2015 Summer Term Alumni Research Scholars Research Report

Jooyoun Hong

1. Introduction

Many engineering applications demand mobility, and wireless system has become much more critical than in the past. High-speed data transformation in wireless environment requires broad frequency spectrum, but the actual frequency range that people can use for communication purpose is very limited. It became absolute necessary to come up with a way to harness the data traffics in given spectrum. One of the ways to improve current wireless system on efficient data transformation is applying the Network Coding scheme in to wireless environment. Under Professor Bobak Nazer's guidance, I studied and performed experiment how this scheme can be developed and used in practice.

2. Backgrounds

A. Network Coding

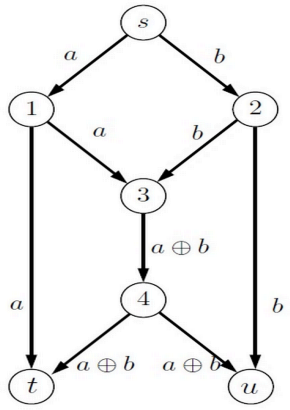


Figure 1. Network Coding

The basic concept of network coding is that it achieves efficient way to send information to the receivers by inserting a relay between the source and the receivers. The relay combines multiple messages from the source into one composite message and broadcast it to the receivers. With the composite message, each receiver can decode the wanted messages with the information they already have. Figure 1 describes the diagram of network coding. This scheme has been implemented for some applications of wired system, and many studies have proven that it is feasible even in wireless environment. The simple comparison between general routing scheme and network coding is the following.

B. *Network Coding in Wireless Environment and Physical Layer Network Coding*

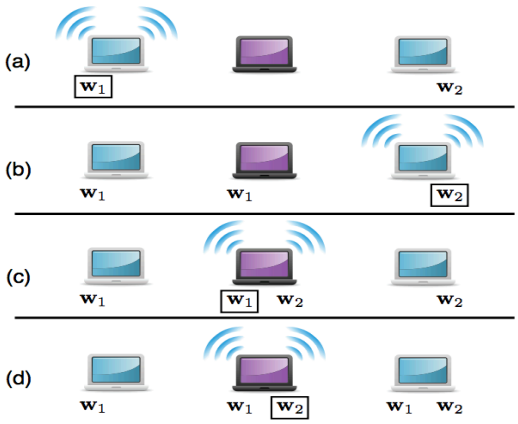


Figure 2. General Routing

Assuming all devices are half duplex, there are two separate end devices try to exchange data to each other through the relay between them. They cannot communicate independently.
With routing strategy, the exchange takes four time slots. (1) Device 1 transmits its message to the relay. (2) Device 2 transmits its message to the relay. (3) The relay sends the message from device 2 to device 1. (4) The relay sends the message from device 1 to device 2. (Figure 2)

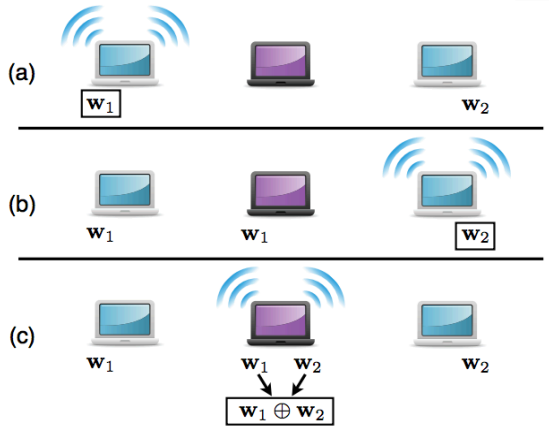


Figure 3. Network Coding

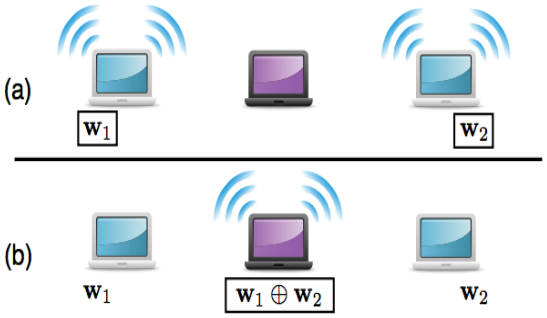


Figure 4. Physical Layer Network Coding

With network coding strategy, the exchange takes three time slots. (1) Device 1 transmits its message to the relay. (2) Device 2 transmits its message to the relay. (3) The relay combines the two messages by logical XOR and broadcasts the composite. Now the both end devices have enough information to extract the wanted messages. (Figure 3.)

