Simulation of IEEE 802.11b Wi-Fi Standard Payload at 2 Mbps and 11 Mbps Data Rates

Kevin Jiang

***Abstract*—The purpose of this paper is to give an overview of the IEEE 802.11b Wi-Fi standard, its channel bandwidths, and its Physical Layer properties. At the end of the paper, a simulation of a payload going from a transmitter to a receiver using the modulation technique of 802.11b at a data rate of 2 Mbps is shown. A simulation of a payload leaving a transmitter at a data rate of 11 Mbps is also shown, as a different modulation technique is used for this data rate.**

Introduction

The IEEE 802.11b Wi-Fi Protocol was the first Wi-Fi protocol to have achieved widespread use. This protocol was released in 1999, where it was quickly adopted by businesses and consumers due to its increased throughput compared to the original 802.11 standard and its cheap pricing. Operating at 2.4 GHz and using DSSS modulation, 802.11b was the foundation upon which later protocols were built upon. In the following sections, detailed descriptions of the bandwidths, packet format, and modulation of 802.11b are provided.

Channel Bandwidths

The 802.11b standard operates in the 2.4 to 2.5 GHz band, which is a microwave frequency band also known as the Industrial Scientific Medical band. This makes the 802.11b less reliable as it is vulnerable to all kinds of signal interference from other microwave devices. The 2.4 GHz band is divided into 14 different bands, each 20 MHz in bandwidth and space 5 MHz apart. These channels overlap to some degree; in fact, the only channels that don't overlap with each other are channels 1, 6, and 11. To mitigate the bleeding of one channel into the other, "spectral masks", which are essentially bandpass filters, are used to ensure that signals from one channel are attenuated to a sufficient degree when they enter the overlapping frequency regions.

Packet Format

To transmit data through the Physical Layer, 802.11b transmits bursts of packets. Packets are sets of data with a predetermined structure. For 802.11b, each packet consists of three fields: The Preamble, the Header, and the Payload. For 802.11b, the three fields function as follows:

-The Preamble contains two subfields: a synch field and a start frame delimiter (SFD) field. The synch field is a sequence of 128 1's that allow the receiver to synchronize and lock on to the signal. The SFD field is a sequence of 16 bits that specify the start of the

frame. The Preamble field is modulated using DBPSK (more on modulation in the following section).

-The Header contains four subfields: a signal field (8 bits), which indicates the data rate of the payload, a service field (8 bits), which specifies the protocol being used, a length field (16 bits) that specifies the size of the frame in bytes, and a control error field (16 bits) used to check for any errors in the header. The Header field maybe modulated with DBPSK or DQPSK.

-The Payload is just the data that is being sent in the packet. Depending on the data rate, the Payload may be modulated with DBPSK, DQPSK, or CCK.

802.11b also uses two different preamble and header types: Long and Short. The Long preamble is twice the length of the Short preamble (72 bits). The Long Header is modulated using DBPSK, while the Short Header is modulated using DQPSK instead of DBPSK, reducing the amount of chips transmitted in the Header by half. Long Preambles and Headers have more bits/chips and thus more error checking capacity, but they also take twice as long to transmit. Short Preambles and Headers are used for packets that require less error checking and more speed.

Modulation Techniques

802.11b supports four data rates: 1Mbps, 2 Mbps, 5.5 Mbps, and 11 Mbps. Each data rate used a different modulation technique for the Payload.

1 Mbps: DBPSK is used, where 1 bit is modulated to 1 symbol (1 or -1). In addition, a Barker code, which is a 11-bit pseudo-noise sequence, is multiplied to the signal after DBPSK modulation. At the receiver, the Barker Code can be multiplied again to decode the signal, because multiplying the Barker Code two times just gives you a sequence of 1's. Since the Barker code is an 11-bit sequence, the transmitted signal becomes 11 times as long. The modulation rate is 1 bit to 11 chips and is named DSSS1M.

2 Mbps: DQPSK is used to modulate 2 bits to 1 symbol. Barker code is used again here. The modulation rate is 2 bits to 11 chips and is named DSSS2M.

