89	0.95
100% No FEC	100	100	100	0
100% NS FEC	100	100	100	0
100% Hybrid FEC	23.62	25.06	24.04	0.45

5.9.3 Deceptive Jamming

Throughput

Similar to the case of random jamming, Figures 5.25 and 5.26 show throughput associated with a deceptive jamming rate of 10% (fair case). The ordinary FEC code could not effectively handle the jamming effect from the beginning of the simulation to about 400 seconds. Beyond this point, though there were intervals of zero bit of data delivery, the ordinary FEC was able to realize high throughput. The case of hybrid FEC code in Figure 5.26 was different as the hybrid code was able to detect the presence of deceptive jammers within the first 200 seconds of the simulation. The outstanding performance of hybrid FEC code is noticeable in the figure that beyond the simulation time of 200 seconds, the hybrid FEC code was able to mitigate the effect of the deceptive jammers though there were short intervals of zero throughput caused by the delays introduced by the action of the recovery block.

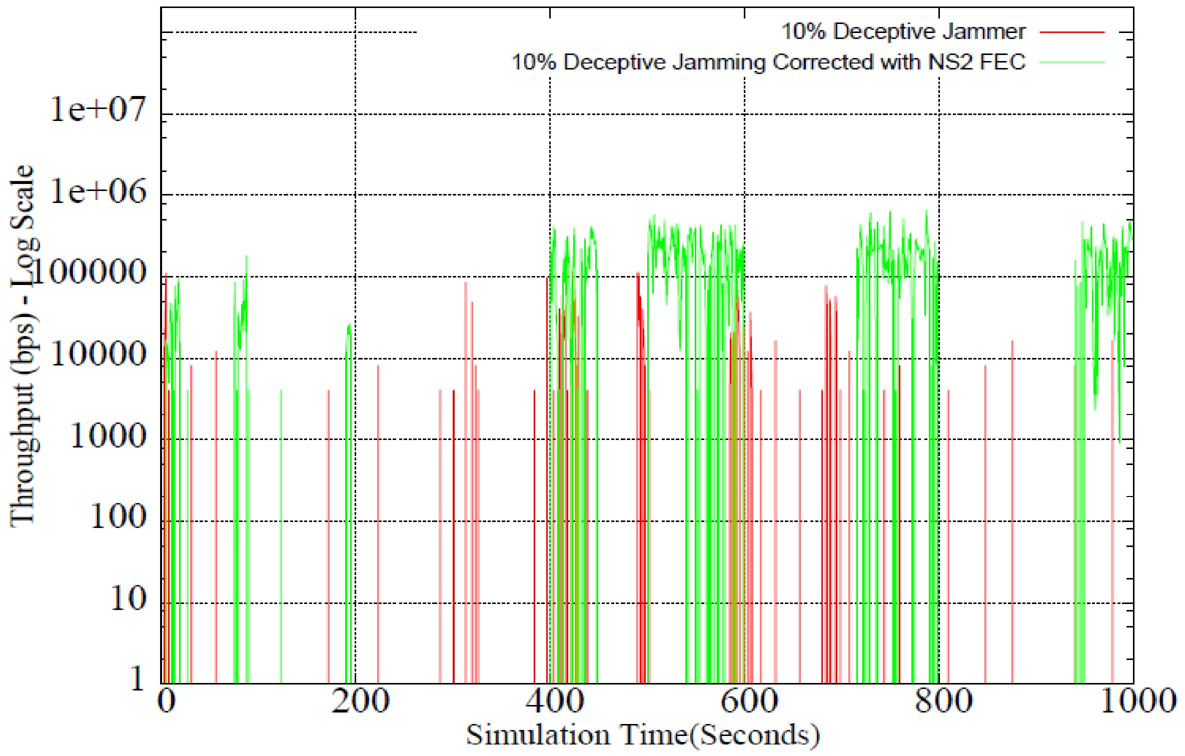


Figure 5.25: FEC Code Performance against 10% Deceptive Jamming

Figures 5.27 and 5.28 depict the deceptive jamming scenario when the rate of jamming was 40% (moderate case). It is noticeable from Figure 5.27 that the ordinary FEC code was only able to mitigate jamming caused by deceptive jammers in some short intervals, while most of the simulation time experienced zero throughput. This means that the throughput of the network dropped drastically as the ordinary FEC code fails at every instance of transmissive value fault and whenever there is a decoding failure of the FEC code. The hybrid FEC code, Figure 5.28, on the other hand exhibit superior performance over the ordinary FEC code as it was able to mitigate the jamming effect more than half of the total simulation period. At the jamming rate of 100% (worst case), while the hybrid FEC code maintains moderately high throughput in Figure 5.29, the throughput of ordinary FEC drops to zero.

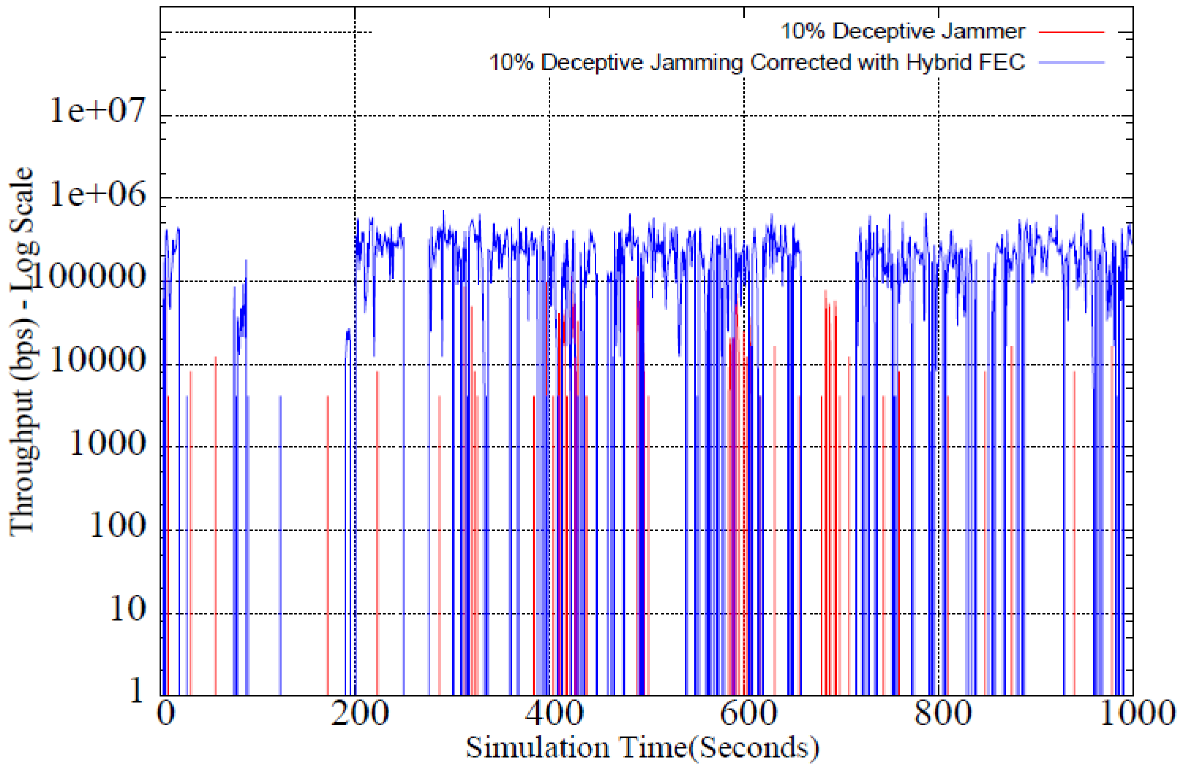


Figure 5.26: **Hybrid FEC Code Performance against 10% Deceptive Jamming**

Packet Delivery Ratio

Figure 5.30 shows the PDR comparison of a CRN using ordinary FEC and that of the hybrid FEC code in a deceptive jamming scenario. It can be seen that the PDR of the ordinary FEC code drops rapidly to about 38% when the jamming rate increases to 60%. This trend continues until the jamming rate of 100% when the PDR of the ordinary FEC drops to zero. Unlike the ordinary FEC, the hybrid FEC maintains a high PDR of about 65% even when the jamming rate is 100%.

