Experimental Study of Multirate Margin in Software Defined Multirate Radio

Tianning Shen
Cleveland State University

EXPERIMENTAL STUDY OF MULTIRATE MARGIN IN SOFTWARE DEFINED MULTIRATE RADIO

TIANNING SHEN

Bachelor of Science in Electrical Engineering

Tianjin University

July, 2006

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

at the

CLEVELAND STATE UNIVERSITY

December, 2009

This thesis has been approved for the

Department of **ELECTRICAL AND COMPUTER ENGINEERING**

and the College of Graduate Studies by

Thesis Committee Chairperson, Dr. Fuqin Xiong

Department/Date

Dr. Chansu Yu

Department/Date

Dr. Pong P. Chu

Department/Date

ACKNOWLEDGMENTS

First of all, I would like to give my gratitude to my advisor, Dr. Xiong, for leading me to the most interesting wireless world, for educating me about the fundamental knowledge of the wireless communication and for his willingness to help me all the time. His encouragement to me and guidance to my thesis are deeply appreciated.

I would like to thank Dr. Yu who helped me a lot about my research work and who always gave me hints to solve the problems in my experimentation. His idea in Multi-hop Transmission is great! I also would like to thank Dr. Chu for imparting the most advanced digital design techniques to me and for his kindness of helping me whenever he is available.

My gratitude will also be given to the fellows and colleagues who worked with me when I was pursuing my master's degree in Cleveland State University. The help and advice from Song, Sachin, Robert, Kushal, Dawei and Honglei are quite valuable and will be highly appreciated.

Last but not the least, I would like to thank my parents who have been supporting me from my childhood till now. Without their help, I cannot make this far.

EXPERIMENTAL STUDY OF MULTIRATE MARGIN IN SOFTWARE DEFINED MULTIRATE RADIO

TIANNING SHEN

ABSTRACT

Due to the recent development of spectrally-efficient modulation schemes, IEEE 802.11 Wifi and IEEE 802.16 WiMax radios support wireless communication at multiple bit rates. While high-rate transmission allows delivering more information in less time, the corresponding performance improvement is less than expected due to the PHY- and MAC-layer overheads, imposed by the 802.11/16 standards. This is particularly true in wireless ad hoc networks as there exist rate-distance and rate-hop count tradeoffs.

The concept of multi-rate margin is proposed in this thesis, which exploits the difference in communication characteristics at different rates and serves as the fundamental ingredient for an opportunistic transmission protocol, targeted to meliorate the ad hoc mobile wireless network performance. In this thesis, the multi-rate margin is analyzed with theoretical derivation, perceived with simulation result using MATLAB and observed through real world testing using USRP and GNU Radio, which is a recent implementation of Software Defined Radio.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYM

ACM	Adaptive Coding and Modulation
DVB-S2	Digital Video Broadcasting C Standard 2
DIFS	DCF Interframe Space
CSMA/CA	Carrier Sensing Multiple Access with Collision Avoidance
USRP	Universal Software Radio Peripheral
MTOP	Multi-hop Opportunistic Transmission
SDR	Software Defined Radio
OFDM	Orthogonal Frequency Division Multiplexing
PSDU	PLCP service data unit
PLCP	Physical Layer Convergence Procedure
PPDU	PHY protocol data unit
SIFS	Short Interframe Space
AMRR	adaptive multi-rate retry
CCA	Clear Channel Assessment
ED	Energy Detect
DSSS	Direct-Sequence Spread Spectrum
CCK	Complementary Code Keying

LNA Low Noise Amplifier

BPF Band Pass Filter

PDR Packet Delivery Ratio

CRC Cyclic Redundancy Code

RSSI Received Signal Strength Indicator

CHAPTER I

INTRODUCTION

1.1 Multi-rate Radio and Adaptive Coding and Modulation (ACM)

The concept of multi-rate radio appears to be a new terminology in the area of wireless communication at the first glance, but in fact the phenomenon of multi-rate radio has already been there when different types of modulation methods are used to transmit a certain amount of data that are supposed to occupy the same spectrum bandwidth. These multiple modulation methods result in multiple bit rates and necessitate the technology of ACM.

Nowadays, ACM is widely used in wireless communication networks, e.g. ACM is specified as a primary technology in physical layer for IEEE 802.11 Standard (Wi-Fi)[1],[2], IEEE 802.16 Standard (WiMax)[3] and even in the most advanced open standards of digital television Digital Video Broadcasting C Standard 2 (DVB-S2). ACM is also selected as one of the major link adaptation strategies.

It is well known that the mobile radio channel fundamentally limits the performance of wireless communication systems. The propagation path between transmitter and receiver can easily change from line of sight to the one that is full of obstacles such as buildings, mountains, etc. Establishing models for wireless channels has historically been one of the most difficult parts in wireless system design because unlike wired communications the parameters of wireless communications always change extremely randomly. One of the important channel models is fading channel, where signal strength changes rapidly over a short period of time or distance[4, 5]. It is imaginable that when wireless communication systems are affected by changing weather conditions, the systems without ACM will not be able to deal with such signal degradation which directly endangers the performance of the communication systems, while the systems installed with ACM will be able to adjust the modulation schemes to protect the communication from being ruined by the weather-related fading, which might be induced by the storm or heavy rain, etc.

One application example of ACM is in DVB-S2, which is shown in Figure 1. In this figure, 8PSK Rate 9/10 is the most efficient coding and modulation combination when the channel condition is good, while QPSK Rate 1/2 is the most robust coding the modulation combination for fading channel or rainy conditions.

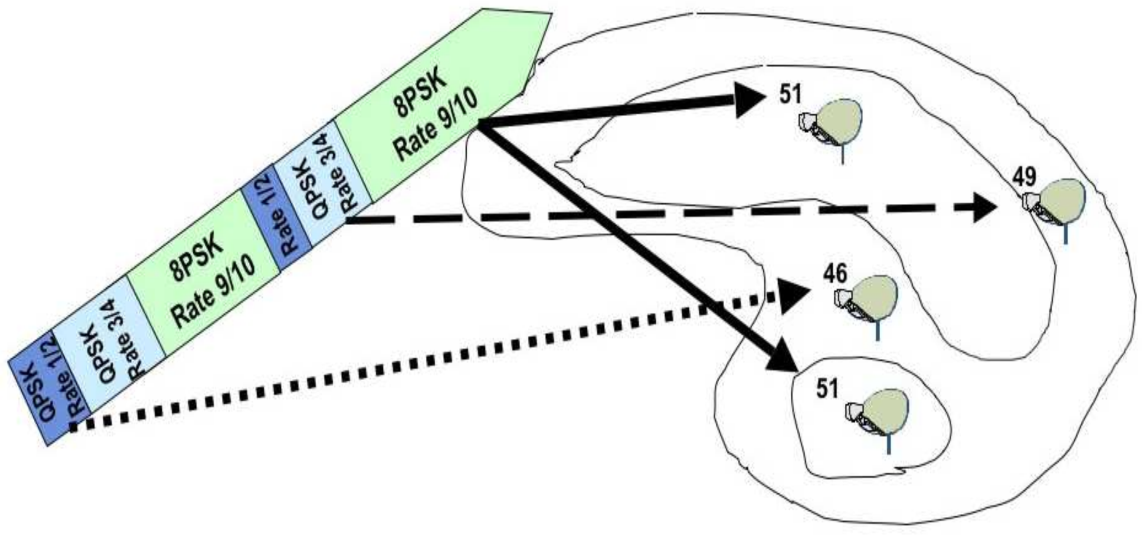


Figure 1: Sample satellite footprint with downlink EIRP values[40]

1.2 Multi-rate Margin

Although higher data rates like 48Mbps and 54Mbps can be used in IEEE 802.11a radios, the network performance does not improve linearly due to rate-independent PHY- and MAC-layer overheads as well as undesirable behavior known as performance anomaly in wireless LANs, which will be discussed in detail in Chapter 2. To address these issues in wireless LANs, opportunistic transmission protocols [7, 8] have been proposed to allow a node to transmit multiple frames back-to-back at high data rates when it captures the chance to use the medium.

However, this may not be effective in Ad Hoc Wireless Networks because it is possible that no data is ready to be transmitted although a node captures the chance. Multi-hop Opportunistic Transmission (MTOP) is an extended version of the conventional opportunistic transmission protocol, proposed by Dr. Yu at Cleveland State University. It allows a frame to be forwarded over multiple hops at high data rates when the first node in the chain captures the chance to use the medium. Eliminating the MAC-layer overhead such as DCF Interframe Space (DIFS) and backoff time will definitely enhance the entire throughput in Ad Hoc Wireless Networks. Note that DIFS and backoff time are imposed by the Carrier Sensing Multiple Access with Collision Avoidance (CSMA/CA) protocol adopted as the MAC protocol in the IEEE 802.11 Standard.

This thesis focuses on demonstrating the existence of and quantifying the Multi-rate Margin in multi-rate radios, which is the basic assumption in the development of MTOP. It will be verified via mathematical analysis, Matlab-based simulation as well as GNU Radio/Universal Software Radio Peripheral (USRP)-based experiments.

1.3 Software Radio and GNU Radio

As mentioned above, this thesis uses GNU Radio and USRP, a recent implementation of Software Radio, as an experimental tool to help us perceive the multi-rate margin in real world [9].

As one of the hot topics of 4 G wireless communication technologies, software radio is absolutely a revolution in radio design in that it is capable of changing radios on the fly, creating new choices for users. One of the major advantages of software radio against hardware based radio comes with the flexibility of using the software. Instead of using a bunch of fixed function gadgets, changing the modes of our own radio can be easily realized by changing the codes which will then be loaded into our software radio. With the help of software radio, our cell phones will be able to give us connectivity using GPRS, 802.11 Wi-Fi, 802.16 WiMax, a satellite hookup or the emerging standard of today.

As one of the representatives in the family of the software radio, GNU Radio is an open source software toolkit of signal processing for building up our own software defined radios. It provides functions to define the transmitted waveforms, to demodulate the received signals and gives us a collection of modulators and demodulators that keep on growing. The type of modulation techniques can be easily modified by changing the argument of a certain modulation option in the command line.

1.4 Contributions and Outline of the Thesis

The contributions of this thesis are: Demonstrate deep understanding of GNU Radio and USRP (Universal Software Radio Peripheral); Design the entire testing scenario and observe the multi-rate margin in real world wireless communication with GNU Radio and USRP; Simulate the IEEE 802.11b self-interference communication system and investigate the multi-rate margin phenomenon in 802.11b mixed networks; Explore the merits of multi-rate margin in MTOP.

This thesis is organized as follows: Chapter 2 shows some background of multi-rate support in IEEE 802.11 Radio and Software Defined Radio (SDR); Chapter 3 presents the multi-rate margin phenomenon existing in the 802.11b radio via mathematical analysis and Matlab-based simulation. Chapter 4 presents the multi-rate margin phenomenon existing in the GNU Radio/USRP-based software radio platform via simulation as well as experiment. Chapter 5 covers the exploration of Multi-rate Margin in MTOP. Finally, conclusions and possible future work will be given in Chapter 6.

CHAPTER II

LITERATURE REVIEW

2.1 Modulation and Coding in communication systems

Modulation and coding are the most important parts or technologies in communication systems. Digital modulation is the process of using the variation of signal to represent the digital symbols and coding scheme is a method to use redundant symbols to protect the information symbols. As can be seen, digital modulation is aimed to increase the bandwidth efficiency but at the same time sacrifices the power efficiency, while coding scheme is designed to improve the power efficiency but simultaneously suffers from bandwidth efficiency loss. The power efficiency of a modulation or coding scheme is straightforwardly defined as the required Eb/No for a certain bit error rate over AWGN channel and bandwidth efficiency of a modulation or coding scheme is defined as Rb/W, where Rb is the bit transmission rate and W is the bandwidth of the transmitted signal [4]. Figure 2 shows the bandwidth efficiency and power efficiency characteristics of three typical modulation schemes, MPSK, MFSK and QAM. As can be seen from Figure 2, the bandwidth efficiency of QAM and

MPSK is rather higher than that of FSK, which is one of the primary reason that QAM and MPSK are chosen as the modulation scheme in IEEE 802.11a and IEEE 802.11b rather than FSK.

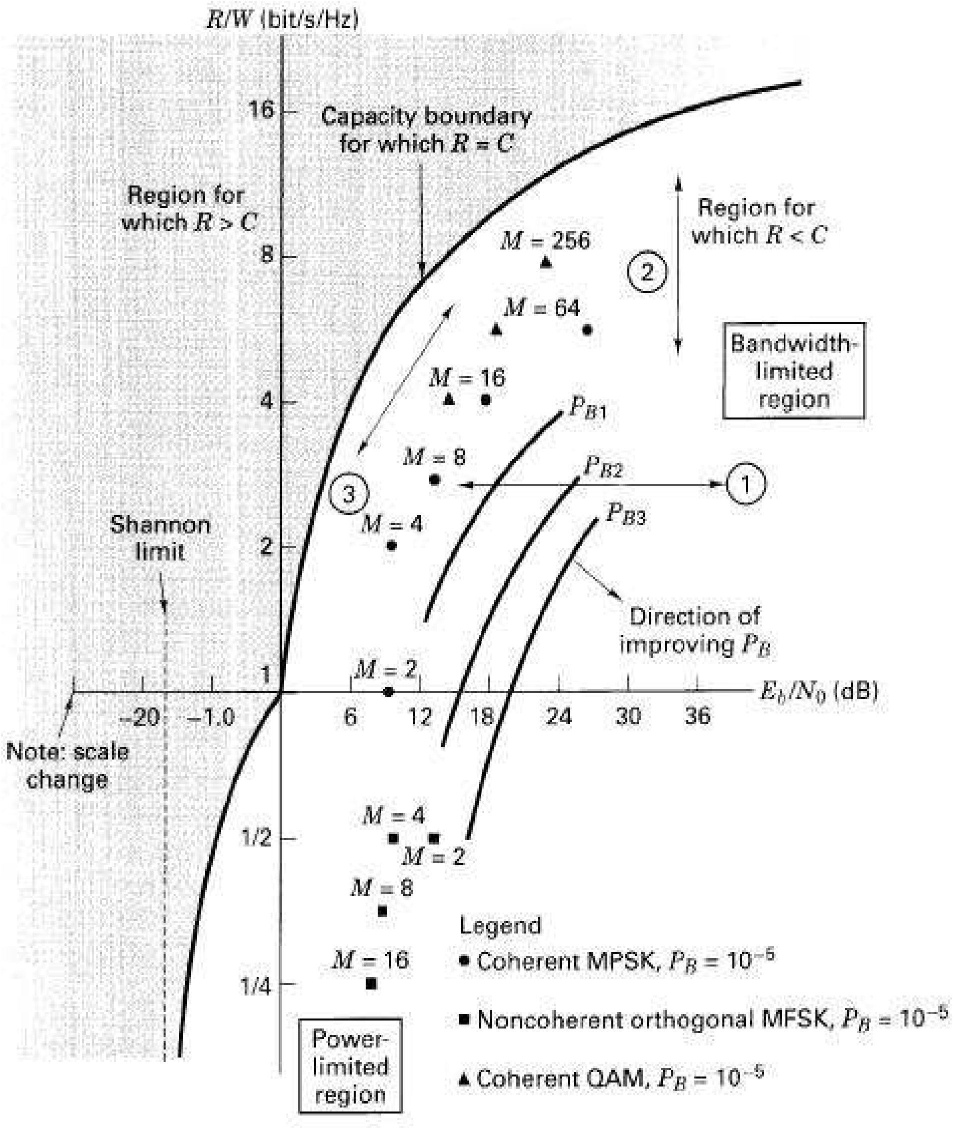
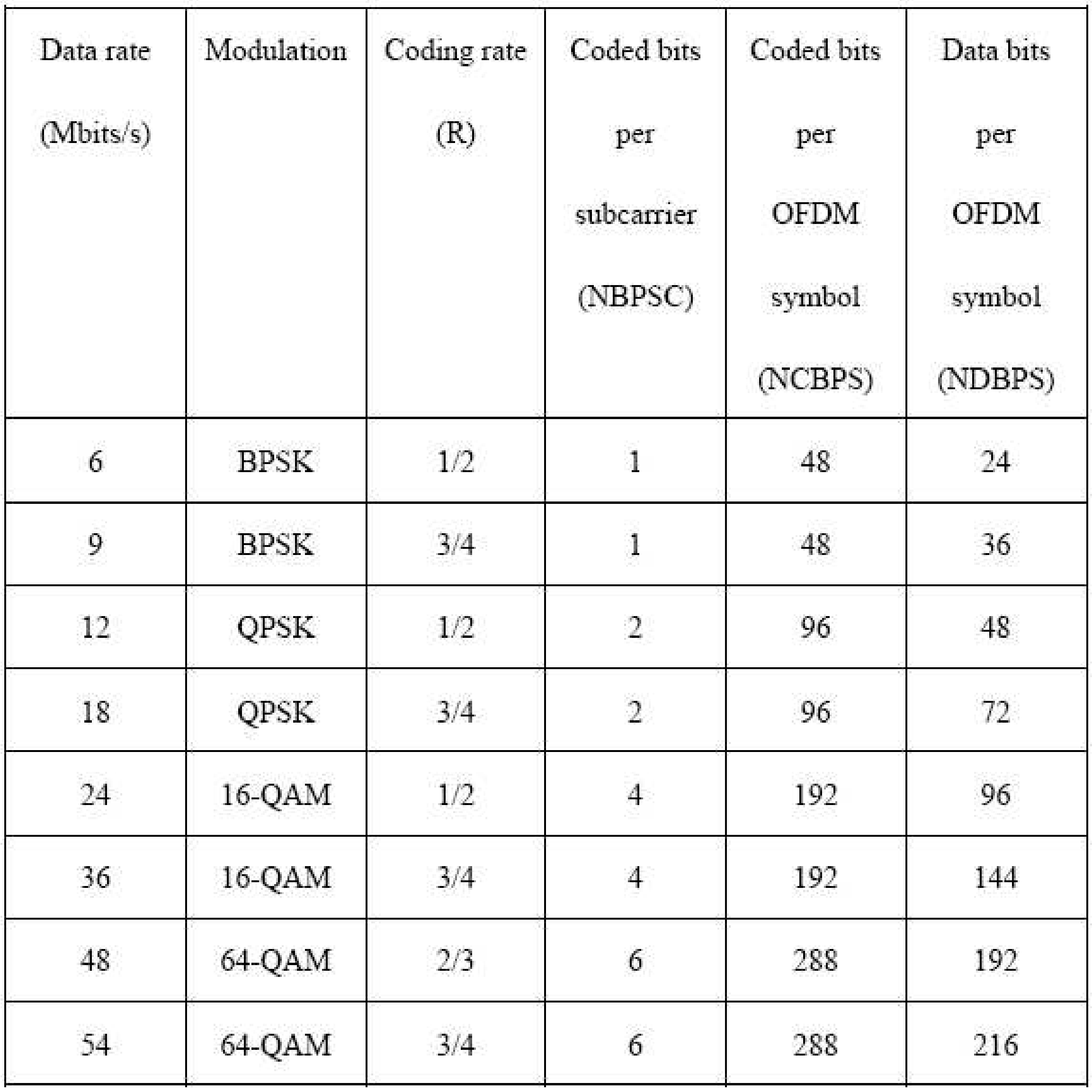


Figure 2: Bandwidth and power efficiency plan [4]

2.2 Multi-rate Support of IEEE 802.11 Radios

2.2.1 Multi-rate and Modulation Schemes in IEEE 802.11 Standards

Multi-rate support is one of the key technologies used in IEEE 802.11 to adjust data transmission rate according to the change in communication environment with the objective of improving the overall performance. Figure 3 shows the modulation schemes and corresponding data rates used in 802.11a, where convolutional coding scheme is selected as the channel coding scheme and Orthogonal Frequency Division Multiplexing (OFDM) is chosen as the multiplexing technique to utilize multi-carrier to transmit signals. The idea of OFDM is to convert a stream of serial data symbols into parallel data symbols and allocate them on a set of orthogonal sub-carriers. Obviously, the length of one symbol duration is extended, which would reduce the intersymbol interference caused by multi-path fading channel. The coding rate in Figure 3 is defined as the number of transmitted information bits over the number of coded bits, e.g. the coding rate of 3/4 means the number of input bits divided by the number of output bits of the convolutional encoder is 3/4. Rather than generating the codeword based on the linear combination of the information bits in block codes, the encoder of the convolutional codes contains memory and the output of the encoder at any time not only depends on the inputs at that time but also on some number of previous inputs [10]. The modulation methods of BPSK and QPSK are similar because both of them use transmitted bits to modulate the phase of the transmitted signal, e.g. a cosine waveform. The only difference between BPSK and QPSK is that the latter one groups two bits into one symbol to modulate the phase while the former one only uses one bit to modulate the phase. Table 1 shows the modulation method of QPSK, which is also used in 802.11b. The QAM modulation method is a kind of non-constant envelop modulation scheme, the constellation of which is shown in Figure 4.



Data rate (Mbits/s)	Modulation	Coding rate (R)	Coded bits per subcarrier (NBPSC)	Coded bits per OFDM symbol (NCBPS)	Data bits per OFDM symbol (NDBPS)
6	BPSK	1/2	1	48	24
9	BPSK	3/4	1	48	36
12	QPSK	1/2	2	96	48
18	QPSK	3/4	2	96	72
24	16-QAM	1/2	4	192	96
36	16-QAM	3/4	4	192	144
48	64-QAM	2/3	6	288	192
54	64-QAM	3/4	6	288	216

Figure 3: Rate dependent parameters of IEEE 802.11a [2](convolutional code is selected as the channel coding scheme and there are 48 subcarriers)

DBPSK (Differential BPSK) and DQPSK (Differential QPSK) are used in 802.11b to transmit signals at 1Mbps and 2Mbps. For DBPSK, each bit is first differentially encoded based on formula 2.1.

$$B(i) = B(i-1) xor A(i) \tag{2.1}$$

Where $A(i)$ is the current information bit and $B(i)$ is the encoded bit at that time.

For DQPSK, the encoding scheme uses similar encoding method as that of DBPSK except it is based on modular 4 additions. The detailed modulation techniques of 802.11b will be discussed in Chapter 3.

Table I: DQPSK encoding table[1]

Dibit Pattern (d0, d1) (d0 is first in time)	Even Symbols Phase Change (+jw)	Odd Symbols Phase Change (+jw)
00	0	π
01	$\pi/2$	$3\pi/2(-\pi/2)$
11	π	0
10	$3\pi/2(-\pi/2)$	$\pi/2$

2.2.2 PHY and MAC layer specification

There are three different physical layers that are supported by IEEE 802.11 standard: one layer based on infrared and two layers on the basis of radio transmission. Based on the physical layer specification in IEEE 802.11 b [1], the PLCP service data unit (PSDU) shall be appended to the Physical Layer Convergence Procedure (PLCP) preamble and header to generate the PHY protocol data unit (PPDU). Two different preambles and headers are defined: the mandatory supported long preamble and header and optional short preamble and header. The preamble and header are used to aid in demodulation and transmission of the PSDU. Figure 5 and 6 show the format for long PPDU and short PPDU.