5.5 Mbps: CCK is used to modulate 4 bits to 8 chips. The coding scheme is shown below.

11 Mbps: CCK is used to modulate 8 bits to 8 chips. The coding scheme is shown below.

The CCK in 802.11b makes use of phases to create vectors with auto correlations of zero. Every group of 8 bits (b7, b6, b5, b4, b3, b2, b1, b0) is transformed into a complex vector with eight terms. The resulting vector has the form:

$$c = \left\{ \begin{array}{c} e^{j(\varphi_1+\varphi_2+\varphi_3+\varphi_4)}, e^{j(\varphi_1+\varphi_3+\varphi_4)}, e^{j(\varphi_1+\varphi_2+\varphi_4)}, e^{j(\varphi_1+\varphi_4)}, \\ e^{j(\varphi_1+\varphi_2+\varphi_3)}, e^{j(\varphi_1+\varphi_3)}, e^{j(\varphi_1+\varphi_2)}, e^{j(\varphi_1)} \end{array} \right\}$$

Each phase term (φ_1, φ_2, φ_3, φ_4) of the complex exponential is determined by looking at the four pairs of bits in the current set of 8 bits. The table below shows which pairs of bits

correspond to which phase term, and what combinations of bits results in which phase value.

Bit Pairs	Phase Term	Bit Combination	Phase Value
(b1, b0)	φ_1	(0,0)	0
(b3, b2)	φ_2	(0, 1)	π
(b5, b4)	φ_3	(1, 0)	π/2
(b7, b6)	φ_4	(1, 1)	- π/2

Simulations

The simulation involves a transmitter and receiver using the 802.11b protocol at 2 Mbps. Only the transmitting techniques were specified, so the receiving technique was written in order to match the BER of the 802.11b transmission as closer to the theoretical BER through an AWGN channel. The simulation involves QPSK modulating 2000 bits to 1000 symbols, using Barker Coding to encode it, and sending it through an AWGN channel. The receiver then decodes the Barker Code by multiplying the received signal again and averaging each group of 11 symbols. The symbols are then demodulated and converted back to bits, where a BER is computed. This process is done through several values of SNR and for 2000 iterations to improve the final BER curve.

The simulation results are as shown:

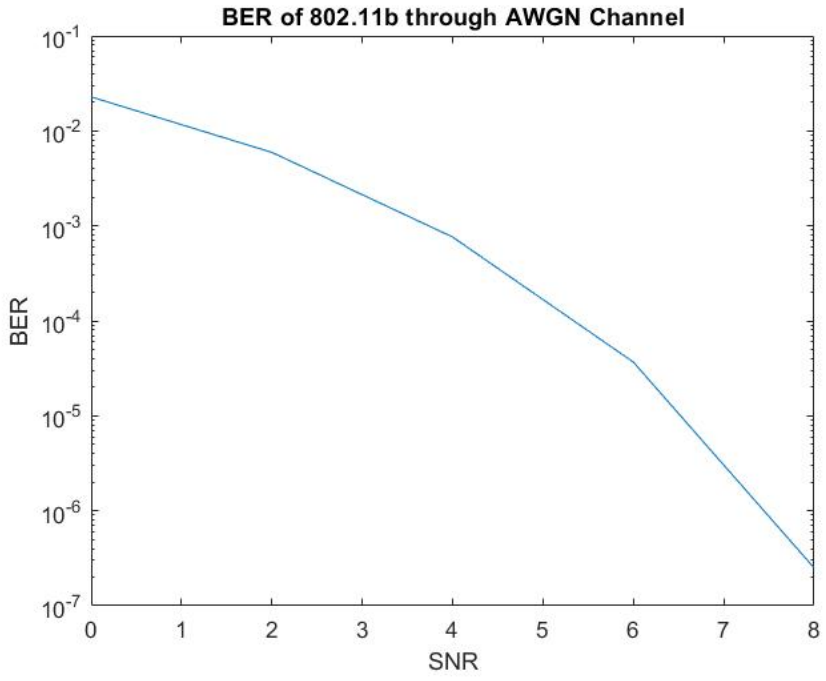

Figure 1: BER of 802.11b through AWGN Channel

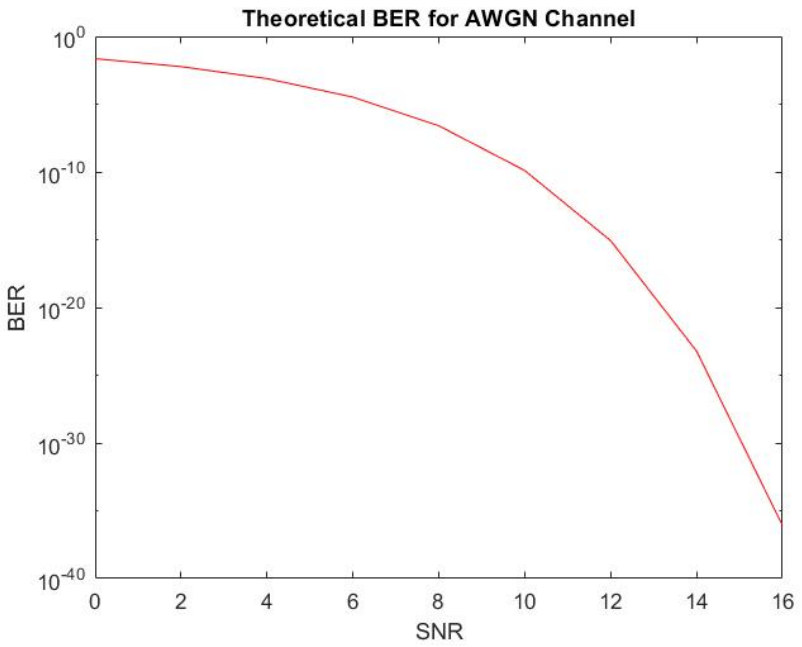

Figure 2: Theoretical BER through AWGN Channel

Only bit error rates for up to a SNR of 8 was generated for the 802.11b simulation, because after that the BER would become too low and drop to 0. This could possibly be improved with more simulations. As shown in Figure 1, a BER on the order of 10^{-7} was achieved at SNR 8. This is the same BER that the theoretical curve achieves at SNR 8 in Figure 2. In fact, when plotting the two curves on the same axes, they perfectly overlap, which was the reason for displaying them on two separate graphs here.

Initially, the Barker Code was so strong that BER was consistently 0 for the 802.11b simulation. This problem was solved at scaling the noise in the AWGN channel to match the higher chip rate of the Barker Code. The method of scaling the noise is:

$$E_S/N_0 \text{ (dB)} = 10\log_{10}(T_{sym}/T_{samp}) + SNR \text{ (dB)}$$

E_S/N_0 is the symbol power to noise power ratio and is used as the SNR of the AWGN channel. For QPSK, T_{sym}/T_{samp} is just log2(M), where M is 4 for QPSK. Because of the Barker Code increasing the chip rate by 11 times, the final sampling period ratio is divided by log2(M)/11.

The simulation of the 11 Mbps has no displayable results. Only the CCK coded signal was generated. Because there was no documentation on how to receive and decode a CCK coded signal, I was unable to write my own receiver to decode the complex CCK code for this project. The code for generating a CCK coded signal is shown in the Appendix.