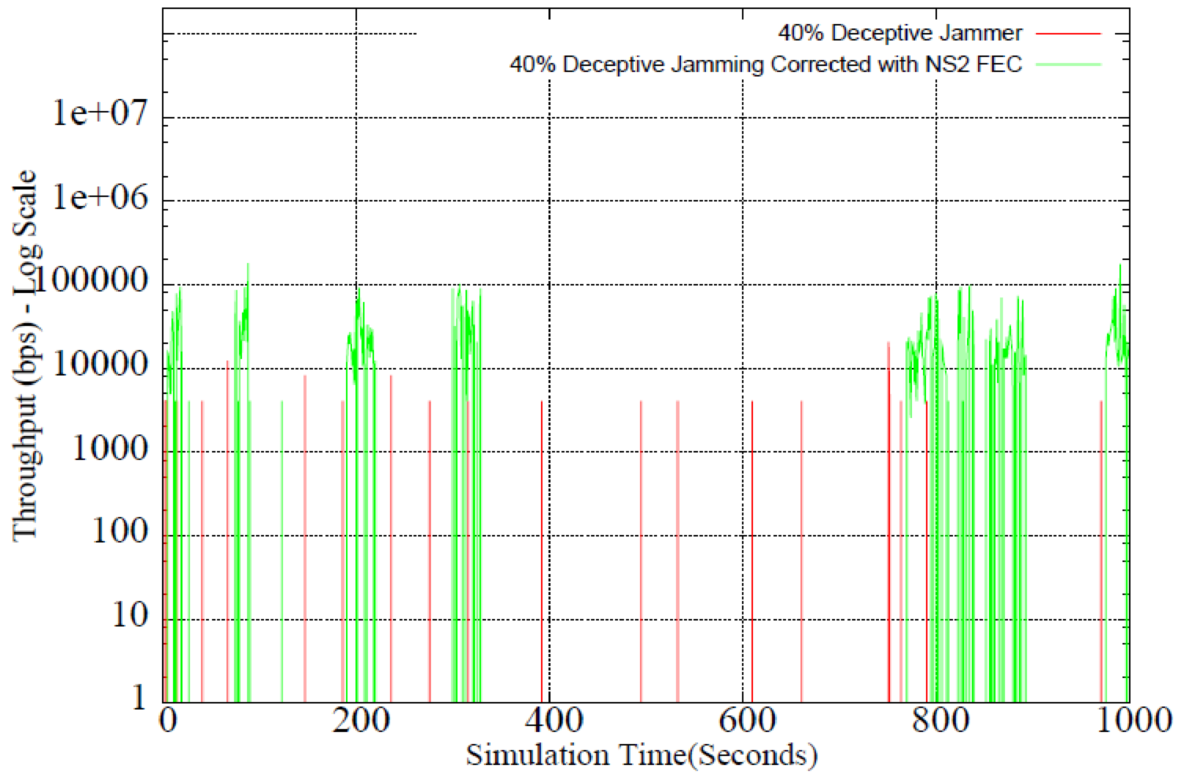


Figure 5.27: **FEC Code Performance against 40% Deceptive Jamming**

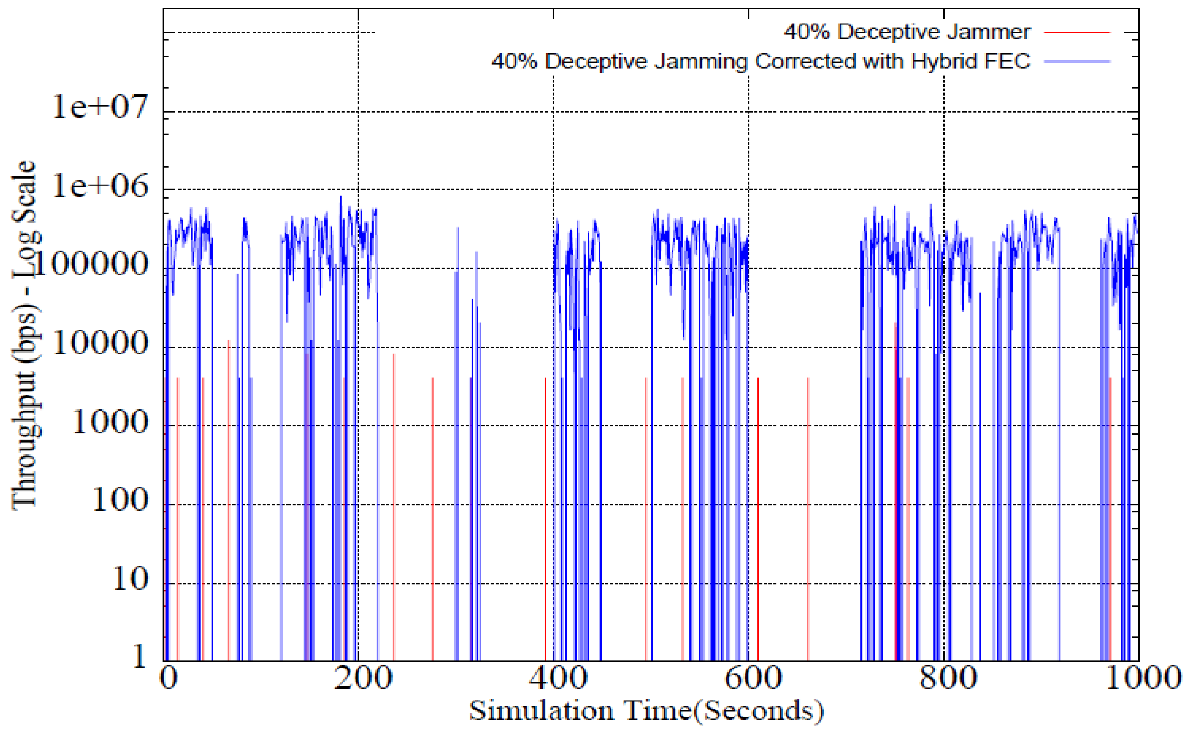


Figure 5.28: **Hybrid FEC Code Performance against 40% Deceptive Jamming**

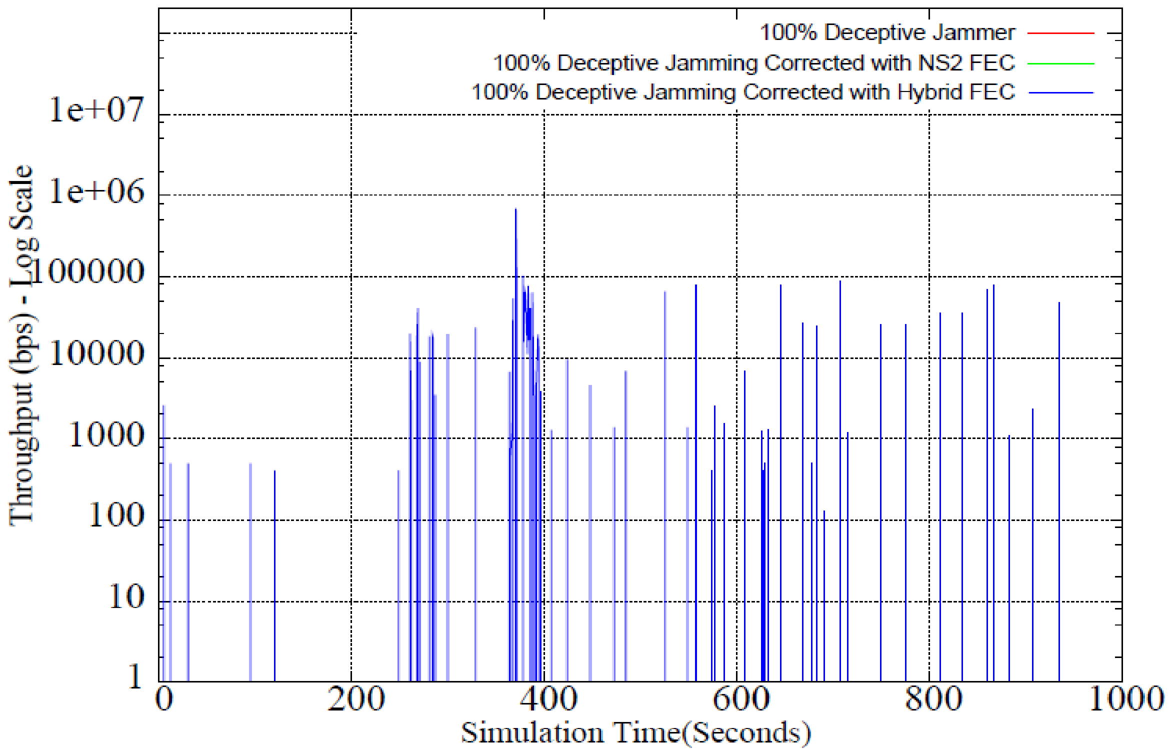


Figure 5.29: **Hybrid FEC Code Performance against 100% Deceptive Jamming**

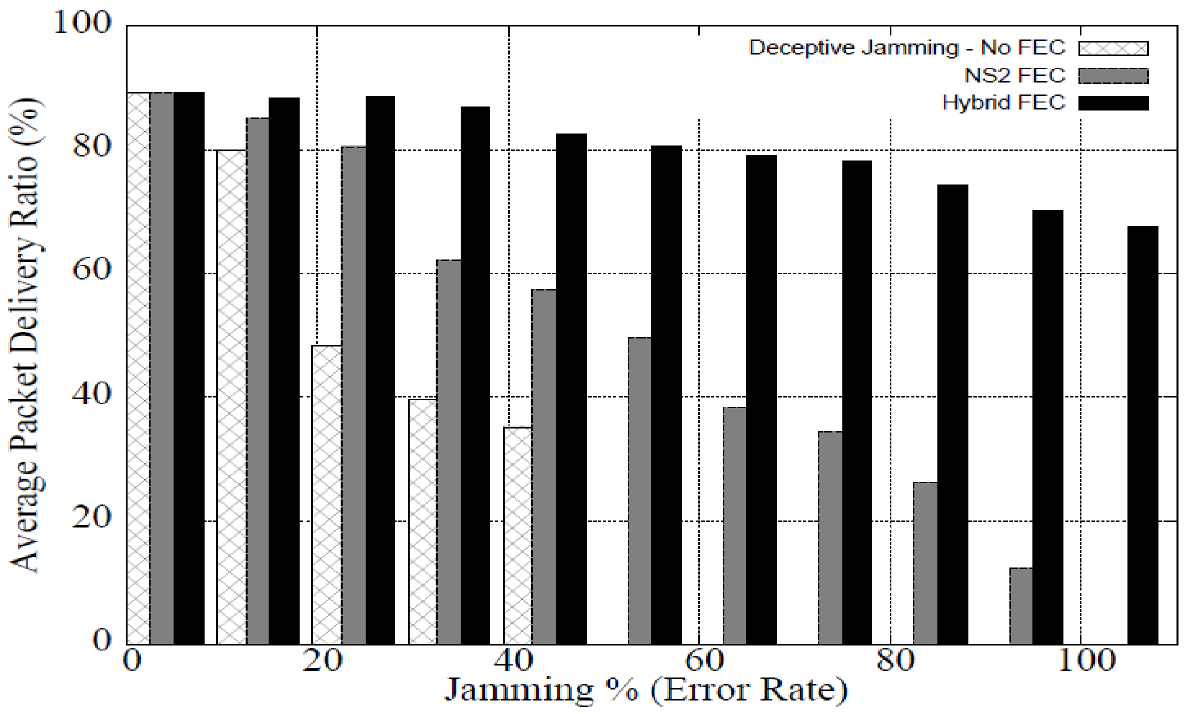


Figure 5.30: **PDR measurement of Hybrid FEC Code Performance against Deceptive Jamming**

Packet Loss Ratio

Figure 5.31 depicts the PLR plot of ordinary FEC code against the hybrid FEC code of a CRN under deceptive jamming attack. It is noticeable from the figure that the PLR of the ordinary FEC code was initially low for low jamming rates, but jumped to about 60% for a jamming rate of about 60% and became worse beyond this point. The hybrid FEC code on the other hand consistently maintains very low PLR for all the jamming rate of the deceptive jammers. This means that the hybrid FEC code