COLLEGE OF ENGINEERING, GUINDY
ANNA UNIVERSITY
CHENNAI 600025

EC5612 - WIRELESS COMMUNICATION AND NETWORKING LABORATORY –SEMESTER VI (R-2019) (January 2023 - June 2023)

RECORD

SUBMITTED BY:

DHARSHIN B

2020105012

In partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

ANNA UNIVERSITY
COLLEGE OF ENGINEERING GUINDY

BONAFIDE CERTIFICATE

NAME: Dharshin B
DEPT: B.E. Electronics and Communication Engineering
REGISTER NUMBER: 2020105012

It is certified that this is the bonafide record of the work done by the above-mentioned student in EC5612 - WIRELESS COMMUNICATION AND NETWORKING LABORATORY - SEMESTERVI (R-2019) during the period January 2023 - June 2023.

Signature of Lab-In-Charge

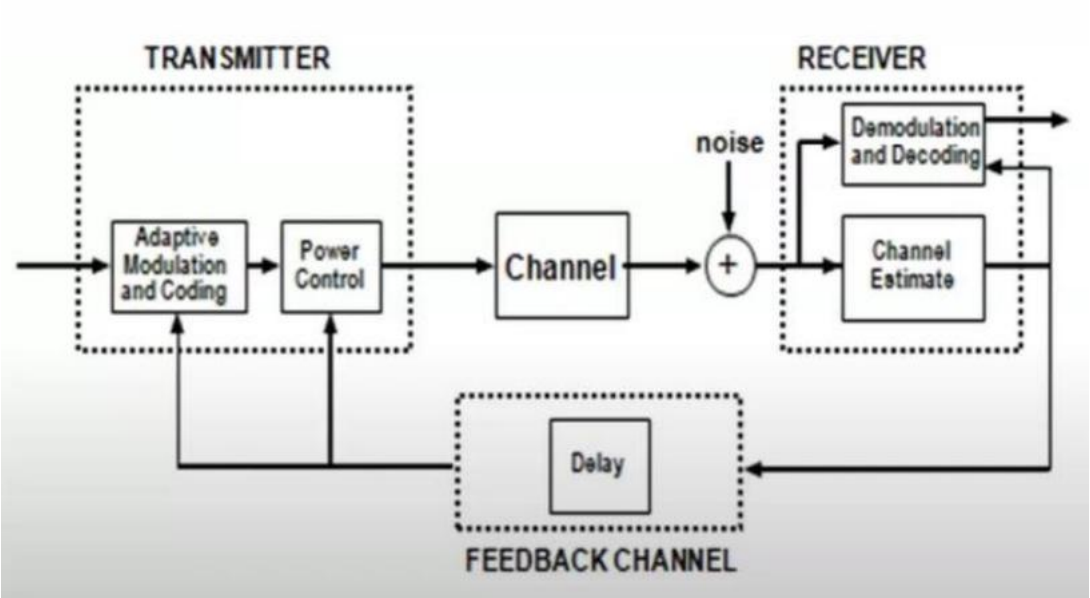
Signature of the Head of the Department

Submitted for the practical exam held on _____

Signature of Internal Examiner

INDEX

BLOCK DIAGRAM:



EXP NO: 01 DATE:	Performance studies of Adaptive Modulation and Coding

AIM:

To perform Adaptive modulation and coding and observe its characteristics.

SOFTWARE REQUIRED:

MATLAB R2022b

THEORY:

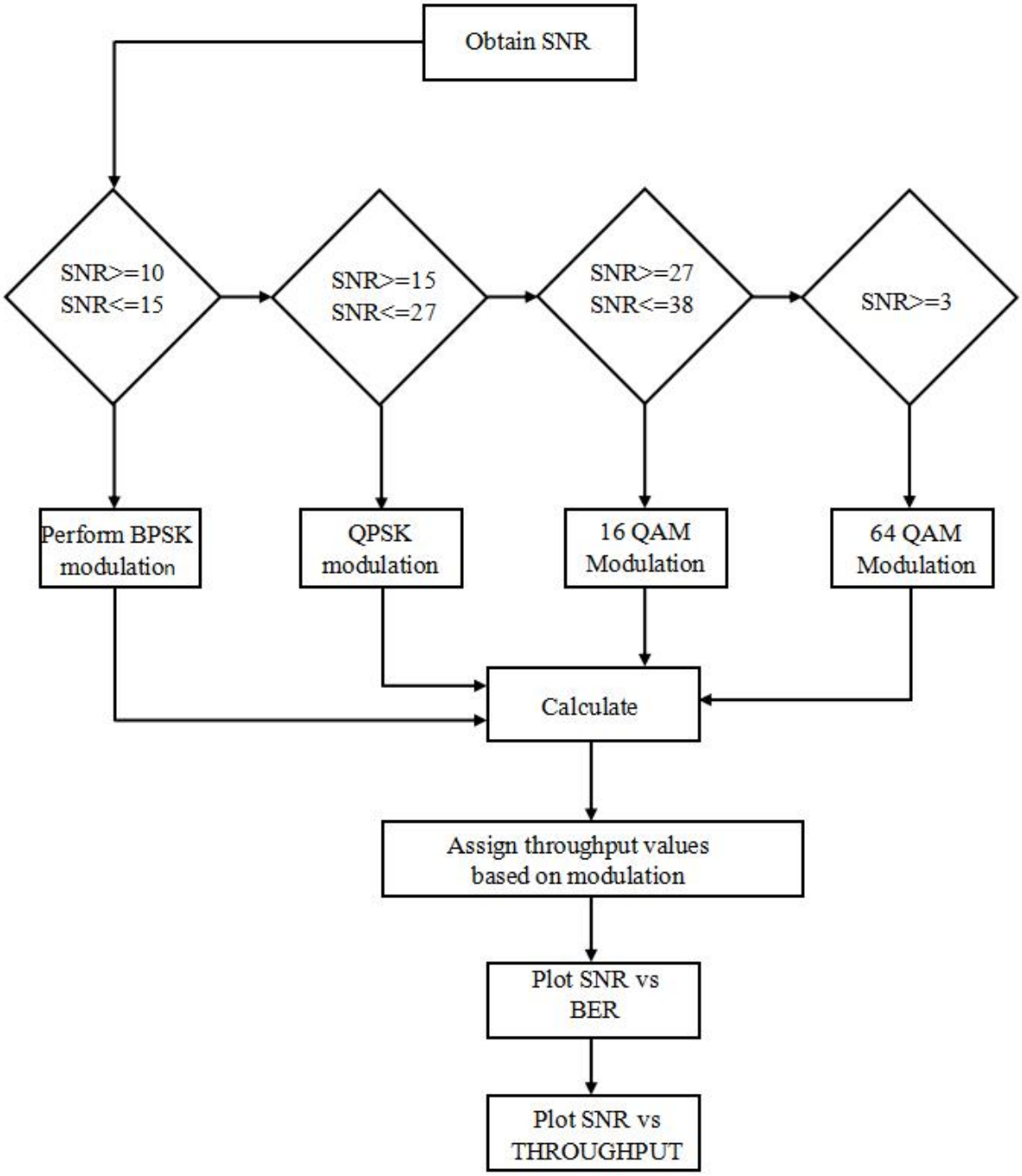
WHAT IS ADAPTIVE MODULATION?

- Adaptive modulation is a technique which allows a radio to change its speed (Modulation rate) as conditions in the radio network changes.
- Interference from outside sources, such as changes in the environment (temperature, tree foliage, moving objects) all effect radio coverage.
- Adaptive modulation enables robust and spectrally efficient transmission over time varying channels.
- The basic premise is to estimate the channel at the receiver and feed this channel back to the transmitter so that the transmission can be adapted relative to channel characteristics.
- Thus, the system can be designed efficiently for the worst-case channel characteristics.

WHY ADAPTIVE MODULATION OVER MODULATION?

- In modulation, there is inefficient utilization of channel and wastage of power which is not the case with adaptive modulation,
- With AMC, data rate and signal power are varied as per the channel conditions for efficient utilization of resources whereas in modulation, they remain fixed.
- AMC guarantees 99% of data transfer even at worst channel condition without any wastage of resources which is not the case with modulation.

FLOWCHART:


Obtain SNR
SNR>=10
SNR<=15
SNR>=15
SNR<=27
SNR>=27
SNR<=38
SNR>=3
Perform BPSK modulation
QPSK modulation
16 QAM Modulation
64 QAM Modulation
Calculate
Assign throughput values based on modulation
Plot SNR vs BER
Plot SNR vs THROUGHPUT

ADVANTAGES AND DISADVANTAGES:

ADVANTAGES:

- AMC allows the system to overcome fading and other interference.
- Maximize the throughput.

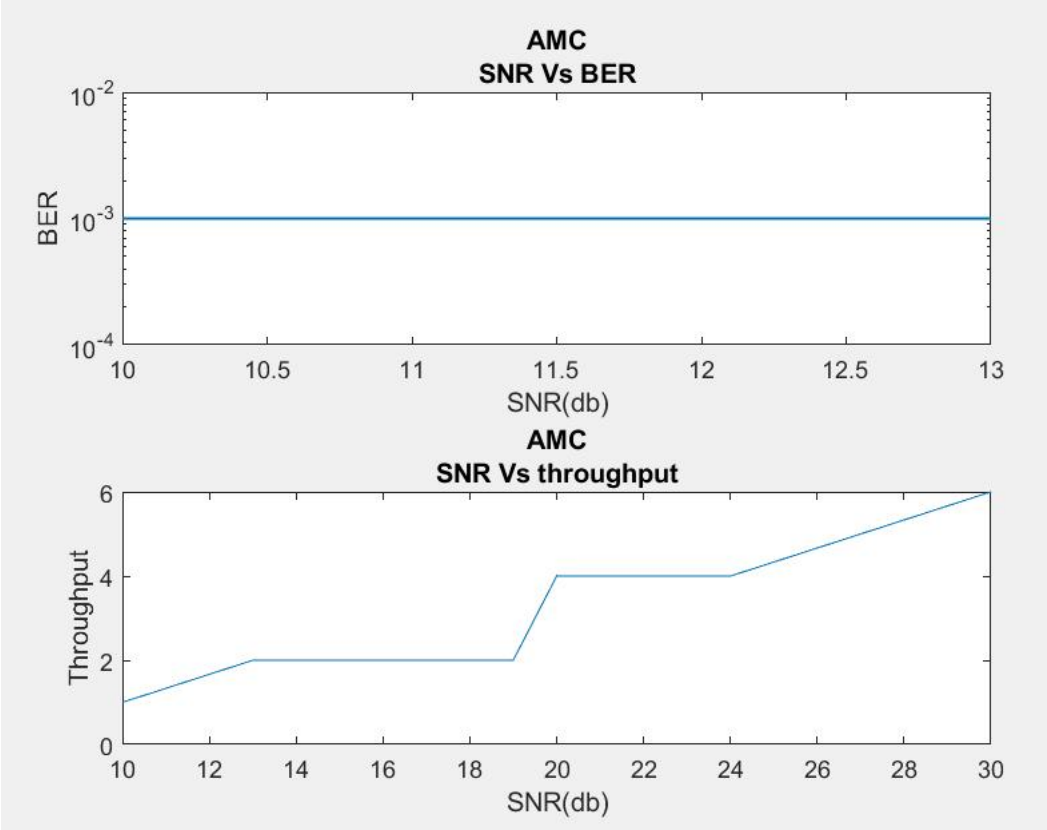
DISADVANTAGES:

- AMC is sensitive to measurement error and delay.
- It requires a feedback system between transmitter and receiver which may not be feasible for some system.
- There are possibilities of wastage of system capacity
- High complexity

ALGORITHM:

1. Define a random sequence of bits for BPSK, QPSK, 16PSK, 64PSK.
2. SNR Range is also defined.
3. If 10<=SNR<=15 use BPSK modulation.
4. If 15<=SNR<=27 use QPSK modulation.
5. If 27<=SNR<=38 use 16PSK modulation.
6. If SNR >38 use 64 PSK Modulation.
7. If BER difference is zero BER=10^-7.
8. Plot a graph with X axis: SNR , Y axis:BER.

MATLAB OUTPUT:



MATLAB CODE:

```
clc;
clear all
close all
snr= randi([10,30],1,15);
snr=sort(snr);
err=[];
for i=1:length(snr)
if(snr(i)>=0 && snr(i)<13)
a=randi([0,1],1,1000);b=pskmod(a,2);
end
if(snr(i)>=13 && snr(i)<20)
a=randi([0,3],1,1000); b=pskmod(a,4);
end
if(snr(i)>=20 && snr(i)<26)
a=randi([0,15],1,1000); b=qammod(a,16);
end
if(snr(i)>=26 && snr(i)<=30)
a=randi([0,63],1,1000);b=qammod(a,64);
end
N0=1/10^(snr(i)/10);
g=awgn(b,snr(i));
n=sqrt(N0/2)*(randn(1,length(a))+1i*randn(1,length(a)));
f=g+n;
if(snr(i)>=10 && snr(i)<13)
d=pskdemod(f,2);
end
if(snr(i)>=13 && snr(i)<20)
d=pskdemod(f,4);
end
if(snr(i)>=20 && snr(i)<26)
d=qamdemod(f,16);
end
if(snr(i)>=26 && snr(i)<=30)
d=qamdemod(f,64);
end
[n,r]=biterr(a,d);

if(r==0)
r=1e-7; end
err=[err r];
end
subplot(2,1,1);semilogy(snr,err,'linewidth',1);
xlabel('SNR(db)');
ylabel('BER'); title({'AMC';'SNR Vs BER'});
thruput=[];
for i=1:length(snr)
if(snr(i)>=10 && snr(i)<13)
thruput=[thruput 1]; end
if(snr(i)>=13 && snr(i)<20)
thruput=[thruput 2]; end
if(snr(i)>=20 && snr(i)<=25)
thruput=[thruput 4]; end
if(snr(i)>=26 && snr(i)<=30)
thruput=[thruput 6];
end
end
subplot(2,1,2); plot(snr ,thruput);
xlabel('SNR(db)'); ylabel('Throughput');
title({'AMC';'SNR Vs throughput'});
```

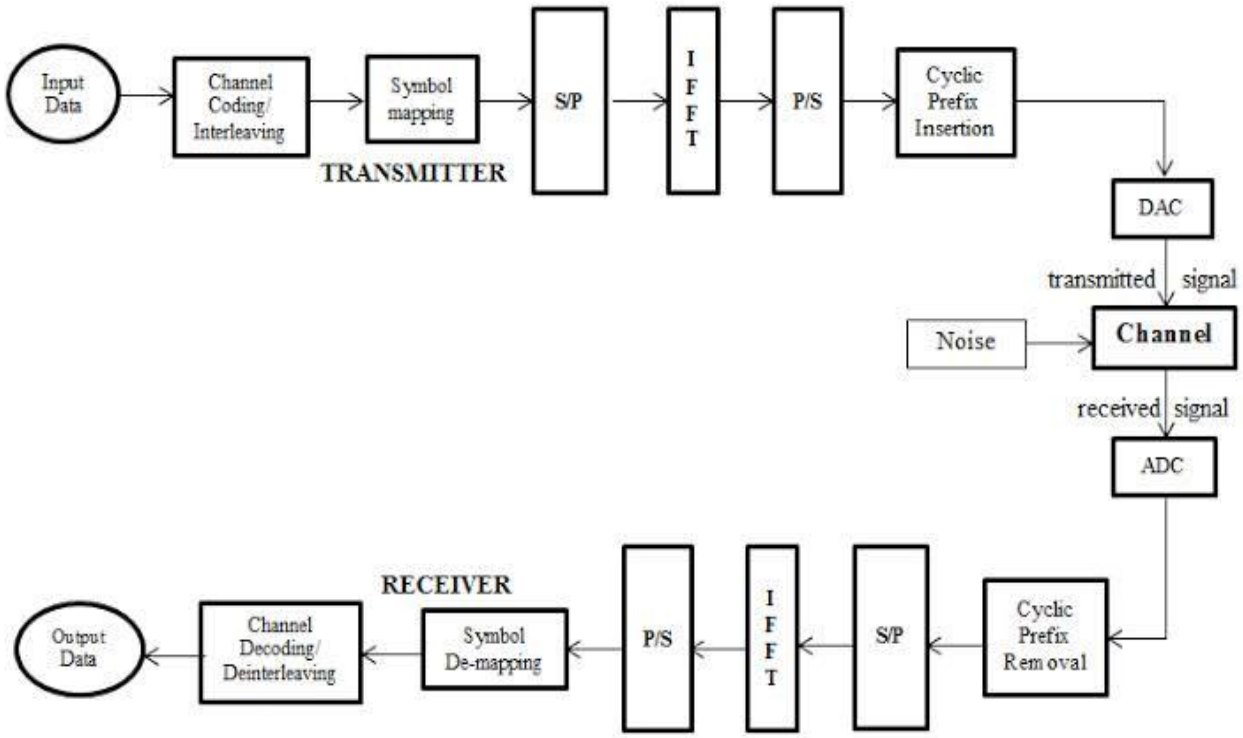
INFERENCE :

1. Thus, different modulation scheme is used for different SNR for lossless transmission of the signal.
2. When the SNR is low, BER is high and hence Throughput is low. Error detection is easier and hence lower order modulation techniques are used.
3. When the SNR is high, BER is low and hence Throughput is high. Error detection becomes difficult and hence higher order modulation techniques are used.

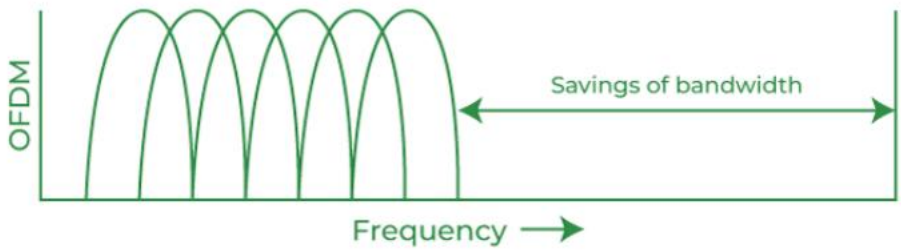
RESULT:

Thus, Adaptive Modulation and Coding has been implemented and analyzed using MATLAB

BLOCK DIAGRAM:



MODEL GRAPH:



EXP NO: 02 DATE:	Orthogonal Frequency Division Multiplexing

AIM:

To implement multicarrier modulation technique (OFDM) using MATLAB.

SOFTWARE REQUIRED:

- MATLAB 2022b

THEORY:

- Orthogonal Frequency Division Multiplexing (OFDM) is a digital multi-carrier modulation scheme that extends the concept of single subcarrier modulation by using multiple subcarriers within the same single channel. Rather than transmit a high-rate stream of data with a single subcarrier, OFDM makes use of a large number of closely spaced orthogonal subcarriers that are transmitted in parallel.

- Each subcarrier is modulated with a conventional digital modulation scheme (such as QPSK, 16QAM, etc.) at low symbol rate. However, the combination of many subcarriers enables data rates similar to conventional single-carrier modulation schemes within equivalent bandwidths.

- The OFDM signal can be described as a set of closely spaced FDM subcarriers. In the frequency domain, each transmitted subcarrier results in a sinc function spectrum with side lobes that produce overlapping spectra between subcarriers. This results in subcarrier interference except at orthogonally spaced frequencies.