However, there is even more efficient method called Physical Layer Network Coding. Since the scheme does not require for the relay to know the individual messages, the relay does not need to spend two time slots listening the two messages separately. If the two signals can be summed in the air after they get transmitted from the sources, the relay only needs one time slot to hear the composite signal. (1) The two sources transmit their messages simultaneously, and the relay receives the composite of the two. (2) The relay broadcasts the composite signal to the end devices. (Figure 4) The entire process, therefore, takes only two time slots. The performance is significantly improved compared to the routing strategy.

Assuming BPSK (bit 1 maps to 1, bit 0 maps to -1), let b_1, b_2 denote the bits and s_1, s_2 denote the symbols from messages of $device_1$ and $device_2$. The relay can transform the combined signals to $b_1 \oplus b_2$ by ML Decision Rule. z denotes Gaussian noise.

1) $b_1 = 0, b_2 = 0 \rightarrow s_1 + s_2 + z = (-1) + (-1) + z = -2 + z \rightarrow b_1 \oplus b_2 = 0$
2) $b_1 = 0, b_2 = 1 \rightarrow s_1 + s_2 + z = (-1) + (+1) + z = 0 + z \rightarrow b_1 \oplus b_2 = 1$
3) $b_1 = 1, b_2 = 0 \rightarrow s_1 + s_2 + z = (+1) + (-1) + z = 0 + z \rightarrow b_1 \oplus b_2 = 1$
4) $b_1 = 1, b_2 = 1 \rightarrow s_1 + s_2 + z = (+1) + (+1) + z = +2 + z \rightarrow b_1 \oplus b_2 = 0$

(Firgure 5)

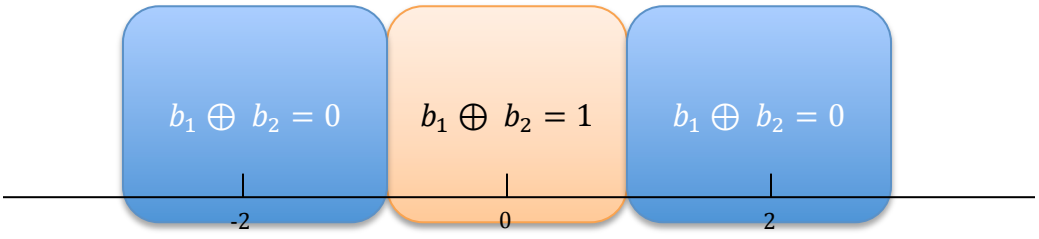


Figure 5. ML Decision

C. Orthogonal Frequency Division Multiplexing (OFDM)

In this research, the system is encoded in OFDM. The one of the most outstanding advantages of using this scheme is that OFDM does multicarrier transmission. Single carrier system is vulnerable to frequency-selective fading due to its wide use of frequency spectrum, and it is highly likely to alter the ML Decision above. Multicarrier system divides the given spectrum by number of the subcarriers so that it achieves flat fading. Each subcarrier has its own carrier frequency, and all of them are orthogonal to each other.

3. WARPLab Experiment

A. SISO (Single Input Single Output) WARPLab Board Experiment

I conducted SISO experiment with the boards to get familiar with WARPlab Boards.

1) 1.5m Distance between the Transmitter and Receiver without Obstacle (Line of Sight)