maintains high recovery rate even when the rate of jamming is as high as 100% showing that the hybrid FEC code performs more efficiently than the ordinary FEC code.

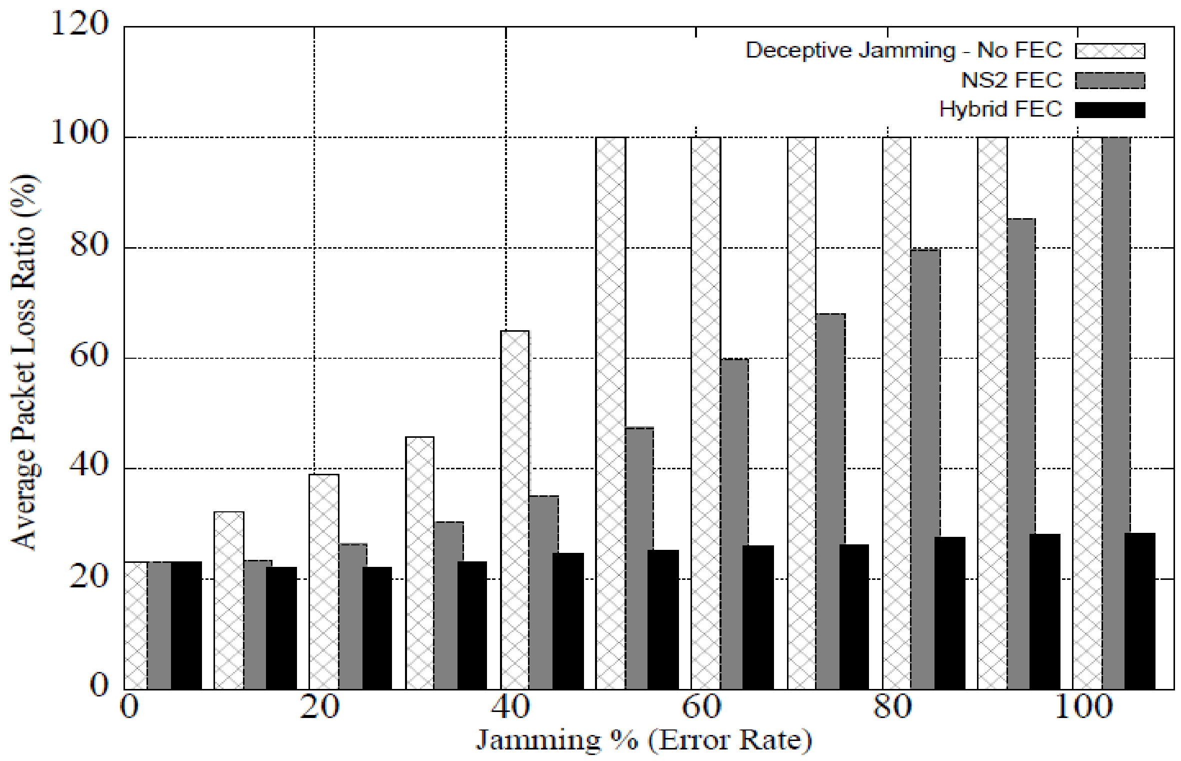


Figure 5.31: **PLR measurement of Hybrid FEC Code Performance against Deceptive Jamming**

5.9.4 Reactive Jamming

Throughput

Similar to the case of deceptive jamming, the fair case of reactive jamming depicted with Figures 5.32 and 5.33 show that the ordinary FEC code could only handle the jamming effect in some short intervals. The throughput achieved is just a little improvement over the case when no FEC code was used. This is because every incidence of a transmissive value fault results to zero throughput of the CRN. To the contrary, the hybrid FEC code in Figure 5.33 was able to handle the jamming effect for most intervals of the simulation. The short intervals of zero throughput were caused by the delays introduced by the action of the recovery block of the hybrid FEC code.