With the illustration in Figure 5 and 6, the side effect of the PHY-layer overhead can be easily analyzed. Considering a PSDU comprised with 512-byte data, since the long PLCP preamble and header are 144 bits and 48bits (192μs), the overall

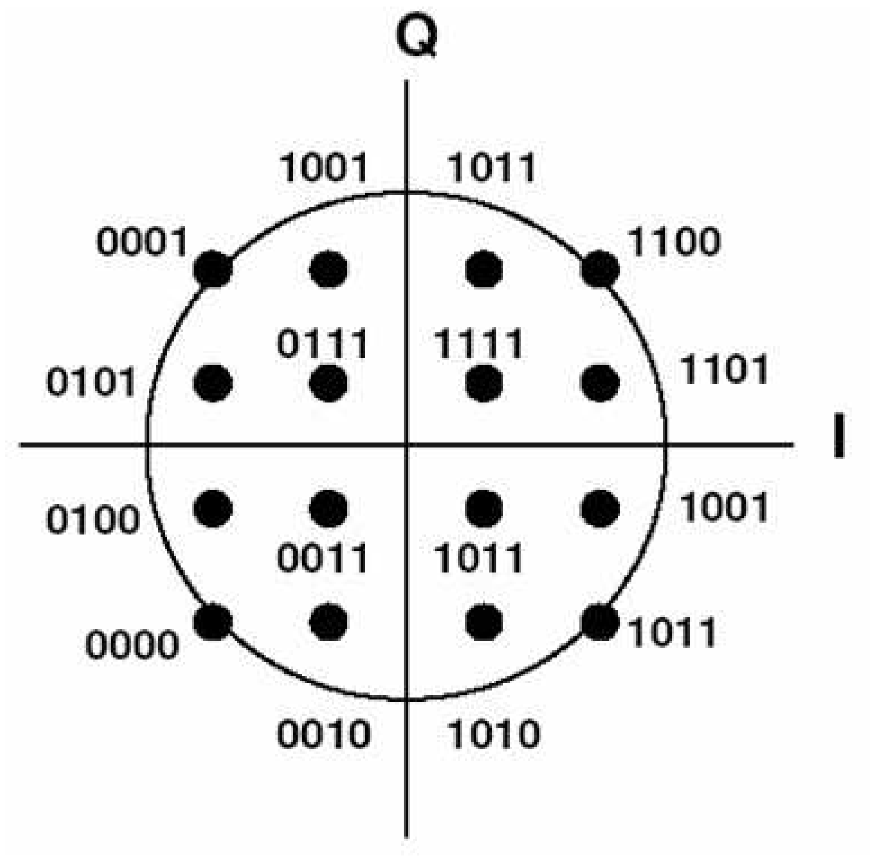


Figure 4: Constellation Picture for 16QAM [4]

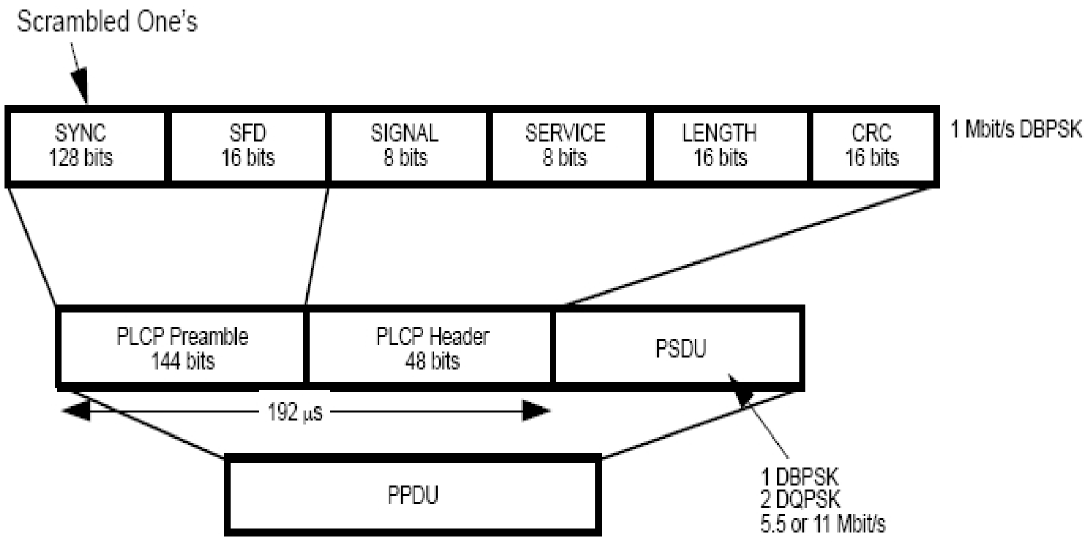


Figure 5: Long PLCP PPDU format [1]

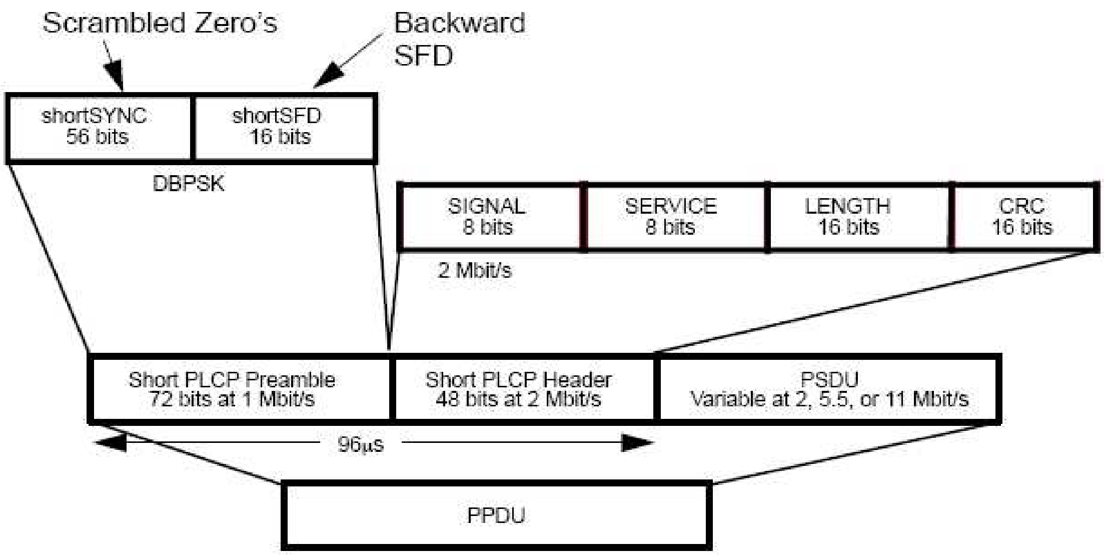


Figure 6: Short PLCP PPDU format [1]

frame size is 4288μs (t_1) at 1Mbps. Since the payload can be transmitted at higher rates, it becomes 2240, 937, and 564μs for 2, 5.5 and 11Mbps, respectively (t_2, $t_{5.5}$, and t_{11}). Consequently, the per-frame PHY overhead due to long PLCP preamble and header is 4.5, 8.6, 20.5 and 34.0% at 1, 2, 5.5 and 11Mbps. Even if the short PLCP preamble and header are selected and transmitting the short PLCP preamble and hearder will cost 96μs, the per-frame PHY overhead due to short PLCP preamble and header is 2.3, 4.5, 11.42 and 20.5 % at 1, 2, 5.5 and 11Mbps. As can be seen from the analysis, the overhead of the short PLCP is still very large.

The mandatory access mechanism of IEEE 802.11 is based on Carrier Sensing Multiple Access with Collision Avoidance (CSMA/CA), which is a random access scheme based on carrier sensing and collision avoidance via random backoff. Figure 7 shows the scenario for the basic CSMA/CA. For the basic CSMA/CA method, when the node senses the media is idle, it waits for DCF Interframe Space (DIFS) period and senses the media again, if the media is still idle, it chooses a random backoff time within a contention window (CW) and delays the transmission for this period of time. If the media is still idle after waiting for the random back off time, the node immediately transmit MPDU. Upon receiving a correct packet at the receiver, the receiving station waits for a Short Interframe Space (SIFS) interval and transmits a positive acknowledgment frame (ACK) back to the source station, implying that the transmission is successful. From Figure 7, the MAC-layer overhead can be easily seen. Suppose each frame transmission contend for the media access based on the random time selection from CW. Since CW_{max} is 31 1023 and slot time is 20μs, the time for contention is 1620 or 320μs (t_c) on the average when CW_{max} is 31. Now, T_i, the time duration for the frame sequence at data rate i, is as follows: DIFS and contention (t_c or 50+320μs), Data (t_i), SIFS (t_{SIFS} or 10s) and ACK (t_{ACK} or 376μs).

$$T_i = t_c + t_i + t_{SIFS} + t_{ACK} \tag{2.2}$$

It totals 5044, 2996, 1693, and 1320μs for 1, 2, 5.5 and 11Mbps, respectively. Considering the payload size, the MAC-layer overhead amounts to 15.0, 25.2, 44.7 and 57.3% for 1, 2, 5.5 and 11Mbps, respectively. In other words, 57.3% of medium time is wasted due to the MAC overhead at 11Mbps, which is increased to as much as 95.0% when CW_{max} reaches 1023. It is evident, therefore, that it is critically important to reduce this overhead, particularly for high-rate transmissions.

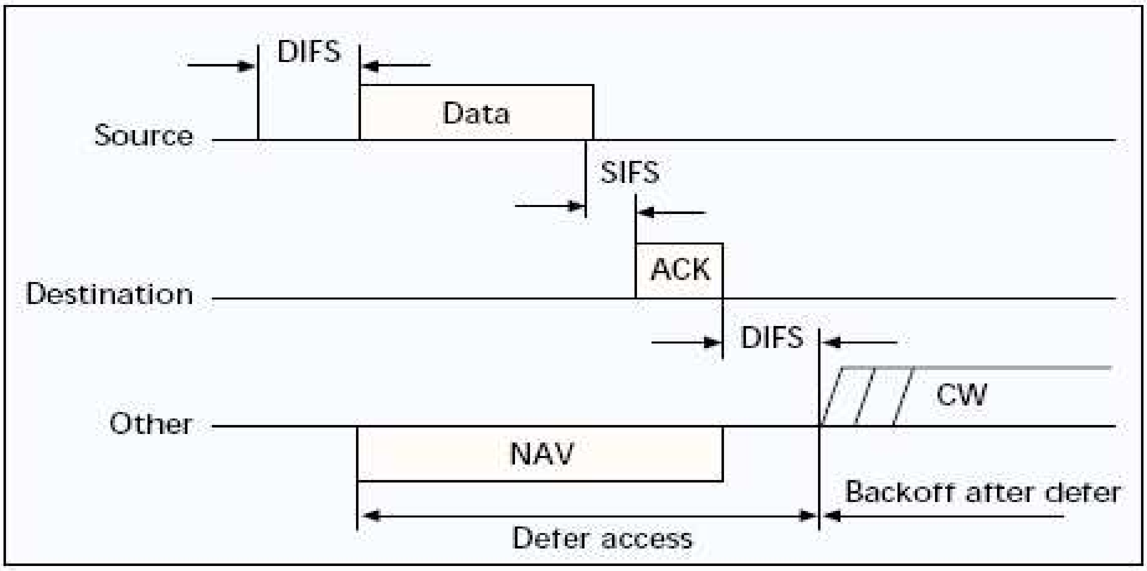


Figure 7: Basic CSMA scenario [46]

2.3 Multi-rate Performance in Wireless Networks

As is known, current WLAN has multi-rate support, where data could be transmitted with different transmission rates. A number of approaches have been proposed for exploiting the multi-rate capability of wireless network. They usually fall into two categories as sender-based and receiver-based approaches.

As one of the sender-based approaches, Auto-Rate Fallback (ARF) [32] aimed to optimize the application throughput in WaveLan II devices, which implemented the 802.11 DSSS standard. A higher transmission rate will be used by the sender after a certain number of successful transmissions between sender and receiver at a relatively lower transmission rate and falls back to a lower rate after one or two consecutive failures. Its two primary disadvantages are that it is not efficient when the channel conditions change very fast because it needs up to 10 successful packet transmissions when the optimum rate adaptation is from one packet to the next in a fast changing channel; it is not efficient when the channel conditions are stable because changing to higher rate after 10 successful transmissions results in retransmission attempts [35]. Other proposals of ARF variations include adaptive ARF [33], adaptive multi-rate retry (AMRR) [33] and estimated rate fall back (ERF) [34].

Receiver-Based Auto Rate [36] involves the use of the RTS/CTS control frames between the source and destination nodes before each data transmission. The receiver uses the received RTS frame to calculate the transmission rate for the upcoming data frame and insert the transmission rate information in the CTS frame, which is supposed to piggyback the data rate information to the source node. The calculation of the transmission rate is based on the received Signal to Noise Ratio (SNR) of the received RTS frame and on a number of SNR thresholds calculated on the assumption of a known wireless channel condition. The flaws of this method mainly stem from the fact that the calculation of SNR thresholds is based on a priori channel model and the incompatibility of this method with the existing 802.11 standard imposes the impossibility on deploying this method in the current 802.11 wireless networks.

A great deal of work on multi-rate adaptation has been reported in the context of multihop networks [42], [43] because it can greatly improve the network throughput. They can be categorized as proactive or on-demand depending on the routing algorithm used. With a proactive multi-rate algorithm, each node maintains link costs to each of its neighbors while taking the multirate capability into account. Link costs used include delay, bandwidth distance product (BDiP) [39], medium time metric (MTM) [42], estimated transmission time (ETT, MITs Roofnet) [45], weighted cumulative ETT (WCETT, MSRs testbed) [44], and bandwidth delay product (BDP, Strix Systems) [44].

2.4 Software Defined Radio

2.4.1 Disadvantages of Hardware Defined Radio

As the traditional approach for the radio design, hardware based design is to use analog circuit to build each element of the radio chain and each element in the radio system just performs a specific function. The typical analog radio receiver block is shown in Figure 8.

First of all, if any of the parameters of the technical requirement changes, the traditional analog radio has to be redesigned to suit the needs of new conditions and even the hardware module has to be replaced and remanufactured. Redesigning, building and manufacturing will cost a lot and take longer time to put the product into the market, which is one of the most significant points that traditional radio is confronting. Second, with hardware based radio, it is quite hard to include many different kinds of services on one device because of the limitations of the elements comprising the hardware radio. Each element is supposed to only realize one specific function, so the hardware based radio cannot work as a CDMA cell phone and at the same time receive DVB signal.

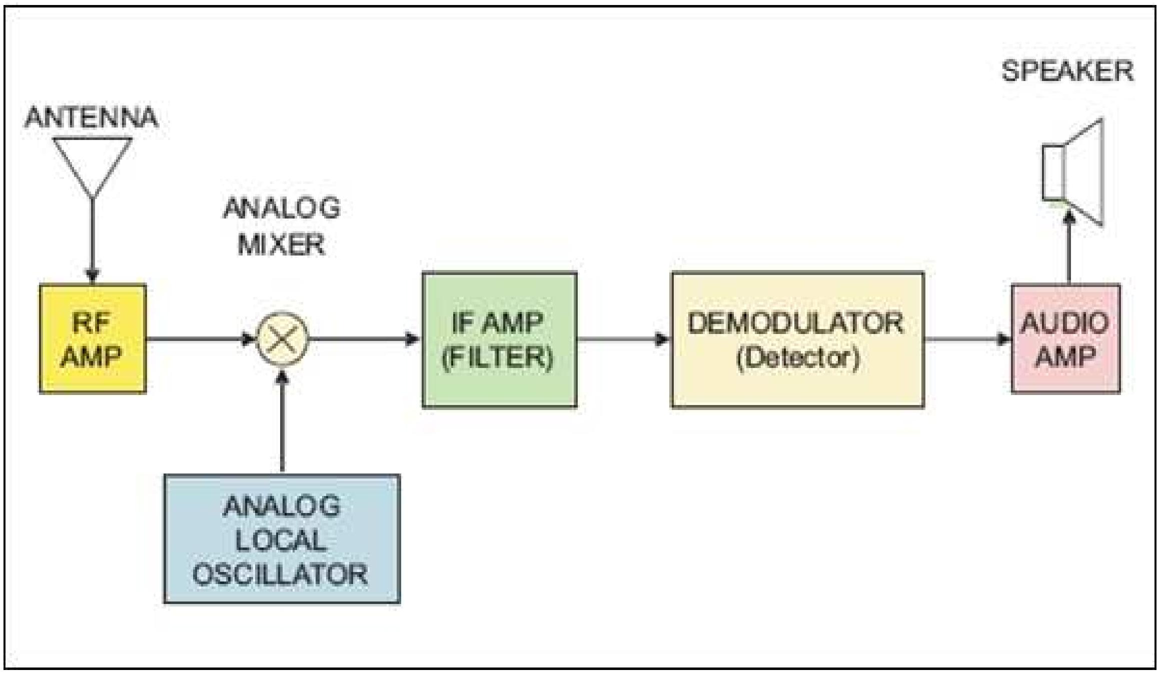


Figure 8: analog radio receiver block diagram[12]

2.4.2 Characteristics of Software Defined Radio

Software Defined Radio is the technique that uses software to realize the function of the traditional radio. It can get code as close to the antenna as possible, define the transmitted waveforms, and demodulate the received waveforms. The most apparent benefit is that we just need to load the relative program to change modulation scheme and coding scheme for different application requirement instead of having to change the dedicated circuit to satisfy different criterion. With only one PC and one USRP, we can tune our hardware to receive FM signal, satellite signal and even HDTV signal, and we can also create mesh network to transmit data among the nodes in the mesh network without relying on so called internet backhaul. The SDR receiver diagram is shown in Figure 9. As can be seen from Figure 9, as the DSP or FPGA can only process digital signals, the A/D converters is required at the receiver to convert the analog signal to digital signal, and after transforming the analog signal to digital signal, the rest of the work such as down sampling and demodulation will be completed using software.

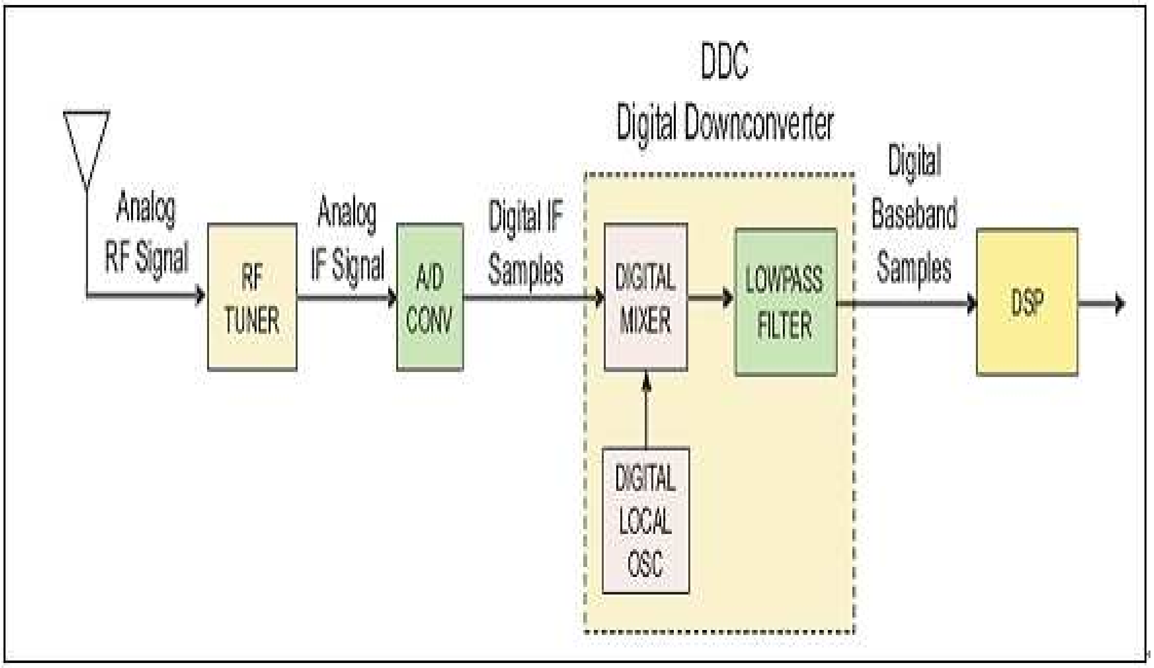


Figure 9: SDR receiver block diagram [12]

CHAPTER III

MULTI-RATE MARGIN IN 802.11

3.1 Overview of Multi-rate Margin

The concept of multi-rate margin arises from the adaptive coding and modulation technique introduced in the first chapter. Figure 10 shows another application of ACM. The required area differs at different rate. This is called multi-rate margin. This thesis is to show it via analysis, simulation and experimentation. When choosing the modulation types in ACM, the QPSK is always chosen for noisy channels and 16QAM is chosen for clear channels because the former is more robust against noise and interference but has lower transmission bit rate compared with the later one. The reason for the above is that different modulation technique has different BER performance and different bandwidth efficiency, both of which are actually contradictory to each other. Improving BER performance will definitely deteriorate its bandwidth efficiency. For the same BER value, the SNR requirement of 16QAM is higher than that of QPSK.

When the propagation path model is decided, the receiver sensitivity, defined as the minimum input signal (S_{min}) required to produce a specified output signal that can ensure the system BER satisfies the application requirement, can be converted to

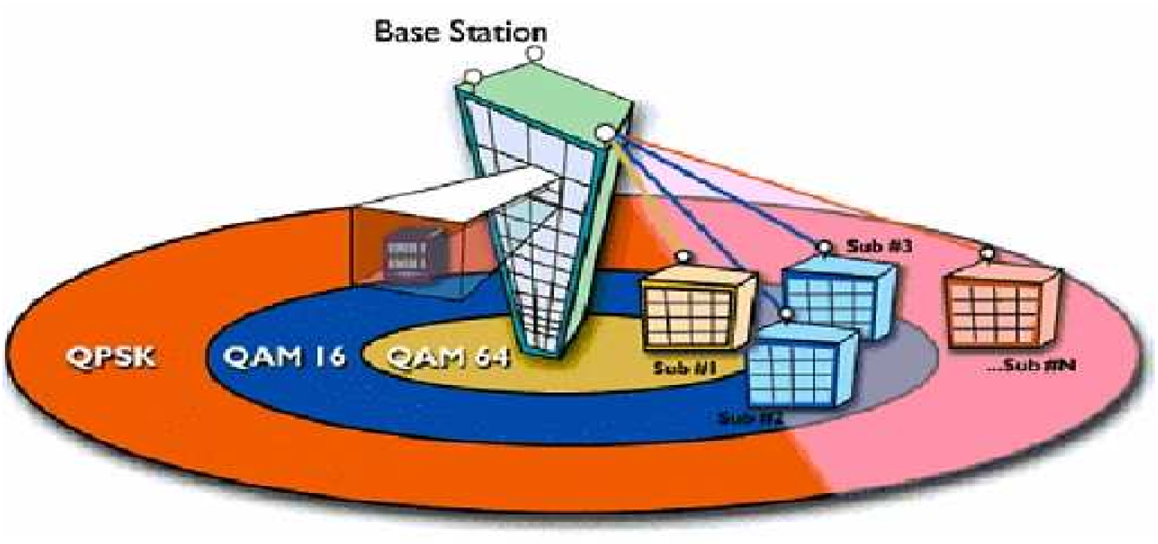


Figure 10: One application of ACM [6]

transmission range(communication range). As the higher the transmission rate, the higher the receiver sensitivity in dBm, the transmission range of lower transmission rate radio is longer than that of higher transmission rate radio. On the other hand, provided that the maximum interference signal strength was derived from receiver sensitivity and SIR, where SIR is taken as the equivalent of SINR, the maximum interference signal strength can be converted to the minimum distance between interferer and receiver. As is known, CSMA is implemented in 802.11 MAC layer, for CSMA to work, certain spatial area around the transmitter is protected via carrier sensing. The carrier sensing range is defined as the sum of interference range and transmission range. The differences among the carrier sensing ranges for different rate radios are called multi-rate margin. This thesis is targeted to show that there exists multi-rate margin via analysis, simulation and real world experimentation.

3.2 Theoretical Analysis for Multi-rate Margin

The idea of observing multi-rate margin in multi-rate radio in fact stems from the phenomenon of different SINR requirements for multi-rate radio such as 802.11a standard which is shown in Table 4. As long as the path loss model is defined, the receiver sensitivity can be converted to communication distance and SINR can be used to calculate the interference range.

Steps to analyze the multi-rate margin are as follow: (i)Estimate the communication range (r_i) based on the receive sensitivity at different rates. (ii)Calculate the SIR requirement based on a certain BER value for multi-rate radios. (iii) Subtract receive sensitivity from the SIR requirement for target BER of 10^{-5} to estimate the maximum tolerable interference for multi-rate radios. (iv) translate the tolerable interference to the minimum RI (receiver to interferer) distance for multi-rate radios ($(1+\delta)r_i$) with the assumption that a path loss model is known. (v)Add the RI distance to the communication range to estimate the minimum TI (transmitter to interferer) distance at different rates ($(2+\delta)r_i$). (vi) Again, translate the TI distance to the required carrier sense threshold at different rates based on the transmit power and path loss model. (vii) Finally, multi-rate margin is the difference between the carrier sense threshold at 1Mbps and the required carrier sense thresholds at high rates. Here characteristics of 802.11b multi-rate radio will be analyzed. Table 5 shows the results.