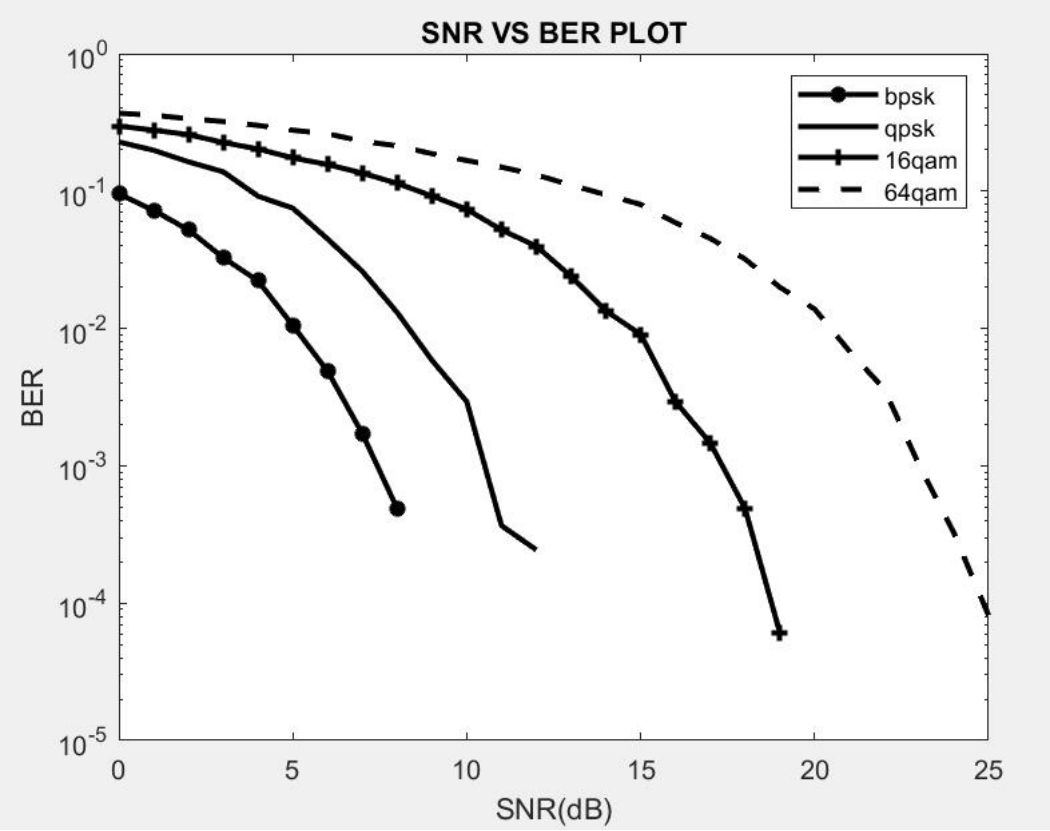
- At orthogonal frequencies, the individual peaks of subcarriers all line up with the nulls of the other subcarriers. This overlap of spectral energy does not interfere with the system's ability to recover the original signal. The receiver multiplies (i.e., correlates) the incoming signal by the known set of sinusoids to recover the original set of bits sent.

- The use of orthogonal subcarriers allows more subcarriers per bandwidth resulting in an increase in spectral efficiency. In a perfect OFDM signal, Orthogonality prevents interference between overlapping carriers.

ALGORITHM:

1. Generate the data points.
2. Modulate the data.
3. Convert columns to row(serial to parallel)
4. Take IFFT for data in the matrix.
5. Add cyclic prefix to create actual OFDM block.
6. Connect rows to column (parallel to serial).
7. Transmit the OFDM Signal.
8. Received signal is converted to parallel.
9. FFT is performed after removing cyclic prefix.
10. Convert the data to serial stream.
11. Demodulate the data.
12. Repeat it for different SNRs
13. Plot the BER vs SNR graph

MATLAB OUTPUT:



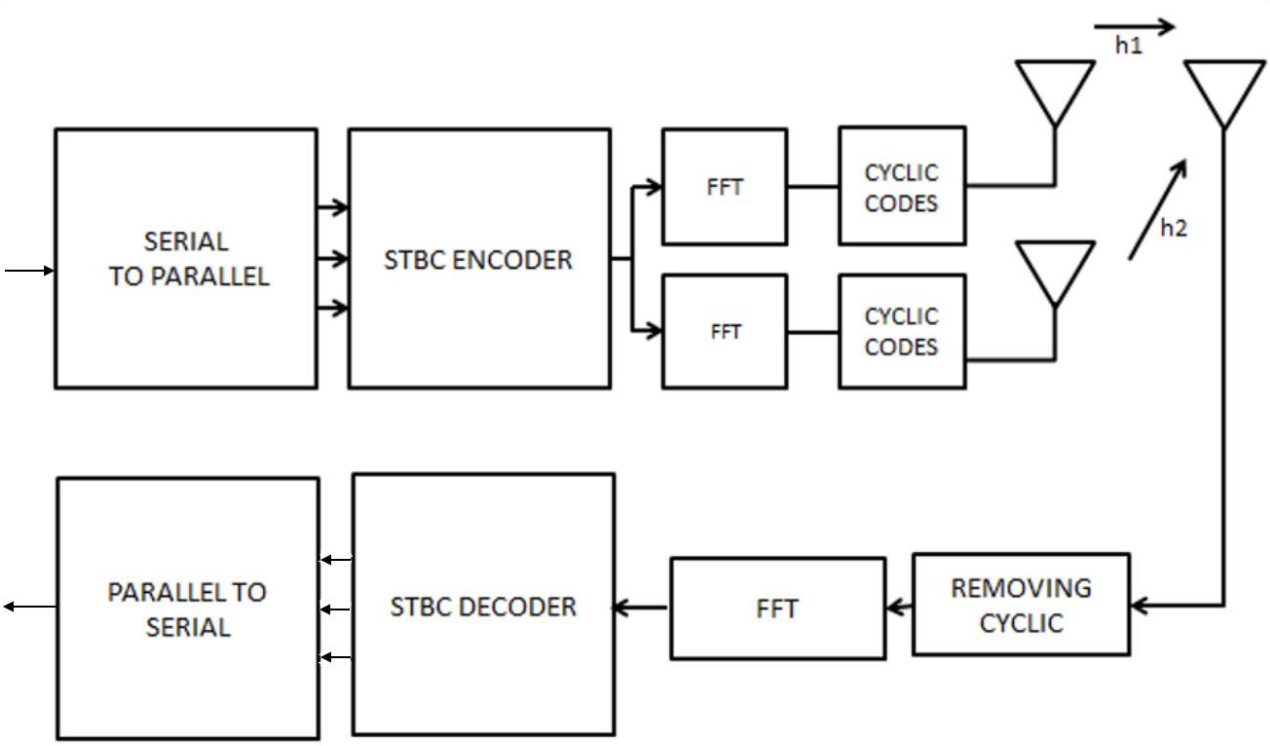
MATLAB CODE:

```
clc; clear close all rng(0) %generating same bit each and every time
n=4096; snr=0:30; biterror=[];
for i=1:4
if i==1 k=2;
x=randi([0 1],1,n); d=x;
y=pskmod(x,k);
elseif i==2 %QPSK
k=4;
x=randi([0 3],1,n);
b=de2bi(x,'left-msb');
c=b.'; d=reshape(c,1,2*n);
y=pskmod(x,k); %input of pskmod should be in decimal
elseif i==3 %16-qam
x=randi([0 15],1,n);
k=16;
b=de2bi(x,'left-msb'); c=b.';
d=reshape(c,1,4*n); %for biterror comparison
y=qammod(x,k);
elseif i==4 %64-qam
x=randi([0 63],1,n);
k=64; b=de2bi(x,'left-msb');
c=b.'; d=reshape(c,1,6*n);
y=qammod(x,k);
end
p=reshape(y,64,length(y)/64);%max no. of subcarriers
q=ifft(p,64);
s=reshape(q,1,length(y));%serial converter be=[];
for j=0:1:30
h=1./sqrt(rand(1,length(y))+i.*sqrt(rand(1,length(y))));
r=h.*s;t=awgn(r,j,'measured');
m=t./h; %flat fading->hx+n
p11=reshape(m,64,length(y)/64); q11=fft(p11,64);
s11=reshape(q11,1,length(y));
if i==1 %BPSK
y11=pskdemod(s11,k); y14=y11;
elseif i==2 %QPSK
y11=pskdemod(s11,k);
y12=de2bi(y11,'left-msb');
y13=y12.';
y14=reshape(y13,1,2*length(y11));
elseif i==3 %16-qam
y11=qamdemod(s11,k);
y12=de2bi(y11,'left-msb'); y13=y12.';
y14=reshape(y13,1,4*length(y11));
elseif i==4 %64-qam
y11=qamdemod(s11,k);
y12=de2bi(y11,'left-msb'); y13=y12.';
y14=reshape(y13,1,6*length(y11));
end
[num1,e1]=biterr(y14,d); be=[be e1];
end
biterror(i,:)=be;
end
semilogy(snr,biterror(1,:),'-*k','linewidth',2);hold on;
semilogy(snr,biterror(2,:),'-k','linewidth',2);hold on;
semilogy(snr,biterror(3,:),'-+k','linewidth',2);hold on;
semilogy(snr,biterror(4,:),'--k','linewidth',2);hold on;
xlabel('SNR(dB)');  ylabel('BER'); title('SNR VS BER PLOT');
legend('bpsk','qpsk','16qam','64qam')
```

RESULT:

Thus, implementation of multicarrier modulation (Orthogonal Frequency Division Multiplexing) is carried out using MATLAB.

BLOCK DIAGRAM:



ALGORITHM:

1. Generate random binary sequence of +1s and -1s.
2. Group them into pairs of two symbols.
3. Code it as per the Alamouti Space Time Code, multiply the symbols with the channels and then add AWGN.
4. Equalize the received symbols.
5. Perform hard decision decoding and count the bit errors.
6. Repeat for multiple values and plot the simulation.

EXP NO: 03 DATE:	Space Time Block Codes

AIM:

To perform space time block codes (Alamouti coding) for a system that has two transmitter and one receiver.

SOFTWARE REQUIRED:

- MATLAB 2022b

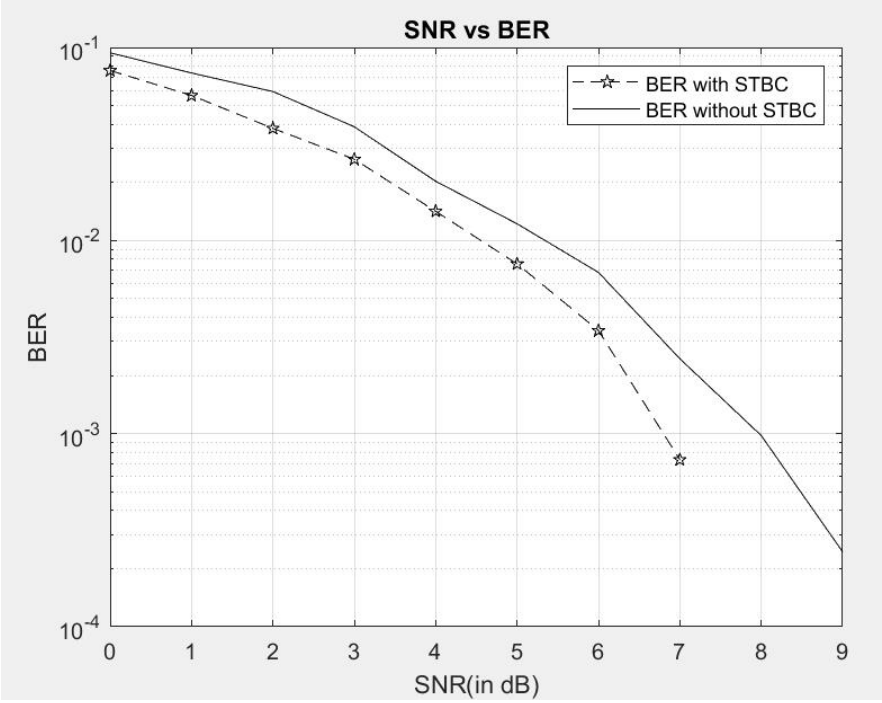
THEORY:

Space-time block codes are used for MIMO systems to enable the transmission of multiple copies of a data stream across a number of antennas and to exploit the various received versions of the data to improve the reliability of data-transfer. Space-time coding combines all the copies of the received signal in an optimal way to extract as much information from each of them as possible.

A space time block code is usually represented by a matrix. Each row represents a time slot and each column represents one antenna's transmissions over time. Space-time block codes (STBC) are a generalized version of Alamouti scheme. These schemes have the same key features. Therefore, these codes are orthogonal and can achieve full transmit diversity specified by the number of transmit antennas. In another word, spacetime block codes are a complex version of Alamouti's space-time code, where the encoding and decoding schemes are the same as there in the Alamouti space-time code in both the transmitter and receiver sides.

At the receiver side, when signals are received, they are first combined and then sent to the maximum likelihood detector where the decision rules are applied.

MATLAB OUTPUT:



MATLAB CODE:

```
clc; clear all; close all;
n = randi([0,1],1,4096);
a = reshape(n,length(n),1);
bpskmod = pskmod(a,2);
snr = 0:1:30; h1 = 2+1j; h2 = 1-2j;
y = []; M= []; Op = [];
OpwoutSTBC = []; q = 1;
for l = 1:length(snr)
for p = 1:2:length((n))-1
c1 = (h1*bpskmod(p,1))+(h2*bpskmod(p+1,1));
M(p,q) = awgn(c1,snr(l),'measured');
c2 = (-h1*conj(bpskmod(p+1,1)))+(h2*conj(bpskmod(p,1)));
M(p+1,q) = awgn(c2,snr(l),'measured');
end
for r = 1:2:(length(n) -1 )
y(r,q) = (conj(h1)*M(r,1)) + h2*conj(M(r+1,1));
y(r+1,q) = (conj(h2)*M(r,1)) - h1*conj(M(r+1,1));
end
t1 = pskdemod(y,2);
Op(l,:) = reshape (t1,1,length(n));
[number,ratio] = biterr(Op,n);
end
semilogy(snr,ratio,'pentagram--C','Color','black')
xlabel("SNR(in dB"); ylabel("BER"); hold on
%without STBC
for l = 1:length(snr)
rec = awgn((h1*bpskmod),snr(l),"measured");
demod = pskdemod(rec,2);
OpwoutSTBC(l,:) = reshape(demod,1,length(n));
[number1,ratio1] = biterr(OpwoutSTBC,n);
end
semilogy(snr,ratio1,'-k')
grid on
legend('BER with STBC','BER without STBC')
xlabel("SNR(in dB"); ylabel("BER");
title("SNR vs BER");
```

ENCODING:

It was designed for a two-transmit antenna system and has the coding matrix:

$$C = \begin{bmatrix} c_1 & c_2 \\ -c_2^* & c_1^* \end{bmatrix}$$

Where * denotes complex conjugate.

It is readily apparent that this is a rate-1 code. It takes two time slots to transmit two symbols. Using the optimal decoding scheme discussed below, the Bit Error Rate (BER) of this STBC is equivalent to 2 n R – branch maximal ratio combining (MRC). This is a result of the perfect orthogonality between the symbols after receive processing – there are two copies of each symbol transmitted and n R copies received.

DECODING:

One particularly attractive feature of orthogonal STBCs is that maximum likelihood decoding can be achieved at the receiver with only linear processing. To consider a decoding method, a model of the wireless communications system is needed.

RESULT:

Thus, the Space Time Block Code using Alamouti coding scheme is studied, and the Bit Error Rate is analyzed.

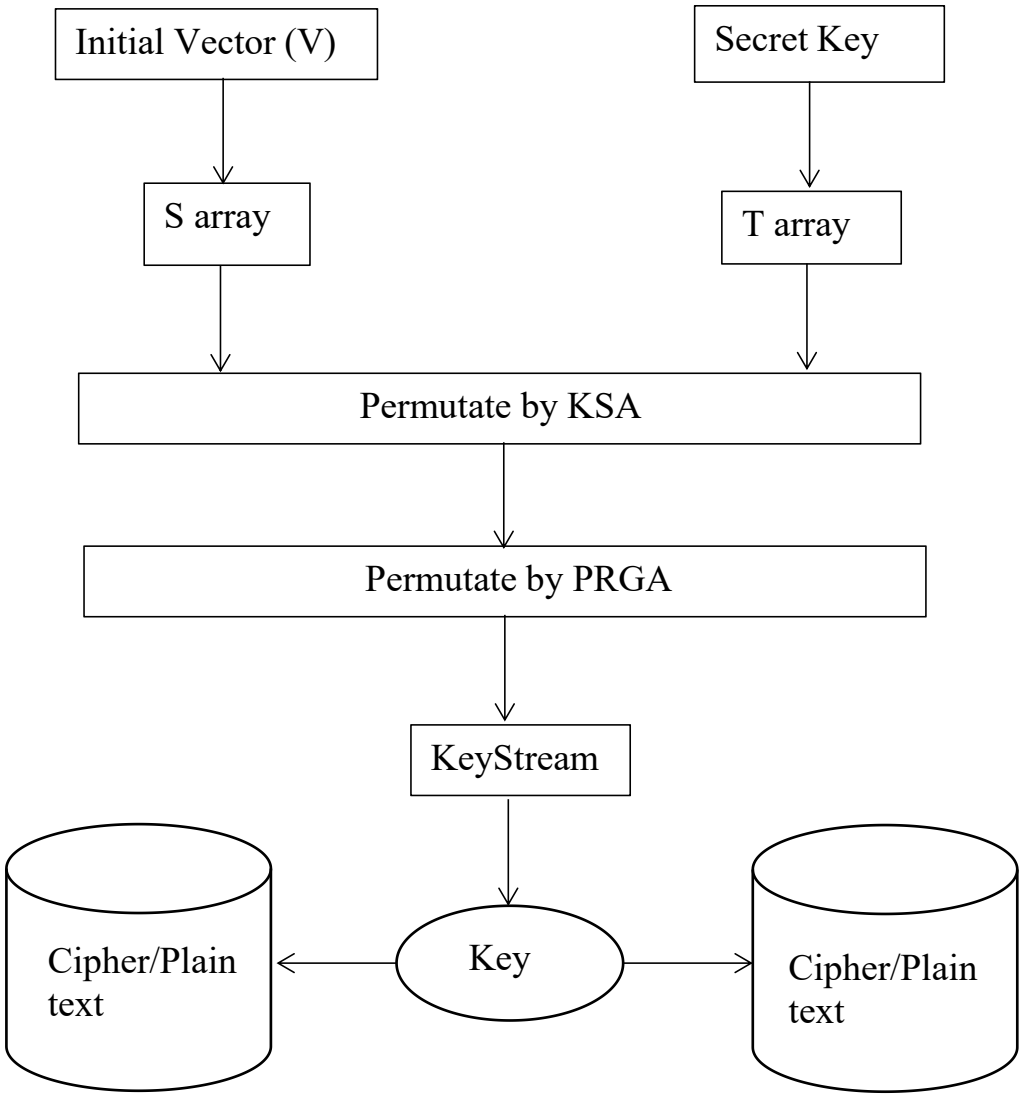
INFERENCE:

1. No need of Channel information since the transmitter antenna is independent of Channel State
2. Since it requires two Transmit antennas, this scheme consumes more power.

CONCLUSION:

The Bit Error Rate is getting reduced while using STBC coding.

BLOCK DIAGRAM:



EXP NO: 04 DATE:	**Implementation of Network Security Protocols – RC4 Encryption Algorithm**

AIM

To implement RC4 stream cipher encryption and decryption in MATLAB.

MATERIALS REQUIRED:

MATLAB R2022b

THEORY:

RC4 means Rivest Cipher 4 invented by Ron Rivest in 1987 for RSA Security. It is a Stream Ciphers. Stream Ciphers operate on a stream of data byte by byte. RC4 stream cipher is one of the most widely used stream ciphers because of its simplicity and speed of operation. It is a variable key-size stream cipher with byte-oriented operations. It is generally used in applications such as Secure Socket Layer (SSL), Transport Layer Security (TLS).