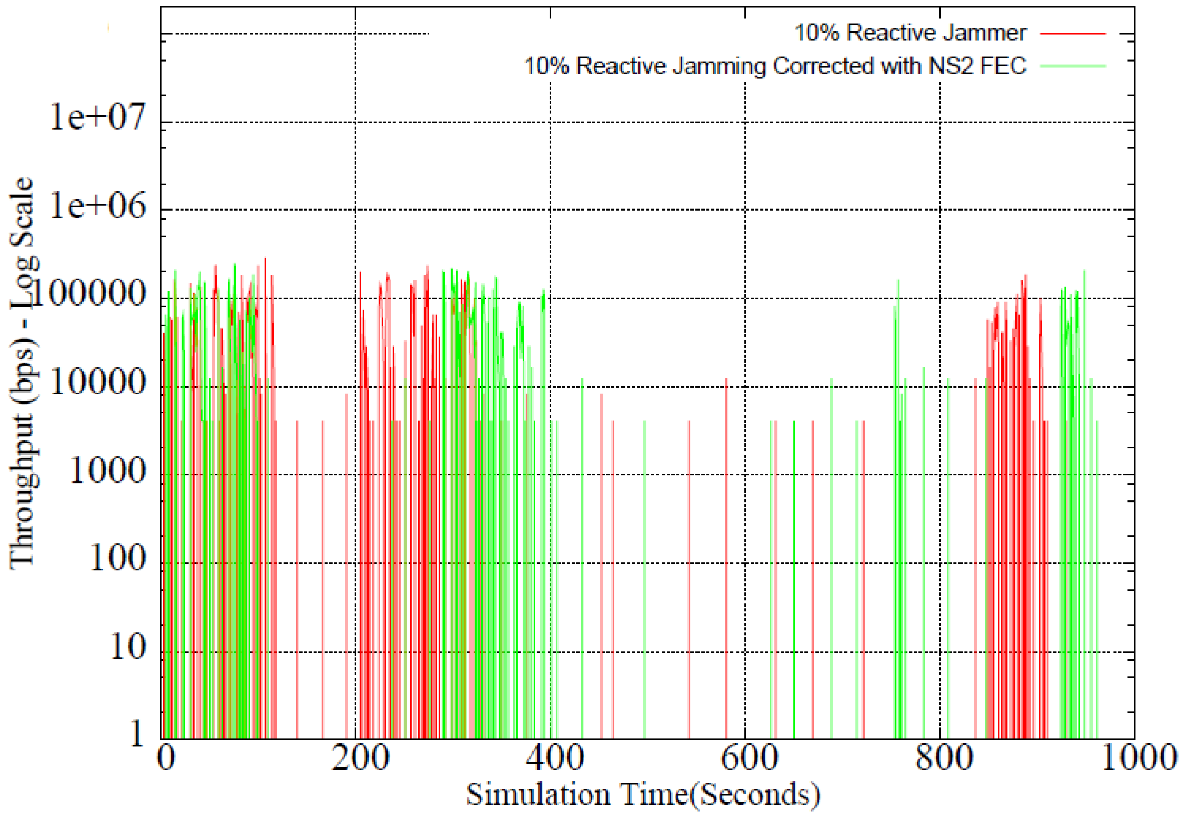


Figure 5.32: **FEC Code Performance against 10% Reactive Jamming**

Figures 5.34 and 5.35 present the reactive jamming scenario when the rate of jamming was 40% (moderate case). It can be observed from the figure that though the hybrid FEC code performed averagely, as it was able to mitigate jamming of the reactive jammers for about half of the simulation time. But the ordinary FEC code failed significantly as the throughput of the network dropped to about zero for most interval of the simulation time. This is due to the cumulative effect of incidences of transmissive value fault and the decoding failure of the FEC code. At the jamming rate of 100% (worst case), while the hybrid FEC code