Table II: For BER equal to 10^{-5}, these are defined SNR and receiver sensitivity to meet BER requirement[25]

Rates(Mbps)	SINR(dB)	Receiver Sensitivity(dBm)
54	24.56	-65
48	24.05	-66
36	18.80	-70
24	17.04	-74
18	10.79	-77
12	9.03	-79
9	7.78	-81
6	6.02	-82

There are three arguments that need to be explained in this Table: receiver sensitivity, SIR requirement, carrier sense threshold (defer threshold).

First, the receiver sensitivity is defined as the minimum input signal (S_{min}) required to produce a specified output signal having a certain signal-to-noise (S/N)

Table III: Characteristics of an 802.11b multi-rate radio. (Transmit power: 15 dBm, indoor radio propagation model with path loss exponent of 3.3 [30].)

Data rate (Mbps)	1	2	5.5	11
Receive sensitivity (dBm)	-94	-91	-87	-82
Range (m)	272	221	167	118
SIR requirement (dB)	2.2	5.2	4.4	7.6
Max. interference (dBm)	-96.2	-96.2	-91.4	-89.6
Min. RI distance (m)	317	317	227	200
TI distance (m)	589	538	394	318
Defer threshold (dBm)	-105.1	-103.8	-99.3	-96.2

ratio to ensure that the BER achieves the systems requirement [37]. Receiver sensitivity indicates how faint an RF signal can be received by the receiver successfully. Indoor path loss model [30] has been used to derive the communication range, i.e. path loss = 40.2 + 20log10(d) if $d < 8$m, and 58.5 + 33log10(d/8), otherwise, which is exactly the framework for step i. Based on the receiver sensitivity, transmission power and path loss model, the communication range can be easily derived.

Second, SINR means how strong the signal is to overshadow the influence of noise and interference in order to achieve a satisfactory BER. A higher-rate communication requires a higher threshold, which means that it is more vulnerable to interference. The SINR requirement at four data rates of 802.11b radio is based on the study in [30]. Figure 11 shows the BER curve for four different data rates. Because the capacity of networks more depends on the interference than noise, the SINR will be replaced by SIR as in [30]. With the BER versus SIR equations in [30] for 802.11b standard, the required SIR for 1, 2, 5.5 and 11Mbps transmission is 2.2, 5.2, 4.4, and 7.6 respectively if the required BER is 10^{-5}.

Carrier sense threshold or defer threshold is defined as a signal strength level such that if the detected signal strength is above that specified level, any transmitter should defer its transmission. The carrier sensing range is an equivalent of the defer range explained in the Figure 12, which comes from the Energy Detect (ED) or defer threshold specified in the 802.11 PHY Clear Channel Assessment (CCA) [38]. With CCA implemented in all of the nodes, the interferer will declare the medium to be

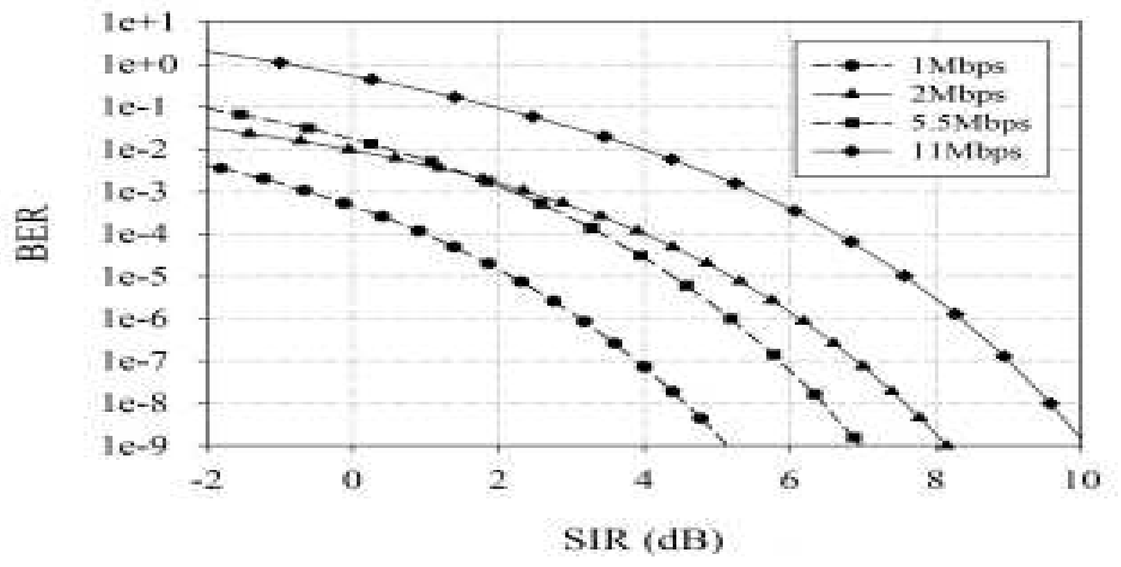


Figure 11: BER versus SIR in 802.11b[30]

busy if the signal strength detected at the interferer is greater than the defer threshold, so the signal transmission from interferer will be inhibited. Below will show how defer threshold is estimated at different data rates of an 802.11b radio.

Assume that the signal strength at the receiver is equal to the receiver sensitivity in Table 3, maximum tolerable interference to meet the SIR requirement can be derived based on that assumption (step (iii)). Assume every node transmits with the same power, using the path loss model mentioned above; the interference range would be easily obtained. Because the receiver does not transmit signal and does not need to sense the media, the collision safe distance can only be assured by sensing the carrier signal from the transmitter. In other words, collisions are avoided when there is no other simultaneous transmitter within 589m ($(2+\delta)r_i$) from an 1 Mbps transmitter, which is obtained by adding the communication range (r1) to the RI distance ($(1+\delta)r_i$) [39] (step (v)). Similarly, nodes within 318m ($(2+\delta)r_i$) from an 11 Mbps transmitter should defer. Again, Figure 12 shows this defer range (DF) at 1 and 11Mbps, respectively.

In single-hop WLAN, the defer threshold is preferably set to be as low as possible to prohibit as many interfering communication attempts as possible and

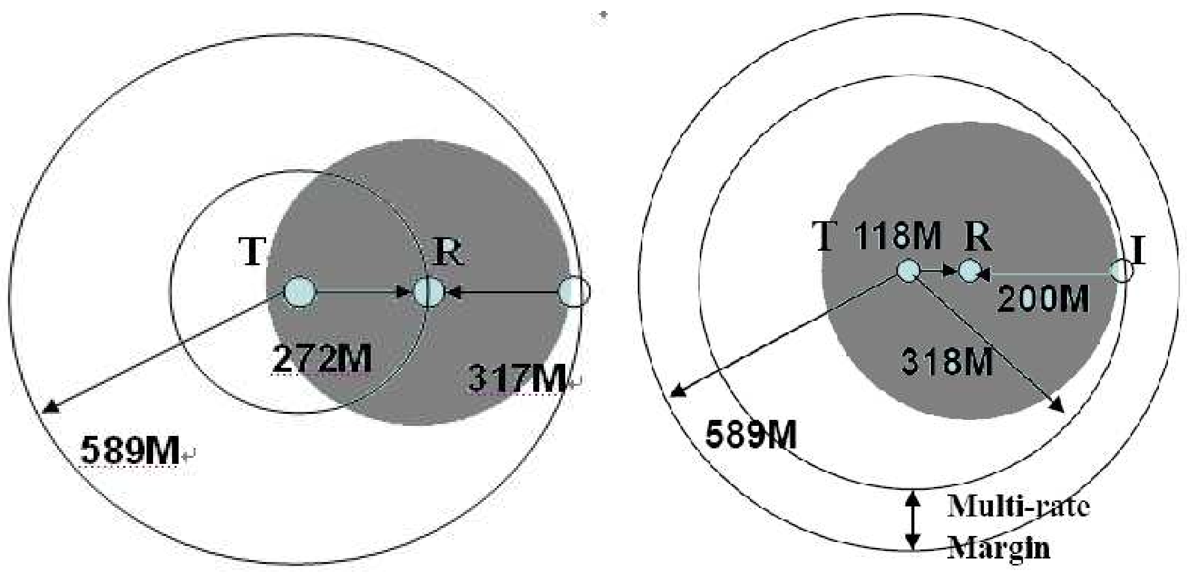


Figure 12: Interference and the required defer range (the required defer range in the right diagram is 271 meters smaller than that in the left but as the same defer range is used in the network, the extra space at 11Mbps mode would be utilized in MTOP.

thus protect the ongoing communication. However, the defer threshold in multi-hop environment has to be chosen with great carefulness because a low defer threshold will disallow more concurrent communications than necessary and thus degrades the spatial use efficiency, while a high defer threshold will induce collisions. However, as shown in Table 3, transmissions at different rates will require different defer thresholds but node cannot change its defer threshold dynamically in practice. Moreover, nodes do not always know the data rate of the signal they receive or overhear. So, to ensure there is no severe collision imposed by the interferer no matter which transmission rate the transmitter is using, the smallest defer threshold should be used in the multi-hop environment [39].

As can be seen from Table 3, the difference between the defer thresholds for 1Mbps and 11Mbps is 8.9dB, which is called multi-rate margin in this thesis, leaving a room for network improvement. The MTOP protocol exploits this margin by allowing a frame to be relayed through 1-2 more hops with a single media access, which will be discussed in details in chapter 5.

3.3 Simulation for Multi-rate Margin

In this section, the interference issue of the physical layer of the IEEE 802.11b standard will be discussed. The physical layer system models for 1, 2, 5.5, and 11 Mbps modes of the IEEE 802.11b standard are described in this section. The transmitter and receiver will be operating in 1, 2, 5.5 and 11Mbps modes and in each mode there will be four different kinds of interferers operating on 1, 2, 5.5 and 11Mbps modes respectively. The interference problem among different operating modes in IEEE 802.11b standard will be analyzed with the help of MATLAB simulation. In the MATLAB simulation, the desired transmitted signal and interference signal are assumed to be synchronized all the time, which means there is neither timing nor frequency offset between received signal and interference signal.

In IEEE 802.11b standard, the first rate is achieved by using differential BPSK (DBPSK) with Direct-Sequence Spread Spectrum (DSSS) and an 11 chip Barker code; the chip rate is 11 M chip/s. The last rate is obtained using Complementary Code Keying (CCK), also at 11 M chip/s. The construction of the communication models used for investigating the interference problem among different modes in 802.11b follows the similar structure that is to investigate the interference issue between 802.15.2 standard and 802.11b standard in[30]. The communications system model for the 1 M bps bit rate is presented in Figure 13, again consisting of the transmitter, the channel, the receiver and the IEEE 802.11b interference source. The details of this model are explained in 3.3.1 through 3.3.4. The CCK system is shown in Figure 17 and discussed in 3.3.3.

3.3.1 802.11b 1Mbps DSSS Model

This system utilizes a spread spectrum scheme to mitigate the effect of interference. The Barker sequence with code length P = 11 is employed to spread the signal. The bit duration, T, is exactly 11 chip periods, T_c long. The processing gain

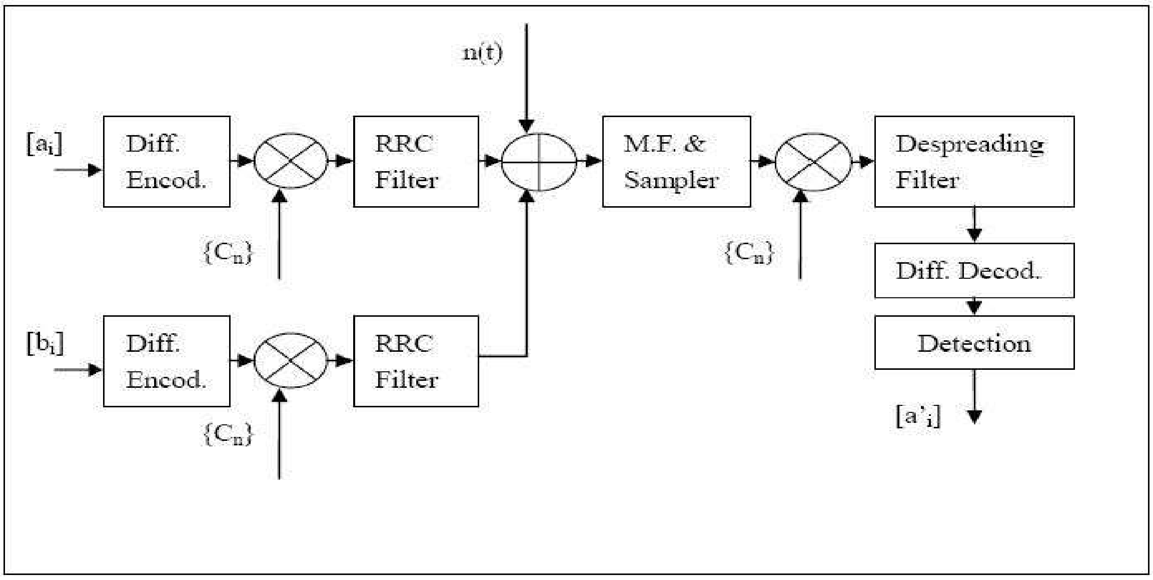


Figure 13: 802.11b DSSS 1Mbps with another 1Mbps interferer system model

(PG) of this system is [27]

PG = R_c/R_b = 11,Where R_b = 1/T is the bit rate, and R_c =1/T_c is the chip rate.

The structure of the communication system is strictly following the specifications of the physical layer in 802.11b standard. As shown in Figure 29, the input data bits are first differentially encoded. The resulting sequence is spread by the Barker code. The output of the spreader is fed to a square-root raised-cosine (RRC) pulse-shaping filter. The impulse response of this filter and how to choose the roll-off factor may be found in Lee[28]. At the receiver, the input samples are first fed into the square-root raised-cosine matched filter. The despreading filter is a rectangular filter that integrates the output of the multiplier during a bit period. The differential decoder calculates the difference between the phase angle of the received symbol and that of the previous one to generate the output bit stream. It is assumed that the carrier frequency of the local oscillator at the receiver is synchronized to that of the transmitter and chip timing of the receiver is synchronized to the transmitter. Because the interference signal transmitter works in the same frequency range and under the same 802.11b standard as the desired signal transmitter, it is assumed that there is no frequency and phase difference between desired signal and interference signal and timing difference between desired signal and interference signal is also ignored. The transmitters structure of the 1Mbps interference signal is exactly the same as that of the desired signals transmitter except that the input bit stream bi is generated randomly, which means the interference bit sequence is independent with the desired transmitted bit sequence. According to[1], the Barker spreading code in the interference signal transmitter is also in the same format as that of desired signal transmitter.

Figure 14 shows the simulation result for the physical layer interference problem between 1Mbps mode and other 3 transmission modes. The communication systems structures when 1Mbps is interfered by other 3 modes have not been plotted out here because the interference blocks of other 3 modes can be easily found from other sections discussing the interference problems of 2Mbps, 5.5Mbps and 11Mbps.

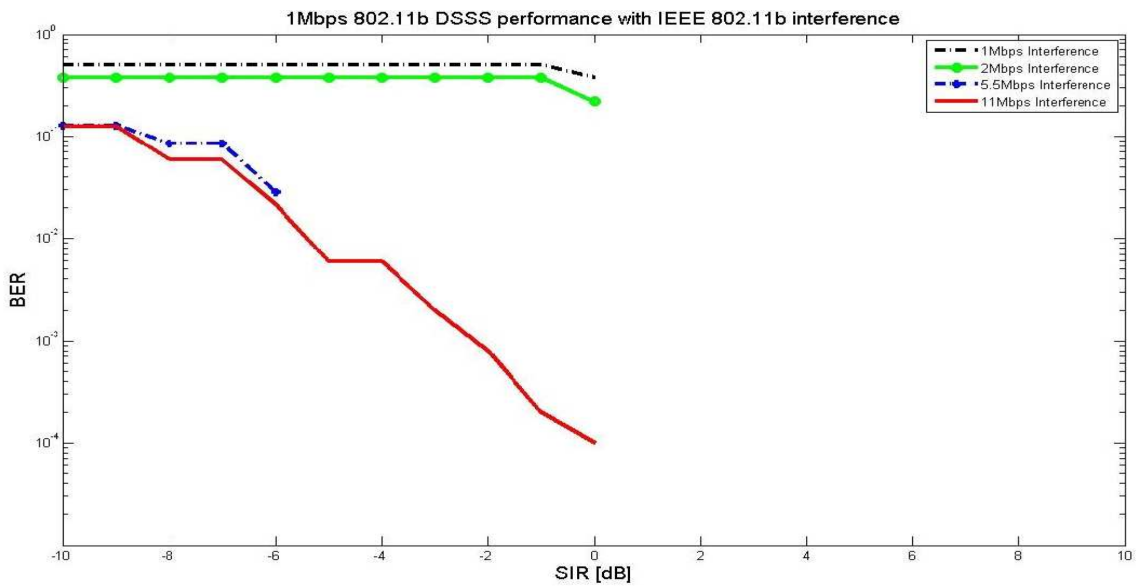


Figure 14: 1Mbps 802.11b performance with IEEE 802.11b interference. AWGN channel. SNR = 35dB

3.3.2 802.11b 2Mbps DSSS Model

The system model for 2Mbps mode is just the duplicate of that for 1Mbps mode except that Gray code and QPSK modulation schemes are used for 2Mbps mode. Figure 15 shows the system model for 2Mbps mode with another 2Mbps mode interfering the desired communication. Figure 16 shows the simulation result for the physical layer interference problem between 2Mbps mode and other 3 transmission modes.

With the only difference between the construction of interferers transmitter and that of the desired signals transmitter being the transmitted bit sequence, the around 50% BER when SIR is below 0dB is expectable, the formula derivation of which will be given below.

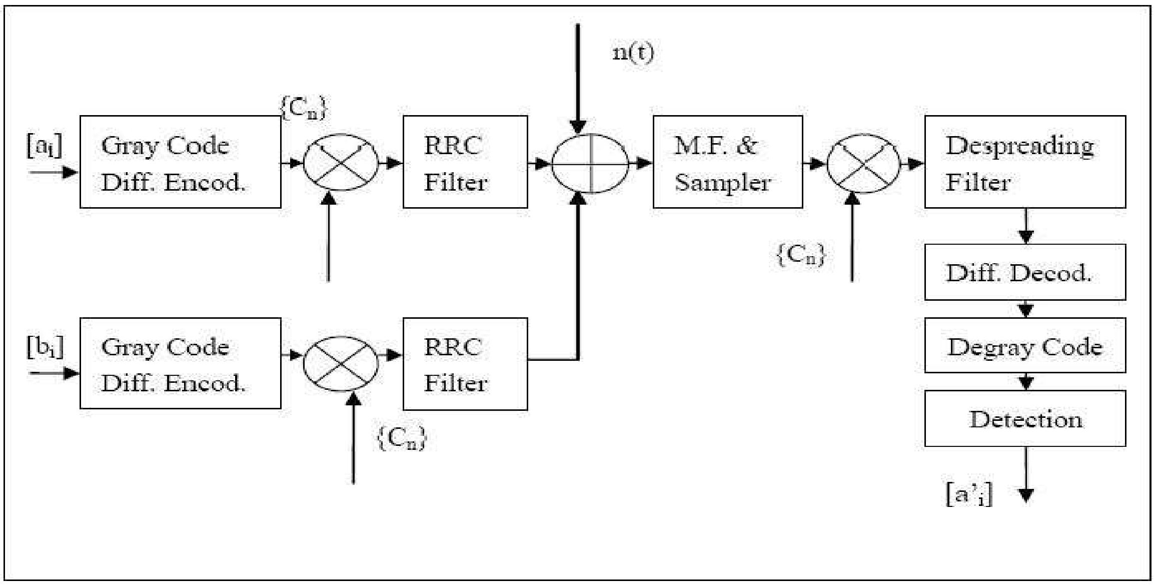


Figure 15: 802.11b DSSS 2Mbps with another 2Mbps interferer system model

3.3.3 802.11b 11Mbps CCK Model

Because both 5.5Mbps mode and 11Mbps mode employ the CCK modulation method and the way the dibits are mapped to phase $\phi 1$ in 11Mbps mode is easier than that in 5.5 Mbps mode, 11Mbps mode is analyzed here before analyzing the 5.5Mbps mode.

For the CCK (complementary codes keying) modulation modes, the spreading code length is 8, which means one symbol consists of exactly 8 complex chips with chipping rate 11M chips per second. The equation 3.1 will be used to derive the CCK

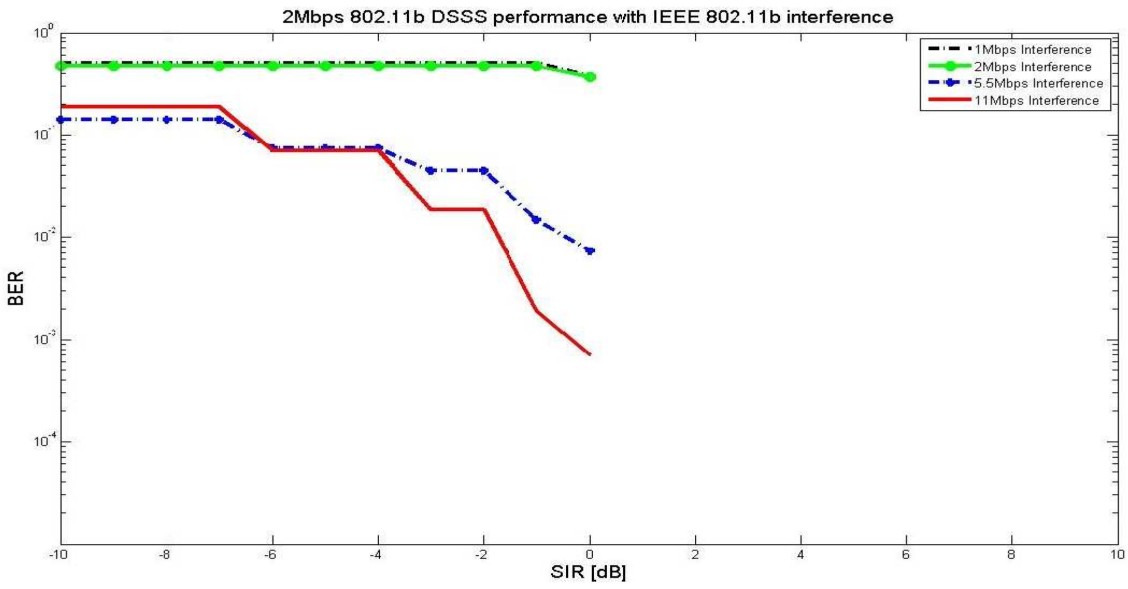


Figure 16: 2Mbps 802.11b performance with IEEE 802.11b interference. AWGN channel. SNR = 35dB

code words for both 5.5Mbps mode and 11Mbps mode.

$$C = \{e^{j(\phi 1+\phi 2+\phi 3+\phi 4)}, e^{j(\phi 1+\phi 3+\phi 4)}, e^{j(\phi 1+\phi 2+\phi 4)}, -e^{j(\phi 1+\phi 4)}, e^{j(\phi 1+\phi 2+\phi 3)}, e^{j(\phi 1+\phi 3)}, -e^{j(\phi 1+\phi 2)}, e^{j(\phi 1)}\} \tag{3.1}$$

where C is the code word, C = {c0 to c7}, $\phi 1$, $\phi 2$, $\phi 3$ and $\phi 4$ are encoded by the input bit sequence based on DBPSK and DQPSK.

Let us take 11Mbps mode as an example. In 11Mbps mode, the input is supposed to be 8 bits sequence. After this CCK encoder, the output will be turned into 8 chips code word based on the equation 3.2 [3].