For encryption:

The user enters the Plaintext and a secret key.
For the secret key entered, the encryption engine creates the keystream using the KSA and PRGA algorithms.
Plaintext is XORed with the generated keystream. Because RC4 is a stream cipher, byte-by-byte XORing is used to generate the encrypted text.
This encrypted text is now sent in encrypted form to the intended recipient.

For Decryption:

The same byte-wise X-OR technique is used on the ciphertext to decrypt it.

ALGORITHM:

KEY SCHEDULING ALGORITHM

```
Initialization
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod K – len];
j = 0;
for i = 0 to 255 do
        j = (j +S[i] + T[i]) mod 256;
        Swap (S[i], S[j]);
```

PSEUDO RANDOM GENERATION ALGORITHM (STREAM GENERATION)

```
i, j = 0;
while (true)
i = (i + 1) mod 256;
j = (j + S[i]) mod 256;
Swap (S[i], S[j]);
T = (S[i] + S[j]) mod 256;
k = S[t];
```

MATLAB OUTPUT:

```
Enter the message for encryption: 'I am a message'
Encrypted Cipher Text:
  Columns 1 through 12

   199   200    62    58    30   167    80   159   132   106   136   188

  Columns 13 through 14

     7   247

  Columns 1 through 12

    73    32    97   109    32    97    32   109   101   115   115    97

  Columns 13 through 14

   103   101

Deciphered message:
I am a message
```

MATLAB CODE:

```
clc; clear all; close all;
s = zeros(1,256);
for i = 1:256
s(i) = i;
end
msg_1 = input('Enter the message for encryption: ');
message = [];
message = [message double(char(msg_1))];
key = [2 4 6 8 2 1 6 1];
t = zeros(1,256);
for i = 1:256
t(i) = mod(i,length(key));
end
%KSA
i = 0; j =0;
for i = 1:256
j = mod(j+s(i)+t(i),256) + 1;
s([i j]) = s([j i]);
j = j-1;
end
%PRGA
i = 0; j = 0; k = [];
for m = 1:length(message)
i = mod(i+1,256);
j = mod(j + s(i),256);
s([i j]) = s([j i]);
k = [k s(mod(s(i) + s(j),256))];
end
encrypted_message = [];
for i = 1:length(message)
encrypted_message = [encrypted_message bitxor(message(i),k(i))];
end
disp("Encrypted Cipher Text: "); disp(char(encrypted_message));
decrypted_message = [];
for i = 1:length(message)
decrypted_message = [decrypted_message bitxor(encrypted_message(i),k(i))];
end
disp(decrypted_message)
disp('Deciphered message: '); disp(char(decrypted_message))
```

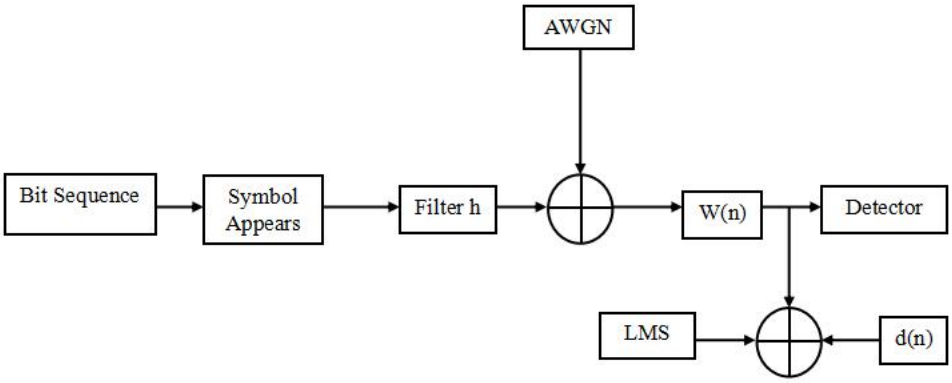
INFERENCE

RC4 is a stream cipher that is simple to use and the speed of operationis also higher. It does not require a lot of memory and can also be implemented on large streams of data. But, RC4 is vulnerable and insecure at times due to which very few applications use it nowadays.

RESULT:

Thus RC4 Encryption stream cipher algorithm is implemented using MATLAB.

BLOCK DIAGRAM:



EXP NO: 05 DATE:	Performance of Equalizers

AIM:

To implement equalization techniques for wireless channels using Least Mean Squares (LMS) and Zero Forcing algorithms

SOFTWARE REQUIRED:

MATLAB 2022b

THEORY:

A channel equalizer is an important component of a communication system and is used to mitigate the ISI (inter symbol interference) introduced by the channel. The equalizer depends upon the channel characteristics. These are usually employed to reduce the depth and duration of the fades experienced by a receiver in a local area which are due to motion. An equalizer within a receiver compensates for the average range of expected channel amplitude and delay characteristics. Equalizers must be adaptive since the channel is generally unknown and varies with time.

Least Mean Squares (**LMS**) algorithms are a class of adaptive filter used to mimic a desired filter by finding the filter coefficients that relate to producing the least mean square of the error signal (difference between the desired and the actual signal). It is a stochastic gradient descent method in that the filter is only adapted based on the error at the current time.

Zero Forcing Equalizers is a linear equalizer which uses unit impulse response to compensate for channel effect. An overall impulse response is equal to one for the defected symbol and zero for all other received symbols. The noise may be increased in this process.

LEAST MEAN SQUARES (LMS) ALGORITHM

ALGORITHM FORMULATION:

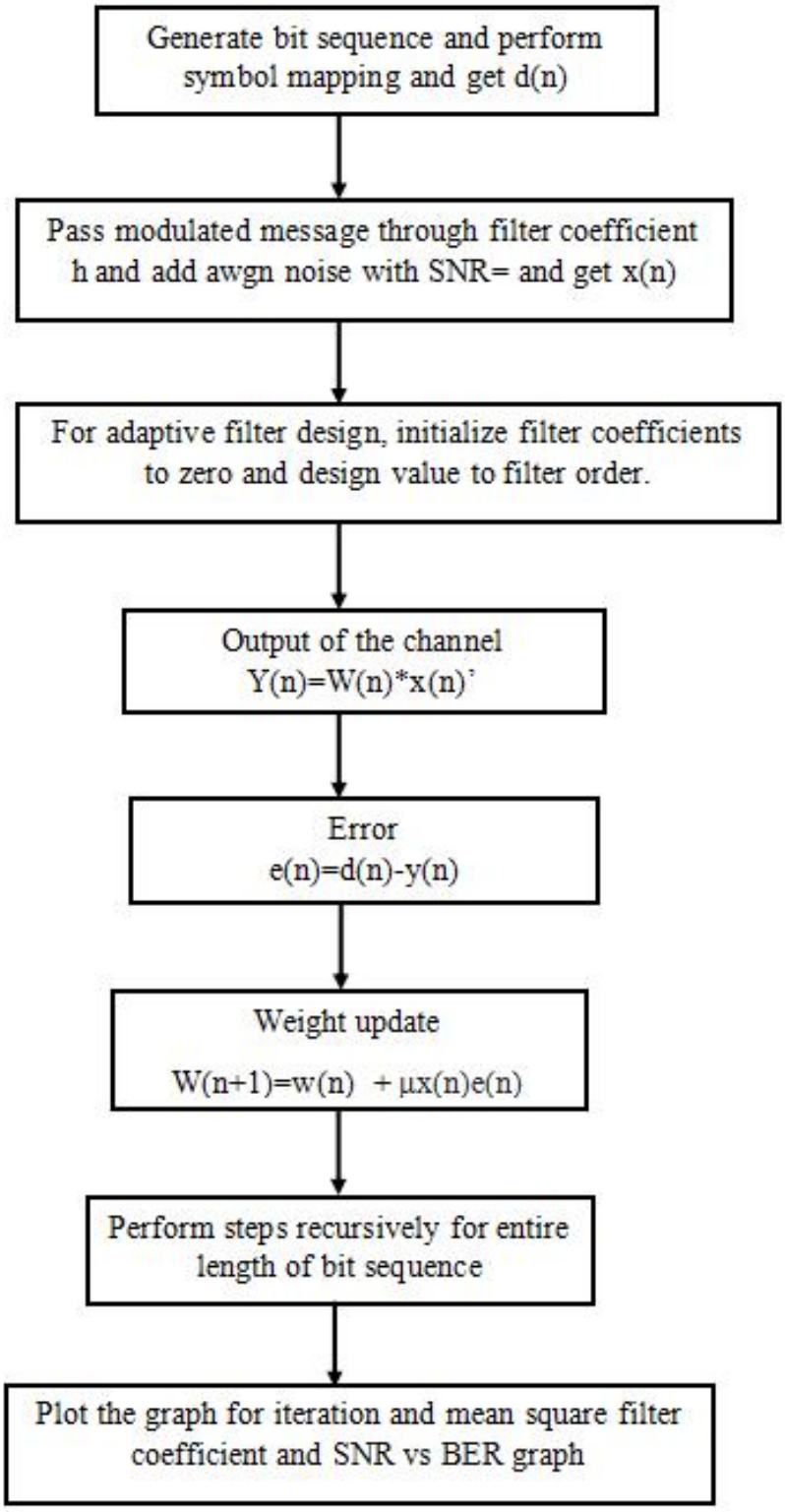
From the method of steepest descent, the weight vector equation is given by,

$$w(n+1) = w(n) + \frac{1}{2}\mu[\nabla(E\{e^2(n)\}]$$

where,

- $w(n)$ → weight update vector matrix on the nth instant
- μ → step-size parameter and controls the convergence characteristics of the LMS algorithm

FLOWCHART: (LMS)



- e^2(n) → mean square error(at the n[th] instant) between the beamformer output y(n) and the reference signal *d(n)* which is given by,

$$e^2(n) = [d(n) - w^H x(n)]^2$$

where,

  - w^H → Hermitian of weight update vector matrix
  - $x(n)$ → Input sequence

- $\nabla(\mathrm{E}\{\mathrm{e}^2(n)\}$ → Gradient performed on the expectation of the mean square error at the n[th] instant

In the method of steepest descent, the biggest problem is the computation involved in finding the values r and R matrices in real time. The LMS algorithm on the other hand simplifies this by using the instantaneous values of covariance matrices r and R instead of their actual values i.e.

$$R(n) = x(n)x^H(n)$$

$$r(n) = d(n)x(n)$$

Therefore, the weight update can be given by the following equation,

$$w(n+1) = w(n) + \mu x(n)[d(n) - x^H(n)w(n)]$$

$$w(n+1) = w(n) + \mu x(n)e(n)$$

The LMS algorithm is initiated with an arbitrary value *w(0)* for the weight vector at *n=0*. The successive corrections of the weight vector eventually lead to the minimum value of the mean squared error.

Therefore, the LMS algorithm can be summarized in following equations;

$$\text{Output} \rightarrow \mathrm{y}(n) = w^H x(n)$$

$$Error \rightarrow e(n) = d(n) - y(n)$$

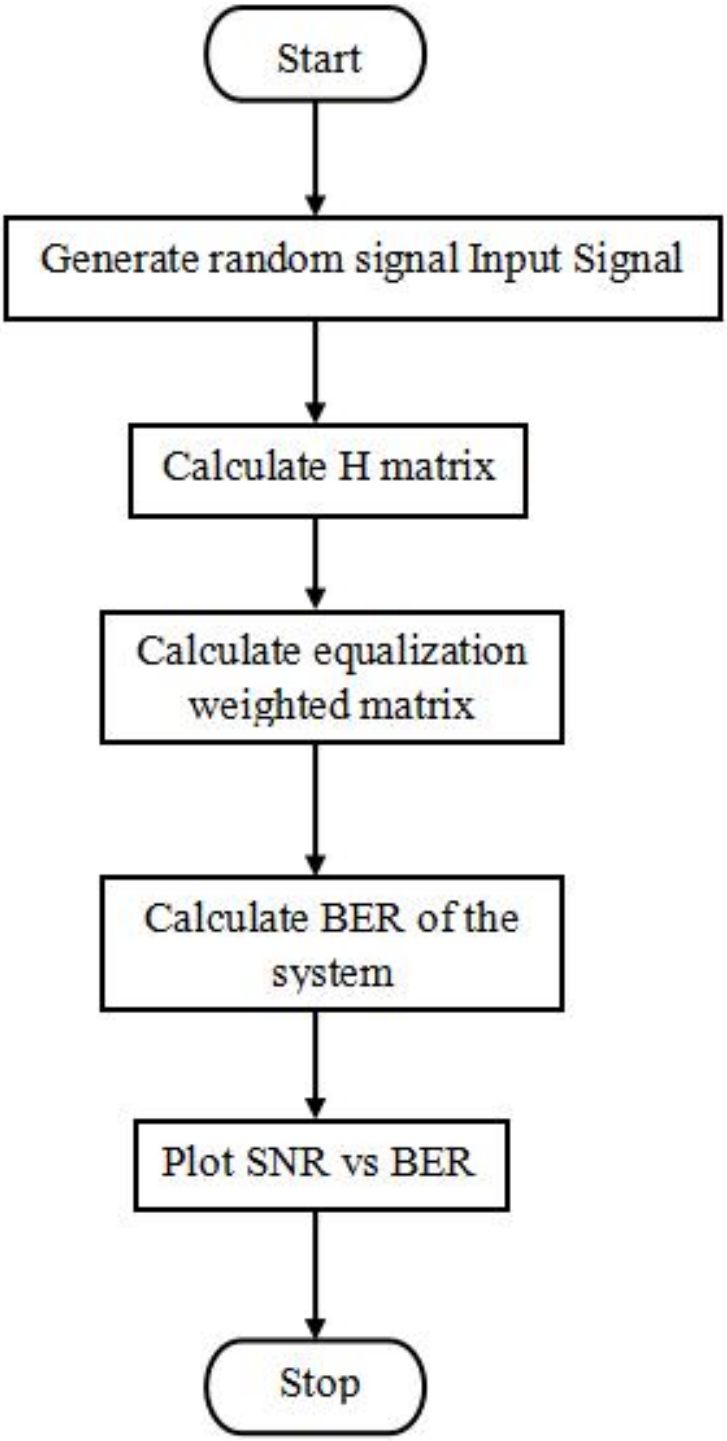
$$Weight \rightarrow w(n+1) = w(n) + \mu x(n)e(n)$$

ZERO FORCING EQUALIZERS

ALGORITHM FORMULATION:

The following equations are used to formulate the algorithm used in Zero Forcing Equalizers:

FLOWCHART: (Zero forcing Equalizers)



$$y_k = h_0 x_k + h_1 x_{k-1} + n_0$$

$$y_{k+2} = h_0 x_{k+2} + h_1 x_{k+1} + n_2$$

$$y_{k+1} = h_0 x_{k+1} + h_1 x_k + n_1$$

$$\begin{bmatrix} y_{k+2} \\ y_{k+1} \\ y_k \end{bmatrix} = \begin{bmatrix} h_0 & h_1 & 0 & 0 \\ 0 & h_0 & h_1 & 0 \\ 0 & 0 & h_0 & h_1 \end{bmatrix} . \begin{bmatrix} x_{k+2} \\ x_{k+1} \\ x_k \\ x_{k-1} \end{bmatrix} + \begin{bmatrix} n_2 \\ n_1 \\ n_0 \end{bmatrix}$$

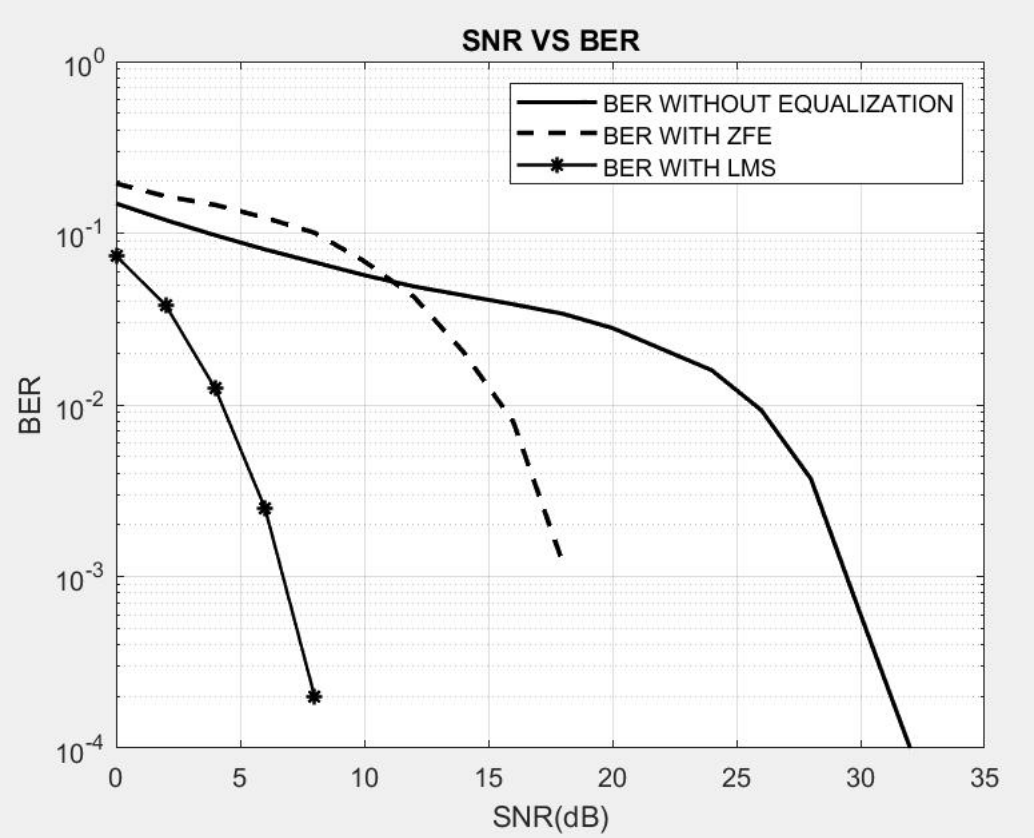
$$\overline{1_2} = C^T H = [0 \quad 0 \quad 1 \quad 0]$$

$$C = (HH^T)^{-1} . \overline{1_2}$$

where,

- $y_k \rightarrow$ Output sequence at instance k
- $x_k \rightarrow$ Input sequence at instant k
- $n_k \rightarrow$ Noise sequence at instant k
- $h_0 \; and \; h_1 \rightarrow$ Equalizing filter coefficients
- $C \rightarrow$ Zero Forcing Constant

MATLAB OUTPUT:



MATLAB CODE:

```
clc; clear all; close all;
message = randi([0,1],1,10000);
snr = 0:2:40; mod = 2;
L = 2; r = 3; % 3 TAP EQUALISER
modulated_bpsk_msg = pskmod(message,mod);
h0 = 1; h1 = 0.7;
H = zeros(r,r+L-1); % ORDER OF H MATRIX
% FORMING H MATRIX FOR 3 TAP EQUALISER
for p = 1:r
H(p,p:p+L-1) = [h0 h1];
end
x = []; % FORMING INPUT MATRIX X
X = modulated_bpsk_msg; X_1 = circshift(X,1);
X_1(1) = 0; X1 = circshift(X,-1);
X1(end) = 0; X2 = circshift(X,-2);
X2(end-1:end)= 0; x = [X2;X1;X;X_1];
C = ((H*H')\H)*[0;0;1;0];
ber_without_ZFE = []; ber_with_ZFE = [];
for p = 1:length(snr)
y = awgn([h0 h1]*[X;X_1],snr(p),'measured');
Noise = y - ([h0 h1]*[X;X_1]);
Noise_1 = circshift(Noise,-1);
Noise_1(end) = 0; Noise_2 = circshift(Noise,-2);
Noise_1(end-1:end) = 0;
Y = (H*x)+ [Noise_2;Noise_1;Noise]; X_PRIME = C.'*Y;
Demodulated_BPSK_msg_with_ZFE =pskdemod(X_PRIME',mod);
Demodulated_BPSK_msg_without_ZFE = pskdemod(y',mod);
[number1,ratio1] = biterr(message,Demodulated_BPSK_msg_with_ZFE');
[number2,ratio2] = biterr(message,Demodulated_BPSK_msg_without_ZFE');
ber_with_ZFE = [ber_with_ZFE,ratio1];
ber_without_ZFE = [ber_without_ZFE,ratio2];
end
semilogy(snr,ber_with_ZFE,'-k','LineWidth',1.6)
hold on
semilogy(snr,ber_without_ZFE,'--k','LineWidth',1.8)
hold on
n = length(message); M = 25; w = zeros(1,M);
wi = zeros(1,M); E = []; mu = 0.0005;
msg_bpsk = pskmod(message,mod);
ber_without_LMS = []; ber_with_LMS = [];
for k = 1:length(snr)
msg_rx = awgn(msg_bpsk, snr(k), 'measured');
for i = M:n
E(i) = msg_bpsk(i) - wi*msg_rx(i:-1:i-M+1)';
wi = wi + 2*mu*E(i)*msg_rx(i:-1:i-M+1);
end
msg_eq = zeros(n,1);
for i = M:n
j = msg_rx(i:-1:i-M+1);
msg_eq(i) = ((wi)*(j)');
end
Demod_with_LMS = pskdemod(msg_eq,mod)';
Demod_without_LMS = pskdemod(msg_rx,mod);
[n1,r1] = biterr(message,Demod_with_LMS);
[n2,r2] = biterr(message,Demod_without_LMS);
ber_without_LMS = [ber_without_LMS,r1];
ber_with_LMS = [ber_with_LMS,r2];
end
grid on; semilogy(snr,ber_with_LMS,'-*k','LineWidth',1.2)
title('SNR VS BER'); xlabel('SNR(dB)');ylabel('BER');
legend('BER WITHOUT EQUALIZATION','BER WITH ZFE','BER WITH LMS');
```

INFERENCE:

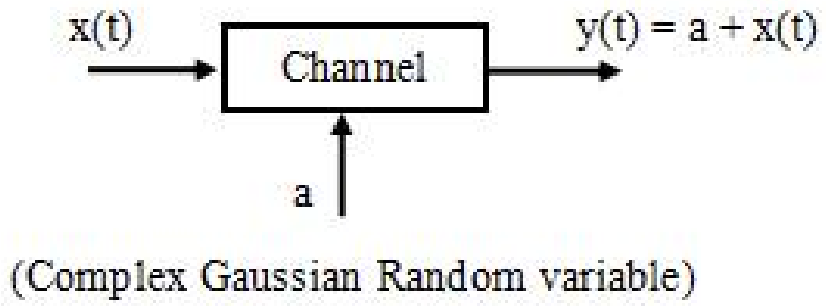
- In the Least Mean Squares (LMS) algorithm of equalization techniques of wireless channels, when the epoch iterations increase, error decreases thereby decreasing the Mean Squared Error (MSE).
- In the case of Zero Forcing Equalizer, it can be observed that it nullifies inter-symbol interference (ISI) in the modulated signals. But noise obtained in the process increases accordingly.
- Hence, comparing both the algorithms, we can observe that Least Mean Squares (LMS) algorithm is efficient than that of Zero Forcing Equalization technique in terms of noise performance.

RESULT:

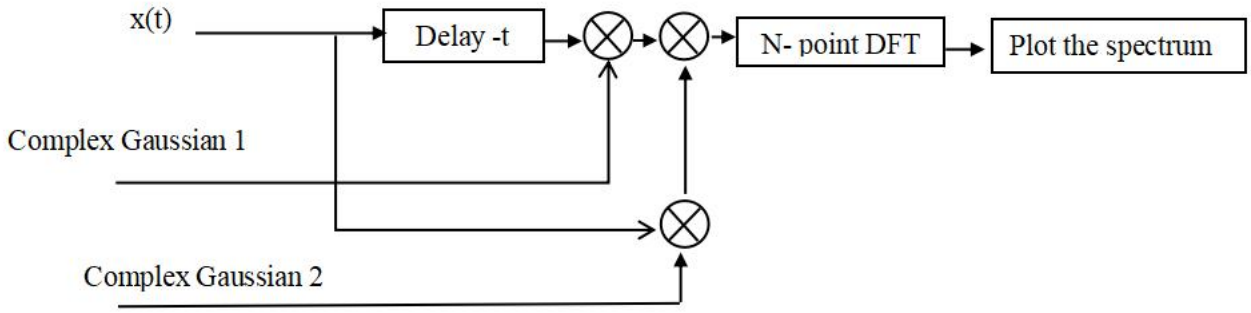
Hence, channel equalization using the Least Mean Squares (LMS) algorithm and Zero Forcing Equalizer are successfully performed using MATLAB and the cost function/BER-SNR graphs for the same are plotted.

BLOCK DIAGRAM:

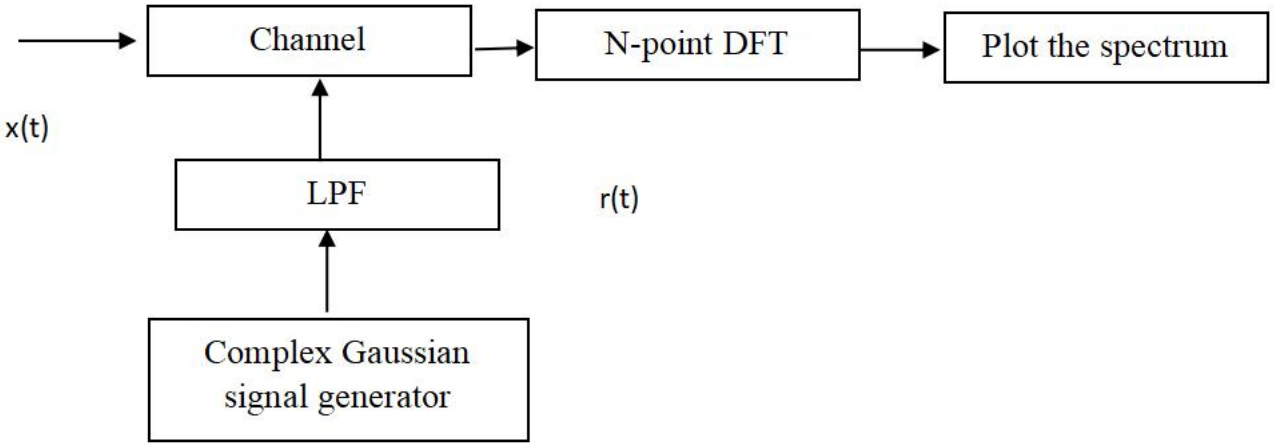
Slow and Flat Fading



Slow and Frequency Selective Fading



Fast and Flat Fading



EXP NO: 06 DATE:	Characterization of Wireless Fading Channels

AIM

To plot the spectra of two tone signal due to slow flat fading, slow frequency selective fading, fast flat fading, fast frequency selective fading, Also to analyse the BER performance of Rayleigh and Rician channel.

SOFTWARE REQUIRED:

MATLAB R2022b

THEORY:

Based on multipath delay spread there are two types of small scale fading viz. flat fading and frequency selective fading. These multipath fading types depend on propagation environment.

FLAT FADING

The wireless channel is said to be flat fading if it has constant gain and linear phase response over a bandwidth which is greater than the bandwidth of the transmitted signal. In this type of fading all the frequency components of the received signal fluctuate in same proportions simultaneously. It is also known as non-selective fading

- Signal BW > Channel BW
- Symbol Period > Delay Spread

The effect of flat fading is seen as decrease in SNR. These flat fading channels are known as amplitude varying channels or narrow band channels.

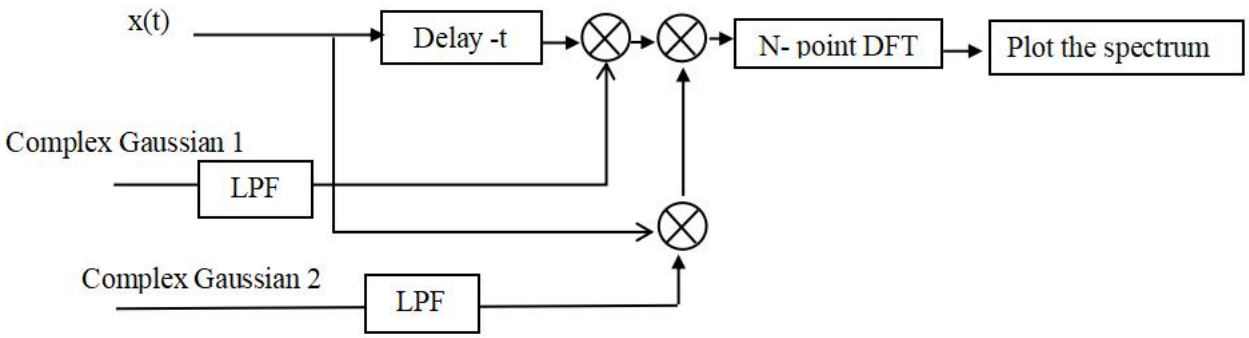
FREQUENCY SELECTIVE FADING

It affects different spectral components of a radio signal with different amplitudes.Hence the name selective fading

- Signal BW> Channel BW
- Symbol period < Delay Spread

Based on doppler spread there are two types of fading viz. fast fading and slow fading. These doppler spread fading types depend on mobile speed i.e.speed of receiver with respect to transmitter.

Fast and Frequency Selective Fading



FAST FADING

The phenomenon of fast fading is represented by rapid fluctuations of signal over small areas (i.e.bandwidth). When the signals arrive from all the directions in the plane, fast fading will be observed for all direction so motion. Fast fading occurs when channel impulse response changes very rapidly within the symbol duration.

- High doppler spread
- Symbol period >Coherence time
- Signal Variation< Channel variation

This parameters result into frequency dispersion or time selective fading due to doppler spreading. Fast fading is result of reflections of local objects and motion of objects relative to those objects.

SLOW FADING

Slow fading is result of shadowing by buildings, hills, mountains and other objects over the path.

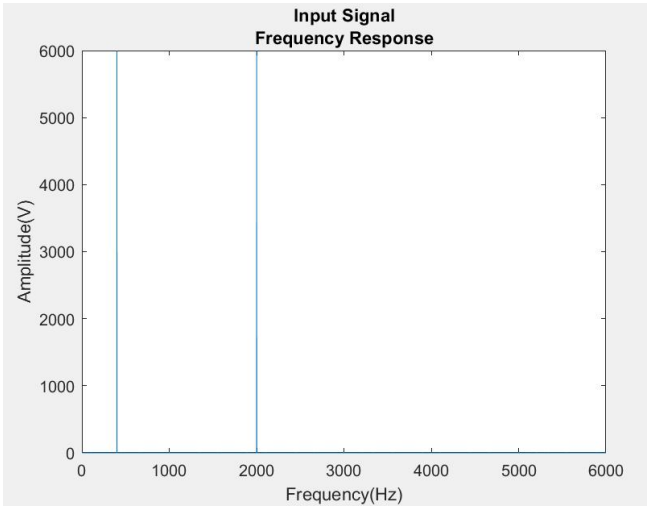
- Low Doppler Spread
- Symbol period << Coherence time
- Signal Variation >Channel Variation

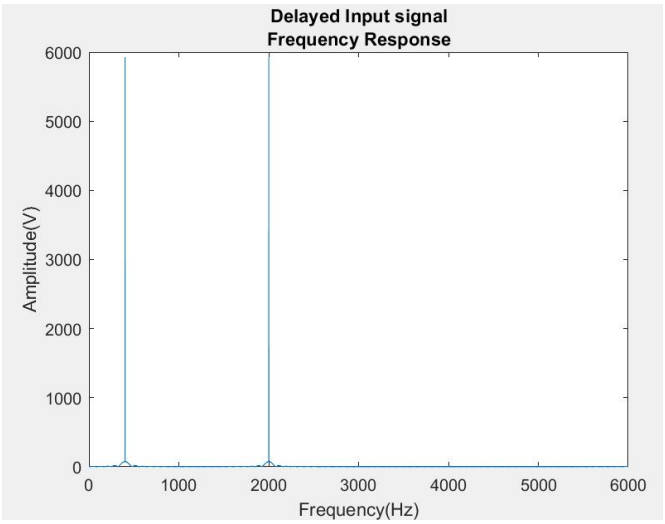
Slow fading results in a loss of SNR. Error correction coding and receiver diversity techniques are used to overcome effects of slow fading.

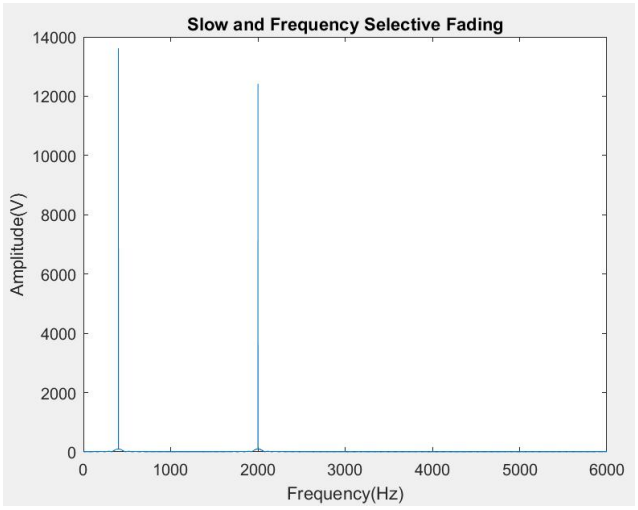
ALGORITHM:

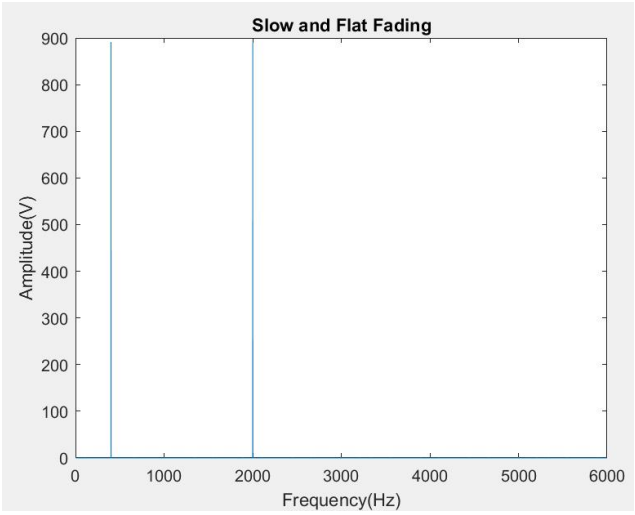
1. Generate a two tone message signal.
2. Fix sampling frequency.
3. Generate a delayed signal.
4. Find the FFT of message signal.
5. For slow & flat fading, generate random complex gaussian variable and multiply with message signal.
6. For slow & frequency selective fading, generate two random complex gaussian variable. Multiply one of the variable with message and the other variable with delayed signal.
7. For fast& flat fading, generate a random complex Gaussian variable and pass it through low pass filter and multiply it with message signal.
8. For fast & frequency selective fading, generate two random complex Gaussian variable. Multiply one of the variable with message and the other variable with delayed signal.Pass it through lowpass filter and multiply it with message signal. Find the FFT of signal and plot it.

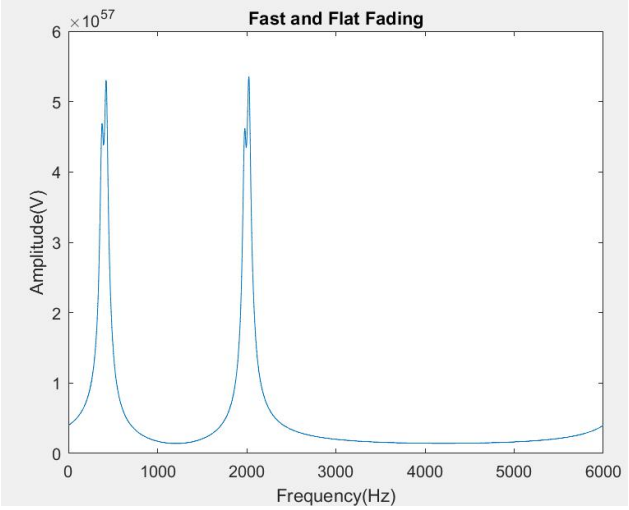
MATLAB OUTPUT:

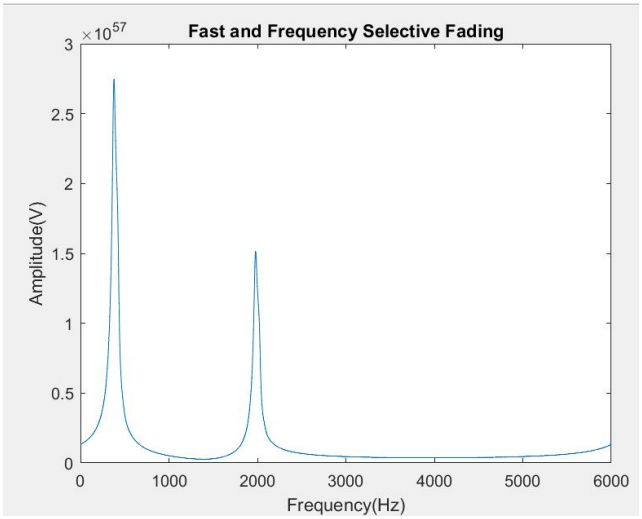










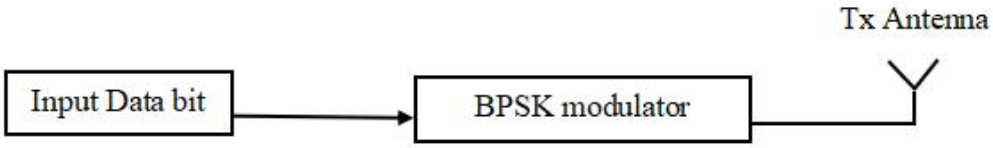


MATLAB CODE:

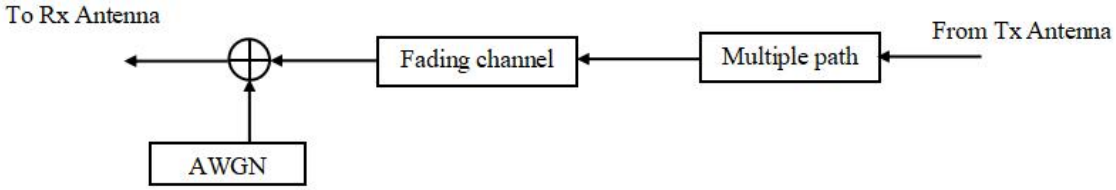
```
clc, clear all; close all;
fs = 6000; f1 = 2000; f2 = 400;
ts = 0:(1/fs):1-(1/fs);
signal1 = exp(complex(0,2*pi*f1*ts));
signal2 = exp(complex(0,2*pi*f2*ts));
message = signal1+signal2;
delayed = [zeros(1,77) message(1:length(message)-77)];
%FREQUENCY RESPONSE
freqres = fft(message); figure;
magnitudeplot = abs(freqres);
plot(1:fs,magnitudeplot);
title({'Input Signal';'Frequency Response'});
xlabel('Frequency(Hz)'); ylabel('Amplitude(V)');
figure; freqres2=fft(delayed);
plot(1:fs,abs(freqres2));
title({'Delayed Input signal';'Frequency Response'});
xlabel('Frequency(Hz)');
ylabel('Amplitude(V)');
%Slow and Flat Fading
h = randn + (1i*randn);
y1 = message.*h;
freqres = fft(y1);
magnitudeplot = abs(fft(y1));
figure;
plot(1:fs,magnitudeplot(1:fs));
title('Slow and Flat Fading');
xlabel('Frequency(Hz)'); ylabel('Amplitude(V)');
%Slow and Frequency Selective Fading
h1 = randn + (1i*randn);
h2 = randn + (1i*randn);
trans2 = (h1.*message) + (h2.*delayed);
magnitudeplot = abs(fft(trans2));
figure; plot(1:fs,magnitudeplot);
title('Slow and Frequency Selective Fading');
xlabel('Frequency(Hz)'); ylabel('Amplitude(V)');
%Fast and Flat Fading
fd = 10;%doppler frequency
hs = randi([0 1],1,length(ts)) +(1i*randi([0 1],1,length(ts)));
[b,a] = butter(12,((2*fd)/1000));
lpf = filter(b,a,hs);
trans3 = lpf.*message;
magnitudeplot = abs(fft(trans3));
figure; plot(1:fs,magnitudeplot);
title('Fast and Flat Fading');
xlabel('Frequency(Hz)'); ylabel('Amplitude(V)');
% Fast and Frequency Selective Fading
ht = randi([0 1],1,length(ts)) +1i*(randi([0 1],1,length(ts)));
lpf1 = filter(b,a,ht); trans4 = (lpf1.*message) +(lpf1.*delayed);
magnitudeplot = abs(fft(trans4));
figure;
plot(1:fs,magnitudeplot);
title('Fast and Frequency Selective Fading');
xlabel('Frequency(Hz)'); ylabel('Amplitude(V)');
```

BLOCK DIAGRAM:

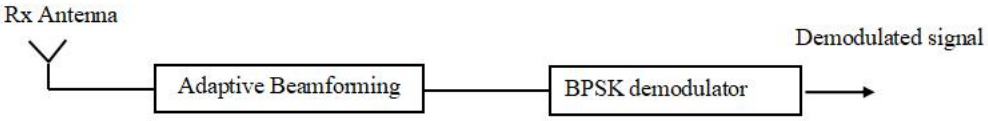
Transmitter:



Channel:



Receiver:



ALGORITHM:

BER VS SNR PERFORMANCE OF CHANNELS

1. Generate BPSK modulated wave.
2. Generate noise sequence that are gaussian distributed.
3. Multiply the modulated wave with the Rayleigh coefficient and add the noise.
4. Detect the received signal, compare it with the input signal to find the error.
5. Plot the BER vs SNR graph for Rayleigh channel.
6. Multiply the modulated wave with the rician coefficient and add the noise.
7. Detect the received signal, compare it with the input signal to find the error.
8. Plot the BER vs SNR graph for rician channel.