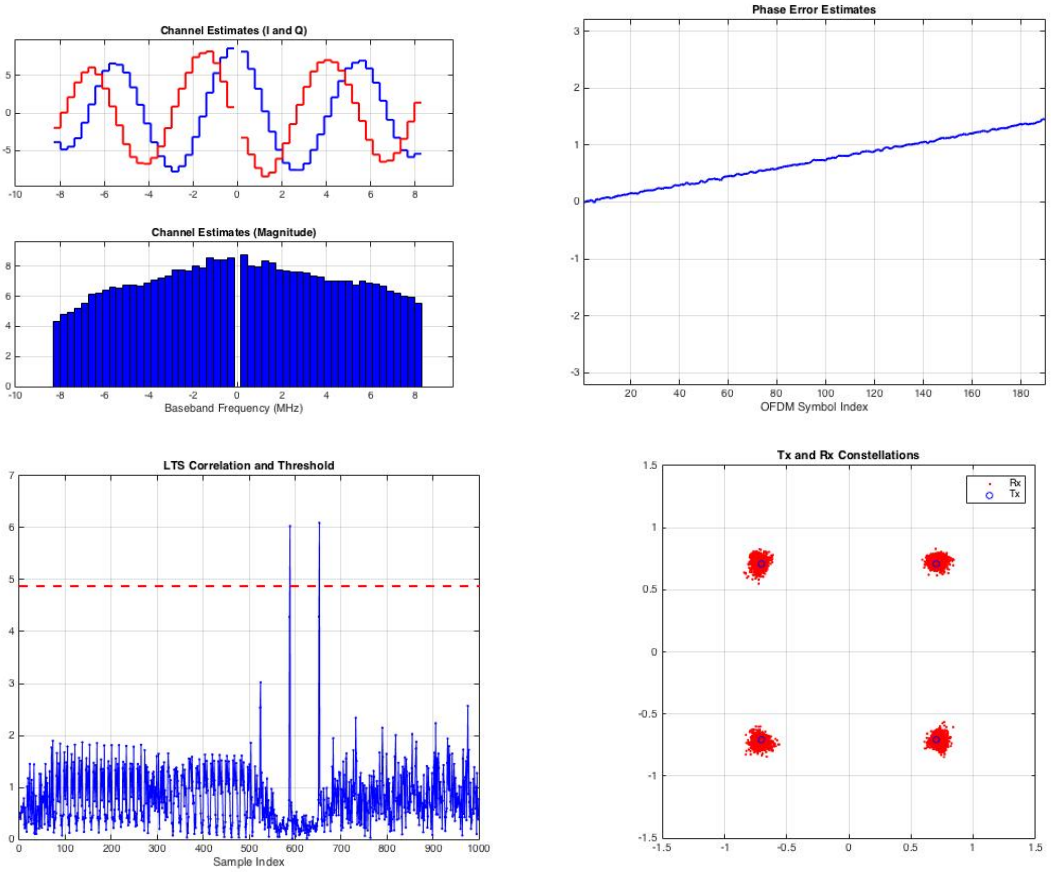


Figure 6. SISO WARP Experiment (LOS)

Since there was no obstacle between the two antennas, the channel is almost flat, and the phase error is very minor. Also there was no noise as high as LTS threshold, and the constellation map has clear division. The bit error was zero.