The 11Mbps mode utilizes CCK as its modulation and demodulation method. The structure of the communication system in Figure 17 strictly followed the specifications of the physical layer in 802.11b standard. Below we will describe how the CCK encoder and CCK decoder function. Suppose 8 bits (d0 to d7; d0 first in time) are transmitted per symbol. The first dibit (d0, d1) encodes $\phi 1$ based on DQPSK, the scheme of which is specified in Table 1. The phase $\phi 1$ is changed according to the preceding symbol. In other words, the phase change for $\phi 1$ is relative to the phase of the preceding DQPSK symbol. All odd-numbered symbols are given an extra 180 degree (π) rotation, in accordance with the DQPSK modulation shown in Table 1[3]. Symbol number starts with 0 for the first 8 chips code word. For example, suppose the reference starting phase is 0, (d0, d1) is (0, 0) and (d8, d9) is (1, 0). Based on the Table 1, the first $\phi 1$ should be 0 and second $\phi 1$ should be $pi/2$ rather than -$\pi/2$

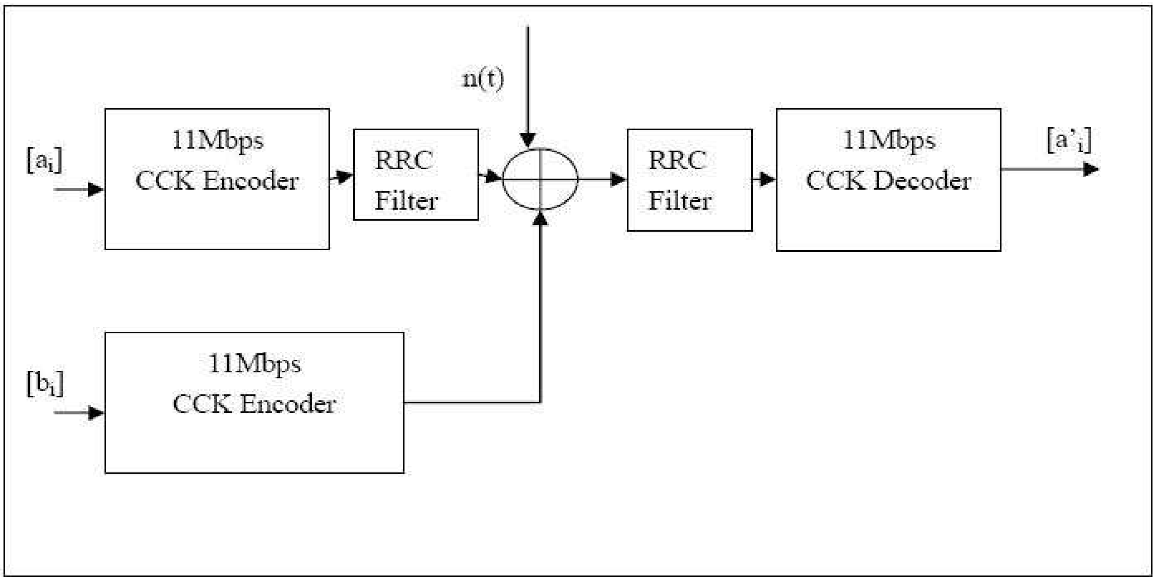


Figure 17: 802.11b 11Mbps with another 11Mbps interferer system model

The data dibits (d2, d3), (d4, d5) and (d6, d7) encode $\phi 2$, $\phi 3$ and $\phi 4$, respectively, based on QPSK encoding scheme, which is shown in Table 4. Gray code is not used in this coding scheme.

Table IV: QPSK encoding scheme[3]

Dibit Pattern (di, d(i+1))(di is first in time)	Phase
00	0
01	$\pi/2$
10	π
11	$3\pi/2$

Rather than implementing the complex optimum maximum likelihood decoder, a less complex sub-optimum algorithms is used for the decoder implementation. The equations proposed by Van Nee[29] are used for our decoding technique.

$$\begin{aligned}
\phi 2 &= arg\{r1r2^* + r3r4^* + r5r6^* + r7r8^*\} \\
\phi 3 &= arg\{r1r3^* + r2r4^* + r5r7^* + r6r8^*\} \\
\phi 4 &= arg\{r1r5^* + r2r6^* + r3r7^* + r4r8^*\} \\
\phi 1 &= arg\{r4e^{-j(\phi 4)} + r6e^{-j(\phi 3)} + r7e^{-j(\phi 2)} + r8\}
\end{aligned} \tag{3.2}$$

where r = [r1 r2 r3 r4 r5 r6 r7 r8] is the received symbol.

After taking the above CCK decoding method, reverse DQPSK and QPSK procedures are applied to the decoded $\phi 1$, $\phi 2$, $\phi 3$ and $\phi 4$ to find out the final decoded bits. The transmitters construction of the 11Mbps interference signal is exactly the same as that of the desired signals transmitter except that the input bit stream bi is generated randomly, which means the interference bit sequence is independent with the desired transmitted bit sequence.

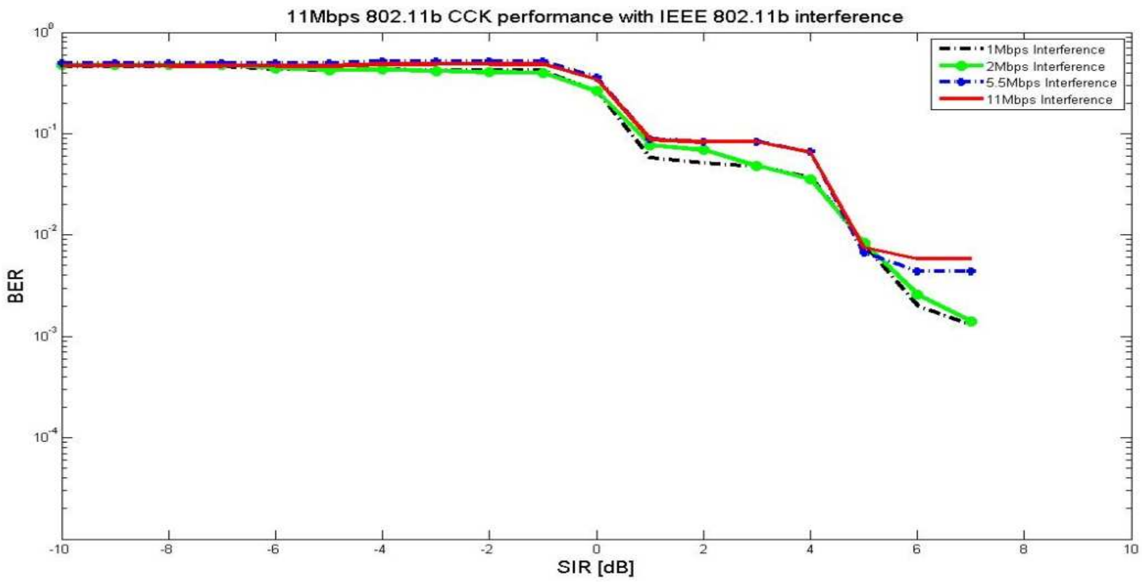


Figure 18: 11Mbps 802.11b performance with IEEE 802.11b interference. AWGN channel. SNR = 35dB

3.3.4 802.11b 5.5Mbps CCK Model

In 5.5Mbps CCK, only 4 bits are supposed to encode $\phi 1$, $\phi 2$, $\phi 3$ and $\phi 4$. The first dibit (d0, d1) is used to encode $\phi 1$ based on DQPSK as in 11Mbps mode while the dibit (d2, d3) encode the $\phi 2$, $\phi 3$ and $\phi 4$ by setting $\phi 2$ = (d2*π) +$\pi/2$, $\phi 3 = 0$

and $\phi4$ = d3 $*\pi$. The encoded $\phi1$, $\phi2$, $\phi3$ and $\phi4$ will then be used to form an 8-chip CCK symbol based on the equation 6.1.

The decoding method for $\phi2$, $\phi3$ and $\phi4$ is exactly the same as that in 11Mbps, while the decoding method for $\phi1$ is a little bit different, which decodes $\phi1$ to be $\arg\{(r4 + r2)e^{-j(\phi4)} + (r7 + r5)e^{-j(\phi2)} + r6 + r8\}$. In the MATLAB simulation for 5.5Mbps and 11Mbps modes, this sub-optimally coherent receiver to decode the received phase is used. Figuer 19 shows the flow graph of the communication system for 5.5Mbps with another 5.5Mbps interferer, while Figuer 20 presents the BER performance.

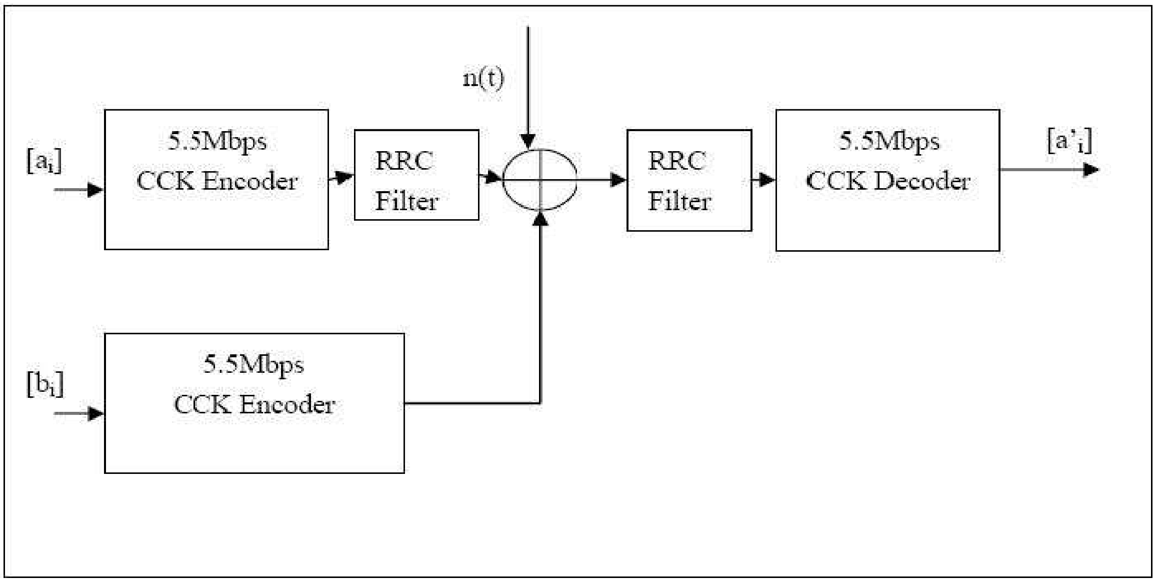


Figure 19: 802.11b 5.5Mbps with another 5.5Mbps interferer system model

3.3.5 Simulation Results

Because the interference signal is assumed to be the AWGN signal in the theoretical analysis of Chapter 3 and the interference signal is assumed to be entirely synchronized with the receiver in terms of timing and carrier frequency in the previous 802.11 b self-interference simulation analysis, the simulation result cannot be exactly the same as the result derived from the theoretical analysis. However, the similarity of the SIR requirements in theoretical analysis and simulation analysis when BER

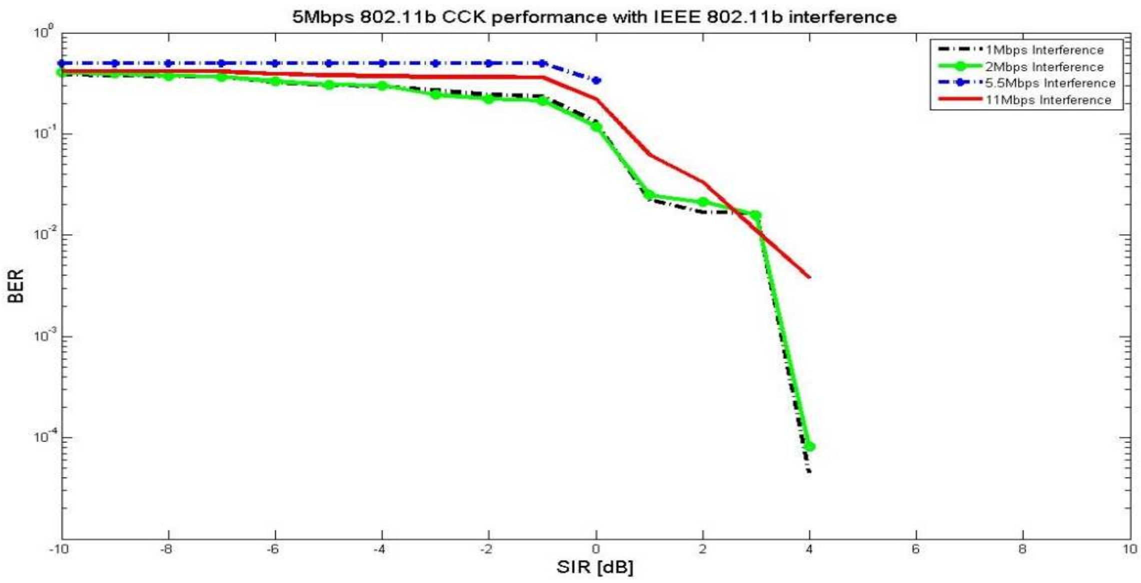


Figure 20: 5.5Mbps 802.11b performance with IEEE 802.11b interference. AWGN channel. SNR = 35dB

is equal to 10^{-5} can be seen from below. From last 4 figures of simulation, it can be observed that the SIR requirement for 1Mbps, 2Mbps, 5.5Mbps and 11Mbps is 0, 0, 4,and 7dB respectively, when BER is required to reach 10^{-5}. Table 5 shows the multi-rate margin analysis based on our simulation result for 802.11 b radios. It can be seen that there is still an 8.1 dB margin between 1Mbps and 11Mbps and two relay nodes could be inserted for 11Mbps scenario if the defer threshold of 1Mbps were used as the defer threshold for both the 1Mbps and 11Mbps system.

Table V: Simulation analysis of the Interference Issues in an 802.11 b multi-rate radio. (Transmit power: 15 dBm, indoor radio propagation model with path loss exponent of 3.3 [30].)

Data rate (Mbps)	1	2	5.5	11
Receive sensitivity (dBm)	-94	-91	-87	-82
Range (m)	272	221	167	118
SIR requirement (dB)	0	0	4	7
Max. interference (dBm)	-94	-91	-91	-89
Min. RI distance (m)	272	221	221	191
TI distance (m)	544	442	388	309
Defer threshold (dBm)	-103.97	-101	-99.13	-95.87

CHAPTER IV

MULTI-RATE MARGIN IN GNU RADIO/USRP-BASED TESTBED

4.1 GNU Radio and USRP

4.1.1 GNU Radio Basics

GNU Radio [26] is an open source software toolkit of signal processing for building up our own software defined radios. It provides functions to define the transmitted waveforms, to demodulate the received signals and gives us a collection of modulators and demodulators that keep on growing. The type of modulation techniques can be easily modified by changing the argument of a certain modulation option in the command line[13].

In GNU Radio, the radio chain resembles a network where the nodes are signal processing blocks and lines represent the data flow. Each signal processing block is written in C++ and Python is used to glue those signal processing blocks together to create your own radio. The way GNU Radio components are linked is shown in Figure 21.

With RF part and ADC/DAC, the analog signal can be received and trans-

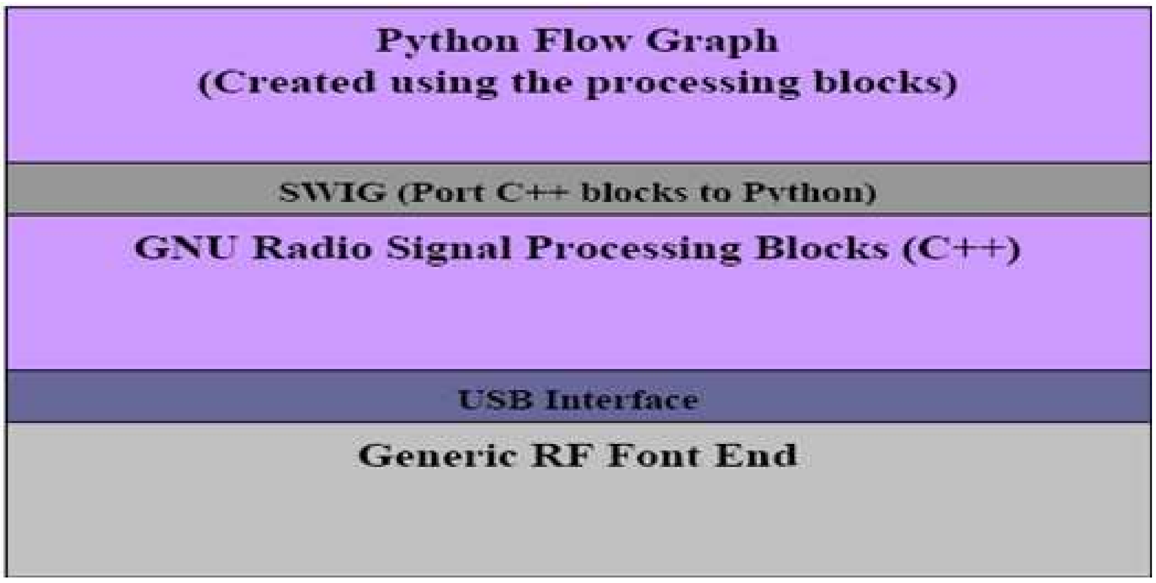

Python Flow Graph
(Created using the processing blocks)
SWIG (Port C++ blocks to Python)
GNU Radio Signal Processing Blocks (C++)
USB Interface
Generic RF Font End

Figure 21: Block Diagram of GNU Radio Components[14]

formed to digital signal, it is necessary to spend some time explaining the role of RF part. Generally speaking, the role of RF part is to translate the signal centered at the high frequency down to a signal centered at the intermediate frequency without distorting the shape of the waveform. As an example, a cable modem tuner module translates a 6 MHz chunk of the spectrum centered between about 50 MHz and 800 MHz down to an output range centered at 5.75 MHz. A typical structure of RF front end after breaking it into small parts is shown below in Figure 22:

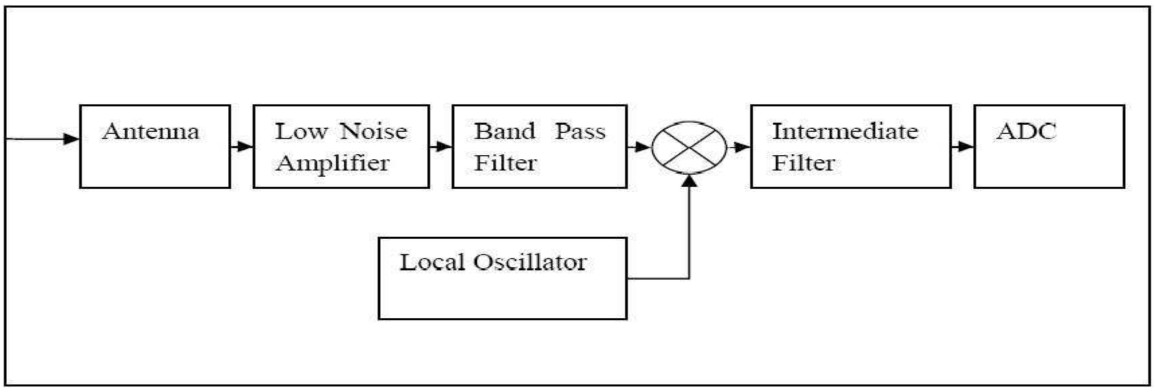


Figure 22: A typical structure of RF front end

The Low Noise Amplifier (LNA) and the Band Pass Filter (BPF) are used to select the bandwidth of interest and amplify the signal. For example, in order to receive the FM stations, you may like to use an LNA and an BPF with a cutoff frequency of 120 MHz[13]. After filtering the noise out of our interested bandwidth, we use local oscillator whose frequency is locked at RF-IF to downconvert the RF band signal to IF band. At the output of the mixer, we get two chunks of spectrum centered at IF and 2 * RF - IF respectively, the later of which would be filtered out through an intermediate filter following the mixer. After this whole process, the ADC is able to process the signal locating at IF band. The reason that we do not connect ADC to antenna to directly process the signal is that based on our current technique, ADC is not fast enough to catch up with 500MHz and even 5GHz for RFX2400 daugterboards. Even if the ADC that could keep up with that fast speed were produced, the power consumed would be very high and the cost for the whole circuit would be much higher than the circuit consisting of RF front end.

4.1.2 USRP

USRP Characteristics

USRP is the hardware component that supports GNU Radio. It loads the code created by software installed on your computer and realizes the commands embedded in the code with a cable connecting your computer to the USB2.0 port on the USRP box. Typically, the USRP consists of a motherboard containing up to four 12-bit, 64M sample/sec ADCs, four 14-bit, 128M sample/sec DACs, a Field Programmable Gate Array (FPGA) and a programmable USB 2.0 controller. Each fully populated USRP motherboard supports four daughterboards, two for receiving and two for transmitting. RF front ends are implemented on the daughterboards. A general setup of the USRP board is shown in Figure 23: For the detailed specifications of the 3 important components, AD/DA converters, daughter boards and the FPGA on the USRP board, interested readers could refer to the reference[15].

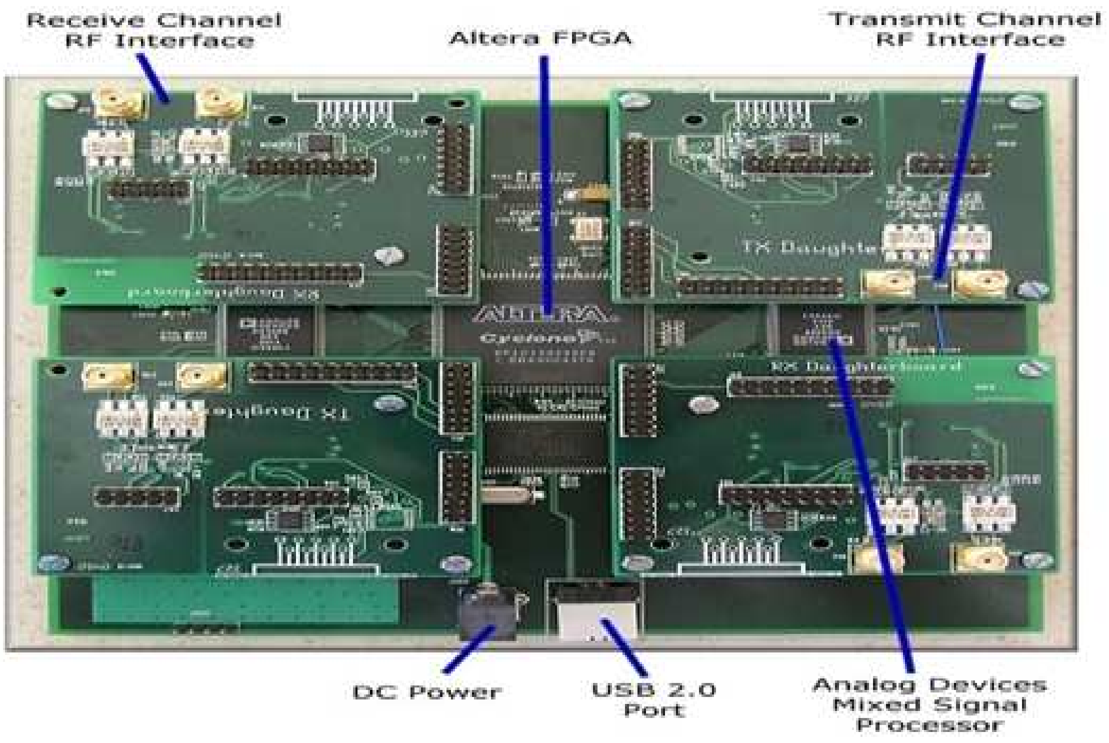

Receive Channel RF Interface
Altera FPGA
Transmit Channel RF Interface
DC Power
USB 2.0 Port
Analog Devices Mixed Signal Processor

Figure 23: USRP Board (One USRP motherboard can support up to 4 daughtboards and there are an FPGA and two AD/DA converters on one motherboard)

A/D and D/A Converters

Because the 4 high-speed 12-bit AD converters can sample at a rate of 64M samples per second, based on Nyquist Criterion it could digitize a band as wide as 32MHz. For the AD/DA converters, some readers may argue that we could also sample the signal whose center frequency is above 32MHz without the RF frond end as long as its single bandwidth is no greater than 32MHz because the chunk of our interested signal would be left and right shifted along the frequency domain by 64MHz when that signal is sampled at a rate of 64M samples per second in the time domain. So, the chunk of our interested signal must be mapped to some places between -32MHz and 32MHz. After synchronization at the FPGA part, the center frequency of this chunk of signal could be tracked, leading precise demodulation at the receiver side. But it would not be an ideal case if we considered the fact that the higher the frequency of the sampled signal, the more the SNR will be degraded by jitter.