RICIAN DISTRIBUTION:

A Rician distribution is one way to model the paths scattered signal stake to a receiver. Specifically, this distribution models line-ofsight scatter — transmissions between two stations in view of each other that have an unobstructed path between them.Line-of-sight scatter includes FM radio waves,microwaves,MRI images in the presence of noise, and satellite transmissions. The distribution also models Rician fading, which is a way to show how signal cancellations affect radio propagation. The probability density function formula is: RAYLEIGH DISTRIBUTION Rayleigh distribution is a continuous probability distribution for positive-valued random variables. The data can be given by the mean value and a lower bound, or by a parameter and a lower bound. These are interconnected by a well-documented relationship given in the literature. For instance, if the mean $\mu=2$ and the lower bound is $\gamma=0.5$,then $\theta=1.59577$ and the standard deviation is $\sigma=1.0454$.

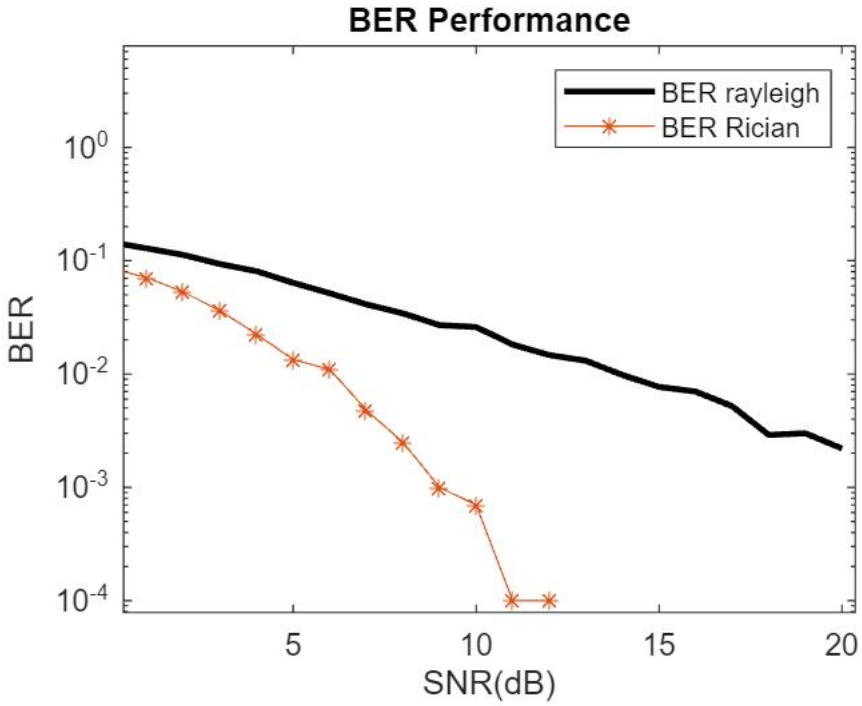
$$f(x \mid \nu, \sigma) = \frac{x}{\sigma^2} \exp\left(\frac{-(x^2+\nu^2)}{2\sigma^2}\right) I_0\left(\frac{x\nu}{\sigma^2}\right),$$

RAYLEIGH DISTRIBUTION

Rayleigh distribution is a continuous probability distribution for positive-valued random variables. The data can be given by the mean value and a lower bound, or by a parameter and a lower bound. These are interconnected by a well-documented relationship given in the literature. For instance, if the mean $\mu=2$ and the lower bound is $\gamma=0.5$,then $\theta=1.59577$ and the standard deviation is $\sigma=1.0454$.

$$f(X) = \frac{X}{\beta^2} \exp\left[-\frac{1}{2}\left(\frac{X}{\beta}\right)^2\right]$$

MATLAB OUTPUT:



MATLAB CODE:

```
%Rician and Rayleigh Channel
clc clear all close all
n = 10000;
i = randi([0,1],1,n);
i1 = 2*i -1;
a = randn(1,n); b = randn(1,n);
rc = 1/sqrt(2)*(sqrt(a.^2+b.^2));
for l = 0:1:20
snr = 10^(l/10); sdev = sqrt(0.5/snr);
N = random('norm',0,sdev,[1,n]);
yrc = rc.*i1+N; YR = (yrc>=0);
ErrorR = sum((xor(YR,i)));
ber_R(l+1) = ErrorR/n;
end
q = 0:1:20;
semilogy(q,ber_R(q+1),'k-','LineWidth',2);
hold on;
axis([0 20 10^-5 1]);
xlabel('SNR(dB)'); ylabel('BER');
k1 = 10; mean = sqrt(k1/(k1+1));
sigma = sqrt(1/(2*(k1+1)));
Nr2 = randn(1,length(i1))*sigma+mean;
Ni2 = randn(1,length(i1))*sigma;
No3 = sqrt(Nr2.^2+Ni2.^2);
for k = 0:1:20
snrl = 10^(k/10); Np = 1/snrl;
sd = sqrt(Np/2); No = random('Normal',0,sd,1,length(i1));
t1 = i1.*No3+No; z1 = t1./No3;
op1 = (z1>0);
Berr(k+1) = sum(xor(op1,i))/n;
end
k = 0:1:20;
semilogy(k,Berr(k+1),'*-');
hold on;
axis([0 20 10^-5 1]);
title(' BER Performance ');
xlabel('SNR(dB)'); ylabel('BER');
legend('BER rayleigh','BER Rician');
```

INFERENCE:

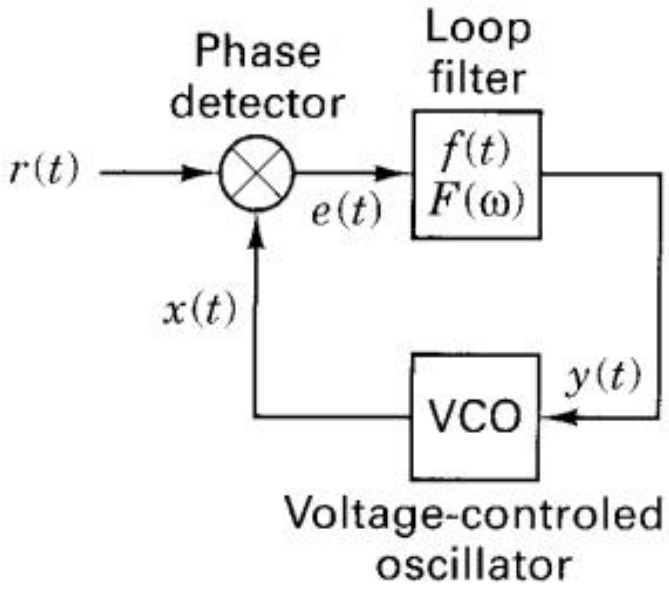
- The amplitude remains the same in slow flat fading.
- The amplitude varies for slow and frequency selective fading.
- The amplitude remains the same but the signal is spread in fast and flat fading.
- The amplitude changes and the signal is spread over the frequencies in fast and frequency selective fading.

RESULT:

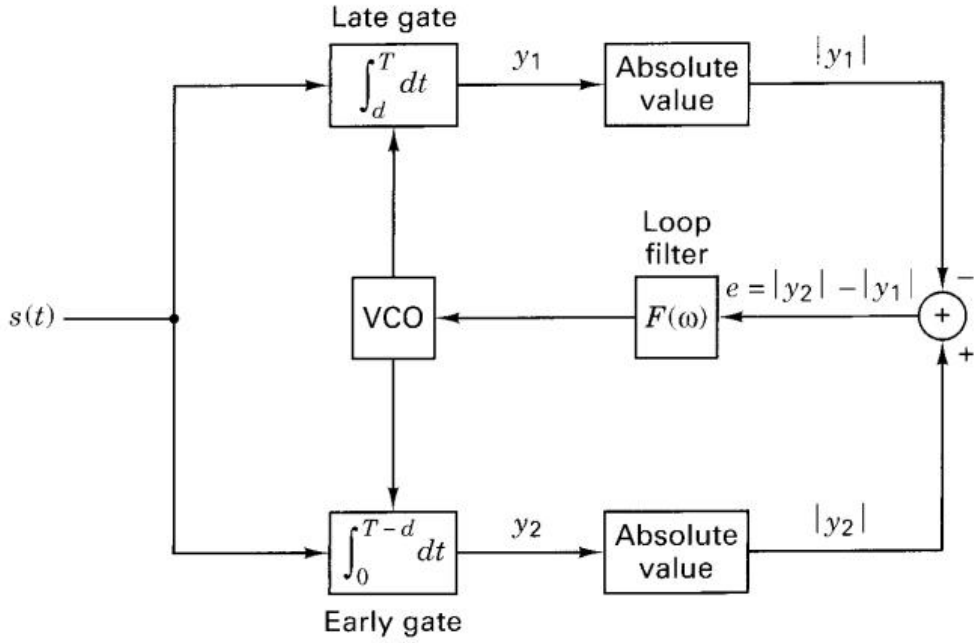
Thus we have simulated and plotted the spectra of fourdifferent types of fading and BER performance of Rayleigh & Rician channel were analyse using MATLAB and verified successfully.

BLOCK DIAGRAM:

Carrier Synchronization:



Symbol Synchronization:



EXP NO: 07 DATE:	Synchronization Techniques

AIM:

To study and simulate the various synchronization techniques in a communication system and to observe the same without synchronization.

SOFTWARE REQUIRED:

Matlab 2022b

THEORY:

The term synchronization is used in two different contexts. In the case of video, it refers to the process of synchronizing display pixel scanning to a synchronization source. In the case of telecommunication, it is the process by which incoming framed data are extracted for decoding with the help of frame alignment signals. In digital communication systems data is not send as a simple stream of bits or bytes but in terms of frames or packets. So, the receiver must be able to recognize the start of the frame by process called frame synchronization. Generally, there are three levels of synchronization in a complete communication system:

- Carrier synchronization
- Symbol synchronization
- Frame synchronization

CARRIER SYNCHRONIZATION

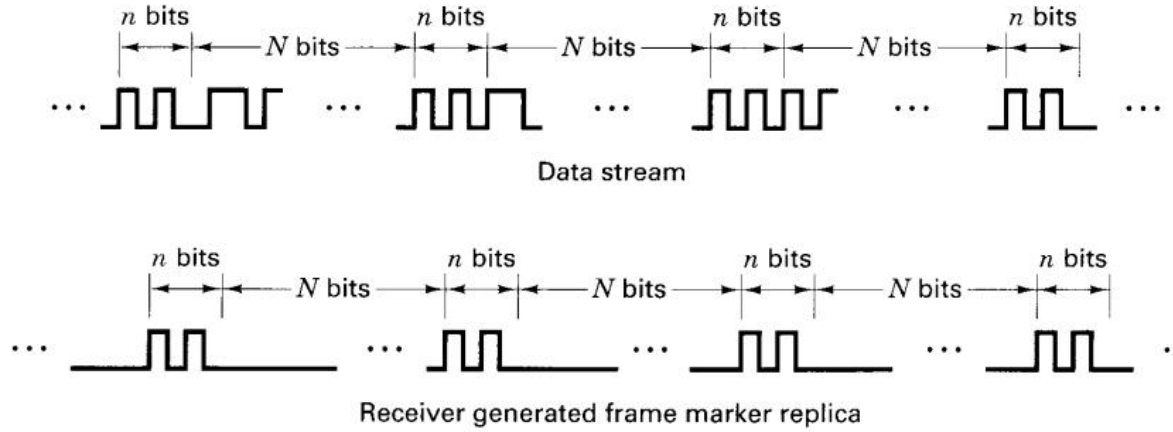
PLL are servo-control loops, whose controlled parameter is the phase of a locally generated replica of the incoming carrier phase. Phase locked loops have three basic components :

1) A phase detector
2) A loop filter
3) A voltage-controlled oscillator (VCO)

A phase detector is a device that measures the difference in phase between the incoming phase and the local replica. As the incoming signal and the local replica change with respect to each other , the phase difference or (phase error) becomes a time varying signal into the loop filter. The loop filter governs the PLL's response to these variations in the error signal. The VCO is a device that produces the carrier replica. The VCO as the name implies produces a sinusoidal signal whose frequency is controlled by a voltage level at the device input.

A VCO is an oscillator whose output frequency is a linear function of its input voltage over some range of input and output. A positive input voltage will cause the VCO output frequency to be greater than its uncontrolled value, ωo, while a negative voltage will cause it to be less. Phase lock is achieved by feeding a filtered version of the phase difference (i.e., the phase error) between the incoming signal r(ȼ) and the output of the VCO, x(1), back to the input of the VCO, y(t).

Frame Synchronization:



ALGORITHM :

Carrier Synchronization :

1. Generate a message signal and carrier signal with phase offset
2. Modulate it using DSB-SC signal
3. Initialize phase vector and time vector
4. Add noise
5. Use carrier synchronization to overcome phase offset
6. Plot the output signal and the phase of the signal

Symbol Synchronization :

1) Generate random bits of length n
2) Transmit the data with symbol in between
3) Until the data meet the symbol generate zero in the receiver
4) Once the symbol is found it should follow the incoming data
5) Plot the transmitted data, synchronized data and unsynchronized data

Frame Synchronization :

1. Convert the message to bit sequence and define a start and stop bit
2. Then bit stuff the data with start and stop bit
3. We will first send the data being synchronized and without being synchronized and analyze the results.

FRAME SYNCHRONIZATION

Here the marker is periodically inserted by the transmitter in each start of each frame of data. At the receiver there is a correlation between the received frame and stored marker pattern and frame synchronization is obtained by examining the correlation values. It is better to choose a marker pattern used in frame synchronization with good autocorrelation properties. Usually in the techniques of frame synchronization, special sequences such as Barker sequences, are used for frame synchronization in digital communication systems. The transmitter and the receiver must agree ahead of time on which frame synchronization scheme they will use. Common frame synchronization schemes are: Framing bit: A common practice in telecommunications, for example in T-carrier, is to insert, in a dedicated time slot within the frame, a non - information bit or framing bit that is used for synchronization of the incoming data with the receiver. In a bit stream, framing bits indicate the beginning or end of a frame. They occur at specified positions in the frame, do not carry information, and are usually repetitive. Sync word framing Some systems use a special sync word at the beginning of every frame. CRC-based framing Some telecommunications hardware uses CRC-based framing.

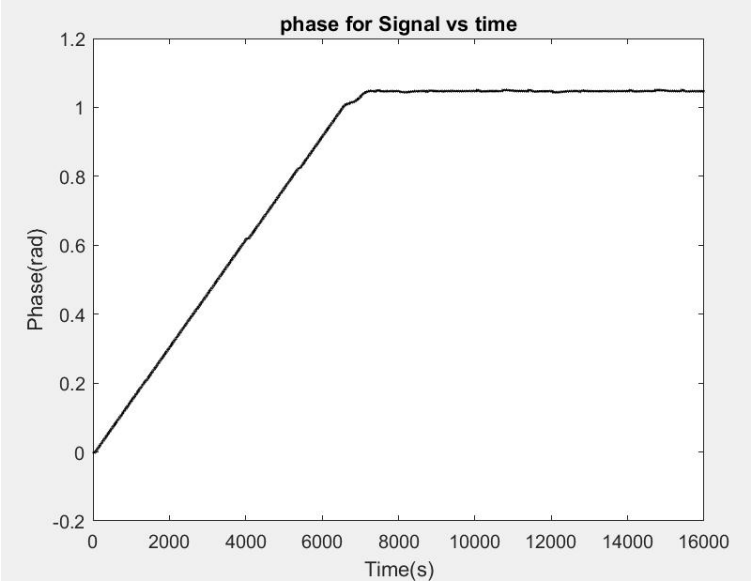
SYMBOL SYNCHRONIZATION :

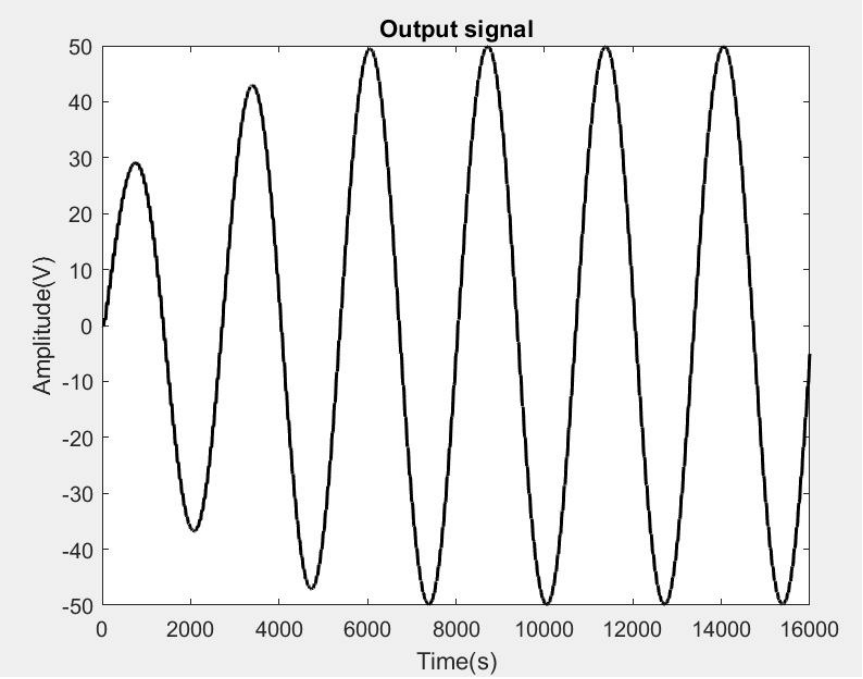
All digital receivers need to be synchronized to the incoming digital symbol transitions in order to achieve optimum demodulation. In the discussion that follows, we will consider several of the basic types of designs of symbol or data synchronizers. For ease of terminology and notation, but the extension to nonbinary baseband signals should be apparent.