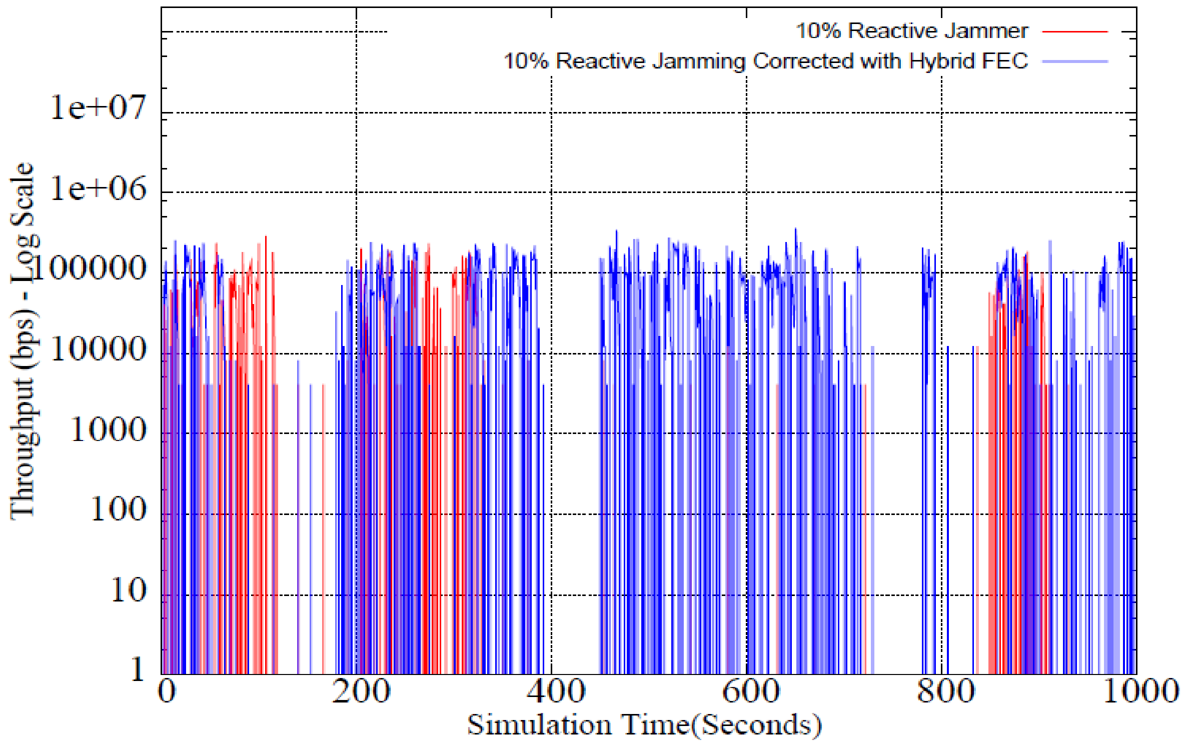


Figure 5.33: **Hybrid FEC Code Performance against 10% Reactive Jamming**

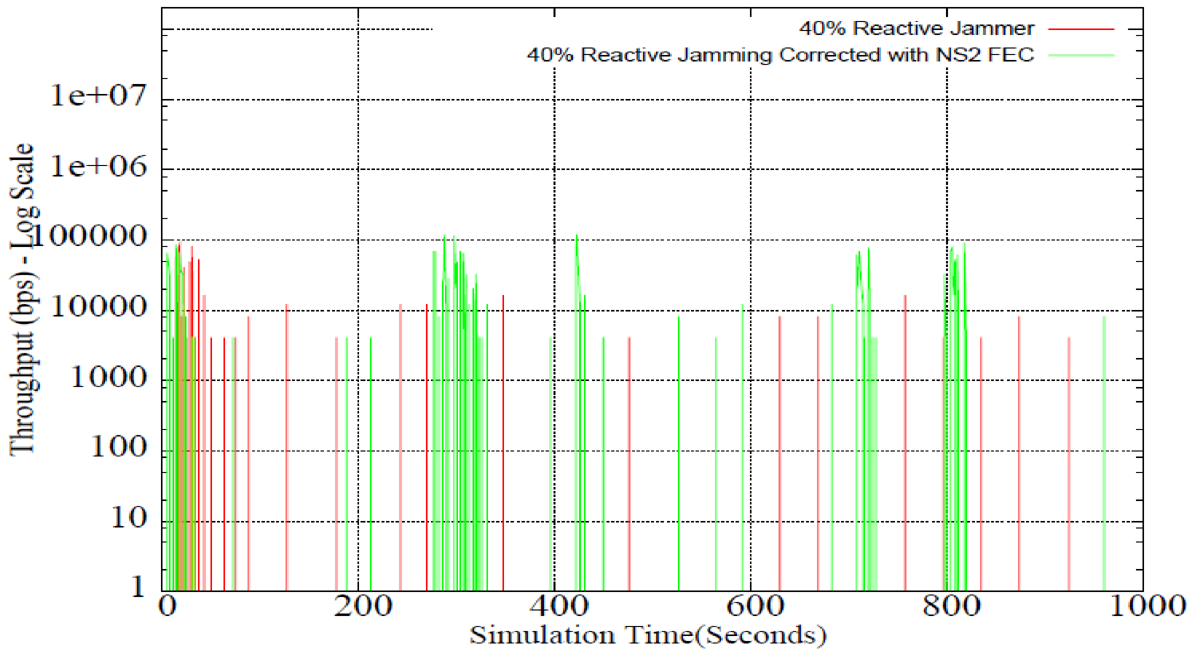


Figure 5.34: **FEC Code Performance against 40% Reactive Jamming**

maintains moderately high throughput, as shown in Figure 5.36, the throughput of ordinary FEC drops to close to zero. This further supports that the hybrid FEC code is more effective than the ordinary FEC code and that it is robust against all categories of jammers identified, even when the CRN is experiencing either partial or total jamming of the network.

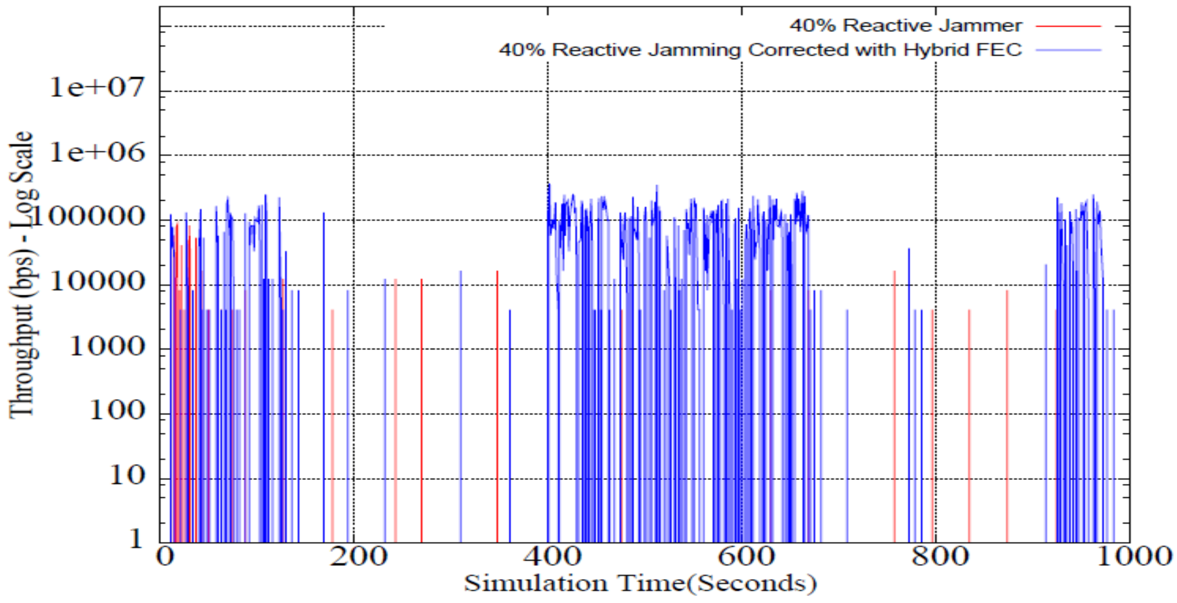


Figure 5.35: **Hybrid FEC Code Performance against 40% Reactive Jamming**

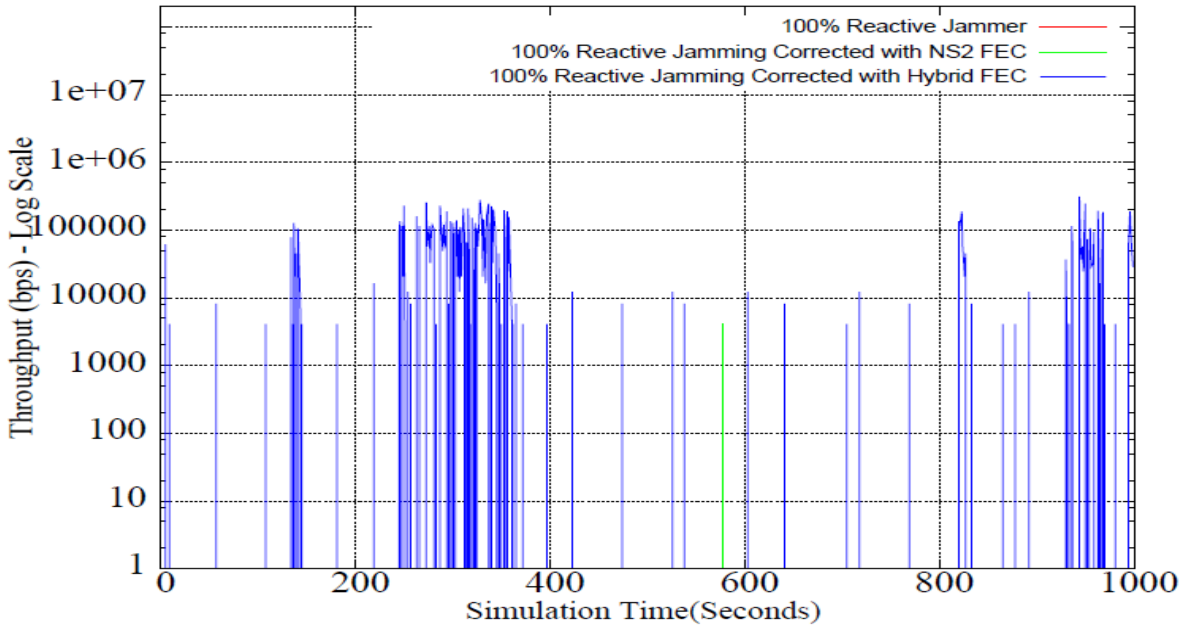


Figure 5.36: **Hybrid FEC Code Performance against 100% Reactive Jamming**

Packet Delivery Ratio

The PDR analysis of the deceptive jammers is depicted with Figure 5.37. The figure essentially compares the PDR of a CRN using ordinary FEC with that of the hybrid FEC code in a reactive jamming scenario. It is noticeable from the figure that the PDR of the ordinary FEC code drops rapidly to about 40% when the jamming rate increases to 60%. Beyond this PDR of the ordinary FEC continues to drop and approaches zero at a jamming rate of 100%. Unlike the ordinary FEC, the hybrid FEC maintains a high PDR of about 70% even when the jamming rate is 100%.