2) 1.5m Distance between the Transmitter and Receiver with Obstacles

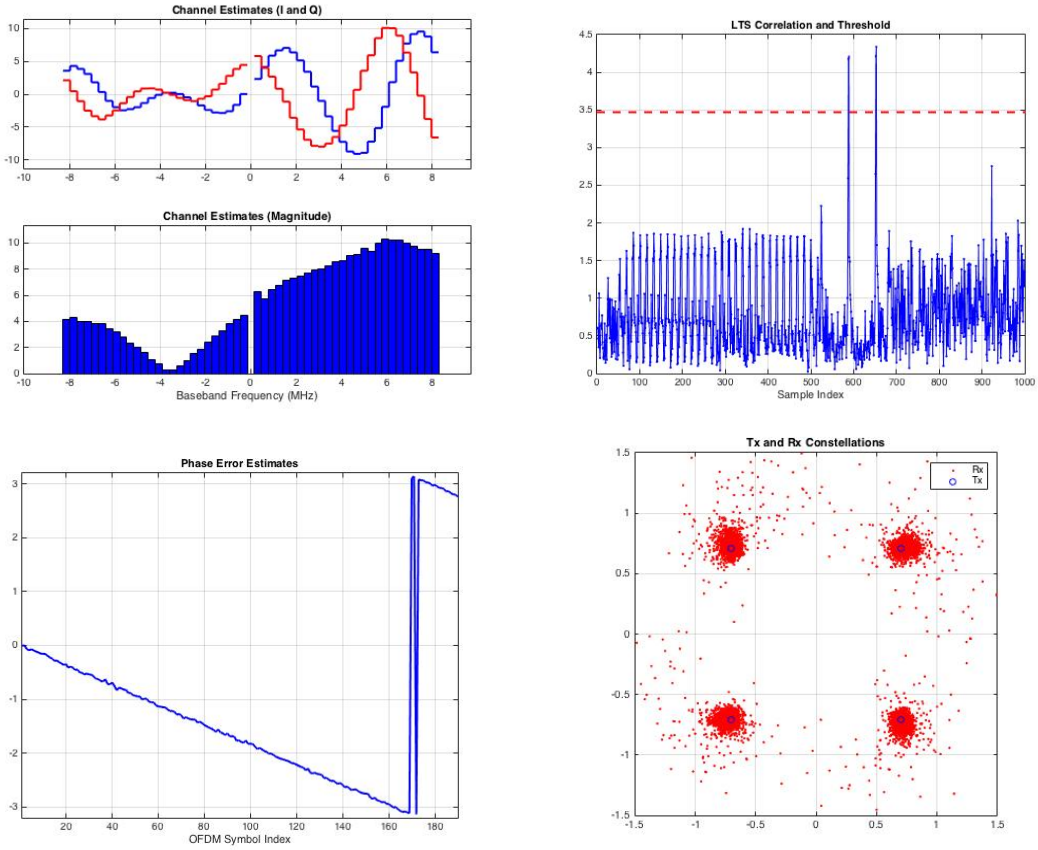
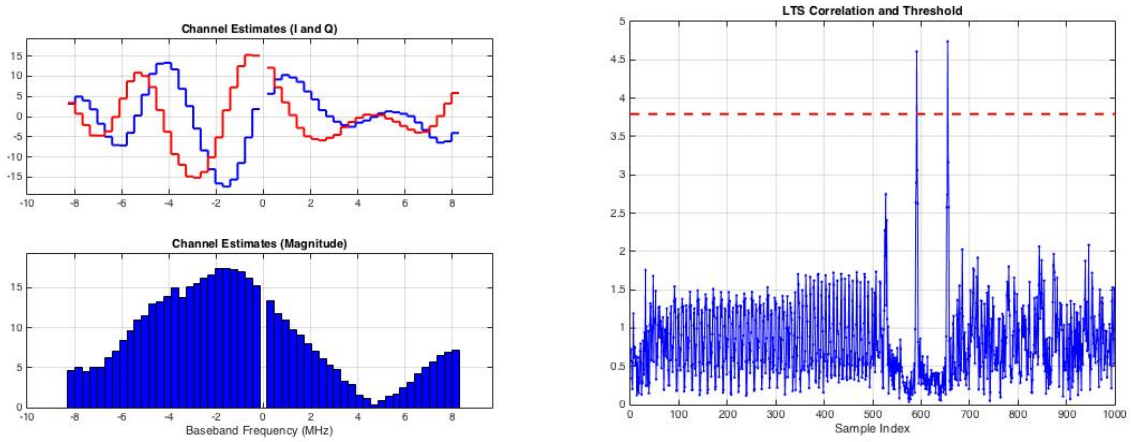


Figure 4. SISO WARP Experiment (1.5m with obstacles)

Same distance as the previous experiment, but I put a wall and other obstacles between the two antennas. The channel estimation (magnitude) around -4MHz is significantly faded resulting distortion of the signal. However, there was no massive noise that reaches the LTS threshold, and constellation map is still neat. The bit error was zero.

3) 2m Distance between the Transmitter and Receiver with Obstacles



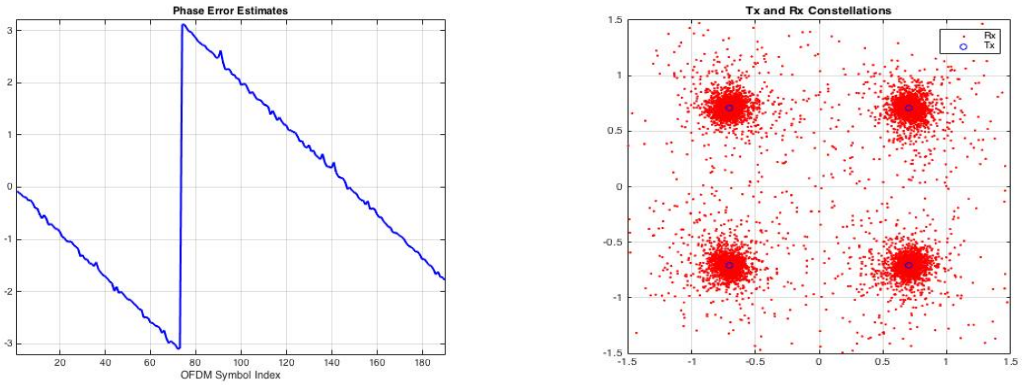


Figure 5. SISO WARP Experiment (2m with obstacles)

The distance between two antennas was 2m, and I put more obstacles than previous experiment. This time, channel estimation (magnitude) around 4MHz was massively faded. The constellation map is much more spread out than the previous one. The bit error was more than zero.