There is a programmable gain amplifier (PGA) before the ADCs to amplify the input signal in order to utilize the entire input range of the ADCs in case the signal is too weak. The gain of the PGA can reach up to 20dB.

Daughter Boards

As can be seen from Figure 11, the four slots on the motherboard can be used to plug in up to 2 RX daughter boards and 2 TX daughter boards. The daughter boards are aimed to hold the RF transmitter and receiver interface.

The daughter board is chosen as RFX2400. The reason to choose this daughter board is that on one hand, the built-in file in GNU Radio we used for our implementation works fine with the RFX2400 daughter board, and on the other hand we have enough RFX2400 available in our lab. The useful features of RFX2400 board are listed below: 30 MHz transmitting and receiving bandwidth; Built-in T/R switching;Built-in analog RSSI measurement;Adjustable transmit power.

FPGA

For the FPGA part, the most important element we should understand is the DDC (digital down converter), the graph of which is shown in Figure 24. First, it down converts the signal from the IF band to the base band. Second, it decimates the signal so that the data rate can be adapted by the USB 2.0 and is reasonable for the computers' computing capability. The decimator can be treated as a low pass filter followed by a downsampler. Suppose the decimation factor is D. If we look at the digital spectrum, the low pass filter selects out the band [-π/D, π/D], and then the downsampler spread the spectrum in [-π/D, π/D] to [-π, π][2]. For example, the bandwidth of FM station is 200 kHz which is constrained by the tone of our human being. After setting the decimation rate to be 250, the data rate across the USB port will be 64MHz/250 = 256 kHz, which is well suited for the 200 kHz bandwidth, which means none of the spectral information will be lost. The maximum sustainable rate across USB is 32MB/sec. All samples sent over the USB interface are in 16-bit signed integers in IQ format, i.e. 16-bit I and 16-bit Q data (complex), which means 4 bytes need to be used to represent one complex sample, resulting in 8M complex samples/sec across the USB, which limits the maximum input spectral bandwidth to be no greater than 8MHz.

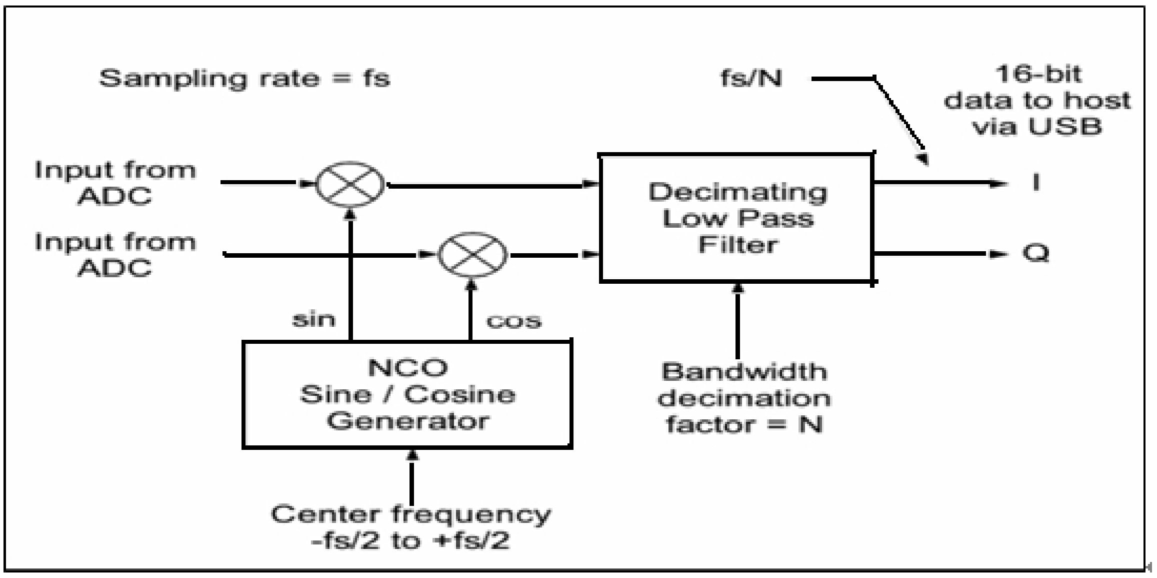

Sampling rate = fs
fs/N
16-bit data to host via USB
Input from ADC
Input from ADC
Decimating Low Pass Filter
I
Q
sin
cos
NCO Sine / Cosine Generator
Bandwidth decimation factor = N
Center frequency -fs/2 to +fs/2

Figure 24: Block Diagram of the Functional Components of FPGA[17]

4.2 Simulation for Multi-rate Margin

4.2.1 Design of Simulation Environment

DBPSK and DQPSK are selected in our experimentation as two different modulation schemes because only these two phase shift keying modulation schemes can be realized using GNU Radio and USRP.

Before adding USRP onto our canvass, the communication system should be simulated in GNU Radio to ensure that DBPSK and DQPSK modulation schemes have been implemented correctly. As is known, each signal processing block is written in C++ and Python file is supposed to connect these signal processing blocks together. Figure 25 shows the signal flow graph in my communication system. A python file has been created to provide users with a control panel where the systems parameters such as modulation mode, the number of transmitted bits, the signal to noise ratio etc. can be imported. This control panel is easy to use and after each run the bit error rate will be displayed in the terminal along with the corresponding signal to noise ratio. Figuer 26 presents a snapshot of the control panel and the python code is listed in A.

As is known, in order to explore the multi-rate margin in multi-rate radio, changing modulation scheme is not the unique method and coding scheme can also be utilized to observe the multi-rate margin. As in GNU Radio, because the implementation template of Reed Solomon code has already existed in the GNU Radio project, it is much easier to implement Reed Solomon code than to implement other coding schemes, Reed Solomon (RS) coding scheme is selected to be our error control coding method in our system. The control panel for RS is similar to that for inspecting modulations but the mydbpsk.py file should be substituted by mydbpsk_rs.py file (A).

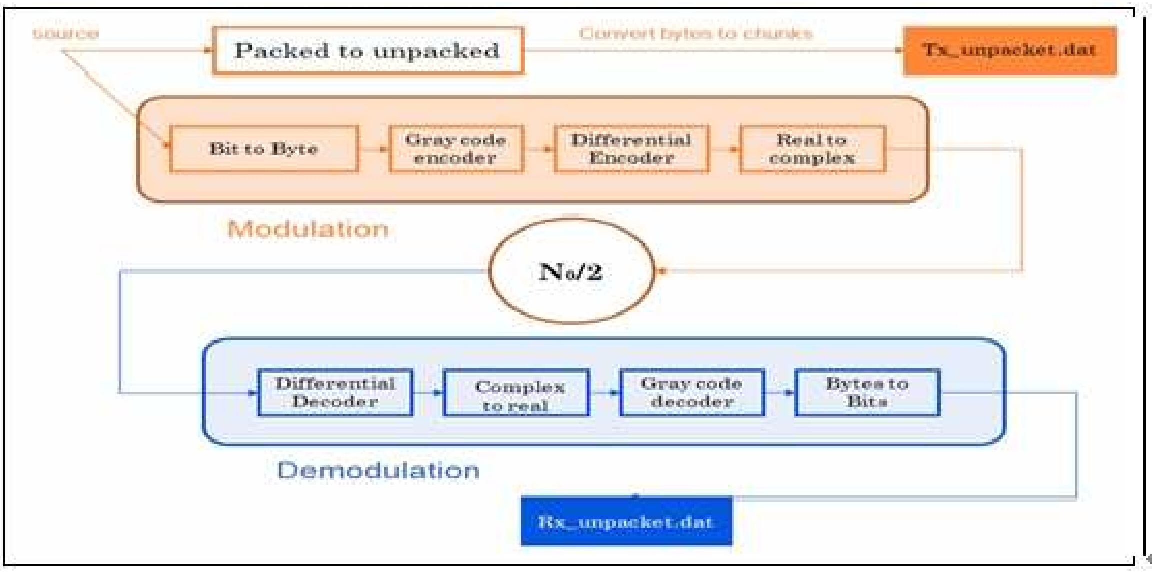


Figure 25: Flow Chart for Testing the BER Performance of our Communication System

The RS code in our system is shortened RS(207, 187), which is supposed to be able to correct 10 errors. Regarding the fact that 256 bytes long packet should be used to help with buffer alignment, some paddings are added to the tail of each packet.

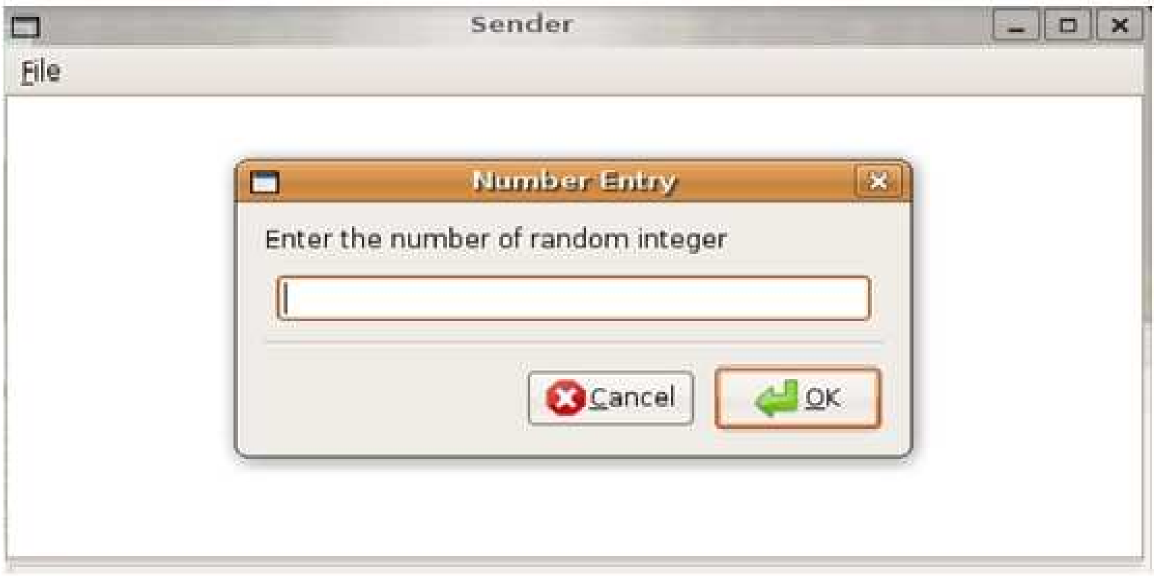


Figure 26: Control Panel to Ease the Simulation Procedure

Several lines of python code should be added into the DBPSK and DQPSK blocks to encode the bit streams before getting them modulated. In our practical transmission system, the preamble and access code, which are used for synchronization at the receiver, cannot be changed because they are the only pair that are supposed to be recognized by the receiver and at the receiver side, each received packet is first demodulated and then the preamble and header part are shredded off, so the coding and decoding schemes are only imposed on the payload part. There are tens of lines that have to be added in packet_utils.py file, which is the most crucial file transforming a piece of payload into a frame that can be transmitted over the air. A shows the modified lines in packet_utils.py.

In fact, another RS code with different coding rate has also been implemented in GNU Radio. In order to implement a new RS code, there are several C++ files that should be modified to fit the needs of new RS code's criterion, which are listed here: atsc.i, atsc_consts.h, atsci_reed_solomon.cc, atsc_rs_decoder.cc, atsc_rs_encoder.cc atsc_types, etc.[25].

The flow graph of our communication system with RS is quite similar to Figure 13 except that the RS encoding block should be appended before the bit-to-byte block and the corresponding RS decoding block should be post-appended after the bytes-to-bits block.

4.2.2 Simulation Results

DBPSK and DQPSK without Reed-Solomon

Before testing the multi-rate margin in the outside, the BER performance of our communication system was analyzed via simulation. The optimum receivers for DBPSK and DQPSK modulation schemes are used in GNU Radio. BER performance of the optimum DBPSK and DQPSK has been shown in Figure 27.

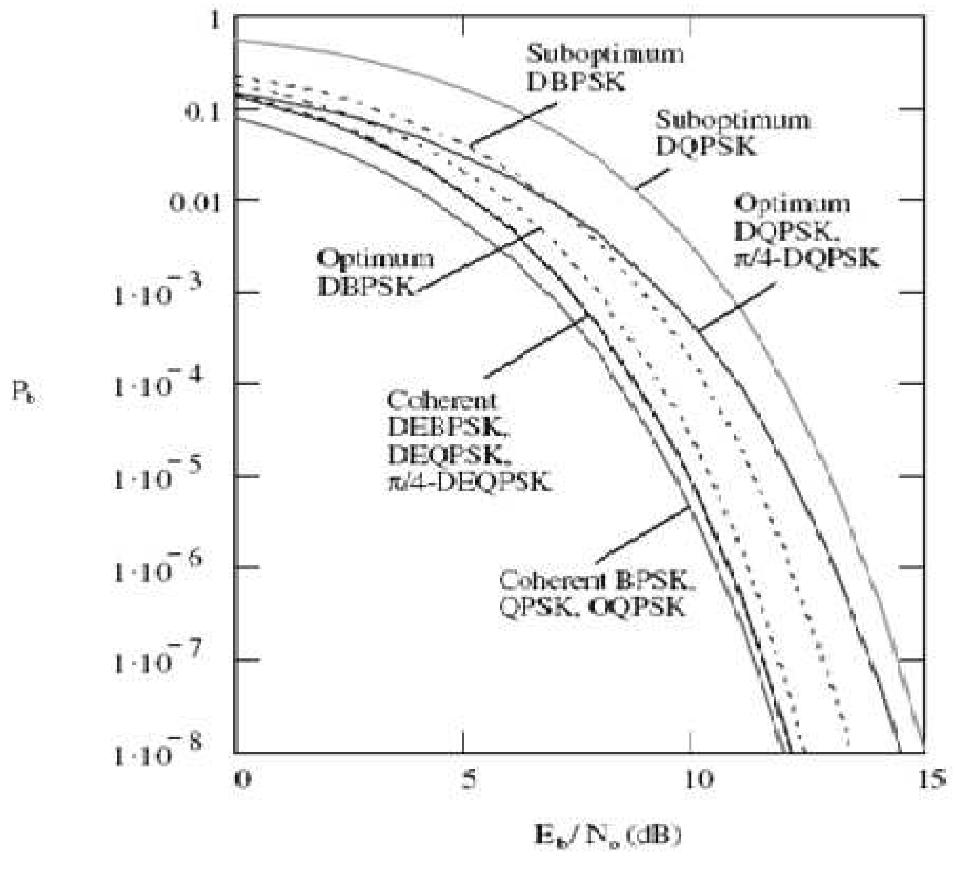


Figure 27: BER Performance for DBPSK and DQPSK [4]

The DBPSK and DQPSK without RS coding system have also been simulated and the same clear BER performance graph can be observed in Figuer 28. Because in outdoor testing, the parameters used to measure the degree of multi-rate margin are

Packet Delivery Ratio (PDR) and SNR, both of which can be easily converted from BER and EbNo respectively, Figuer 29 is plotted out to show the simulation curve of PDR versus SNR. As can be seen from Figure 29, 95% PDR is chosen to be the acceptable value and the corresponding SIR for DBPSK and DQPSK is 7.55dB and 11.55dB respectively.

Based on the receive sensitivity of 802.11 b radios discussed in Chapter 3 and the SIR values mentioned in the previous paragraph, we can easily see that the allowed maximum interference signal strength for DBPSK and DQPSK is -101.55dBm and -102.55dBm, indicating that the interference distance is 459 meters for DBPSK and 492 for DQPSK, respectively. The carrier sensing range for DBPSK and DQPSK would be 731 meters and 713 meters, which predicts that a multi-rate margin should emerge when real world transmission is engaged.

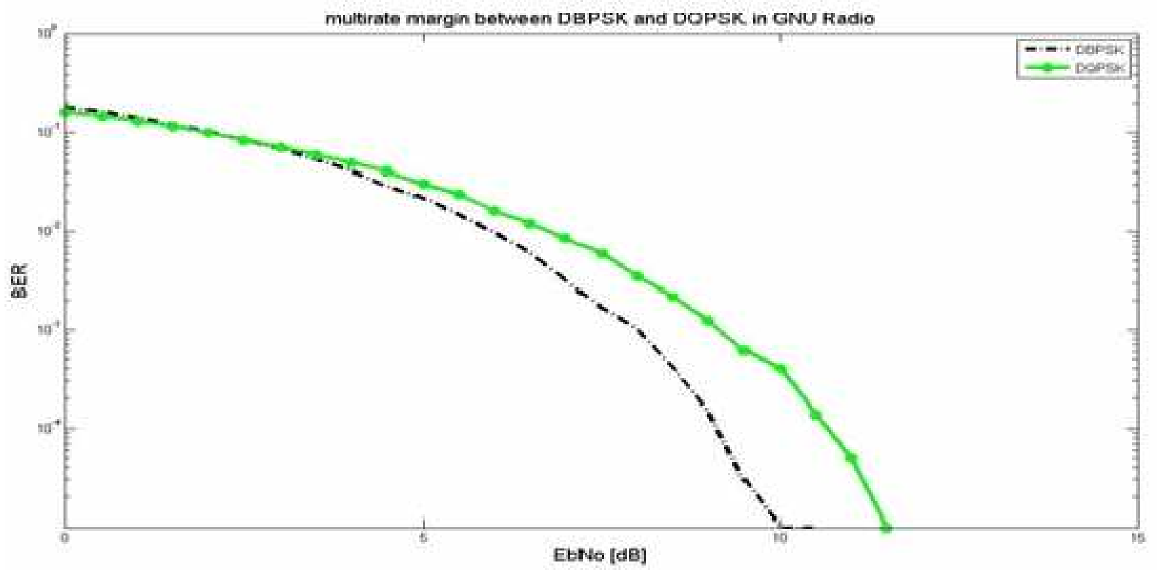


Figure 28: BER Simulation Result for DBPSK and DQPSK without RS code

DBPSK and DQPSK with Reed-Solomon

As can be seen from Figuer 30 and 31, there are still around 4dB multi-rate margin between DBPSK and DQPSK with reed Solomon coding scheme and both of the PDR and BER curves with RS coding scheme have been shifted to the left with regard to the PDR and BER curves without RS coding scheme, say around 3 dB for

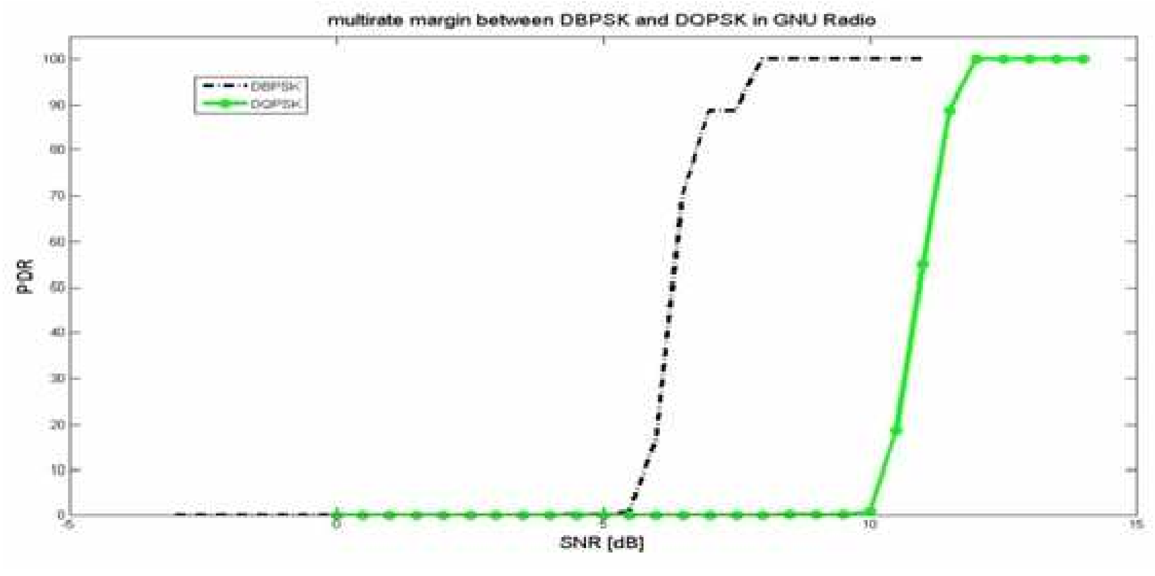


Figure 29: PDR Simulation Result for DBPSK and DQPSK without RS code

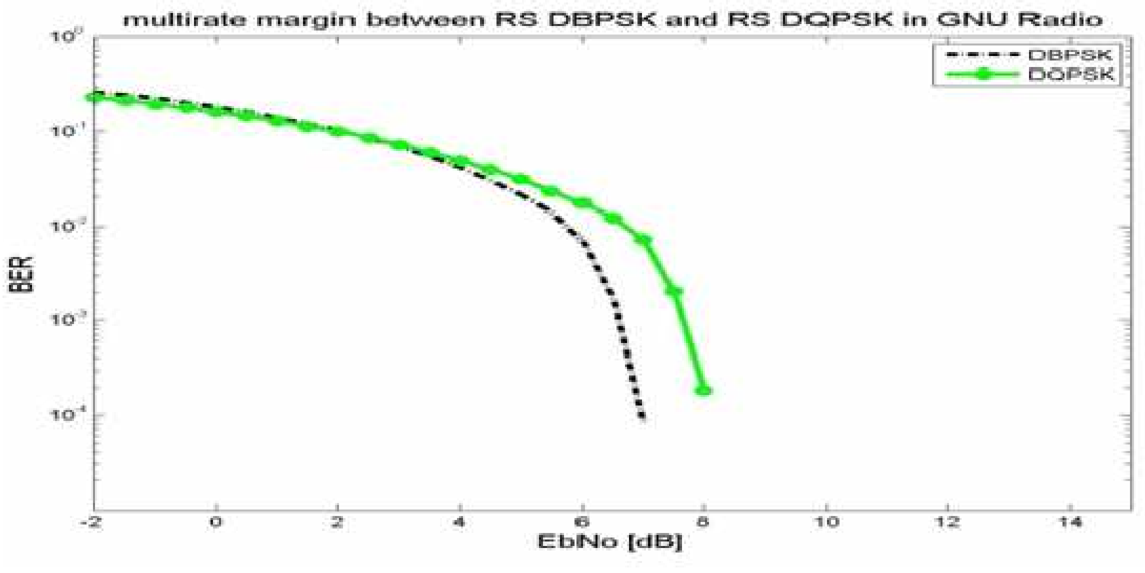


Figure 30: BER Simulation Result for DBPSK and DQPSK with RS code

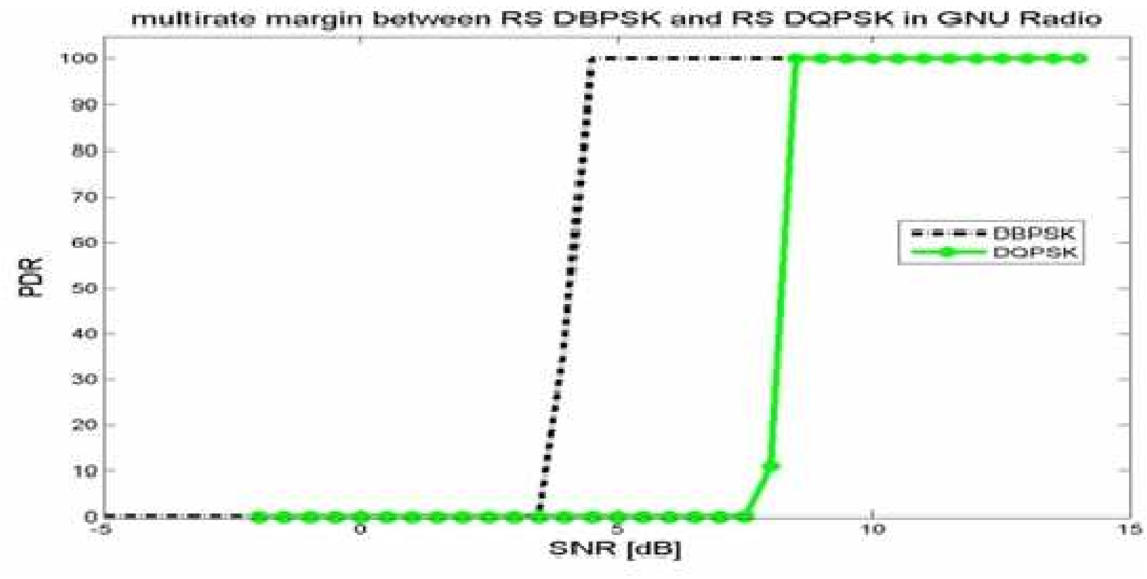


Figure 31: PDR Simulation Result for DBPSK and DQPSK with RS code

DQPSK modulation scheme. Using the same analysis procedure as previous section, 645 meters and 620 meters are estimated to be the carrier sensing range for DBPSK and DQPSK respectively. So, there would still be a 25 meters multi-rate margin if Reed Solomon coding were implemented in our system.