Packet Loss Ratio

Figure 5.38 compares the PLR plot of ordinary FEC code with that of the hybrid FEC code of a CRN under reactive jamming attack. It is noticeable from the figure that the PLR of the ordinary FEC code increases correspondingly as the rate of jamming increases. This shows that there was no significant improvement made by the introduction of the ordinary FEC code with respect to PLR for the different rate of jamming. The hybrid FEC code on the other hand consistently maintains very low PLR for all the jamming rate of the reactive jammers even at 100% jamming. This shows that the hybrid FEC code was able to maintain high recovery rate even when the rate of jamming is as high as 100%, meaning that the hybrid FEC code performs more efficiently than the ordinary FEC code.

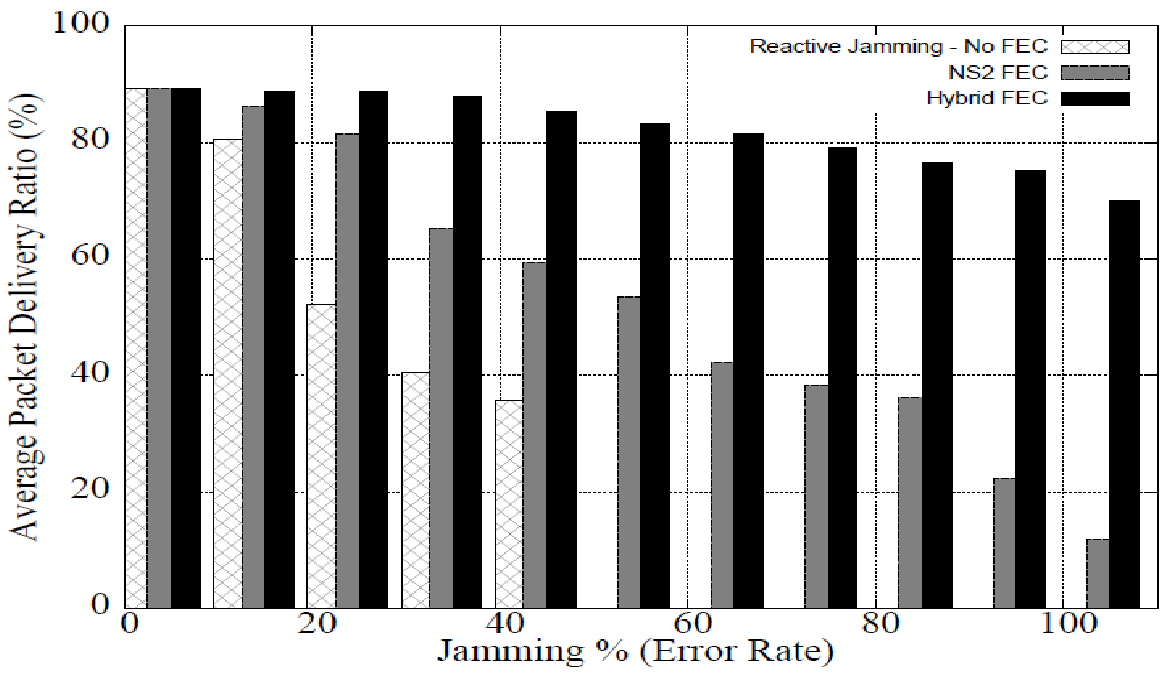


Figure 5.37: **PDR measurement of Hybrid FEC Code Performance against Reactive Jamming**

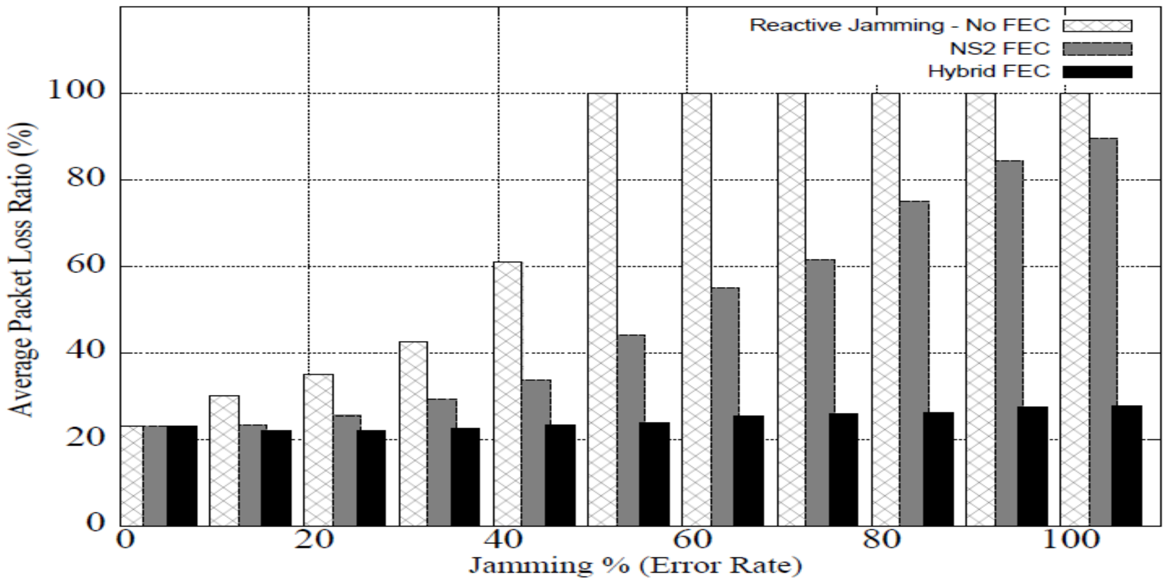


Figure 5.38: **PLR measurement of Hybrid FEC Code Performance against Reactive Jamming**

5.10 Contributions

Given the description of the research description above, the contribution of this research work are restated to include the following:

1. Formulation of a CRN jamming taxonomy that considers both transmissive and omissive value faults. The result of the formulation was presented in [20].
2. Simulation of different jamming types that consider hybrid fault models.
3. Quantifying of the impact of jamming attacks in a hybrid fault model classified CRNs. The result of the jamming impact measurement was presented in [21]
4. Traditional approaches and the faults they can deal with are put into context.
5. A hybrid forward error correction (FEC) code that is capable of mitigating all jamming types identified in Fault-Model-Classified CRNs considering value faults is specified. The result of its performance was presented in [22].

The hybrid FEC code with its effectiveness and the computational feasibility presents a practical solution to the survivability of CR networks under jamming attacks.

5.11 Discussion

In this chapter, we reviewed different anti-jamming strategies that have been deployed in both traditional wireless networks and Cognitive Radio networks with the intent of determining their suitability in combating jamming in CRN. We discovered that non of the strategies are capable of mitigating jamming that produces value faults in CRN. We therefore specified an efficient hybrid FEC code that is capable of mitigating jamming in this adversarial model. We also show through simulation and analysis that our solution is robust against any type of jamming even when the jammers rate of jamming is close to 100%.

Chapter 6

SUMMARY AND FUTURE WORK

6.1 Summary

This research provides a complete approach to dealing with jamming attacks in fault-model based Cognitive Radio Networks. After providing the background of relevant topics, jamming was investigated in the context of hybrid fault models for different jamming strategies. Specifically, we identified fault scenarios and fault types with the expectation that this will enable us to accommodate jamming faults of different severity in a CSS CR network. Attacks on the CR networks were addressed in the context of hybrid fault models using a classification that considers transmissive and omissive value faults. This made it possible to adequately specify what mechanisms can handle these faults.

As a result of the classification, we investigated the fault potential of a combined architectural model for CSS in CR networks by extending it to fault types that include transmissive and omissive value faults. We simulated different types of CR jammers that exhibit these omissive and transmissive fault potentials in NS-2 to study the impact of different jamming strategies on the MAC layer and packet transmissions. It could be shown that the impact of CR jammers on fault-model-classified CRN is huge as the networks collapse when the jamming rate of CR jammers is just about 30%. We therefore explored existing solutions and investigated their capabilities in mitigating CR jammers and concluded that existing solutions for mitigating jamming, both for conventional wireless networks and CR networks, including the use of FEC codes, are not sufficient to handle jamming attacks capable of producing value faults.