B. warp_7_4_0_miso_OFDM_2xWARP.m

This code simulates the physical layer network coding strategy with OFDM. Once the relay receives the composite of the two signals, it can never guess what the originals were. Therefore, it is important to check if the two signals are synchronized enough to make the correct ML decision rule, and the preamble of signals plays significant role. Preamble consists of STS (Short Training Symbols) and LTS (Long Training Symbols). Each transmitted message has two LTS at certain indexes of the sample. Figure 9 is LTS Correlation plot from this code without any noise. The LTS from the first message is 353, 417 and the second message is 545, 609. These indexes are changeable depending on simulations, but it is important that these four LTS share distance of 64, 128, and 64. Especially the second LTS from the first message and the first LTS from the second message should have distance that is as close to 128 as possible to achieve correct ML Decision.

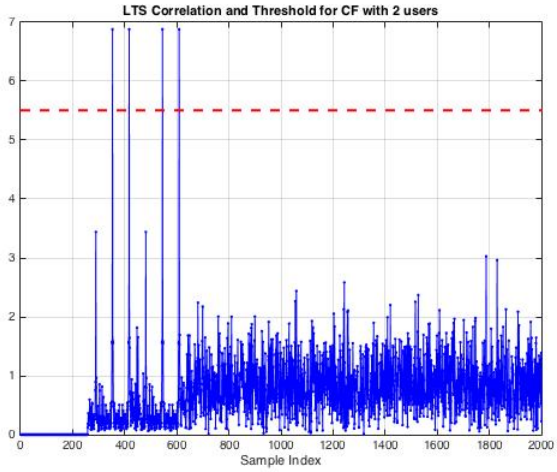


Figure 9. Noise Free LTS Correlation

These specified distances also allow the system to separate the original preambles from significant noises. In practice, a lot of noises are high enough to go above the LTS thresholds. Main role of the LTS is to notify the system that the payload will be delivered soon, and if the noise that can go above LTS threshold and the system misunderstands that noise as LTS, the payload cannot be delivered correctly. The chance for the noise that is high as LTS also to share the same distances with other LTS is very rare. Checking the distances between potential LTS's allows picking up the real LTS. Assuming there will be a lot of noises that go above the threshold, it is needed to study efficient algorithm to calculate the distances and find real LTS.

Along with the preamble, pilots are inserted to each signal to measure CFO and estimate the channel. CFO is calculated based on the difference between original pilots and transmitted pilots of each signal. Currently CFO is fixed by averaging the two CFO's then pushing the signals back by that amount. Measuring the phase difference between the two signals themselves also seems to be critical, so it needs to be studied further. The code generates the bit error rate by comparing xored bits of original signals, and decoded received signal.