Appendix

Code for 2Mbps simulation:

```
% Wireless Comm Project 2
% Simulation of IEEE 802.11b 2Mbps Transmitter & Reciever for 1 ms
clear;close all;clc
nSym = 1000; % 2 Mbps is 1 Msymbols/s, in a period of 1 ms 1000 symbols are
sent
numIter = 2000; % The simulation is run 1000 times to generate a better BER
curve
SNR_Vec = 0:2:16; % Vector containing values of SNR to iterate through
lenSNR = length(SNR_Vec);
M = 4; % We are using QPSK for 2 Mbps
ber = zeros(1, lenSNR); % Vector to hold mean bit error rates
berVec = zeros(numIter, lenSNR); % Vector for holding ber at each SNR for all
iterations
barkcode = transpose([1, -1, 1, 1, -1, 1, 1, 1, -1, -1, -1]); % The Barker Code
% Run the simulation numIter amount of times
for i = 1:numIter
bits = randi([0 1], 1, nSym*log2(M)); % Generate random bits, twice the number
of symbols
bitLen = length(bits);
% Convert pairs of binary to decimal before using QPSK
msg = zeros(1, bitLen/log2(M));
msgindex = 1;
for bitindex = 1:log2(M):(bitLen)
msg(msgindex) = bi2de(bits(bitindex:bitindex+log2(M)-1), 'left-msb');
msgindex = msgindex+1;
end
    for j = 1:lenSNR % one iteration of the simulation at each SNR Value
    tx = qammod(msg,M); % QPSK modulate the signal
    txCoded = tx.*barkcode; % Multiply by Barker code
    txNoisy = awgn(txCoded,SNR_Vec(j)+ 10*log10(log2(M)/11)); % AWGN Channel,
scaled to fit chip rate of Barker Code
    rxDecoded = mean(txNoisy.*barkcode,1); % Multiply again by Barker Code and
find the mean
    rxDemod = qamdemod(rxDecoded,M); % Demodulate
    % Converting the symbols back to bits
    rxMSG = zeros(1, bitLen);
    rxMSGindex = 1;
    for rxDemindex = 1:1:length(rxDemod)
    rxMSG(rxMSGindex:rxMSGindex+log2(M)-1) = de2bi(rxDemod(rxDemindex),
log2(M), 2, 'left-msb');
    rxMSGindex = rxMSGindex + log2(M);
    end
    % Compute and store the BER for this iteration
    [~, berVec(i,j)] = biterr(bits(1:end), rxMSG(1:end)); % We're interested in
the BER, which is the 2nd output of BITERR
    end % End SNR iteration

end % End numIter iteration
% Compute and plot the mean BER
ber(1,:) = mean(berVec,1);
figure;
semilogy(SNR_Vec, ber(1,:));
title('BER of 802.11b through AWGN Channel');
xlabel('SNR');
```

```
ylabel('BER');
% Compute the theoretical BER for an AWGN channel
M = 4;
berTheory = berawgn(SNR_Vec + 10*log10(log2(M)),'psk',M,'nondiff');
figure;
semilogy(SNR_Vec,berTheory,'r')
title('Theoretical BER for AWGN Channel');
xlabel('SNR');
ylabel('BER');
```

Code for 11Mbps simulation:

```
% CCK transmitter only implementation according to 802.11b methods
clear;close all;clc
numIter = 1; % The number of iterations of the simulation
nSym = 11000/2; % Let's say this is a simulation of 11 Mbps running for 0.01 s,
the number of symbols is half of number of bits
bit2phase = [0, pi, pi/2, -pi/2]; % Phase array for CCK
chan = 1; % Only applying AWGN channel
M = 4;
bits = randi([0 1], 1, nSym*log2(M)); % Generate random bits
bitLen = length(bits);
chips = zeros(1, bitLen); % cck chips
dibits = zeros(1,4); % storage array for current iteration of 8 bits, converted
to decimal
phi = zeros(1,4); % storage array for current iteration of phases
% Code the signal using CCK
for cckindex = 1:8:(bitLen-7) % Convert 8 bits to four phases. Each two bits
can represent one phase
    for dibitindex = 1:4 % Find the decimal number associated with each two
bits, convert to phase
        dibits(dibitindex) = bi2de(bits(cckindex+2*(dibitindex-
1):cckindex+1+2*(dibitindex-1)), 'left-msb');
        phi(dibitindex) = bit2phase(dibits(dibitindex)+1); % Decimal is
converted to phase
    end

chips(cckindex:cckindex+7)=[exp(1j*sum(phi)),exp(1j*(phi(1)+phi(3)+phi(4))),...
        exp(1j*(phi(1)+phi(2)+phi(4))),-
exp(1j*(phi(1)+phi(4))),exp(1j*(phi(1)+phi(2)+phi(3))),...
        exp(1j*(phi(1)+phi(3))),-exp(1j*(phi(1)+phi(2))),exp(1j*phi(1))]; % Use
the cck formula to turn phases into 8 chips
end
```