A solution was proposed that is a hybrid FEC code incorporating data integrity checking into an efficient forward error correction mechanism. The algorithm was specified as a hybrid FEC code utilizing a Raptor code together with SHA-2 and was simulated in NS-2. Using relevant performance measurement metrics, we analyzed the performance of our proposed solution and compared it with the ordinary FEC code. We discovered that the proposed hybrid FEC code was superior in performance compared to the ordinary FEC code and that the algorithm is very efficient as it was able to maintain high recovery rates by providing consistent low packet loss ratio (PLR), even at 100% jamming rate. The hybrid FEC code with its effectiveness and the computational feasibility presents a practical solution to the survivability of CR networks under jamming attacks.

6.2 Future Work

Several directions for future research are envisioned. There is a need to investigate in detail the performance of fault-model classified CRNs in the presence of other threats such as the spectrum sensing data falsification attack, or primary user emulation attack. Furthermore, we suggest investigating strategies that will help in reducing delays introduced by the hybrid FEC code. Specifically, delay associated with the amount of encoded symbols of the Raptor code necessary to reduce the probability of a decoding failure, as well as the time to migrate control messages from a jammed channel to another channel free of jamming. It is also suggested to quantify the overhead of the new hybrid FEC code with a view of exploring solution strategies towards decreasing the overhead. Lastly, we suggest investigating the impact of the jammers on the primary users of the CRNs.

REFERENCES

[1] FCC Spectrum Policy Task Force, *Report of the spectrum efficiency working group*, ET Docket no. 02-135, Nov. 2002. [Online]. Available: http://www.fcc.gov/sptf/reports.html

[2] J. Mitola and G. Q. Maguire, *Cognitive Radio: Making Software Radios More Personal*, IEEE Pers. Commun. 6(4) (1999)13-18.

[3] I. F. Akyildiz, et al., *NeXt generation/ dynamic spectrum access / cognitive radio wireless networks: A survey, Computer Networks* 50 (13) (2006) 2127-2159.

[4] I. F. Akyildiz, B. F. Lo, and R. Balakrishnan, *Cooperative Spectrum Sensing in Cognitive Radio Networks: A Survey*, Physical Communication (Elsevier) Journal, vol. 4, no. 1, pp. 40-62, March 2011.

[5] M. Acharya and D. Thuente, *Intelligent jamming attacks, counterattacks and (counter)2 attacks in 802.11b wireless networks*, Proceeding of OPNETWORK-2005 Conf., Washington DC, USA, Aug. 2005.

[6] International Telecommunication Union (ITU), E.408 : *Telecommunication networks security requirements.* [Online]. Available: http://www.itu.int/rec/T-REC-E.408-200405-I/en

[7] D. Powell, *Failure Mode Assumptions and Assumption Coverage,* Proc. of the 22nd +- Annual IEEE Int. Symposium on Fault-Tolerant Computing, pp. 386-395, Boston, MA, July 8-10, 1992

[8] M. Li, I. Koutsopoulos, and R. Poovendran. *Optimal jamming attacks and network defense policies in wireless sensor networks.* In 26th IEEE International Conference on Computer Communications, pp. 1307-1315, May 2007.

[9] Asterjadhi, M. Zorzi, *JENNA: A Jamming Evasive Network coding Neighbor discovery Algorithm for Cognitive Radio Networks,* In IEEE ICC Workshop on Cooperative and Cognitive Mobile Networks (CoCoNet3), Cape Town, South Africa, May, 2010

[10] M. Strasser, C. Popper, and S. Capkun. *Efficient Uncoordinated FHSS Anti-jamming Communication*. Proceedings of the 10th ACM international Symposium on Mobile Ad hoc Networking and Computing, pp. 207-218, 2009.

[11] T. Yucek and H. Arslan, *A Survey of Spectrum Sensing Algorithms for Cognitive Radio Applications*, IEEE Communications Surveys Tutorials, Vol. 11, No. 1, pp. 116- 130, 2009.

[12] D. Cabric, S. Mishra and R. Brodersen, *Implementation Issues in Spectrum Sensing for Cognitive Radios*, In Proc. of Asilomar Conf. on Signals, Systems, and Computers, vol. 1, pp. 772-776, 2004.

[13] A. Ghasemi and E. Sousa, *Collaborative Spectrum Sensing for Opportunistic Access in Fading Environments*, in Proc. of IEEE DySPAN 2005, pp. 131-136, 2005.

[14] S. Mishra, A. Sahai and R. Brodersen, *Cooperative sensing among cognitive radios*, in Proc. of IEEE ICC 2006, vol. 4, pp. 1658-1663, 2006.

[15] M.H. Azadmanesh, and R.M. Kieckhafer, *Exploiting Omissive Faults in Synchronous Approximate Agreement,* IEEE Trans. Computers, 49(10), pp. 1031-1042, Oct. 2000.

[16] F.J. Meyer, and D.K. Pradhan, *Consensus with Dual Failure Modes*, Proc. Seventeenth Fault-tolerant Computing Symposium, pp. 48-54, Jul 1987.

[17] K. Fazel and S. Kaiser. *Multi-Carrier and Spread Spectrum Systems: From OFDM and MC-CDMA to LTE and WiMAX*. Wiley, 2008.

[18] Baldini, G., Sturman, T., and Biswas, A. R., et al., *Security Aspects in Software Defined Radio and Cognitive Radio Networks : A Survey and A Way Ahead,* IEEE Communications Surveys and Tutorials 14(2): 355-379 (2012).

[19] E. Lance and G.K. Kaleh. *A Diversity Scheme for a Phase-Coherent Frequency Hopping Spread-Spectrum System.* IEEE Transactions on Communications, 45(9):1123 - 1129, Sep 1997.

[20] V. Balogun and A. Krings, *On The Impact of Jamming Attacks on Cooperative Spectrum Sensing in Cognitive Radio Networks*, Proc. 8th Annual Cyber Security and Information Intelligence Research Workshop, Oak Ridge National Laboratory, January 8 - 10, 2013.

[21] V. Balogun and A. Krings, *An Empirical Measurement of Jamming Attacks in CSS Cognitive Radio Networks*, Submitted to the 27th IEEE Annual Canadian Conference on Electrical and Computer Engineering (CCECE 2014), Toronto, Canada, May 5 - 8, 2014.

[22] V. Balogun and A. Krings, *Mitigating Constant Jamming in Cognitive Radio Networks using Hybrid FEC Code*, Submitted to the 28th IEEE International Conference on

Advanced Information Networking and Applications (AINA-2014) Victoria, Canada, May 13 - 16, 2014.

[23] Todor Mladenov, S. Nooshabadi and K. Kim, *Implementation and Evaluation of Raptor Codes on Embedded Systems*, IEEE Transactions on Computers, vol. 60, no. 12, 2011 .

[24] T. S. Rappaport. *Wireless communications, principles and practice*. Prentice Hall, 1996.

[25] J. Burbank, *Security in Cognitive Radio Networks: The Required Evolution in Approaches to Wireless Network Security*, Third International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CrownCom), 15-17 May, 2008.

[26] J. L. Burbank, et al., *A Common Lexicon and Design Issues Surrounding Cognitive Radio Networks Operating in the Presence of Jamming*, IEEE MILCOM 2008; 1-7.

[27] W. Cadeau and X. Li, *Anti-jamming Performance of Cognitive Radio Networks under Multiple Uncoordinated Jammers in Fading Environment*, Proc. of the 46th Annual CISS, Princeton Univ., NJ, March 2012.

[28] Qi Dong and Donggang Liu, *Adaptive Jamming-Resistant Broadcast Systems with Partial Channel Sharing*, Proc. 2010 International Conference on Distributed Computing Systems, Genova, Italy.

[29] Philipp M. Eittenberger, Todor Mladenov, and Todor Mladenov, *Raptor Codes for P2P Streaming*, 20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 2012.