4. MATLAB Code (summer_research_hong.m)

```
%% Summer 2015 Jooyoun Hong

N_SYMS = 10000;     % Number of Symbols
N_BITS = 164;       % Number of Bits

CFO_CORR = 0;
N_USER = 2;


NOISE = 'CFO'; % FREE, AWGN, CFO

N_SAMPLE = floor(N_SYMS/N_BITS); % Number of Samples per Bits

carr_freq = 4 * N_BITS;


% Random Bit Generation

tx_data_1 = randi(2, 1, N_BITS) - 1;
tx_data_2 = randi(2, 1, N_BITS) - 1;
if (N_USER == 4)
    tx_data_3 = randi(2, 1, N_BITS) - 1;
    tx_data_4 = randi(2, 1, N_BITS) - 1;
end

% Upsampling
perfect_filter = ones(1,N_SAMPLE); % square one
filtered_data_1 = kron(tx_data_1, perfect_filter);
filtered_data_2 = kron(tx_data_2, perfect_filter);
if (N_USER == 4)
   filtered_data_3 = kron(tx_data_3, perfect_filter);
   filtered_data_4 = kron(tx_data_4, perfect_filter);
end


Nb = N_SAMPLE * N_BITS;
t = 1:N_SYMS;
bi_sampled_data_1 = zeros(1, N_SYMS);
bi_sampled_data_1(1:Nb) = filtered_data_1(1:Nb);
bi_sampled_data_2 = zeros(1, N_SYMS);
bi_sampled_data_2(1:Nb) = filtered_data_2(1:Nb);
```

```
42
43  if(N_USER == 4)
44      bi_sampled_data_3 = zeros(1, N_SYMS);
45      bi_sampled_data_3(1:Nb) = filtered_data_3(1:Nb);
46      bi_sampled_data_4 = zeros(1, N_SYMS);
47      bi_sampled_data_4(1:Nb) = filtered_data_4(1:Nb);
48  end
49
50  %% BPSK
51
52  modvec_bpsk   =  (1/sqrt(2))  .* [-1 1];
53  mod_fcn_bpsk  = @(x) complex(modvec_bpsk(1+x),0);
54
55
56  % Map the data values on to complex symbols
57  % BPSK
58  sampled_data_1 = arrayfun(mod_fcn_bpsk, bi_sampled_data_1);
59  sampled_data_2 = arrayfun(mod_fcn_bpsk, bi_sampled_data_2);
60  if (N_USER == 4)
61     sampled_data_3 = arrayfun(mod_fcn_bpsk, bi_sampled_data_3);
62     sampled_data_4 = arrayfun(mod_fcn_bpsk, bi_sampled_data_4);
63  end
64
65
66  signal_1 = sampled_data_1.*exp(i*2*pi*carr_freq/N_SYMS*t);
67  signal_2 = sampled_data_2.*exp(i*2*pi*carr_freq/N_SYMS*t);
68
69  if (N_USER == 4)
70      signal_3 = sampled_data_3.*exp(i*2*pi*carr_freq/N_SYMS*t);
71      signal_4 = sampled_data_4.*exp(i*2*pi*carr_freq/N_SYMS*t);
72  end
73  %%---------------------------------------- NOISE SELECTION ------------------------------------------------
74
75  switch (NOISE)
76  case 'FREE'
77      tx_payload_1 = signal_1;
78      tx_payload_2 = signal_2;
79
80      if (N_USER == 4)
81          tx_payload_3 = signal_3;
82          tx_payload_4 = signal_4;
83      end
84
85  case 'CFO'
86      payload_cfo_1 = 2e-5;
87      payload_cfo_2 = 4e-5;
88      tx_payload_1 = signal_1 .* exp(i*2*pi*payload_cfo_1*t);
89      tx_payload_2 = signal_2 .* exp(i*2*pi*payload_cfo_2*t);
90
91      if(N_USER == 4)
92          payload_cfo_3 = 3e-5;
93          payload_cfo_4 = 4e-4;
94          tx_payload_3 = signal_3 .* exp(i*2*pi*payload_cfo_3*t);
95          tx_payload_4 = signal_4 .* exp(i*2*pi*payload_cfo_4*t);
96      end
97
98  end
99
100
101 %%------------------------------------------------- RX -------------------------------------------------
102 rx_signal_original = tx_payload_1 + tx_payload_2;
103 if (N_USER == 4)
104     rx_signal_original = rx_signal_original + tx_payload_3 + tx_payload_4;
105 end
106
107
108 if (CFO_CORR == 1)
109     if(N_USER == 2)
110         cfo_ave = mean([payload_cfo_1;payload_cfo_2]);
111     elseif (N_USER == 4)
112         cfo_ave = mean([payload_cfo_1;payload_cfo_2;payload_cfo_3;payload_cfo_4]);