4.3 Experiment for Multi-rate Margin

4.3.1 Design and Implementation of Testbed

Design Procedure

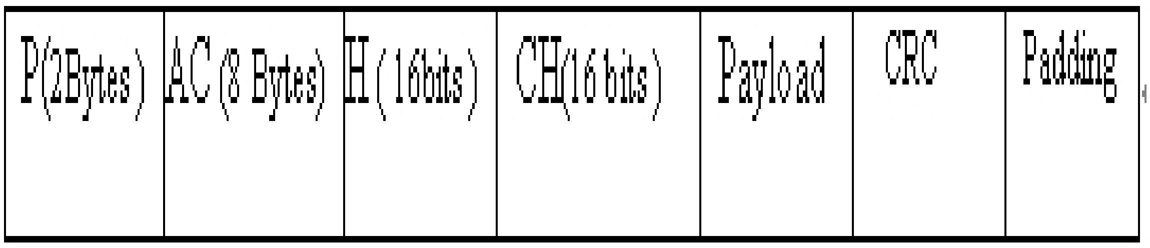
Although the simulation environment of our communication system is designed successfully in Section 4.2, transmitting real signals over the air needs us to connect USRP and our general processor together.

First of all, in GNU Radio, benchmark_rx.py and benchmark_tx.py under the folder of gnuradio/gnuradio-examples/python/digital are usually used as the basic files to transmit and receive real signals. All of the other Python files that are used to transmit signal are modified based on these two files. The primary purpose of benchmark_tx.py is to provide an interface for us to specify the parameters for the communication system, to generate the transmitted data and to call the send_pkt function in transmit_path module. The primary purpose of benchmark_rx.py is to get the receiver ready for the signal specified by the input on the command line and display the true or false condition of the received packet. Three important parameters for transmitter as well as receiver are center frequency(f), bit rate(r) and samples per symbol(S). The relationship between these three elements is explained below.

The most important element for a communication system is the bit rate or so-called coded bit rate. For our GNU Radio and USRP system, the bit rate is specified as: bit rate = converter_rate / interpolation rate / samples_per_symbol * bits_per_symbol, where the converter_rate, standing for the sampling frequency of the DAC and ADC in our implementation is set to the default value, which is 128MHz for the Tx part and 64MHz for the Rx part and interpolation rate should be replaced with decimation rate for the receiver side. There are some tricks for setting the samples_per_symbol value, the range of which is from 2 to 7 defined in GNU Radio. According to Nyquist sampling theory, two samples per symbol is enough for the receiver to restore the transmitted symbol as long as the bandwidth of the symbol is finite in the frequency domain and the aliasing could be neglected after shifting the spectrum to left and right by f_s which is two times the cutoff frequency of the symbol. Considering the factors we explained above, samples_per_symbol is chosen to be 4, which has been in fact verified by my testing: with the same distance between two USRPs, the PER was almost 100% when samples_per_symbol was 2 while the PER was almost 0% when samples_per_symbol was 4 under DQPSK modulation scheme. Because the interpolation rate or decimation rate is chosen from a specific range respectively, e.g. decimation rate can be changed from 8 to 256 with a step of 2, we chose 200kbps for the case of DBPSK and 400kbps for the DQPSK case and both of these two rates can be reached accurately.

Second, the transmit_path module is to glue the USRP hardware module with the software signal processing blocks such as modulator and amplifier and to set some parameters such as amplitude amplification factor and interpolation rate. The receive_path module is just to demodulate the received signal following the reverse path of that in transmit_path. It is the function called make_packet in packet_utils.py module that transforms a payload into a frame. What is actually transmitted over the air is the packet called pkt packed in the packet_utils.py file, which consists of preamble (P), access code (AC), header1 (H), header2 (CH), payload and crc. Figuer 32 shows how each frame is constructed: where copied header is just a duplicated one of its previous header, where the lower 12 bits indicates the length of the payload or how many bytes the payload contains, the higher 4 bits indicates the whitener_offset. The reason of adding padding code in the tail is to generate sufficient padding so that each packet sent across the USB ultimately comprises of a multiple of 512 bytes. Note: the length of the complex data type is 8. After reading the code about how Cyclic Redundancy Code (CRC) works in GNU Radio, the fact that there is no error correcting effect in CRC in the transmitted packet is confirmed. The way CRC in the packet_utils.py module works is: at the receiver end, the payload in the received packet is used to generate an expected CRC which is supposed to be compared with the CRC attached behind the payload and if the expected CRC were different from the attached CRC, this packet would be considered to be a false packet, which should be dropped. So, the effect of CRC is only detection in the demo file, but considering the statistical theory, if the length of payload were large enough, say 1500 bytes per packet, it would be a meaningful detection method since with only 4 bytes in CRC, if there were an error in CRC it would be highly possible that the message in payload happened to have some errors, producing a wrong expected CRC, which would induce a False Packet judgment.

The unmake_packet function in packet_utils.py module will extract the payload and CRC from the received packet and justify the correctness of the demodulated payload. If the payload is wrong, it will signal the receive_path module that this packet is false.



P(2Bytes)	AC (8 Bytes)	H (16bits)	CH(16 bits)	Payload	CRC	Padding

Figure 32: Transmitted frame (padding code is to add some padding at the end of the packet to fit the needs of USRP)

Implementation Procedure

Received Signal Strength Indicator (RSSI) is one of the most important parameters in our testing scenario: the RSSI of each received packet has been read and the sum of them is divided by the number of the total received packets to get an average value. All of the RSSI values are stored in the file RSSIM.dat, from which an average value will be calculated by running RSSIM.py file (A). There are 3 ways to read the RSSI value of the received packet in the USRP system: 1) Analog RSSI (we can read it using AUX ADC); 2) Digital RSSI in FPGA (from output of ADCs); 3) Digital RSSI in host (computed however you like, from the channel zed signal sent over the bus by the USRP)[18]. At the beginning, the value read was an analog signal value after the lowpass filter on the board, but after doing some testing, it was found that anything falling within the 15MHz-20MHz bandwidth of the lowpass filter would cause a rise in the RSSI value. In other words, even when there is no packet transmitting between two USRPs, the RSSI value read is greater than 160 while the maximum RSSI value is only 4096. The changing range of RSSI value is so small that it could not reflect the trivial change in the received signal power. Based on the observation above, another method was adopted, which was to calculate the filtered squared magnitude of each received packet because if carrier sense algorithms were implemented in the benchmark files, there must be some kind of received signal's magnitude calculation method in the files in order to compare the strength of the received signal with a threshold to realize carrier sensing. So the RSSI values stored in the RSSIM.dat file stand for the magnitude of the received signal.

Second, the bit streams used for testing should be randomly generated, so a random number generator (seed.py file) has been created to generate a sequence of random code, the length of which can be defined by the user (A). Before doing outdoor experimentation, a tx_payload file was generated stemming from the randomly generated bit streams and that tx_payload file would be saved at the receiver side in order to calculate the number of bit errors in each received packet.

Third, the ber.py file has been created to calculate the bit error rate of all the transmitted packets (A).

Forth, tx_run.py and rx_run.py files have been created to automate the whole testing procedure (A). Fifth, in order to check out the latest code from the development trunk, which is updated the moment a developer puts his code into the trunk, we should input the command $ svn co http://gnuradio.org/svn/gnuradio/trunk GNU Radio in the terminal to download the latest version of the code.

It is possible to use one PC to handle two USRPs at the same time as long as the w argument is set to 0 for the transmitter and to 1 for the receiver, but one thing that should be taken into account is the maximum bandwidth for each of the USRP system would be diminished by half.

In fact, the other two parameters, costas-alpha and gain-mu, regulating the frequency/phase tracking loop and timing recovery gain[19], are the real culprits of the malfunction of the DQPSK system. Although DBPSK demodulator functions very well with the default values of these two parameters, DQPSK demodulator is unable to demodulate any correct packet with its default values of these parameters. The key point is to chose a good starting point for parameters, costas-alpha and gain-mu so that the DQPSK demodulator also functions as well as DBPSK's. A for loop to search the appropriate starting points of those two parameters was created using shell script language. After running that program, it is known that costas-alpha should be set to 0.05 and gain-mu to 0.001, which did produce reasonable results for both of these two modulation schemes after our indoor testing.

4.3.2 Experiment Results

Since radio propagation and its channel dynamics cannot easily be captured using analytical or simulation models, we tried to observe the multi-rate margin based on a small-scale test bed using USRP [15]and GNU Radio [16].

The following are the details of the experiment as similarly done in [20]. (i) The testbed includes 3 USRP systems (version 5b), 3 RFX2400 transceivers (2.3-2.9 GHz) and GNU Radio software (version 3.1.3). (ii) Modulation schemes used are DBPSK and DQPSK. (iii) Carrier frequency and bandwidth are 2.479 GHz and 200 KHz, respectively. Therefore, the maximum data rate is 200Kbps and 400Kbps for DBPSK and DQPSK, respectively. A smaller bandwidth and data rates are used partly due to bandwidth constraints imposed by the USRP and USB port [21]. (iv)Transmitter amplitude is set to 8,000, which is smaller than the default value (12,000). This is to make the communication range no farther than 300 feet. Considering the three-node scenario with transmitter, receiver and interferer as in Figure 8, we needed an open space of at least 500 feet long. (v) Packet size is 1,500 bytes and 3,300 packets were transmitted for each experiment. (vi)Carrier sensing is effectively disabled to know the effect of interference more clearly. In fact, the implementation of carrier sense-based medium access protocol is known to be difficult for the current USRP/GNU Radio platform. Since USRP communicates via USB (2.0) port with a host system that runs GNU Radio, there exists a non-negligible delay between the instance that USRP senses the medium idle and another instance that the host system recognizes it[22, 23, 27].

Our goal was to validate Table 5, particularly the margin (8.9 dB discussed in Section 4.2) with two data rates supported by DBPSK and DQPSK modulation schemes. The experiment has been conducted in two phases.

First, in order to obtain communication range (r) with DBPSK and DQPSK, we set up two USRP systems and measured RSSI versus distance and PDR versus RSSI. In this experiment, the RX is fixed at the center of a circle and TX is moved from 300ft towards RX. At each position, 3,300 packets are received at the RX and the measured RSSI is the average RSSI of these packets. The first 200 packets are ignored in case there would be synchronization problem. The relationship between Distance and RSSI is shown in Figuer 33 and the relationship between RSSI and PDR is shown in Figure 34. According to our experimental results shown in Figuer 33 and 34, r for DBPSK and DQPSK is estimated as 215ft and 150ft, respectively. Note that 95% PDR is used to estimate the maximum communication range.

Second, in order to obtain interference range ($(1+\delta)$r), we set up three USRPs, a transmitter (T), a receiver (R) and an interferer (I) on a straight line (T-R-I) as in Figure 12. The distance between T and R is fixed at 80% of the communication range that we have already found in our first step, i.e., 172ft and 120ft for DBPSK and DQPSK, respectively. Since the communication distance is only 80% of the maximum communication range, the communication performance between T and R would be very good and PDR would be greater than 95%. But when I approaches R, the interference signal will disrupt the communication between T and R. The interference range is defined as the distance between R and I when the PDR from T to R stays steady at 95%. In our experiment, as I approaches R, it would cause a stronger interference. SIR at node R decreases and so is PDR. Figure 35 and 36 show PDR versus RSSI (from I to R) and PDR versus SIR. Note that the recorded RSSI is proportional to the mean power value of each received packet. The SIR at the receiver is calculated as RSSI from the sender, which is in the form of decibel, minus RSSI from the interferer [24].

According to the experiment results, we observed that the degree of capture is more significant at low data rate (DBPSK) as similarly observed in[19]. From Figure 33 and Figure 35, interference range ($(1+\delta)$r) is estimated as 220ft and 235ft for DBPSK and DQPSK, respectively, and the required CS range ($(2+\delta)$r) is about

392ft and 355ft. Comparing with Table 3, we can conclude that the same trend as well as the additional margin at a high rate (392 versus 355 ft) has been observed.

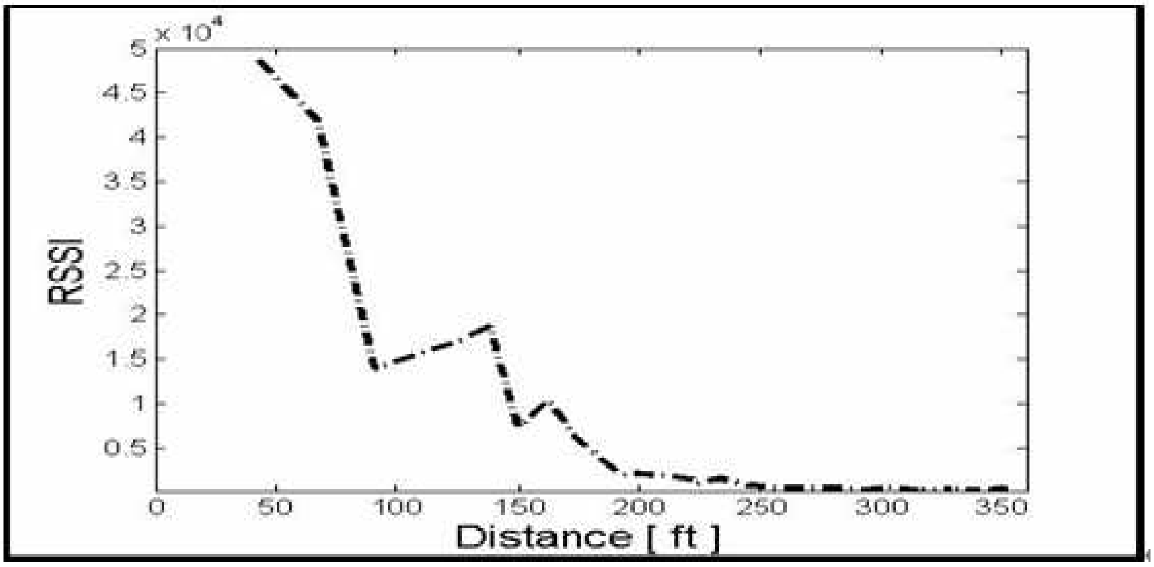


Figure 33: RSSI versus Distance

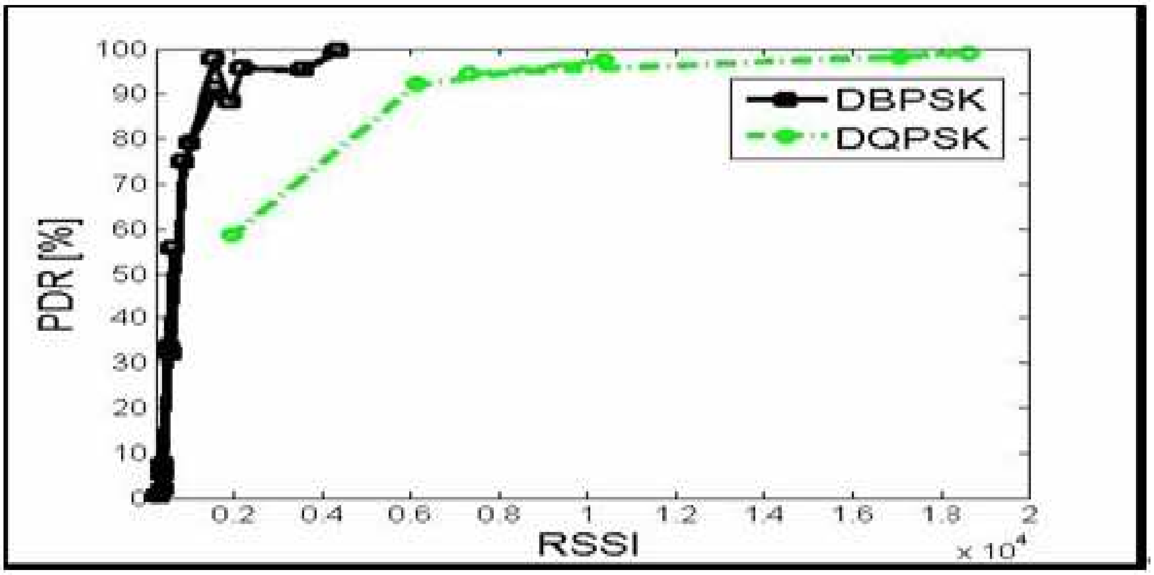


Figure 34: PDR versus RSSI

Because the theoretical data such as receiver sensitivity for our communication system is not available, the simulation result will be used as the reference for our experimental outcomes. In fact, the simulation analysis in Section 4.2.2 and Figure 29 definitely consolidates the correctness of our outdoor experimentation. From Figure 29 and Figure 36, three important conclusions can be achieved. First, the

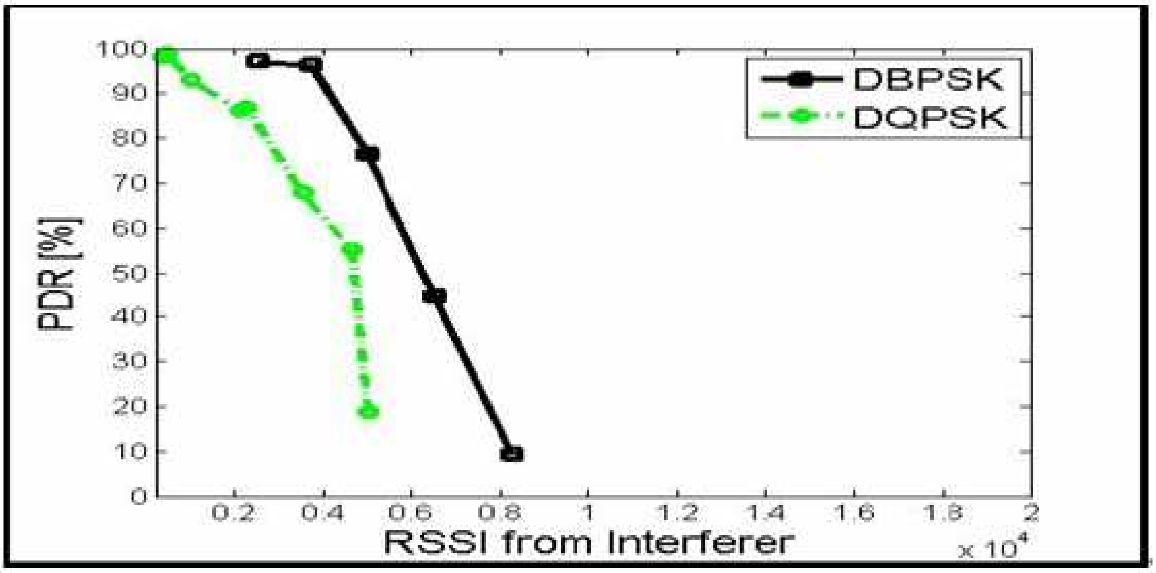


Figure 35: PDR versus RSSI from Interference

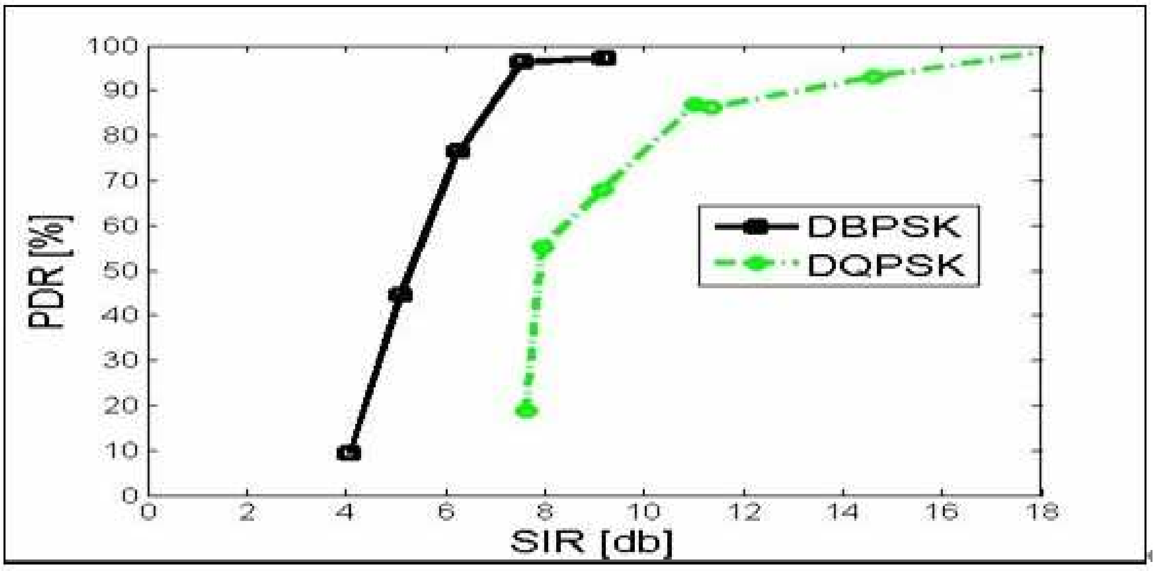


Figure 36: PDR versus SIR

trends of the curves in these two figures are roughly the same. Second, in theoretical curve, the PDR reaches 100% when SNR is around 8dB for DBPSK and 12dB for DQPSK, while PDR attains 100% when SIR is around 10dB for DBPSK and 16dB for DQPSK. In other words, the curves obtained from outdoor testing have been right shifted around 3dB which is conceivable because it is more difficult for receiver to demodulate packets when there is background noise, multi-path or other propagation effects nearby[20]. Third or the most important observation is that the gap between the DBPSK curve and DQPSK curve in simulation figure and the figure obtained from outdoor experimentation are almost the same, around 4.5 dB.

The transmission range difference for DBPSK and DQPSK implemented with RS coding scheme has also been observed in our outdoor testing, which is shown in Figuer 37. From this figure, the communication range for DBPSK with RS (207, 187) is approximately 310ft and the communication range for DQPSK with RS (207, 187) is around 220ft. So, apart from changing different modulation schemes, adapting different coding schemes is an alternative to observe multi-rate margin. For example, with DBPSK modulation scheme and RS (207, 187) coding scheme, the communication distance is extended by 44.19%, while the bandwidth efficiency is decreased by 10%. So, as can be imagined from Figure 10, the DBPSK modulation scheme could be used in inner layer communication system, while the DBPSK with RS coding scheme could be used in outer layer communication system.