This class of synchronizers is called non-data-aided (NDA) synchronizers. There is another class of symbol synchronizers that use known information about the data stream. This knowledge may be obtained by feeding back decisions on received data, or because a known sequence has been injected into the data stream. Data-aided (DA) techniques have become more important and prevalent with the increasing use of bandwidth-efficient modulation. This is especially true with the class of continuous phase modulations. Data-aided techniques will be considered in somewhat more detail in the succeeding section.

The symbol synchronizers that will be considered here can be classified into two basic groups. The first group consists of the open-loop synchronizers. These circuits recover a replica of the transmitter data clock output directly from operations on the incoming data stream. The second group comprises the closed-loop synchronizers. Closed-loop data synchronizers attempt to lock a local data clock to the incoming signal by use of comparative measurements on the local and incoming signals. Closed-loop methods tend to be more accurate, but they are much more costly and complex.

MATLAB OUTPUT:





MATLAB CODE:

Carrier Synchronization

```
clc
clear all
close all
% GENERATING SIGNAL
t = 0:16000; % time scale
fs = 4e5; % Sampling Frequency
fc = 4000; % Carrier Frequency
m = sin(2*pi*150*t/fs); % sample message signal
c = cos(2*pi*fc*t/fs + pi/3); % carrier signal with phase offset
st = m.*c; % DSB-SC signal
% RECEIVER PART
N = length(st);
t = 0:1:N-1; % Time vector
phi = zeros(1,N); % Phase vector of VCO initialize
s1 = zeros(1,N);
s2 = zeros(1,N);
y1 = zeros(1,N);
y2 = zeros(1,N);
for i = 1:N
if i>1
phi(i) = phi(i-1) - (5*10^-5)*pi*sign(y1(i-1)*y2(i-1));
end
s1(i) = st(i) * cos(2*pi*fc*t(i)/fs + phi(i));
s2(i) = st(i) * sin(2*pi*fc*t(i)/fs + phi(i));
if i<=100
for j=1:i
y1(i) = y1(i) + s1(j);
y2(i) = y2(i) + s2(j);
end
else
for j = i-99:i
y1(i) = y1(i) + s1(j);
y2(i) = y2(i) + s2(j);
end
end
end
figure;
plot(t,y1,'k','Linewidth',1.5);title('Output signal');
xlabel('Time(s)');ylabel('Amplitude(V)');
figure;
plot(t,phi,'k','Linewidth',1.5);title('phase for Signal vs time');
xlabel('Time(s)');ylabel('Phase(rad)');
phase = phi(end);
disp(phase)
```

MATLAB OUTPUT:

```
bits =

  4×7 char array

    '1101000'
    '1100101'
    '1101100'
    '1110000'


bit_stream =

    '11010001100101111011001110000'
```

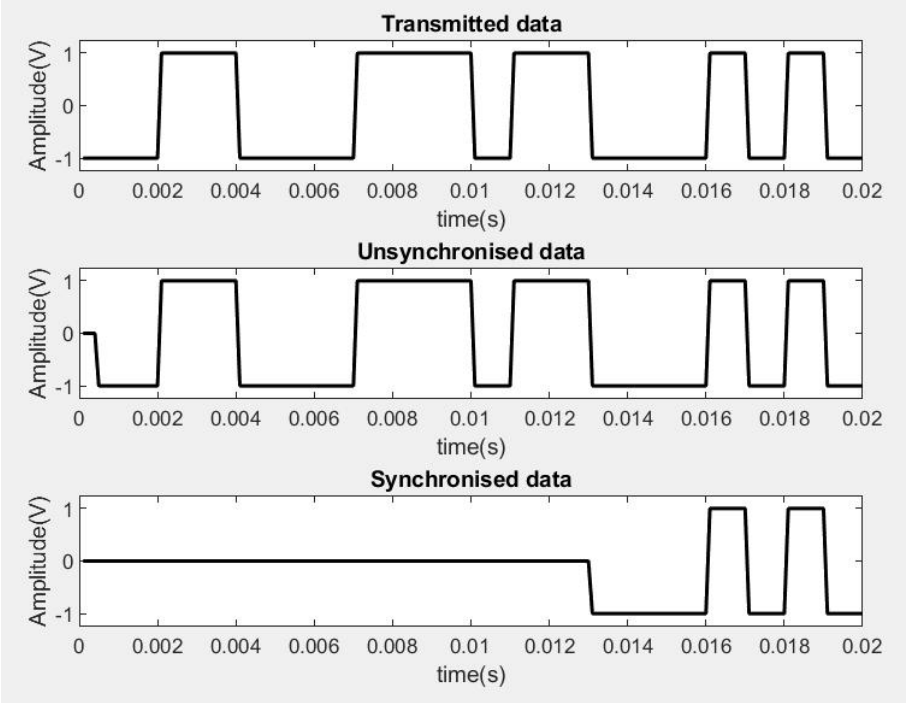
```
------With Frame Sychronisation-----------

bit_stuffed =

    '1111010000011110010100111101100001111100000'


received_msg =

  4×1 char array

    'h'
    'e'
    'l'
    'p'
```

```
------Frames not synchronized-----------

bit_stuffed =

    '1110100010011100101100111011001001111000010'


received_msg =

  4×1 char array

    'Q'
    'K'
    'Y'
    'a'
```

Frame Synchronization

```
clc
clear all
close all

% encoding message
bits = dec2bin('help')
bit_stream = reshape(bits.',1,[]);
bit_stream
%defining start and stop bits
mark='11';
space = '00';
%-------------------------FRAME SYNCHRONIZED---------
% bit stuffing
disp("------With Frame Sychronisation-----------")
[r,c] = size(bits);
bit_stuffed = [];
for i = 1:7:length(bit_stream)
bit_stuffed = [bit_stuffed mark bit_stream(i:i+6) space];
end %
bit_stuffed
% frame synchronized decoding
bits_recover1 = [];
for j = 1:11:length (bit_stuffed)
if bit_stuffed(j:j+1) == mark
bits_recover1 = [bits_recover1 bit_stuffed(j+2 : j+8)];
end
end
%recovering the msg
bits_ordered1 = (reshape(bits_recover1,c,r)).';
received_msg = char(bin2dec (bits_ordered1))
%-------------FRAME DE SYNCHRONIZED-----------------
disp("------Frames not synchronized-----------")
bit_stream = [bit_stream(2:length(bit_stream)) '1'];
% bit stuffing
[r,c] = size(bits);
bit_stuffed = [];
for i = 1:7:length(bit_stream)
bit_stuffed = [bit_stuffed mark bit_stream(i:i+6) space];
end
bit_stuffed
% decoding without frame synchronization
bits_recover2= [];
for j = 1:11:length (bit_stuffed)
if bit_stuffed(j:j+1) == mark
bits_recover2 = [bits_recover2 bit_stuffed(j+2 : j+8)];
end
end
% recovering the msg
bits_ordered2 = (reshape(bits_recover2,c,r)).';
received_msg = char(bin2dec (bits_ordered2))
```

MATLAB OUTPUT:




```
     1

     6

     6

    15

lock1
case1
    24

     2

     6

     6

    34

lock1
case1
```

```
    43

     3

     6

     6

    53

    63

lock1
case1
    72

     4

     6

     6

    82

    92
```

```
lock1
case1
   101

     5

     6

     6

   111

   121

   131

lock1
Locked at final position :
   131

Synchronised at time
    0.1310
```

Symbol Synchronization

```
clc;clear all;close all;
n=20;up=10;t=up;T=5;d=4;Tb=0.001;
bs1=randi([0 1],1,n);%bs1=[0 1 1 0 1 0 1 1 0 1];
bs=bs1;bs(bs==0)=-1; u=ones(1,up);
bs_up1=upsample(bs,up);
bs1_up=bs_up1(1,1:length(bs_up1)-(up-1));
bs_up=conv(bs1_up,u); test=bs_up(T:end);
s=[zeros(1,(T-1)), test];
for i=1:length(bs_up(3:end))
test_b=bs_up(T:end);
P1=[u((d+1):end), zeros(1,d)];
P2=[zeros(1,d), u(1:(t-d))];
Pe=abs(sum(test_b(1:t).*P1));
Pl=abs(sum(test_b(1:t).*P2));
disp(i);disp(Pe); disp(Pl);
if (Pe==Pl) q=1;
for p=1:3
T=T+t; disp(T); test_b=bs_up(T:end);
Pe=abs(sum(test_b(1:t).*P1));
Pl=abs(sum(test_b(1:t).*P2));
if (Pe==Pl)
q=q+1;
continue;
else
break
end
end
disp('lock1');
if (q==4)
break;
end
end
if (Pe>Pl)
disp('case1');
T=T+t-1;
disp(T);
elseif (Pe<Pl)
disp('case2');
T=T+t+1; disp(T);
end
end
disp('Locked at final position :'); disp(T);
disp('Synchronised at time'); disp(T*Tb);
t1=Tb/10:Tb/10:Tb*n;
figure
subplot(3,1,1); plot(t1,bs_up,'k','Linewidth',1.5);
ylim([-1.25 1.25]);
xlabel('time(s)'); ylabel('Amplitude(V)');
title('Transmitted data');
%s=[zeros(1,(T-1)), test];
subplot(3,1,2); plot(t1,s,'k','Linewidth',1.5);
ylim([-1.25 1.25]);
xlabel('time(s)'); ylabel('Amplitude(V)');
title('Unsynchronised data');
s1=[zeros(1,(T-1)), test_b];
subplot(3,1,3);plot(t1,s1,'k','Linewidth',1.5);
ylim([-1.25 1.25]);
xlabel('time(s)'); ylabel('Amplitude(V)');
title('Synchronised data');
```

INFERENCE :

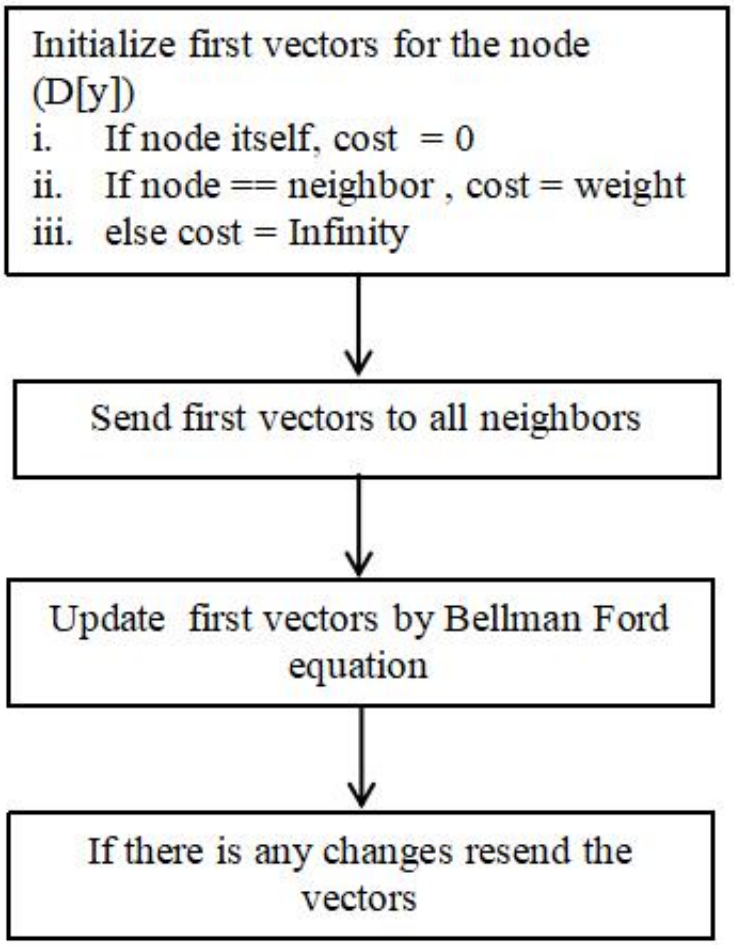
- Synchronization techniques are essential for error – free decoding of transmitted signal.
- PLL circuits are essential for phase corrections in a carrier synchronization techniques.
- Start and stop bits define the frames of a bit steam in frame synchronization.

RESULT:

Thus various synchronisation techniques are studied and implemented using matlab 2022b.

FLOWCHART:

DISTANCE VECTOR ROUTING:



EXP NO: 08 DATE:	Network Routing Algorithms

AIM:

To study the network routing algorithms like distance vector routing, Link state algorithm and Path – vector routing.

SOFTWARE REQUIRED:

MATLAB R2022b

THEORY:

A routing algorithm is **a procedure that lays down the route or path to transfer data packets from source to the destination**. They help in directing Internet traffic efficiently. After a data packet leaves its source, it can choose among the many different paths to reach its destination. Routing is the process of forwarding the packets from source to the destination but the best route to send the packets is determined by the routing algorithm.

Here we will see Distance Vector Routing and Link State Routing Algorithms.

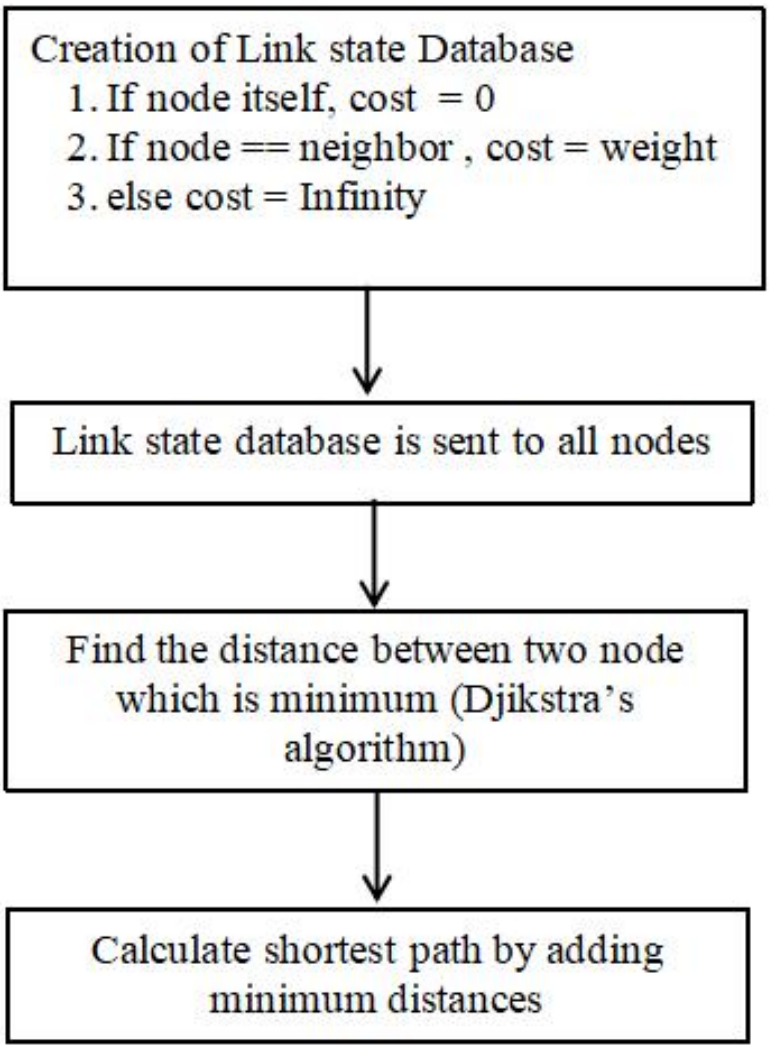
DISTANCE VECTOR ROUTING:

Distance vector routing is **an asynchronous algorithm in which node x sends the copy of its distance vector to all its neighbors**. When node x receives the new distance vector from one of its neighboring vector, v, it saves the distance vector of v and uses the Bellman-Ford equation to update its own distance vector.

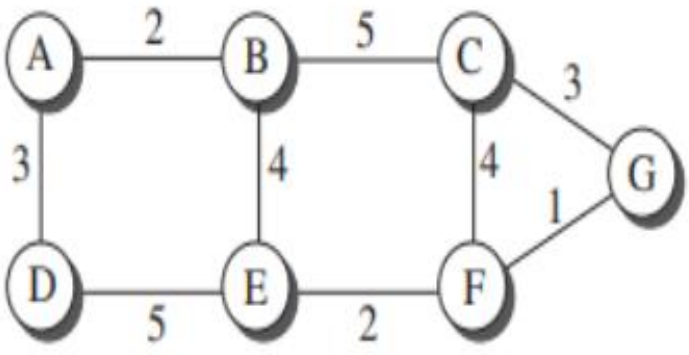
LINK STATE ROUTING:

The Link State Routing Algorithm is **an interior protocol used by every router to share the information or knowledge about the rest of the routers on the network**. The link state routing algorithm is a distributed algorithm using which every router computes its routing table. Each node uses **Dijkstra's algorithm** on the graph to calculate the optimal routes to all nodes. The Link state routing algorithm is also known as Dijkstra's algorithm which is used to find the shortest path from one node to every other node in the network.