113     end
114
115 rx_signal_original = rx_signal_original.*exp(-1i*2*pi*cfo_ave*t);
116 end
117
118 rx_signal = rx_signal_original.*exp(-1i*2*pi*carr_freq/N_SYMS*t); % undo signal frequency
119
120 %% ---------------------------------------- DOWNSAMPLING ------------------------------------------------
121 count = 0;
122 rx_data_1 = zeros(1,N_BITS);
123
124 for count = 0:N_BITS-1
125     rx_data_1(count + 1) = mean(rx_signal((1+(count*floor(N_SYMS/N_BITS))):(floor(N_SYMS/N_BITS)+(count*floor(N_SYMS/N_BITS)))
        ));
126     count = count + 1;
127 end
128
```

```
%%---------------------------------------------- Decode ------------------------------------------------------

if(N_USER == 2)
    a1 = 1;
    a2 = 1;
elseif(N_USER == 4)
    a1 = 0;
    a2 = 0;
end

demod_fcn_bpsk = @(x) double(mod(round(( real(x)/(sqrt(2)) + (a1+a2)/2)), 2) );
rx_data = arrayfun(demod_fcn_bpsk, rx_data_1);

if(N_USER == 2)
    tx_data = bitxor(tx_data_1, tx_data_2);
elseif(N_USER == 4)
    tx_data_xor_1 = bitxor(tx_data_1, tx_data_2);
    tx_data_xor_2 = bitxor(tx_data_3, tx_data_4);
    tx_data = bitxor(tx_data_xor_1, tx_data_xor_2);
end

errorsum = sum (rx_data ~= tx_data)



cf = 0;

if (N_USER == 2)
    cf = cf + 1;
    figure(cf); clf;
    subplot(3,1,1);
    plot(t, real(signal_1), 'linewidth', 2, 'color','b');
    hold on;
    plot(t, sampled_data_1, 'linewidth', 1, 'color','r');
    axis([1,N_SYMS,-2.5,2.5]);
    xlabel('n');
    legend('BPSK','Binary Message');

    subplot(3,1,2);
    plot(t, real(signal_2), 'linewidth', 2, 'color','b');
    hold on;
    plot(t, sampled_data_2, 'linewidth', 1, 'color','r');
    axis([1,N_SYMS,-2.5,2.5]);
    xlabel('n');
    legend('BPSK','Binary Message');

    subplot(3,1,3)
    plot(t, real(rx_signal_original), 'linewidth', 2, 'color','b');
    hold on;
    plot(t, rx_signal, 'linewidth', 1, 'color','r');
    axis([1,N_SYMS,-2.5,2.5]);
    xlabel('n');
    legend('BPSK','Binary Message');
elseif (N_USER == 4)
    cf = cf + 1;
    figure(cf); clf;
    subplot(5,1,1);
    plot(t, real(signal_1), 'linewidth', 2, 'color','b');
    hold on;
    plot(t, sampled_data_1, 'linewidth', 1, 'color','r');
    axis([1,N_SYMS,-2.5,2.5]);
    xlabel('n');
    legend('BPSK','Binary Message');

    subplot(5,1,2);
    plot(t, real(signal_2), 'linewidth', 2, 'color','b');
    hold on;
    plot(t, sampled_data_2, 'linewidth', 1, 'color','r');
    axis([1,N_SYMS,-2.5,2.5]);
    xlabel('n');
    legend('BPSK','Binary Message');

    subplot(5,1,3);
    plot(t, real(signal_3), 'linewidth', 2, 'color','b');
    hold on;
    plot(t, sampled_data_3, 'linewidth', 1, 'color','r');
    axis([1,N_SYMS,-2.5,2.5]);
    xlabel('n');
    legend('BPSK','Binary Message');

    subplot(5,1,4);
    plot(t, real(signal_4), 'linewidth', 2, 'color','b');
    hold on;
    plot(t, sampled_data_4, 'linewidth', 1, 'color','r');
    axis([1,N_SYMS,-2.5,2.5]);
    xlabel('n');
    legend('BPSK','Binary Message');
```

```
216
217     subplot(5,1,5)
218     plot(t, real(rx_signal_original), 'linewidth', 2, 'color','b');
219     hold on;
220     plot(t, rx_signal, 'linewidth', 1, 'color','r');
221     axis([1,N_SYMS,-2.5,2.5]);
222     xlabel('n');
223     legend('BPSK','Binary Message');
224 end
```