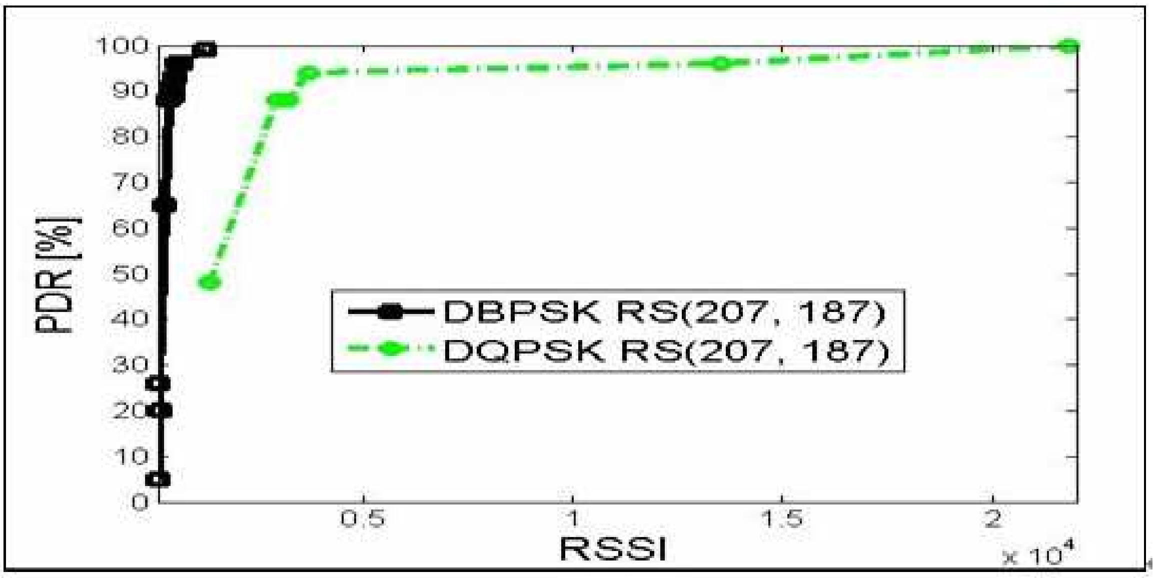


Figure 37: PDR versus RSSI with Reed Solomon Coding

CHAPTER V

EXPLOITATION OF MULTI-RATE MARGIN

Multi-rate Margin could be exploited by MTOP, which has been briefly discussed in Chapter 1. To better understand the special characteristics of MTOP, Figure 38 has been drawn to illustrate the transmission characteristics of MTOP over other transmission protocols. Using the idea of multi-rate margin in MTOP, a frame could be relayed through several intermediate nodes without inter-frame gaps such as DIFS and backoff time. It does not seem to be possible considering the nodes contention when frames are relayed among intermediate nodes, but the strategy employed in MTOP will prove the feasibility of this method.

5.1 Multi-rate Transmission Opportunity (MTOP)

As can be seen from Table 3, the difference between the carrier sensing range of 1Mbps mode and that of 11Mbps mode is 271 meters. This 271 meters space can actually used for two extra node to relay the transmitted frame without DCF Inter-frame Space (DIFS) and backoff time because the relaying action will be protected

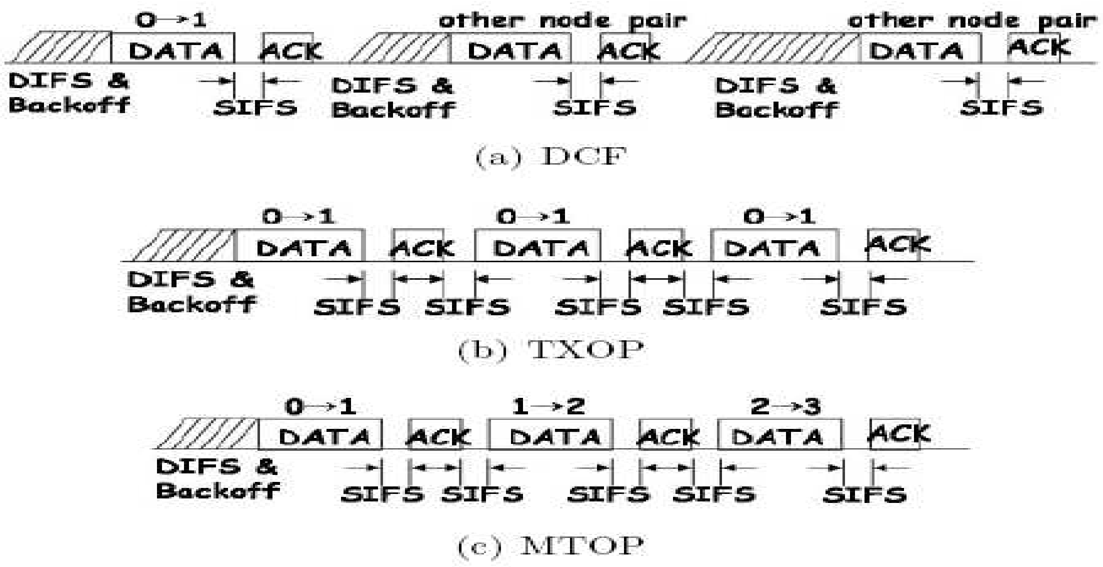


Figure 38: DCF, TXOP and MTOP packet transmission protocol[41]

by the carrier sensing range of 1Mbps mode. To better illustrate the MTOP idea, Figure 12 has been plotted out.

The idea of MTOP is to relay one packet via several nodes only with the inter-packet gap of SIFS. This will efficiently improve the overall throughput for the mesh network particularly at high rate because all the nodes that are supposed to relay the packet are sitting within the protected arch and transmitting packets without considering the interference; this will give us a chance to omit the DCF Interframe Space (DIFS) and backoff time between each pair of the relaying nodes. For example, in the high rate communication in Figure 12, a hop node could be inserted on the right side of the receiver and the distance between the hop node and the receiver is 118 meters. With 589 meters set to be the defer range, the first, second and third hop communications would not be disturbed by any interferer because both of them are being protected by the defer range, while in the 1 Mbps case, there is no more space to insert any more node for relaying.

A small scale test bed consisting of USRP and GNU Radio is again used to verify the efficiency of MTOP idea. The MAC layer implementation of MTOP and the layout of our scenario will be discussed below.

5.2 The MTOP Scenario Layout and result analysis

Current implementation of GNU Radio does packet transmission and reception using USRP based on packets streaming i.e. there is no carrier sense mechanism implemented in the benchmark programs for packet transmission and reception. It was modified to include carrier sense algorithm[31]. Further this code was modified to include MTOP algorithm too. The python code for implementing the CSMA and MTOP algorithm in the benchmark programs for sending, forwarding and receiving the packets will be found in A.

Two sets of experiments were conducted to exhibit the properties of MTOP: the first one is supposed to prove the second hop of transmission will not be impacted by an interferer sitting outside the carrier sense range of transmitting node i.e. node A; the second one is to observe the effect of MTOP on communication inside the multi-rate wireless network boundaries.

First set consisted of observing wireless communication among three nodes with CSMA and with/without MTOP algorithm. In the first part of this exercise, nodes A, B and C used DQPSK modulation scheme at 400 Kbps and interferer node D used DBPSK modulation at 200 Kbps. Since MTOP affects only the working of intermediate hop, node B was selected for implementing MTOP for this prototype. Measurements were recorded for both the scenarios in which node B was working on CSMA with and without MTOP implementation. In this exercise, firstly, PDR and throughput measurements for node C were recorded without MTOP implementation on intermediate hop node i.e. node B. In the second part of this exercise, measurements were recorded with MTOP implementation on node B.

Table VI: Table representing PDR and throughput improvement [41]

	PDR%	Throughput(Kbps)
Without MTOP	87.53	53.64
With MTOP	90.54	55.31

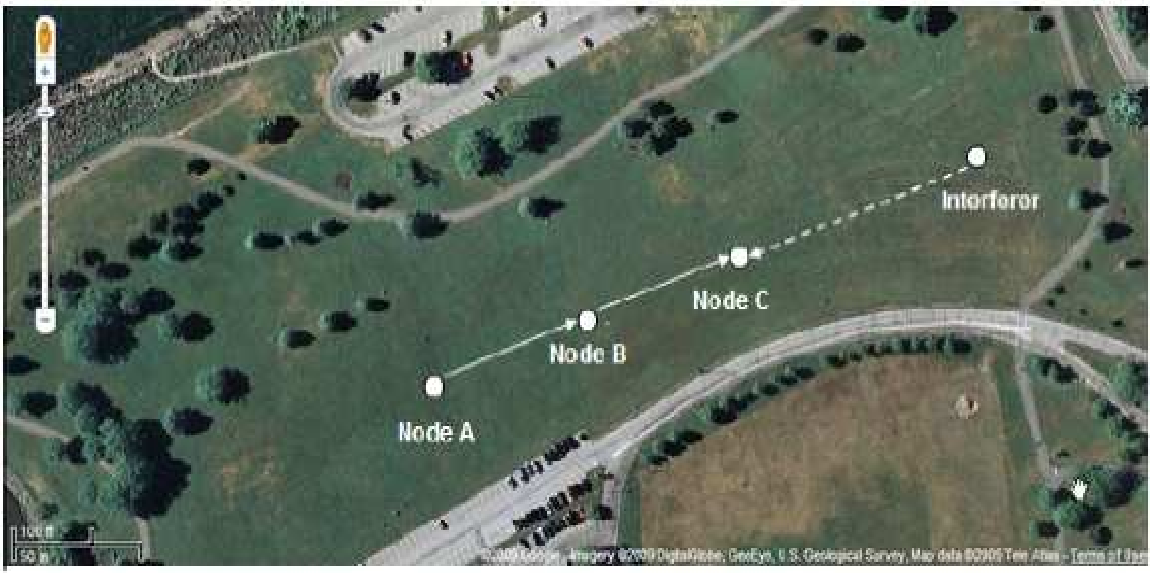


Figure 39: USRP Placement in Edgewater Park, Cleveland [41]

From Table 6, it can be seen that the PDR has been improved by 3.03% when nodes communicate with MTOP against nodes without MTOP. It proves the argument that no adverse impact has been induced on wireless network performance when intermediate nodes send a packet without waiting for next contention period (after DCF Interframe Space (DIFS)). In addition to PDR, throughput with MTOP in effect was also higher by 1.67% than without MTOP scenario, proving our argument of improvement in end-to-end communication scenario.

The second set of experiments was done to observe the effect of MTOP on communication inside the multi-rate wireless network boundaries. Without MTOP algorithm, each node competes for the medium and transmits the packet on medium availability. This restricts the ability to transmit the data packet over multiple hops of communication and reach the destination in less time. Throughput improvement is the main objective of MTOP. Using MTOP, a packet can be transmitted much faster over a series of hops without possible packet collision.

In this exercise, we wanted to prove that MTOP improves the network performance through increased throughput without increased collision. A set of PDR and throughput measurements was taken in this scenario with all the nodes inside the

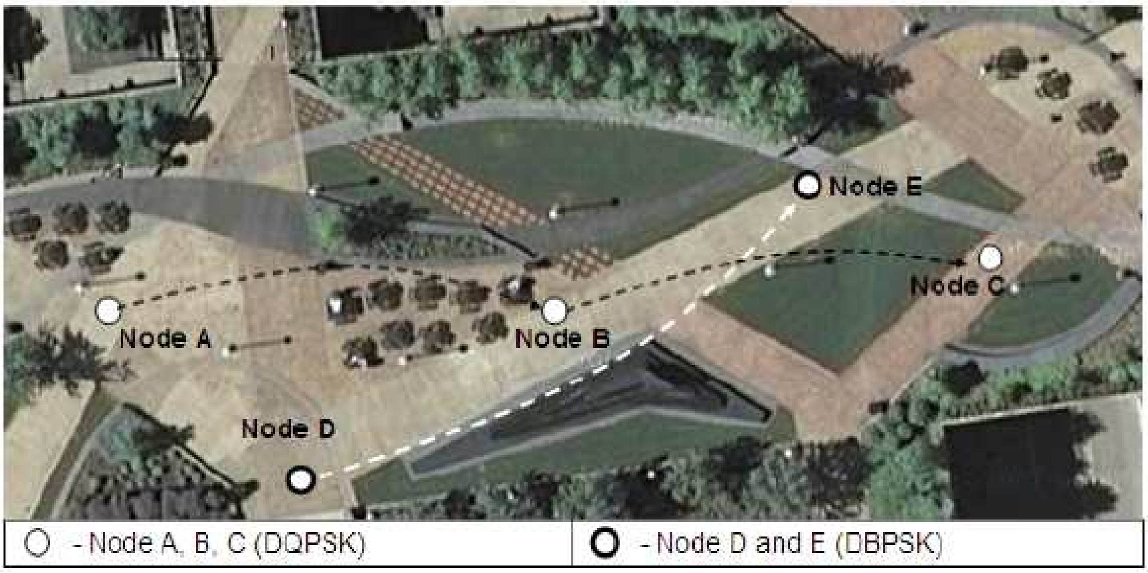


Figure 40: Real Test setup in front of Rhodes Tower of Cleveland State University [41]

carrier sense range of node A. Node A, B and C were kept in a line with intermediate distance of 90 ft. between nearby nodes. Node D and E were kept in the vicinity of node B and C with a distance of 80 ft. between them. The testing setup is shown with a top view of the location, in Figure 40, where this experiment was actually performed. Firstly, the measurements were taken without MTOP implementation on node B followed by another set of measurements with MTOP implementation. Under the MTOP implementation, node B forwards the packets received from node A, to node C which is the packet destination node. Frame length of each packet transmitted was kept as 1000 bytes and each measurement was performed multiple times to exclude any effect of environmental disturbances i.e. wind. It was observed that throughput was improved by 13.19% after implementing MTOP in the wireless scenario, without any impact on PDR which was seen as almost constant. Further details of the measurements are listed in Tables 7 and 8.

Table VII: PDR and throughput recorded for node C and E for out of carrier range interference[41]

	Node C		Node E	
	PDR%	Throughput(Kbps)	PDR%	Throughput(Kbps)
Without MTOP	90.22	48.37	71.94	38.83
With MTOP	93.15	54.15	76.62	44.54

Table VIII: Average PDR and throughput for out of carrier range interference[41]

	PDR%	Throughput(Kbps)
Without MTOP	81.08	43.59
With MTOP	84.88	49.34

CHAPTER VI

SUMMARY AND FUTURE RESEARCH

6.1 Conclusions

Multi-rate margin is the fundamental ingredient for multi-hop opportunistic transmission (MTOP) protocol, which is targeted to improve communication performance in wireless network environment. In this thesis, the concept of multi-rate margin is introduced based on the multi-rate analysis in IEEE 802.11b radios after the multi-rate support in IEEE 802.11 radios is investigated; Deep understanding to Software Defined Radio (SDR) and GNU Radio has been provided; the multi-rate margin has been analyzed from theoretical point of view, investigated via MATLAB communication system simulation and tested through real world wireless communication in outdoor testing with the help of GNU Radio and USRP; the multi-rate margin is also exploited for providing proof of concept of MTOP and even the self-interference problem of 802.11b radios has been investigated, the simulation result of which could be utilized for performance enhancement in multi-hop networks based on the idea of multi-rate margin.

The around 4.5dB gap has been observed between PDR versus SIR performance curves of DBPSK and DQPSK communication systems from the outdoor experimentation, the result of which is exactly verified by the communication system simulation using MATLAB. Apart from the multi-rate margin between DBPSK and DQPSK, the similar trend is also observed when two kinds of modulation schemes are implemented with Reed Solomon coding schemes. The multi-rate margin between DBPSK and DBPSK encoded with RS codes is also apparently observed. MTOP has also been considered as performing better compared with a simple ad hoc network communication scenario.

This thesis shows a promising approach for improving the wireless network performance investigated by Dr. Chansu Yu from Cleveland State University. An attempt has been made to provide the real world transmission result to support the multi-rate margin, which has been consolidated by a variety of simulation and theoretical analysis.

6.2 Future Research

Instead of changing modulation schemes to observe multi-rate margin, other coding schemes can also be utilized to observe multi-rate margin, say convolutional coding schemes. It will be easier to see the trade off between the advantages resulting from multi-rate margin and the loss in the bandwidth efficiency. While the RS code is good at error detection, the convolutional code more suits to correcting errors.

To further confirm the benefits of MTOP idea, NS-2 network simulation is necessary to get launched because experimental findings always need simulation result for support.

BIBLIOGRAPHY

[1] IEEE Std 802.11b-1999(R2003) Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications High Speed Physical Layer Extension in the 2.4GHz Band.

[2] IEEE Std 802.11a-1999(R2003) Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications High Speed Physical Layer in the 5GHz Band.

[3] Jeffrey G. Andrews, Arunabha Ghosh, Rias Muhamed,"Fundamentals of WiMAX: Understanding Broadband Wireless Networking," Prentice Hall, 2007.

[4] Fuqin Xiong, "Digital Modulation Techniques," Artech House, Boston/London, 2000.

[5] Theodore S. Rappaport, "Wireless Communications: Principles and Practice," Upper Saddle River, NJ: Prentice Hall, 1996.

[6] Sam W. Ho, "Adaptive Modulation (QPSK, QAM)," Intel Corporation, 2004.

[7] M. Heusse, F. Rousseau, G. Berger-Sabbatel and A. Duda, "Performance anomaly of 802.11b," IEEE INFOCOM, 2003.

[8] G. Tan and J. Guttag, "Time-based Fairness Improves Performance in Multi-rate Wireless LANs," The USENIX Annual Technical Conference, 2004.

[9] Eric Blossom, "Exploring GNU Radio," 2004.

[10] Shu Lin and Daniel J. Costello, "Error Control Coding(2nd Edition)," Prentice Hall, NJ, 2004.

[11] I. Tinnirello and S. Choi, "Temporal fairness provisioning in multirate contention-based 802.11e WLANs," IEEE WoWMoM, 2005.

[12] Rodger H Hosking, “Software Defined Radio Handbook,” Upper Saddle River, NJ, June 2009.

[13] Nortre Dame, “Entering the World of GNU Software Radio,” https://radioware.nd.edu/documentation/basic-gnuradio/entering-the-world-of-gnu-software-radio

[14] Naveen Manicka, “GNU Radio Testbed,” Master Thesis, University of Delaware, Spring 2007.

[15] University of Notre Dame, “The Usrp Board,” https://radioware.nd.edu/documentation/hardware/the-usrp-board

[16] Universal Software Radio Platform (USRP), http://www.ettus.com/

[17] GNU Radio Project, www.gnuradio.org/trac/

[18] http://gnuradio.org/trac/wiki/UsrpFAQ/RSSI

[19] Sklar, B., “Digital Communications: Fundamentals and Applications,” Englewood Cliffs, New Jersey: Prentice Hall, 1988.

[20] A. Jow, C. Schurgers and D. Palmer, “CalRadio: A Portable,Flexible 802.11 Wireless Research Platform,” International Workshop on System Evaluation for Mobile Platforms (MobiEval), 2007.

[21] K. A. Jamieson, “The SoftPHY Abstraction: from Packets to Symbols in Wireless Network Design,” Ph.D. Dissertation, MIT, June 2008.

[22] R. Dhar, G. George, A. Malani and P. Steenkiste, “Supporting Integrated MAC and PHY Software Development for the USRP SDR,” 1st IEEE Workshop on Networking Technologies for Software Defined Radio Networks, 2006.

[23] T. Schmid, O. Sekkat and M. B. Srivastava, “An experimental study of network performance impact of increased latency in software defined radios,” ACM WinTECH, 2007.

[24] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon and Y. Choi, "An Experimental Study on the Capture Effect in 802.11a Networks," ACM WinTech, 2007.

[25] J. Yee and H. Pezeshki-Esfahani, "Understanding wireless LAN performance trade-offs," CommsDesign.com, Nov. 2002.

[26] Nortre Dame, "Advanced GNU Radio," https://radioware.nd.edu/documentation/advanced-gnuradio

[27] Proakis, J. G., "Digital Communications," New York, NY, McGraw-Hill, 1995, pp. 1C928.

[28] Lee, J. S., and Miller, L. E., "CDMA Engineering Handbook," Boston, MA, Artech House, 1998, pp. 1C1228.

[29] Van Nee and R. D. J., "OFDM codes for peak-to-average power reduction and error correction," Proceedings IEEE Global Telecommunications Conference, London, UK, Vol. 1, pp. 740C744, November 1996.

[30] IEEE Std 802.15.2-2003.pdf Part 15.2: Coexistence of Wireless Personal Area Networks with Other Wireless Devices Operating in Unlicensed Frequency Bands.

[31] http://acert.ir.bbn.com/projects/adroit

[32] A. Kamerman and L. Monteban, "WaveLAN-II: A High-performance wireless LAN for the unlicensed band," Bell Lab Technical Journal, pages 118-133, Summer 1997.

[33] M. Lacage, M.H. Manshaei and T. Turletti, "IEEE 802.11 Rate Adaptation: A Practical Approach," ACM MSWiM, 2004.

[34] A. Akella, G. Judd, P. Steenkiste and S. Seshan, "Self Management in Chaotic Wireless Deployments," ACM MobiCom, 2005.

[35] Mathieu Lacage, Mohammad Hossein Manshaei and Thierry TUrletti, “IEEE 802.11 Rate Adaptation: A Practical Approach,” MSWiM’04, October 4-6. 2004.

[36] G. Holland, N. Vaidya, and P. Bahl, “A Rate-Adaptive MAC Protocol for Multi-Hop Wireless Networks,” In Proceeding ACM MOBICOM, July 2001.

[37] www.tscm.com/rcvr_sen.pdf

[38] R. Gummadi, D. Wetherall, B. Greenstein and S. Seshan, “Understanding and Mitigating the Impact of RF Interference on 802.11 Networks,” ACM SIGCOMM, 2007.

[39] H. Zhai and Y. Fang, “Physical Carrier Sensing and Spatial Reuse in Multirate and Multihop Wireless Ad Hoc Networks,” IEEE INFOCOM, 2006.

[40] http://www.hughes.com/HUGHES/Rooms/DisplayPages/LayoutInitial? Container=com.webridge.entity.Entity[OID[09EA078968A3E64FB8A00B6DFBE8650]]

[41] Sachin Hirve, “Multihop Transmission Opportunistic Protocol on Software Radio,” Master Thesis, Cleveland State University, 2009.

[42] B. Awerbuch, D. Holmer and H. Rubens, “High throughput route selection in multi-rate ad hoc wireless networks,” Technical report, Johns Hopkins University, 2003.

[43] J. Bicket, D. Aguayo, S. Biswas and R. Morris, “Architecture and Evaluation of an Unplanned 802.11b Mesh Network,” IEEE/ACM MobiCom, 2005.

[44] R. Draves, J. Padhye and B. Zill, “Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks,” ACM MobiCom, 2004.

[45] strixsystems.com, “Solving the Wireless Mesh Multi-Hop Dilemma,” 2008.

[46] Brian P. Crow, Indra Widjaja, Jeong Geun Kim and Prescott T. Sakai, “IEEE 802.11 Wireless Local Area Networks,” IEEE Communications Magazine, September 1997.