LINK STATE ROUTING:



EXAMPLE NETWORK :



a. The weighted graph

ALGORITHM:

```
Distance_Vector_Routing ( )
  {
  // Initialize (create initial vectors for the node)
  D[myself ] = 0
  for (y = 1 to N)
  {   if (y is a neighbor)
  D[y] = c[myself ][y]
    else
  D[y] = ∞  }
    send vector {D[1], D[2], .... , D[N]} to all neighbors
    // Update (improve the vector with the vector received from a neighbor)
    repeat (forever)
  { wait (for a vector Dw from a neighbor w or any change in the link)
    for (y = 1 to N) {
  D[y] = min [D[y], (c[myself ][w] + Dw[y ])]    } // Bellman-Ford equation
  if (any change in the vector)
  send vector {D[1], D[2], .... , D[N]} to all neighbors
  }
  } // End of Distance Vector

Dijkstra's Algorithm ( )
  {
  // Initialization
  Tree = {root} // Tree is made only of the root
  for (y = 1 to N) // N is the number of nodes
  { if (y is the root)
  D[y] = 0 // D[y] is shortest distance from root to node y
  else if (y is a neighbor)
  D[y] = c[root][y] // c[x][y] is cost between nodes x and y in LSDB
  else D[y] = ∞
  }
  // Calculation
  repeat
  { find a node w, with D[w] minimum among all nodes not in the Tree
  Tree = Tree ∪ {w} // Add w to tree
  // Update distances for all neighbors of w
  for (every node x, which is a neighbor of w and not in the Tree)
  { D[x] = min{D[x], (D[w] + c[w][x])} }
  } until (all nodes included in the Tree)
  } // End of Dijkstra
```

MATLAB OUTPUT:

```
First vector of node A:
     0     2   Inf     3   Inf   Inf   Inf

First vector of node B:
     2     0     5   Inf     4   Inf   Inf

First vector of node C:
   Inf     5     0   Inf   Inf     4     3

First vector of node D:
     3   Inf   Inf     0     5   Inf   Inf

First vector of node E:
   Inf     4   Inf     5     0     2   Inf

First vector of node F:
   Inf   Inf     4   Inf     2     0     1

First vector of node G:
   Inf   Inf     3   Inf   Inf     1     0


Distance vector of node A:
     0     2     7     3     6     8     9

Distance vector of node B:
     2     0     5     5     4     6     7

Distance vector of node C:
     7     5     0    10     6     4     3

Distance vector of node D:
     3     5    10     0     5     7     8

Distance vector of node E:
     6     4     6     5     0     2     3

Distance vector of node F:
     8     6     4     7     2     0     1

Distance vector of node G:
     9     7     3     8     3     1     0
```

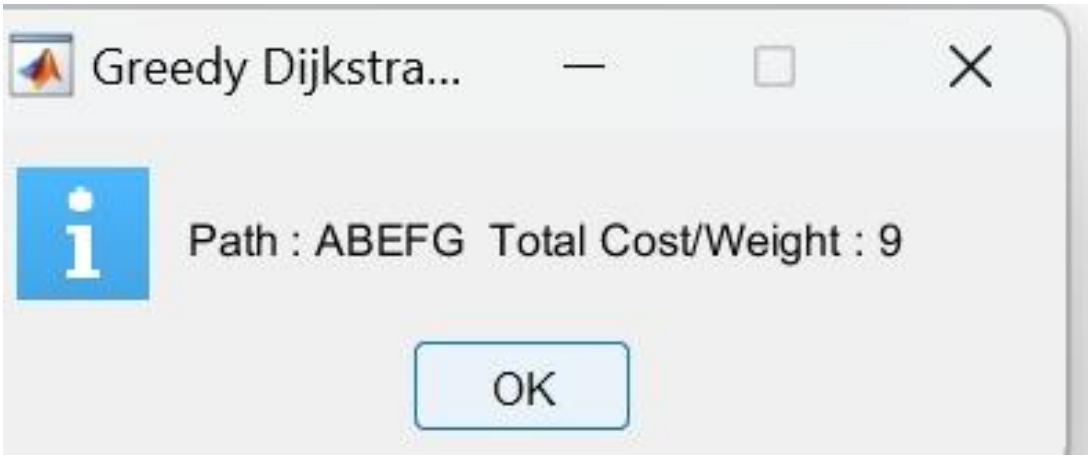
MATLAB CODE:

Distance vector routing:

```
clc
clear all
close all
% %Nodes in the given system
% n = input('Please enter the number of nodes less than 26 = ');
% if (n>26)
%     error('We can calculate upto 26 nodes only')
% end
% disp("Enter 'inf' if it is not the neighbouring node: ")
%
% % Defining node numbers in terms of Alphabets
% alphabets= char(65:1:65+n);
% adj=zeros(n,n);
% for p=1:n
%     for q=1:n
%         adj(p,q)=input(['Enter Weight between [',alphabets(p),']','[',alphabets(q),'] = ']);
%     end
% end

alphabets = char(65:1:65+7);
adj = [0     2   Inf     3   Inf   Inf   Inf;  2     0     5   Inf     4   Inf   Inf; Inf     5
0   Inf   Inf     4     3;
    3   Inf   Inf     0     5   Inf   Inf; Inf     4   Inf     5     0     2   Inf; Inf   Inf
4   Inf     2     0     1;
    Inf   Inf     3   Inf   Inf     1     0];
n = 7;
for i = 1:n
    message = ['First vector of node ',alphabets(i),':'];
    disp(message)
    disp('--')
    disp(adj(i,:))
end

for i = 1:n
    neighbor = find(adj(i,:) > 0 & adj(i,:) < inf);
    for j = 1:n
        if (adj(i,j) == inf)
            for k = 1: length(neighbor)
                %Bellman Ford Equation
                adj(i,j) = min(adj(neighbor(k),j)+adj(i,neighbor(k)),adj(i,j));
            end
        end
    end
end

for i = 1:n
    for j = n:-1:1
        adj(i,j) = adj(j,i);
    end
end
disp('--------------------------------------------------------------------------------')
for i = 1:n
    message = ['Distance vector of node ',alphabets(i),':'];
    disp(message)
    disp('--')
    disp(adj(i,:))
end
```

MATLAB OUTPUT:

```
Link state database
     0     2   Inf     3   Inf   Inf   Inf
     2     0     5   Inf     4   Inf   Inf
   Inf     5     0   Inf   Inf     4     3
     3   Inf   Inf     0     5   Inf   Inf
   Inf     4   Inf     5     0     2   Inf
   Inf   Inf     4   Inf     2     0     1
   Inf   Inf     3   Inf   Inf     1     0

Please enter the source node = 1
Please enter the destination node = 7
```



Link State Routing: (Djikstra's Algorithm)

```
clc
Clear all
% % Taking number of input nodes from user
% n=input('Please enter the number of nodes less than 26 = ');
%
% % Checking for bad input
n = 7;
% Defining node numbers interms of Alphabets
alphabets= char(65:1:65+n);
adj=zeros(n,n);
% for p=1:n
%     for q=1:n
%         adj(p,q)=input(['Enter Weight between [',alphabets(p),']','[',alphabets(q),'] = ']);
%     end
% end

adj = [0   2  Inf  3 Inf  Inf  Inf;  2   0   5   Inf   4   Inf   Inf; Inf   5   0   Inf   Inf
4   3;  3   Inf  Inf   0   5   Inf   Inf; Inf   4   Inf   5   0   2   Inf; Inf  Inf   4   Inf
2    0   1; Inf  Inf   3   Inf  Inf   1   0];
disp("Link state database")
disp(adj)
%Coversting status of all nodes to Tentative
for p=1:n
    a(p).status=0;
end

temp=[];
acc=[];

% Taking input from user regarding source & destination nodes

s=input('Please enter the source node = ');
t=input('Please enter the destination node = ');
% converting source node to status PERMANANT
a(s).status=1;
path=s;
% Main algorithm of Greedy Dijkstra
for k=1:inf
for h=1:n
 if a(h).status==0   % If Unknown
     temp=[temp adj(s,h)];
 else
     temp=[temp inf];
 end
end
[aa,bb]=min(temp);
acc=[acc aa];
temp=[];
s=bb;
a(s).status=1;
path=[path s];
if s==t
    break
end
end
% Creating dialogue box for Output
helpdlg(['Path : ' alphabets(path) '  Total Cost/Weight : '...
       num2str(sum(acc))],'Greedy Dijkstra Algorithm')
```

INFERENCE:

Distance vector protocols send their entire routing table to directly connected neighbors. Link state protocols send information about directly connected links to all the routers in the network.

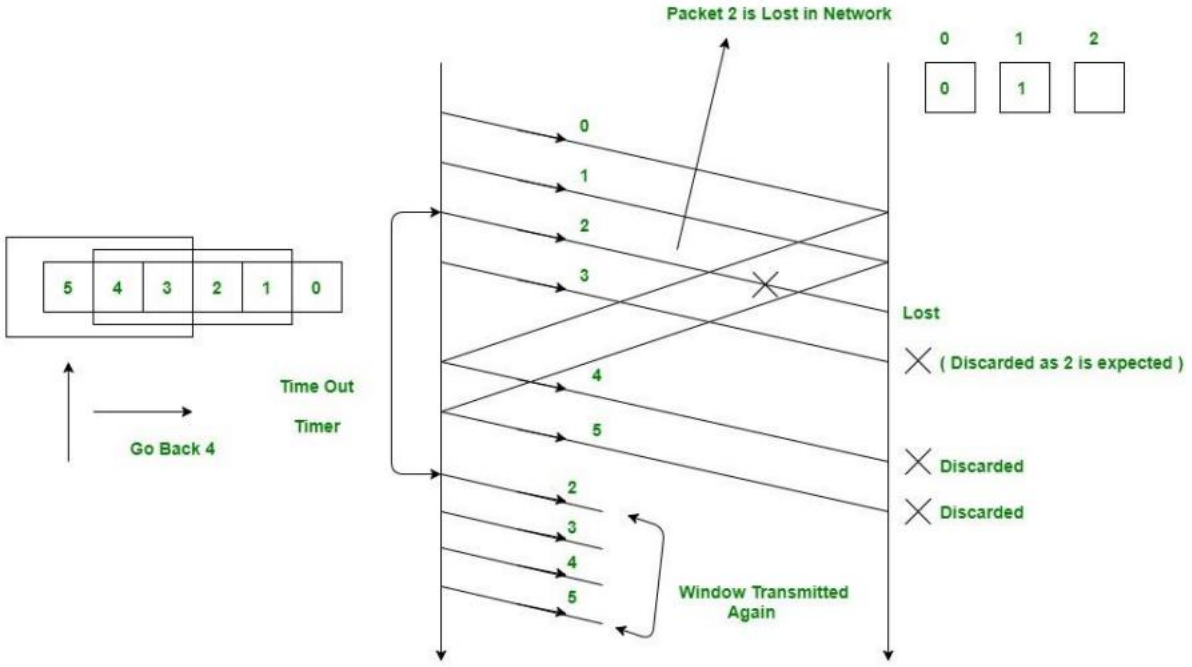
RESULT:

Hence, different routing algorithm is analysed through Matlab.

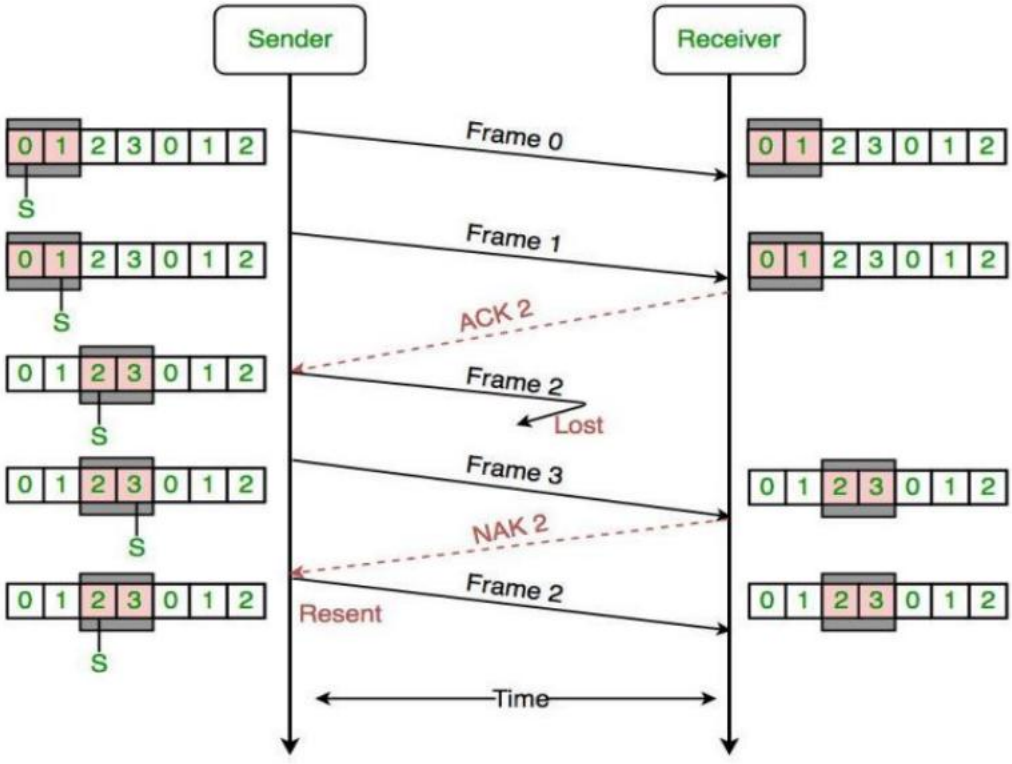
CONCLUSION:

- Distance vector protocols have slow convergence and suffer from the count-to-infinity problem.
- Link state convergence occurs faster than distance vector convergence. This is because link state establishes a neighbor relationship with directly connected peers and shares routing information with its neighbors only when there are changes in the network topology.

GO BACK N protocol:



SELECTIVE REPEAT protocol:



EXP NO: 09 DATE:	LLC Protocols

AIM:

To implement

- Go Back 'N' Protocol
- Selective Repeat Protocol

and to determine the throughput and delay for each.

SOFTWARE REQUIRED:

NETSIM Software

THEORY:

GO BACK 'N' PROTOCOL:

Go Back 'N' is a connection-oriented transmission. The sender transmits the frames continuously. Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size. The sender has a window i.e., a buffer to store the frames. This buffer size is the number of frames to be transmitted continuously. The size of the window depends on the protocol designer

SELECTIVE REPEAT PROTOCOL:

It is similar to Go Back 'N' Protocol, but the sender send frame only after the reception of ACK signal. It may be used as a protocol for delivery and ACK for message units for delivery of subdivided message. It is used as a protocol for delivery of message sender continuous to send frames specifies by windows size even after becoming frameless. Once the sender has sent all the frame in its windows, it resends the frame number given by ACK and continuous where it left off.

GO BACK ‘N’ PROTOCOL

LAYOUT:

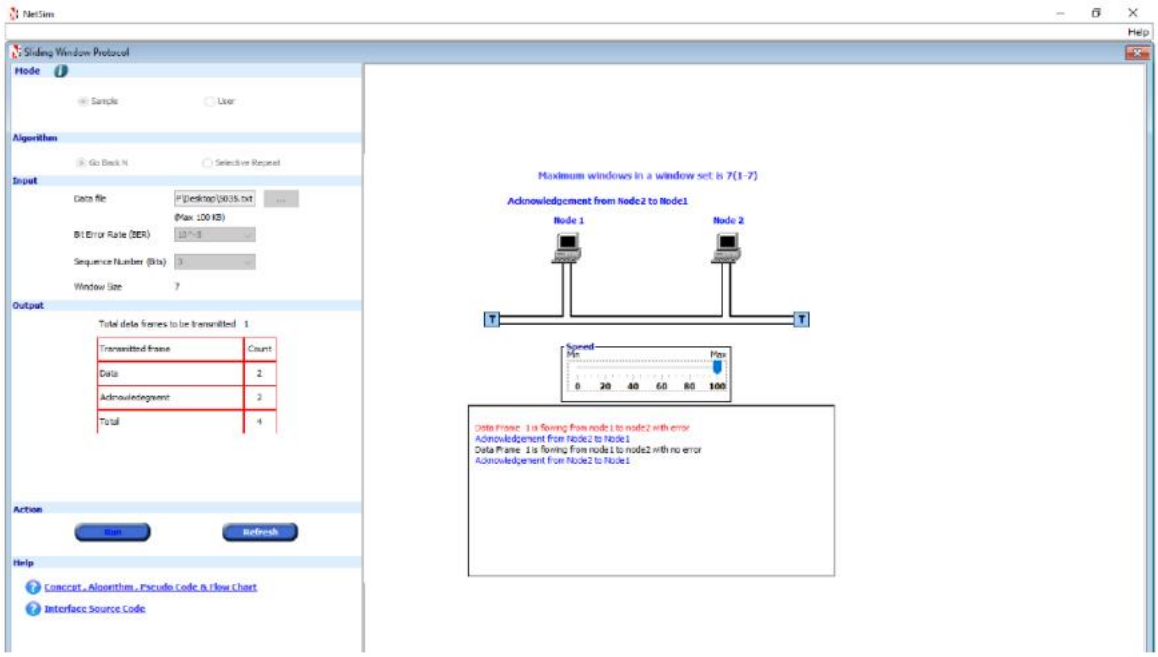


TABLE: (Sequence number : 3 bits)

BER	TOTAL DATA FRAMES TO BE TRANSMITTED	DATA	ACKNOWLEDGEMENT	TOTAL
10^-5	1	2	2	4
10^-6	1	2	2	4
10^-7	1	1	1	2
10^-8	1	1	1	2
10^-9	1	1	1	2
NO ERROR	1	1	1	2

ALGORITHM:

GO BACK 'N' PROTOCOL:

- The source code transmits the frames continuously.
- Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size.
- The source code has a window i.e., a buffer to store the frames. This buffer size is the number of frames to be transmitted continuously. The size of the window depends on the protocol designer.
- For the first frame, the receiving node forms a positive acknowledgement if the frame is received without any error.
- If subsequent frames are received without error (up to window size) cumulative positive acknowledgement is formed.
- If the subsequent frame is received with error, the cumulative acknowledgement error-free frames are transmitted. If, In the same window two frames or more frames are received with error, the second and the subsequent error frames are neglected. Similarly, even the frames received without error after the receipt of a frame with error are neglected.
- The source code re-transmits all frames of window from the first error frame.
- If the frames are errorless in the next transmission and if the acknowledgement is error-free, the window slides by the number of error-free frames being transmitted.
- If the acknowledgement is transmitted with error, all the frames of window at source are re-transmitted and window doesn't slide.
- This concept of replacing the transmission from the first error frame in the window is called as Go Back-N transmission flow control protocol.

SELECTIVE REPEAT PROTOCOL

LAYOUT:

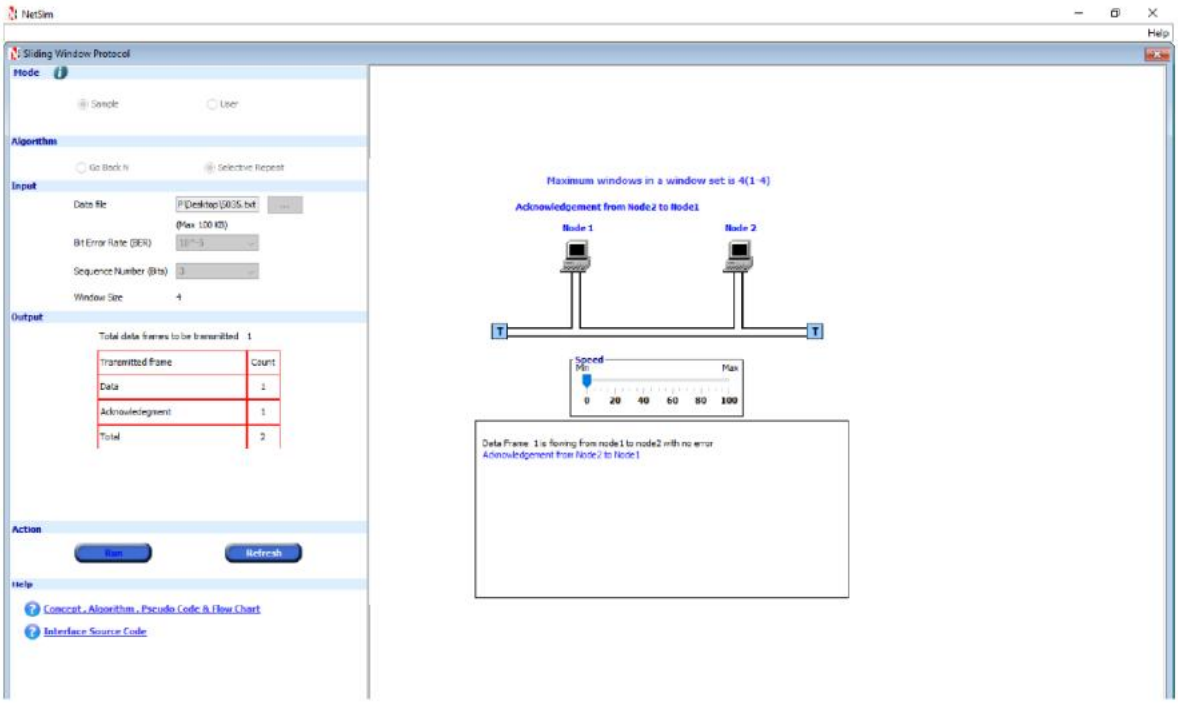


TABLE: (Sequence number : 3 bits)

BER	TOTAL DATA FRAMES TO BE TRANSMITTED	DATA	ACKNOWLEDGEMENT	TOTAL
10^-5	1	1	1	2
10^-6	1	1	1	2
10^-7	1	1	1	2
10^-8	1	1	1	2
10^-9	1	1	1	2
NO ERROR	1	1	1	2

SELECTIVE REPEAT PROTOCOL:

- The source node transmits the frames continuously.
- Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size.
- The source node has a window i.e., buffer to store the frames. This buffer size is the number of frames to be transmitted continuously.
- The receiver has a buffer to store the received frames. The size of the buffer depends upon the window size defined by the protocol designer.
- The source node transmits frames continuously till the window size is exhausted. If any of the frames are received with error only those frames are requested for retransmission (with a negative acknowledgement).
- If all the frames are received without error, a cumulative positive acknowledgement is sent.
- If there is an error in frame 3, an acknowledgement for the frame 2 is sent and then only frame 3 is retransmitted. Now the window slides to get the next frames to the window.
- If acknowledgement is transmitted with error, all the frames of window are retransmitted. Else ordinary window sliding takes place. (*In implementation part, Acknowledgement error is not considered).
- If all the frames transmitted are errorless the next transmission is carried out for the new window.
- This concept of repeating the transmission for the error frames only is called Selective Repeat transmission flow control protocol.