The code generates random bits according to the value of parameter N_BITS. The code can simulate four users by setting N_USER to four. The number of symbols (N_SYMS) is set to be 10000, and the number of samples per one bit (N_SAMPLE) is $\frac{N\_SYMS}{N\_BITS}$. In the upsampling part, I used the perfect square filter, but in practice, Raised Cosine Filter is commonly used. The MATLAB function 'kron' duplicates (samples) each bit as many as N_SAMPLE.

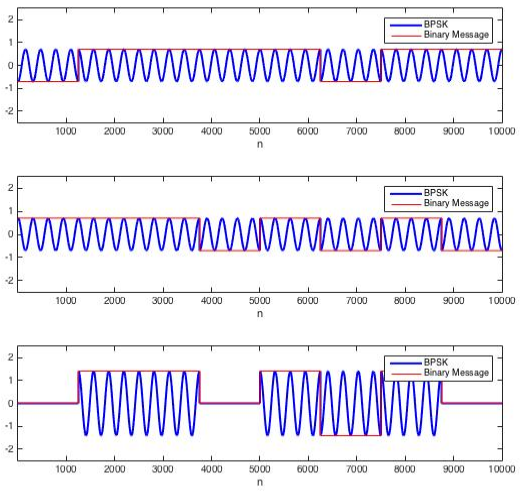


Figure 10. Noise Free 8 Bit Transmission

BPSK Modulation scheme maps bit 0 to symbol -1 and bit 1 to symbol 1. Every symbol is multiplied by $e^{j2\pi f_c t}$, where f_c denotes carrier frequency. f_c is calculated $\frac{4*N\_BITS}{N\_SYMS}$. As the sinusoid wave gets multiplied by 1's and -1's, the signal flips up and down. This can be seen in Figure 10 where N_BITS is 8 without any noise. The two plots from the top are signals transmitted from each source, and the last plot describes when these two signals are added together (assuming perfect synchronization.) Also it is supposed that the two signals are sharing the same carrier frequency for synchronization purpose as well.

The signals also can be affected by CFO. Each symbol's phase would be off by $cfo * t$, resulting that phase off is gradually increasing as the timeline goes. To compensate this, the average of the two CFO's are applied reversely to the received signal (line108 – 116). Figure 11 and Figure 12 below plot signals affected by CFO. Figure 11 describes the received signal without CFO Correction, and it results bit error around 40% . Figure 12 shows the received signals with CFO correction, and it results no bit error.

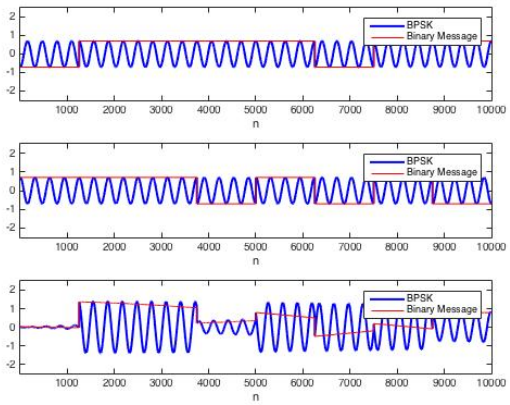


Figure 11. CFO Signals without CF Correction

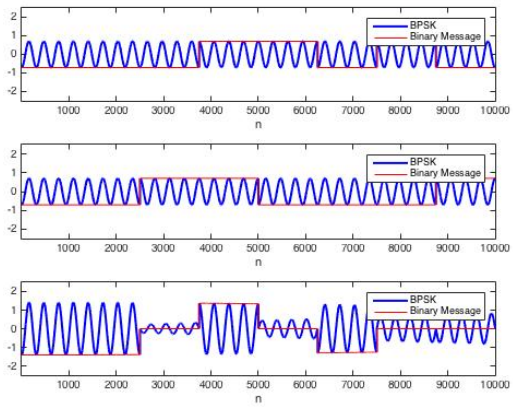


Figure 12. CFO Signals with CFO Correction

In the downsampling part, the code gathers all of the samples for each symbol and takes the average of them. This is not the same downsampling in real practice, but the signal distortion on the channel does not tend to be flat, and taking average of all the samples makes more sense as this can compensate some of the samples that got affected much more than the others (line120 – 127).

This code should be further developed as it may not represents realistically error, and preambles and pilots, which are critical for synchronization and CFO measurement, are not embedded in the signals. Moreover, currently the code (including WARPLab code) does not measure the phase difference between the multiple transmitted signals, but doing so might be useful as it could help improving synchronization. Advanced mapping method such as QPSK and 16-QAM also should be added, because BPSK only can represent 1 bit per symbol. QPSK and 16-QAM's symbol can include 4bits and 16bits.

5. Conclusion

The research subject is very interesting because advanced engineering applications that will come out in near future cannot perform desirably unless a solution to break the challenge of limited frequency spectrum is discovered. As mentioned above, physical layer network coding can significantly improve the throughput of the system compared to what is being used now. There will be many obstacles that are even too soon to discuss. However, once this research gathers more practical data and studies further, the efficient and realistic way to achieve this will appear.