APPENDIX

APPENDIX A

PROGRAM

A.1 Program: Python code for outdoor testing and GNU Radio simulation

A.1.1 Program: Rx Program collecting data

```
#!/usr/bin/env python

#

# Copyright 2005,2006,2007 Free Software Foundation, Inc.

#

# This file is part of GNU Radio

#

# GNU Radio is free software; you can redistribute it and/or modify

# it under the terms of the GNU General Public License as published by

# the Free Software Foundation; either version 3, or (at your option)

# any later version.

#

# GNU Radio is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of‘
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GNU Radio; see the file COPYING.  If not, write to
# the Free Software Foundation, Inc., 51 Franklin Street,
# Boston, MA 02110-1301, USA.
#

from gnuradio import gr, gru, modulation_utils
from gnuradio import usrp
from gnuradio import eng_notation
from gnuradio.eng_option import eng_option
from optparse import OptionParser

import random
import struct
import sys
import time

# from current dir
from receive_path import receive_path
import fusb_options

import cPickle as p

#import os
```

```
#print os.getpid()
#raw_input('Attach and press enter: ')

class my_top_block(gr.top_block):
    def __init__(self, demodulator, rx_callback, options):
        gr.top_block.__init__(self)
        self.rxpath = receive_path(demodulator, rx_callback, options)
#print self.rxpath.u.read_aux_adc(0, 0)
#self.rxpath.u.read_aux_adc(0, 0)
        self.connect(self.rxpath)

# ///////////////////////////////////////////////////////////////////////////////
#                                   main
# ///////////////////////////////////////////////////////////////////////////////

global n_rcvd, n_right

def main():
    global n_rcvd, n_right, rx_sequence, rssi_m, rssi, per, tb, MAX
    MAX = 1325
    rx_sequence = {}
    rssi = {}
    rssi_m = {}
    per = []
    n_rcvd = 0
    n_right = 0
    txpayload = 'txpayload.dat'
    f2 = file(txpayload)
    txdata = p.load(f2)

    #print txdata

    f2.close

    pktnos_tx = txdata.keys()

    pktnos_tx.sort()

    txno_max = pktnos_tx[-1]  # the max pktno allowed

    #print txno_max




    def rx_callback(ok, payload):

        global n_rcvd, n_right, rx_sequence

#delay = 0.0001


        (pktno,) = struct.unpack('!H', payload[0:2])

        n_rcvd += 1

        #rx_sequence[pktno] = payload # modified for BER

#rssi[pktno] = tb.rxpath.u.read_aux_adc(0, 0) # modified for RSSI the type

#of return is int

#rssi_m[pktno] = tb.rxpath.probe.level()

#while tb.rxpath.carrier_sensed():

         #   rssi_m[pktno] = tb.rxpath.probe.level()

        if n_rcvd > 200: # erase #0-#199 packets

    rx_sequence[pktno] =  payload

    rssi_m[pktno] = tb.rxpath.probe.level()

            if ok:

                n_right += 1

    PER = float(n_right)/float(pktno-200) # modified for PER which stands
```

```
#for PDR
    #print pktno
    per.append(PER)
    #if pktno > (MAX - 3):
            # print "ok = %5s  pktno = %4d  n_rcvd = %4d  n_right = %4d PER
= %.4f" % (
            #     ok, pktno, n_rcvd, n_right, PER)        # modified for PER
    #if pktno > 900:
    print "ok = %5s  pktno = %4d  n_rcvd = %4d  n_right = %4d PDR = %.4f"
% (
                 ok, pktno, n_rcvd, n_right, PER)

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
# put all the received sequence in a file
perfile = 'per.dat'
f = file(perfile, 'w')
p.dump(per,f)
f.close()
rxpayload = 'rxpayload.dat'
     f = file(rxpayload, 'w')
p.dump(rx_sequence, f)
     f.close()
rssifile = 'RSSI.dat'
     f = file(rssifile, 'w')
p.dump(rssi, f)
```

```
     f.close()
rssimfile = 'RSSIM.dat'
f = file(rssimfile, 'w')
p.dump(rssi_m, f)
f.close()
pass
    except RuntimeError:
# put all the received sequence in a file
perfile = 'per.dat'
f = file(perfile, 'w')
p.dump(per,f)
f.close()
rxpayload = 'rxpayload.dat'
     f = file(rxpayload, 'w')
p.dump(rx_sequence, f)
     f.close()
rssimfile = 'RSSIM.dat'
f = file(rssimfile, 'w')
p.dump(rssi_m, f)
f.close()
pass
    finally:
# put all the received sequence in a file
perfile = 'per.dat'
f = file(perfile, 'w')
p.dump(per,f)
f.close()
rxpayload = 'rxpayload.dat'
```

```
     f = file(rxpayload, 'w')
p.dump(rx_sequence, f)
     f.close()
rssimfile = 'RSSIM.dat'
f = file(rssimfile, 'w')
p.dump(rssi_m, f)
f.close()
        pass}
```

A.1.2 Program: Tx Program sending data

```
#!/usr/bin/env python
#
# Copyright 2005, 2006, 2007 Free Software Foundation, Inc.
#
# This file is part of GNU Radio
#
# GNU Radio is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 3, or (at your option)
# any later version.
#
# GNU Radio is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with GNU Radio; see the file COPYING.  If not, write to
# the Free Software Foundation, Inc., 51 Franklin Street,
# Boston, MA 02110-1301, USA.
#

from gnuradio import gr, gru, modulation_utils
from gnuradio import usrp
from gnuradio import eng_notation
from gnuradio.eng_option import eng_option
from optparse import OptionParser

import random, time, struct, sys

# from current dir
from transmit_path import transmit_path
import fusb_options

import cPickle as p

#import os
#print os.getpid()
#raw_input('Attach and press enter')

class my_top_block(gr.top_block):
    def __init__(self, modulator, options):
        gr.top_block.__init__(self)
        self.txpath = transmit_path(modulator, options)
        self.connect(self.txpath)
```

```
# ////////////////////////////////////////////////////////////////////////////
#                                    main
# ////////////////////////////////////////////////////////////////////////////

def main():
    global arrImg
    arrImg = []

    def send_pkt(payload='', eof=False):
        return tb.txpath.send_pkt(payload, eof)

    def rx_callback(ok, payload):
        print "ok = %r, payload = '%s'" % (ok, payload)

    #def source_seed(charnum): # modified
# charnum = int(charnum)
# #arrImg = []
# for i in range(0, charnum):
 #        data = random.randint(0,255) # generate charnum random characters
from 0 to 255
  #       arrImg.append(data)
#arrImg.append('')

# print len(arrImg) let us use source.dat file as the random generator which
is known by both the Tx and Rx.
    source_data = 'source.dat'
    f = file(source_data)
    arrImg = p.load(f)
    print len(arrImg)
    f.close()


    mods = modulation_utils.type_1_mods()#The mod/demod code registers
when they are loaded.
#See for example the last two lines in
#gnuradio-core/src/gnuradio/blks2impl/dbpsk.py


    parser = OptionParser(option_class=eng_option, conflict_handler="resolve")
    expert_grp = parser.add_option_group("Expert")


    parser.add_option("-m", "--modulation", type="choice", choices=mods.keys(),
                      default='gmsk',
                      help="Select modulation from: %s [default=%%default]"
                            % (', '.join(mods.keys()),))


    parser.add_option("-s", "--size", type="eng_float", default=1500,
                      help="set packet size [default=%default]") # how many
in one packet
    parser.add_option("-M", "--megabytes", type="eng_float", default=1.0,
                      help="set megabytes to transmit [default=%default]") #
of the bytes being sent
    parser.add_option("","--discontinuous", action="store_true", default=False,
                      help="enable discontinous transmission
(bursts of 5 packets)")
    parser.add_option("","--random-num", action="store_true", default=False,
                      help="enable random char generation")
    parser.add_option("","--from-file", default=None,
                  help="use file for packet contents")
    #parser.add_option("","--charnum", type="eng_float", default=125000,
    #                  help="set the number of random characters
[default=%default]")
# modified



    transmit_path.add_options(parser, expert_grp)


    for mod in mods.values():
        mod.add_options(expert_grp)


    fusb_options.add_options(expert_grp)
    (options, args) = parser.parse_args ()


    if len(args) != 0:
        parser.print_help()
        sys.exit(1)


    if options.tx_freq is None:
        sys.stderr.write("You must specify -f FREQ or --freq FREQ\n")
        parser.print_help(sys.stderr)
        sys.exit(1)


    if options.from_file is not None:


        source_file = open(options.from_file, 'r')
    #if options.random_num is not False:
```

```
#source_seed(options.charnum)

    # build the graph
    tb = my_top_block(mods[options.modulation], options) # dbpsk_mod
1st call

    r = gr.enable_realtime_scheduling()
    if r != gr.RT_OK:
        print "Warning: failed to enable realtime scheduling"

    tb.start()                      # start flow graph   UP TO HERE

    # generate and send packets
    nbytes = int(1e6 * options.megabytes)
    n = 0
    pktno = 0
    pkt_size = int(options.size)

    # generate a dictionary to store transmitted pktno and corresponding
payload
    tx_sequence = {}

    while n < nbytes:
        if options.from_file is None and options.random_num is False:
            data = (pkt_size - 2) * chr(pktno & 0xff) # duplicate the
char being transmitted (pkt_size-2) times with each byte standing
for one char, say'0'
        elif options.from_file is not None:
```

```
# is "0x00" 1&0xff = 1 chr(1)=0x01
            data = source_file.read(pkt_size - 2)
            if data == '':
                break;
else:
    data = ''
    for m in range(0,pkt_size-2):
        datan = arrImg[pktno * (pkt_size - 2) + m]
     #if datan == '':
                 #   break;
datan = chr(datan & 0xff)
data = data + datan



        payload = struct.pack('!H', pktno & 0xffff) + data
tx_sequence[pktno] = payload #modified
        #rx_callback(True, payload)  # modified
        send_pkt(payload)
        n += len(payload) # including pktno
        sys.stderr.write('.')
        if options.discontinuous and pktno % 5 == 4:
            time.sleep(1)
        pktno += 1

    # put all the transmitted sequence in a file
    txpayload = 'txpayload.dat'
    f = file(txpayload, 'w')
    p.dump(tx_sequence, f)
    f.close()

    send_pkt(eof=True)


    tb.wait()                       # wait for it to finish

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        pass
```

APPENDIX B

PROGRAM

B.1 Program: MATLAB code for communication system simulation

B.1.1 Program: simulation for DBSPK with RS (255,235)

```
function [BER] = test(num_bit, M, bitrate, fs, fc )
%------------------------------------------
% bit generator
%------------------------------------------
bits = rand(1, num_bit) > 0.5;
bits = double(bits);
%-------------------------------------
% reed-solomon encoder
%-------------------------------------
m=8;        %number of bits each symbol from input
t=10;          %error correction ability from input
x_msg = bi2de(reshape(bits,m,length(bits)/m).', 'left-msb');
x_msg = x_msg.';

n=2^m-1-48;    %code length only here modified!!!
k=n-2*t;      %information length 3 187
x_msg = reshape(x_msg, k, length(x_msg)/k).';
msg = gf(x_msg, m);
code = rsenc(msg, n, k);
code = reshape(code.', 1, length(bits)/m/k*n);
code_uint32 = code.x;
en_bits = de2bi(double(code_uint32), m, 'left-msb'); % convert to bits
[x,y]=size(en_bits);
en_bits = reshape(en_bits.', 1, x.*y); % reshape it to get a row vector
%------------------------------------------
% gray code
%------------------------------------------
for j=1:length(en_bits)
    if en_bits(j) == 0
        gr_en_bits(j) = 0;
    else
        gr_en_bits(j) = 1;
    end
end
%------------------------------------------
% differential encoding
%------------------------------------------
len_bits = length(gr_en_bits);
d(1)=0;   %-- reference bit is 0
```

```
for j = 1:len_bits

    d(j+1)= xor(gr_en_bits(j), d(j));

end

diff = d;

%------------------------------------------

% baseband BPSK modulate

%------------------------------------------

h = modem.pskmod(M); %Create a modulator object

mod_sym = modulate(h,diff); % Modulate the signal bits

%------------------------------------------

% Pulse Shaping and RRC Filter

%------------------------------------------

width = 1/bitrate; % bit duration time [s]

points = fs * width; % the number of points for each bit,

num = rcosine(1,points,'sqrt'); % Transfer function of filter

xpulse = rcosflt(mod_sym,1,points,'filter',num);% Filter the data.

xpulse = xpulse.';

%----------------------------------------

%IF modulation

%----------------------------------------

dt = width/points; % sampling period

t = dt .* [0 : length(xpulse) - 1];

y_tx = xpulse.*exp(i.*2.*pi.*fc.*t);


%-------------------------------------------

% channel

%-------------------------------------------

%k = log2(M); % bits per symbol
```

```
%EbNo = 20;

EbNo = 10; % for bpsk Eb is Es

Num = length(EbNo); % measure the length of EbNo

for j = 1:Num

    snr = EbNo(j) + 10*log10(log2(M)) - 10*log10(points);

    y_noisy = awgn(y_tx,snr,'measured', 'db');

    y_rx = y_noisy;

%---------------------------------------------

%(receive filter) match filter

%---------------------------------------------

    y_match = rcosflt(y_rx.* exp (-i.*2.*pi.*fc.*t),1, points,'Fs/filter',num);


    length(y_match);

    y_match = y_match.';

%------------------------------------------

% Sample it at time T

%------------------------------------------

    decision_t = [(6*points+1) : points : ((length(mod_sym)+5)*points+1)];

    y_decision = y_match(decision_t);

%----------------------------------------------

%differential decoder differentiation also induces noise???

%---------------------------------------------

    for m_de= 1:(length(y_decision)-1)

        decoded(m_de)=conj(y_decision(m_de)).*y_decision(m_de+1);

    end

%---------------------------------------------

%demapping

%---------------------------------------------
```

```
    demap_bit = demodulate(modem.pskdemod(M),decoded);


%--------------------------------------------
%degray code
%--------------------------------------------
    for j_demap=1:length(demap_bit)
        if demap_bit(j_demap) == 0
            degray_bits(j_demap) = 0;
        else
            degray_bits(j_demap) = 1;
        end
    end
%----------------------------------
% Reed-Solomon Decoder
%----------------------------------
    de_bits = bi2de(reshape(degray_bits, m, length(degray_bits)/m).’,
    ’left-msb’);
    de_bits = gf(de_bits,m);
    de_sym = reshape(de_bits.’, n, length(de_bits)/n);
    decoded_rs = rsdec(de_sym.’, n, k);
    decoded_rs = reshape(decoded_rs.’, 1, length(bits)/m);
    decoded_bits = de2bi(double(decoded_rs.x),m, ’left-msb’); %convert to bits
    decoded_bits = reshape(decoded_bits.’, 1, length(bits));
    %figure(4); stem(z_bit);


    [num_err(j), BER(j)] = biterr(bits, decoded_bits);
    %end
end
```

```
ebno = power(10, EbNo/10);

a1 = 2.*qfunc(sqrt(2.*ebno)).*(1-qfunc(sqrt(2.*ebno)));

a3 = 2.*qfunc(sqrt(2.*power(10, 7/10))).*(1-qfunc(sqrt(2.*power(10, 7/10))))

a2 = qfunc(sqrt(2.*ebno));

L = semilogy(EbNo, BER, EbNo, a1, EbNo, a2);

legend('Simulated BER for DBPSK with RS','Theoretical BER for DBPSK',

'Theoretical BER of BPSK');

set(L, 'LineWidth', [2]);

axis([0 10 1e-3 1e-1]);

title('BER performance of coherently detected differentially encoded BPSK',

'FontSize', 16);

X = xlabel('Eb/No [db]');

set(X, 'FontSize', 14);

Y = ylabel('BER');

set(Y, 'FontSize', 14);
```

B.1.2 Program: simulation for 1Mbps with 1Mbps 802.11b interference

```
%% 802.11b 1Mbps PHY link with 802.11b interference.

% This M code simulates DBPSK modulation and barker code spreading

% on a perfectly synchonized 802.11b link with other 802.11b devices

% interfering the communication. It calculates the BER

% rate at each SIR and plots the result with the assumption that SNR is

% equal to 35dB.


function [BER,BER_2, SIR] = test_inter(num_bit, M, bitrate, fs, fc )


% Specify a number of system constants.
% Spreading parameters
Barker=[1 -1  1  1 -1  1  1  1 -1 -1 -1].'; % Barker sequence
Barker2=[1 1 1 -1 -1 -1 1 -1 -1 1 -1].';
SpreadingRate=length(Barker);              % Spreading rate

% Upsampling rate
SamplesPerChip=8; %fs/(bitrate*11)
SNR = 35;
%% ------------------------------------------
% bit generator
%------------------------------------------
%rand('seed', 1);
bits_src = rand(1, num_bit) > 0.5; %instead of using randint which generates
% a random scalar that is either 0 or 1, with equal probability,
                              %we use rand since rand returns a pseudorandom,
%scalar value drawn from a uniform distribution on the unit interval.
bits_src = double(bits_src);
numofone = sum(bits_src)

bits_inter = rand(1, num_bit) > 0.5;
bits_inter = double(bits_inter);

bits_inter_2 = rand(1, 2*num_bit) > 0.5;
bits_inter_2 = double(bits_inter_2);

%% ------------------------------------------
% differential encoding & modulate & spreading
```

```
%-------------------------------------------
len_bits = length(bits_src);
% gray code
en_xsym = bi2de(reshape(bits_inter_2,2,len_bits).', 'left-msb');
en_xsym = en_xsym.';
for k=1:length(en_xsym)
    if en_xsym(k) == 0
        gr_en_xsym(k) = 0;
    elseif en_xsym(k) == 1
        gr_en_xsym(k) = 1;
    elseif en_xsym(k) == 2
        gr_en_xsym(k) = 3;
    else
        gr_en_xsym(k) = 2;
    end
end

d1(1)=0;   %-- reference bit is 0
d2(1)=0;
d3(1)=0;
for k = 1:len_bits
    d1(k+1)= xor(bits_src(k), d1(k));
    d2(k+1)= xor(bits_inter(k), d2(k));
end
for k = 1:len_bits
    d3(k+1)= mod((gr_en_xsym(k) + d3(k)), 4);
end
diff1 = d1(2:len_bits+1);
```

```
diff2 = d2(2:len_bits+1);

diff3 = d3(2:len_bits+1);

diff1 = 1-2*diff1; % 1-> -1

diff2 = 1-2*diff2; % 0-> 1

h = modem.pskmod(4); %Create a modulator object

mod_sym = modulate(h,diff3); % Modulate the signal bits


SpreadChips = reshape(Barker*diff1,[],1).';

SpreadInterChips = reshape(Barker*diff2,[],1).';

SpreadInterChips_2 = reshape(Barker*mod_sym,[],1).';


%% -----------------------------------------------------

% RRC Filter

%-------------------------------------------------------------

h = rcosine(1,SamplesPerChip,'sqrt', 0.01); % Transfer function of filter

Tx_Samples = rcosflt(SpreadChips,11*bitrate,SamplesPerChip*11*bitrate,

'filter',h);% Filter the data. Normalize power due to upsampling

length(Tx_Samples)

Tx_Inter_Samples = rcosflt(SpreadInterChips,

11*bitrate,SamplesPerChip*11*bitrate,'filter',h);% Filter the data.

length(Tx_Inter_Samples)

Tx_Inter_Samples_2 = rcosflt(SpreadInterChips_2,

11*bitrate,SamplesPerChip*11*bitrate,'filter',h);% Filter the data.

Tx_Samples=Tx_Samples.';

Tx_Inter_Samples=Tx_Inter_Samples.';

Tx_Inter_Samples_2=Tx_Inter_Samples_2.';

%% IF & Receiver Design

bit_width = 1/bitrate;
```

```
dt = bit_width/11/SamplesPerChip; % sampling period

t = dt .* [0 : length(Tx_Samples) - 1];


Ep = sum(Tx_Samples.^2)/length(Tx_Samples);

Tx_Samples_Complex = Tx_Samples.*exp(i.*2.*pi.*fc.*t)/sqrt(Ep);

Signal_Power = var(Tx_Samples_Complex)


Ep2 = sum(Tx_Inter_Samples.^2)/length(Tx_Inter_Samples);

Tx_Inter_Samples_Complex = Tx_Inter_Samples.*exp(i.*2.*pi.*fc.*t)./sqrt(Ep2);

Signal_Power2 = var(Tx_Inter_Samples_Complex)


Ep3 = sum(Tx_Inter_Samples_2.*conj(Tx_Inter_Samples_2))/

length(Tx_Inter_Samples_2);

Tx_Inter_Samples_2 = Tx_Inter_Samples_2.*exp(i.*2.*pi.*fc.*t)./sqrt(Ep3);

Signal_Power3 = var(Tx_Inter_Samples_2)

noise = 1/sqrt(2)*[randn(1,length(Tx_Samples_Complex)) +

j*randn(1,length(Tx_Samples_Complex))];

noise_signal = sqrt(Signal_Power./power(10, 0.1*SNR)).*noise;

10*log10(Signal_Power/var(noise_signal))

SIR = 0;

for k3 = 1:length(SIR)

    Amp = sqrt(Signal_Power./power(10, 0.1*SIR(k3)));

    Tx_Inter_Samples_Complex_Amp = Tx_Inter_Samples_Complex.*Amp; % fd =1MHz

    var(Tx_Samples_Complex)

    var(Tx_Inter_Samples_Complex_Amp)

    10*log10(var(Tx_Samples_Complex)/var(Tx_Inter_Samples_Complex_Amp))

    Rx_Samples_Input = Tx_Samples_Complex + Tx_Inter_Samples_Complex_Amp ;
    Rx_Match = rcosflt(Rx_Samples_Input.* exp (-i.*2.*pi.*fc.*t),
11*bitrate,SamplesPerChip*11*bitrate,'Fs/filter',h);
    length(Rx_Match);
    Rx_Match = Rx_Match.';
    decision_t = [(6*SamplesPerChip+1) : SamplesPerChip :
((length(SpreadChips)+5)*SamplesPerChip+1)]; % delay = 3
    Rx_Samples = Rx_Match(decision_t);
    % Despread - sample symbol
    Rx_Symbols=Barker.'*reshape(Rx_Samples,SpreadingRate,num_bit);
 % Multiply by Barker
    Rx_Symbols = Rx_Symbols/SpreadingRate;   % Make a row and normalize
    % demodulate
    for n = 1: length(Rx_Symbols)
        if real(Rx_Symbols(n)) > 0
            Rx_En_Bits(n) = 0;
        else
            Rx_En_Bits(n) = 1;
        end
    end
    numofone= sum(Rx_En_Bits)
    % Diff. Decod.
    Rx_En_Bits = [0, Rx_En_Bits];
    for m= 1:(length(Rx_En_Bits)-1)
        Rx_Bits(m) = xor(Rx_En_Bits(m+1), Rx_En_Bits(m));
    end
    length(bits_src);
    Rx_Bits;
    ErrorBits(k3) = sum(bits_src~=Rx_Bits);
    clear Rx_En_Bits;
end


for k = 1:length(SIR)
    Amp = sqrt(Signal_Power./power(10, 0.1*SIR(k)));
    Tx_Inter_Samples_Complex_Two = Tx_Inter_Samples_2.*Amp; % fd =1MHz
    Rx_Samples_Input_2 = Tx_Samples_Complex +
Tx_Inter_Samples_Complex_Two+ noise_signal;


    Rx_Match_2 = rcosflt(Rx_Samples_Input_2.* exp (-i.*2.*pi.*fc.*t),
11*bitrate,SamplesPerChip*11*bitrate,'Fs/filter',h);
% Filter the received data, we only need its real part;
    length(Rx_Match_2);
    Rx_Match_2 = Rx_Match_2.';
    decision_t = [(6*SamplesPerChip+1) : SamplesPerChip :
((length(SpreadChips)+5)*SamplesPerChip+1)]; % delay = 3
    Rx_Samples_2 = Rx_Match_2(decision_t);
    % Despread - sample symbol
    Rx_Symbols_2=Barker.'*reshape(Rx_Samples_2,SpreadingRate,num_bit);
% Multiply by Barker
    Rx_Symbols_2 = Rx_Symbols_2/SpreadingRate;   % Make a row and normalize
    % demodulate
    for n = 1: length(Rx_Symbols_2)
        if real(Rx_Symbols_2(n)) > 0
            Rx_En_Bits_2(n) = 0;
        else
            Rx_En_Bits_2(n) = 1;
        end
    end
    % Diff. Decod.
    Rx_En_Bits_2 = [0, Rx_En_Bits_2];
    for m= 1:(length(Rx_En_Bits_2)-1)
        Rx_Bits_2(m) = xor(Rx_En_Bits_2(m+1), Rx_En_Bits_2(m));
    end
    length(bits_src);
    ErrorBits_2(k) = sum(bits_src~=Rx_Bits_2);
    clear Rx_En_Bits_2;
end
BER = ErrorBits./length(bits_src)
BER_2 = ErrorBits_2./length(bits_src)
figure(1);
L = semilogy(SIR, BER, SIR, BER_2);
title('1Mbps 802.11b DSSS performance with IEEE 802.11b interference');
set(L, 'LineWidth', [2]);
axis([-20 5 1e-3 1e-0]);
X = xlabel('SIR [db]');
set(X, 'FontSize', 14);
Y = ylabel('BER');
set(Y, 'FontSize', 14);
```