[30] Axel Krings, *Design for Survivability: A Tradeoff Space*, Proc. 4th Cyber Security and Information Intelligence Research Workshop, CSIIRW 2008, Oak Ridge National Laboratory, May 12-14, 2008.

[31] L. Lamport, et.al., *The Byzantine Generals Problem*, ACM Transactions on Programming Languages and Systems, Vol. 4, No. 3, pp. 382-401, July 1982.

[32] M. Luby, *LT-codes*, in Proceedings of the 43rd Annual IEEE Symposium on the Foundations of Computer Science (FOCS), pp. 271-282, 2002.

[33] J. Mitola III, *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*, Doctor of Technology dissertation, Royal Inst. Technol. (KTH), Stockholm, Sweden, 2000.

[34] S. Zhou, G.B. Giannakis, and A. Swami. *Digital Multi-Carrier Spread Spectrum versus Direct Sequence Spread Spectrum for Resistance to Jamming and Multipath.* IEEE Transactions onCommunications, 50(4):643-655, Apr 2002.

[35] K.Pelechrinis, M.Iliofotou and S.V.Krishnamurthy, *Denial of Service Attacks in Wireless Networks: The case of Jammers*, In IEEE Communication Surveys and Tutorials, pp (99):113, April 2011.

[36] R.A. Poisel. *Modern Communications Jamming Principles and Techniques*. Artech House Publ., 2003.

[37] T. Issariyakul and E. Hossain. *Introduction to Network Simulator NS2*. Springer Science+Business Media, LLC, New York, NY 10013, USA 2009.

[38] Shokrollahi, A. *LDPC Codes: An introduction.* Available: www.ipm.ac.ir/IPM/homepage/Amin2.pdf. [2003]

[39] A. Shokrollahi, *Raptor Codes*, IEEE/ACM Transaction on Networking, 14(SI):2551-2567, 2006.

[40] N. Sklavos and O. Koufopavlou, *Implementation of the SHA-2 Hash Family Standard Using FPGAs,* J. of Supercomputing, Vol. 31, No 3, pp. 227-248, 2005.

[41] P. Tague, D. Slater, G. Noubir, and R. Poovendran, *Quantifying the impact of efficient cross-layer jamming attacks via network traffic flows*, Network Security Lab (NSL), University of Washington, Tech. Rep., 2009. Available: www.ee.washington.edu/research/nsl/papers/TR005.pdf

[42] P. Tague, *Improving Anti-jamming Capability and Increasing Jamming Impact with Mobility Control*, In 6th IEEE International Workshop on Wireless and Sensor Networks Security (WSNS), Nov. 2010.

[43] P. Thambidurai, and Y. K. Park, *Interactive Consistency with Multiple Failure Modes,* Proc. 7th Symp. on Reliable Distributed Systems, Columbus, OH, pp. 93-100, Oct. 1988.

[44] Yongle Wu, Beibei Wang, K. J. Ray Liu, and T. Charles Clancy, *Anti-Jamming Games in Multi-Channel Cognitive Radio Networks*, IEEE Journal on Selected Areas in Communications, Vol. 30, NO. 1, January 2012.

[45] Y. Wu, B. Wang, and K. J. Ray Liu. *Optimal Defense against Jamming Attacks in Cognitive Radio Networks Using the Markov Decision Process Approach.* In Proceedings of GLOBECOM'2010. pp.1-5, 2010.

[46] The Network Simulator version 2 (NS-2). Available: http://www.isi.edu/ nsnam/NS2/.

[47] Jing Zhong and Jialiang Li, *Cognitive Radio Cognitive Network (CRCN),* Available: http://stuweb.ee.mtu.edu/ ljialian/index.htm

[48] National Institute of Standards and Technology (NIST), *Secure Hash Standard (SHS)*, Federal Information Processing Standards Publication, March 2012. Available: http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

[49] Wenyuan Xu, et al. *The feasibility of launching and detecting jamming attacks in wireless networks*, Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, May 25-27, 2005, Urbana-Champaign, IL, USA.

[50] M.K. Simon, J.K. Omura, R.A. Scholtz, and B.K. Levitt. *Spread Spectrum Communications Handbook.* McGraw-Hill, Inc., 2001.

[51] H. Zimmermann. *OSI Reference Model -The ISO Model of Architecture for Open Systems Interconnection.* IEEE Transactions on Communications 28 (4): 425-432, April 1980.

[52] R. Chen, J. Park and J. H. Reed, *Defense against Primary User Emulation Attacks in Cognitive Radio Networks,* IEEE Journal on Selected Areas in Communications, Vol. 26, No. 1, pp. 25-37, 2008.

[53] Q. Liu, J. Gao, Y. Guo and S. Liu, *Consensus-Based Cooperative Spectrum Sensing with Improved Robustness Against SSDF Attacks*, Frequenz, Vol. 65, pp. 103-107, May 2011.

[54] Bakhtiari, S.; Safavi-Naini, R.; and Pieprzyk, J. *Cryptographic Hash Functions: A Survey.* Technical Report 95-09, Department of Computer Science, University of Wollongong, July 1995. Available: http://citeseer.ist.psu.edu/bakhtiari95cryptographic.html.

[55] W. Zhang and K. Letaief, *Cooperative Spectrum Sensing with Transmit and Relay Diversity in Cognitive Radio Networks,* IEEE Transactions on Wireless Communications 7 (12) (2008) 4761-4766.

[56] K. Cheun, K. Choi, H. Lim, and K. Lee. *Antijamming Performance of a Multicarrier Direct-Sequence Spread-Spectrum System.* IEEE Transactions on Communications, 47(12):1781-1784, Dec 1999.

[57] V. Bharghavan, A. Demers, S. Shenker and L. Zhang, *MACAW: Media access protocol for wireless lans,* In In Proceedings of the ACM SIGCOMM Conference, 1994.

[58] S. Liu, L. Lazos, and M. Krunz, *Thwarting Control-Channel Jamming Attacks from Inside Jammers,* IEEE Transaction on Mobile Computing, Vol. 11, No. 9, pp. 1545-1558, Sept. 2012.

[59] R. W. Doran, *The Gray Code*, CDMTCS Research Reports, CDMTCS-304, March 2007. Available: http://www.cs.auckland.ac.nz/research/groups/CDMTCS/researchreports/304bob.pdf

[60] M. Gudmundson, *Correlation Model for Shadow Fading in Mobile Radio Systems,* Electronics Letters, Vol. 27, No. 23, pp. 2145-2146, 1991.

[61] H. Zhang and Y. Li. *Anti-Jamming Property of Clustered OFDM for Dispersive Channels*. In IEEE Military Communications Conference, vol. 1, pP. 336-340 Vol.1, Oct. 2003.

[62] A. Ghasemi and E. Sousa, *Asymptotic Performance of Collaborative Spectrum Sensing Under Correlated log-normal Shadowing,* IEEE Communications Letters, Vol. 11, No. 1, pp. 34-36, 2007.

[63] Todd K. Moon, *Error Correction Coding: Mathematical Methods and Algorithms*, Wiley-Interscience, 2005

[64] K. Bian and J.-M. Park, *Security Vulnerabilities in IEEE 802.22,* The Fourth International Wireless Internet Conference (WICON 2008), Nov. 2008.

[65] B. Randell and J. Xu, *Recovery Blocks.* In: Marciniak, J.J, ed. Encyclopedia of Software Engineering. New York, USA: Wiley, 1994, pp.1037-1038.

[66] J.T. Chiang and Y. Hu. *Dynamic Jamming Mitigation for Wireless Broadcast Networks.* In the IEEE 27th Conference on Computer Communications, pp. 1211-1219, April 2008.

[67] S. Anand, Z. Jin, and K. P. Subbalakshmi*, An Analytical Model for Primary User Emulation Attacks in Cognitive Radio Networks*, Proceedings of the IEEE Symposium of New Frontiers in Dynamic Spectrum Access Networks (DySPAN ’2008), Oct. 2008.