PROCEDURE:

1. Open NETSIM.
2. Click Programming >> Transmission Flow Control.
3. Select Sample.
4. Select Go Back-N transmission.
5. Enter input data and BER.
6. Click Link to execute the program.
7. Repeat steps 1 to 3.
8. Select Selective Repeat protocol.
9. Enter input data and BER. Click Link to execute the program.

INFERENCE:

It is found that Selective Repeat Protocol is the most optimum out of these two LLC protocols. But the complexity equipment's are high. So, a trade off exists the total number of transmissions taken place and complexity.

RESULT:

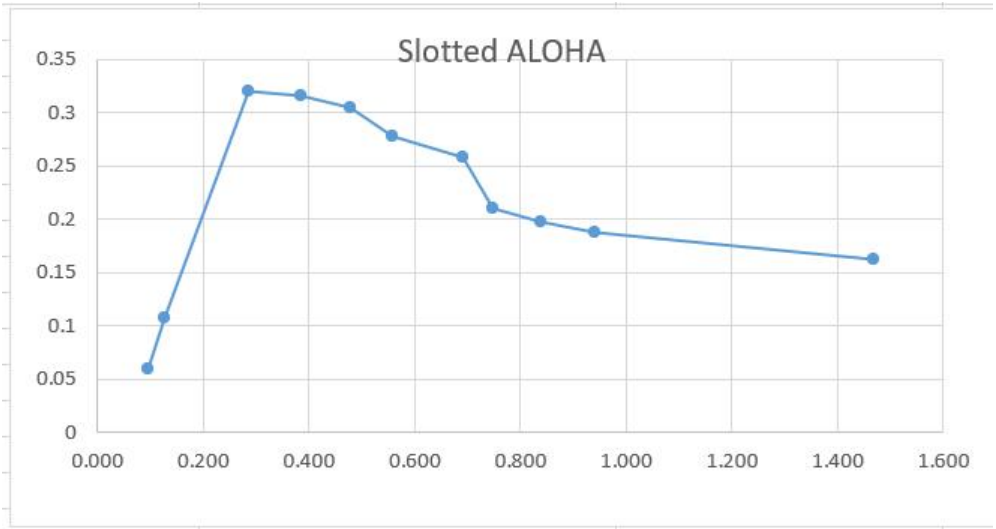
Thus, the LLC protocols such as "Go Back 'N'" and "Selective Repeat" were studied using NETSIM and their performance were analysed.

CONCLUSION:

Both the LLC protocols were studied and it can be concluded that Selective Repeat is more effective but also more complex as compared to Go back n protocol.

SLOTTED ALOHA:

Number of nodes generating traffic	Throughput (in Mbps)	Total number of packets transmitted	Throughput per packet time	Number of packets transmitted per packet time
1	0.589019	793	0.0589019	0.095
2	1.067784	1062	0.1067784	0.127
3	3.1992	2386	0.31992	0.286
4	3.15	3207	0.315	0.385
5	3.044	3984	0.3044	0.478
6	2.783153	4657	0.2783153	0.559
7	2.5747	5775	0.25747	0.693
8	2.10549	6241	0.210549	0.749
9	1.975212	6994	0.1975212	0.839
10	1.872166	7841	0.1872166	0.941
15	1.624065	12236	0.1624065	1.468
21	1.89	16702	0.189	2.004
24	1.702	18623	0.1702	2.235



EXP NO: 10 DATE:	MAC Protocols

SLOTTED ALOHA:

AIM:

To study and analyze slotted ALOHA system and plot the characteristic curve of throughput versus offer traffic for the same.

SOFTWARE REQUIRED:

NETSIM

THEORY:

ALOHA provides a wireless data network. It is a multiple access protocol (this protocol is for allocating a multiple access channel). There are two main versions of ALOHA: pure and slotted. They differ with respect to whether or not time is divided up into discrete slots into which all frames must fit.

In slotted ALOHA, time is divided up into discrete intervals, each interval corresponding to one frame. In Slotted ALOHA, a computer is required to wait for the beginning of the next slot in order to send the next packet. The probability of no other traffic being initiated during the entire vulnerable period is given by e^{-G} which leads to

$$S = G\ e^{-G}$$

S → the mean of the Poisson distribution with which frames are being generated. (frames per frame time)

G → the mean of the Poisson distribution followed by the transmission attempts per frame time.

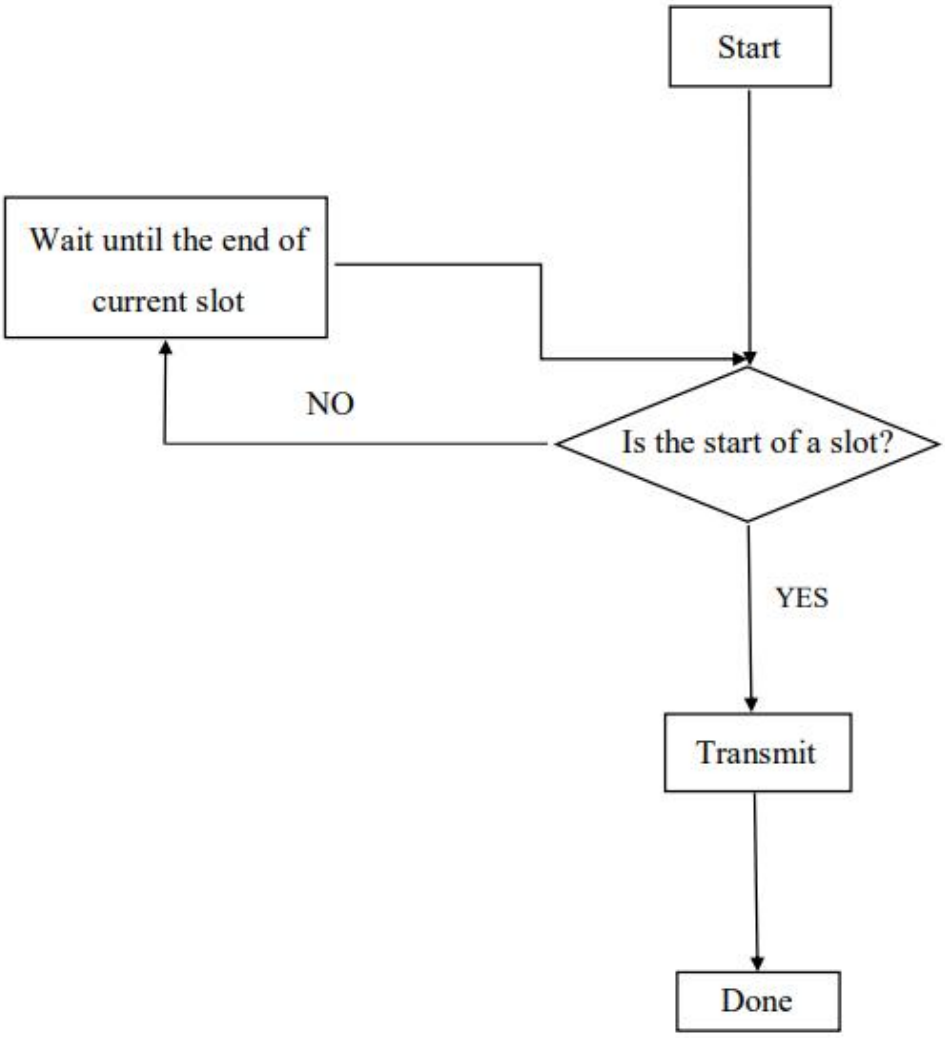
For reasonable throughput S should lie between 0 and 1.

FORMULA:

Attempts per packet time (G) can be calculated as follows;

$$G = \frac{\text{No. of packets transmitted x PT}}{\text{ST x 1000}}$$

FLOWCHART


Start
Wait until the end of current slot
NO
Is the start of a slot?
YES
Transmit
Done

G → Attempts per packet time
PT → Packet time (in seconds)
ST → Simulation time (in seconds)

The throughput (in Mbps) per packet time can be obtained as follows:

$$S = \frac{\text{Throughput(in Mbps)} * 1000 * PT}{PS * 8}$$

S → Throughput per packet time
PT → Packet time (in milliseconds)
PS → Packet size (in bytes)

Packet size (PS) =1472 (Data Size) + 28 (Overheads) = 1500 bytes

$$PT = \frac{packet\ size(bits)}{bandwidth(Mbps)}$$

Packet time = 1.8 millisec (Bandwidth=10Mbps)

PROCEDURE:

- For creating scenarios, we select a new file for slotted ALOHA using legacy network.
- The requires components for slotted ALOHA is available. Initially 2 nodes are dropped and application is created.
- Edit the wireless node properties for generating traffic.

TCP → Disable
Slot length → 1500
Data rate → 10Mbps

Edit the application icon properties as

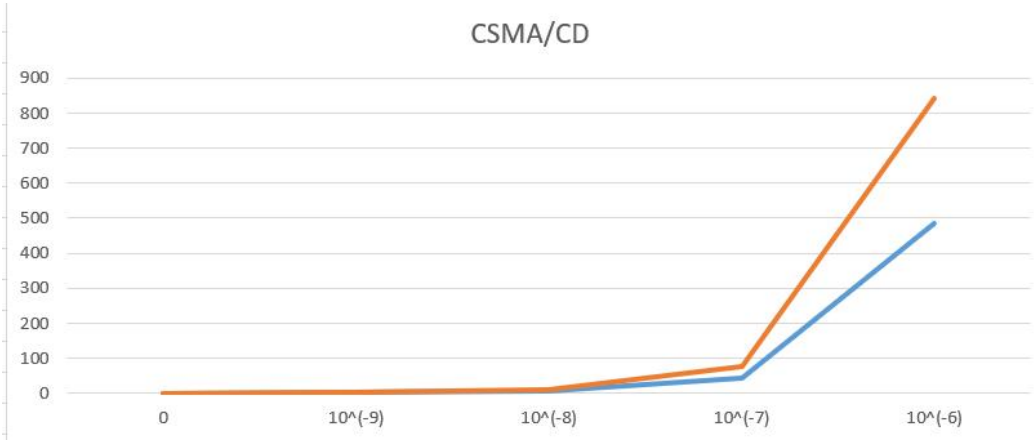
Application method → Unicast
Application type → Custom
Source ID → 1
Destination ID → 2
Packet size → Exponential 1472
Inter arrival time → Exponential 20000 μs

- Set the simulation time as 10s.
- Now repeat the process for different no. of nodes 3, 4, 6, 8 so on.

CSMA/CD:

BER	packets errored (one node)	packets errored (two node)	packets generated (one node)	packets generated (two node)
0	0	0	39999	79998
10^(-9)	2	2	39999	79998
10^(-8)	6	10	39999	79998
10^(-7)	45	75	39999	79998
10^(-6)	485	841	39999	79998

Transmitted packet time vs Throughput per packet time



- Note and tabulate the n values for various nodes.

INFERENCE:

From the graph of No. of packets, transmitted packet time vs throughput per packet time, we can infer that there if a steady increase in graph and later it decreases.

When average no. of packets per slot increases, the throughput per slot also increases. Conversely since there is a high probability of collision as the no. of nodes N increases, the throughput per slot decreases.

CSMA/CD:

AIM:

To understand the impact of bit error rate on packet error and investigate the impact of error on a simple hub based CSMA / CD network.

SOFTWARE REQUIRED:

NETSIM

THEORY:

Bit error rate (BER): The bit error rate or bit error ratio is the number of bit errors divided by the total number of transferred bits during a studied time interval i.e.

$$\text{BER} = \frac{Bit\ error}{Total\ no.\ of\ bits\ transmitted}$$

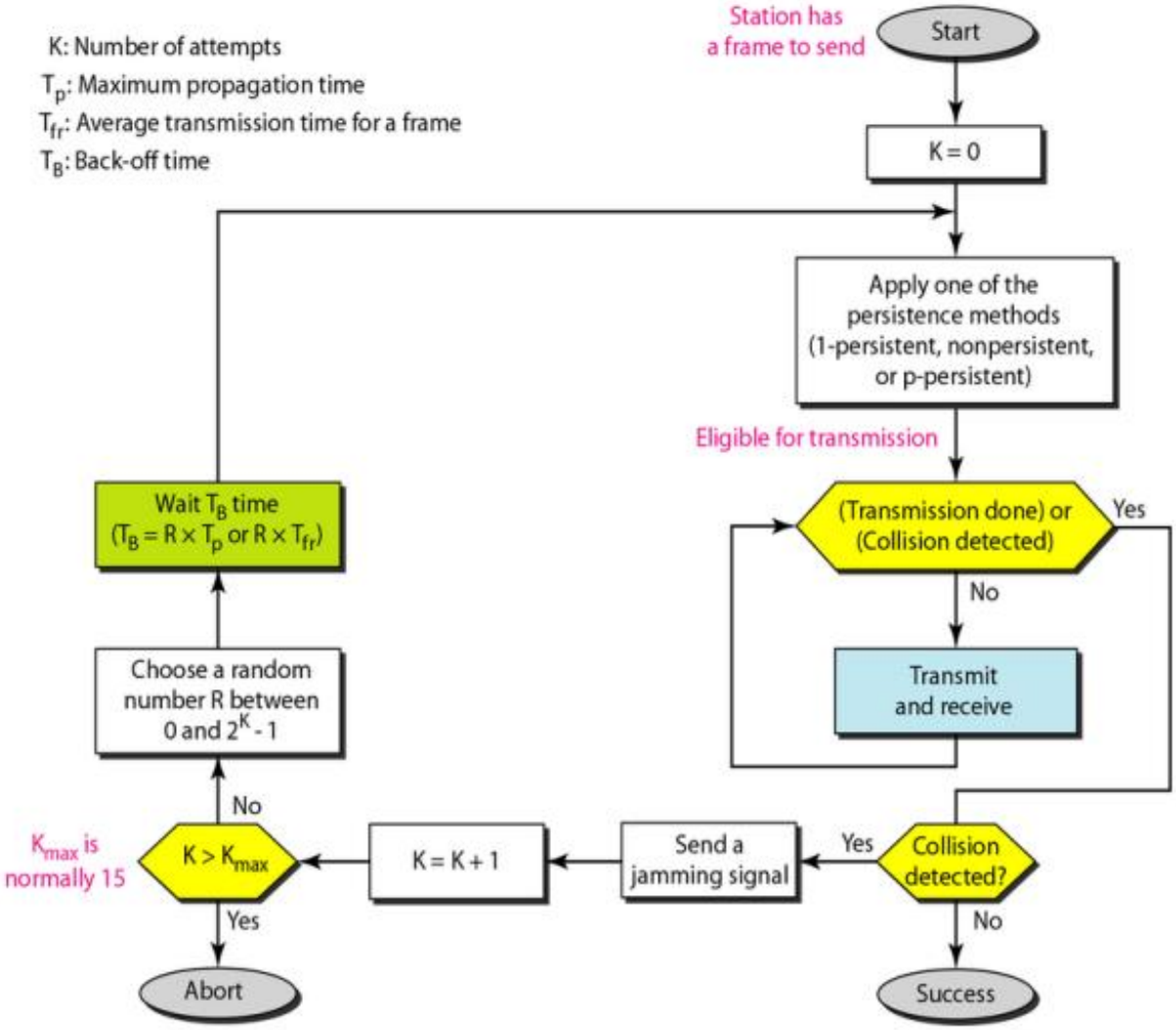
As BER coupler, the effect of all the components in the system given the actual performances of the system that has to be tested. Bit error probability is the expectation value of the BER. The BER can be considered as an approximate estimate of the bit error probability. This estimate is accurate for a long time interval and a high number of bit errors.

A packet is incorrectly received even if one bit is errorless

$$P_p = 1 - [1 - P_e]^N$$

P_p → Packet error probability
P_e → Bit error probability
N → length of N bits in a packet.

FLOWCHART


K: Number of attempts
T_p: Maximum propagation time
T_{fr}: Average transmission time for a frame
T_B: Back-off time
Station has a frame to send
Start
K = 0
Apply one of the persistence methods (1-persistent, nonpersistent, or p-persistent)
Eligible for transmission
(Transmission done) or (Collision detected)
Yes
No
Transmit and receive
Collision detected?
Yes
Send a jamming signal
K = K + 1
$K > K_{max}$
No
Yes
K_{max} is normally 15
Choose a random number R between 0 and $2^K - 1$
Wait T_B time ($T_B = R \times T_p$ or $R \times T_{fr}$)
No
Success
Abort

PROCEDURE:

- For creating a new scenario we select legacy networks and then CSMA/CD.
- From the available components in the network, drag and drop one hub and two nodes, Node 1 and 2.
- Connect them using wires. Their properties are:

Link properties	Sample 1	Sample 2	Sample 3	Sample 4	Sample 5
Data rate(Mbps)	10	10	10	10	10
Bit error rate (BER)	No error	10^{-9}	10^{-8}	10^{-7}	10^{-6}

And set of application

Application method → Unicast
- Application type → Custom
- Source ID → 1
- Destination ID → 2

Packet size → Constant 1472
- Packet interval time → Constant 2500 μs

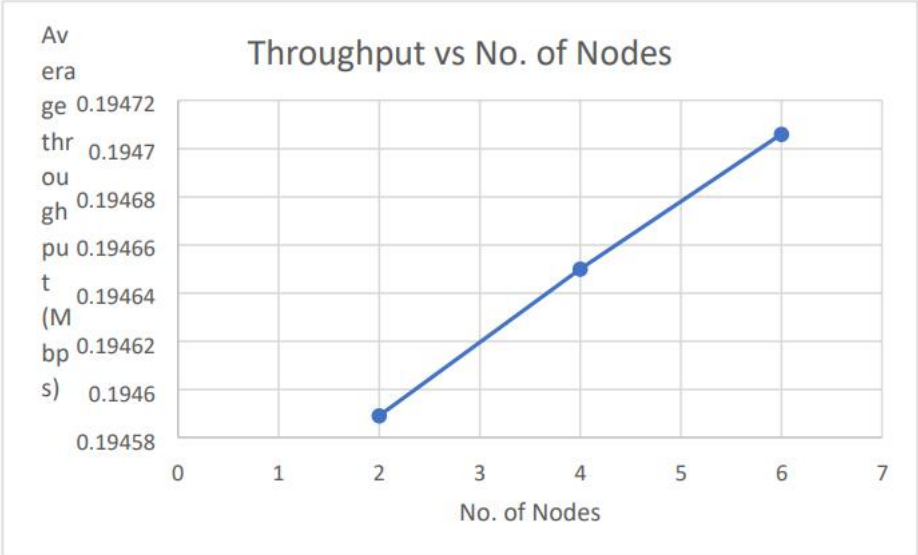
- Run the simulation and note down the values.
- For the next step, click and drop one bit with 4 nodes and connect them using wires.
- Set the same properties as above and note down the table for packet generated.
- Plot the graph for packets errored and bit error rate for both node transmission.

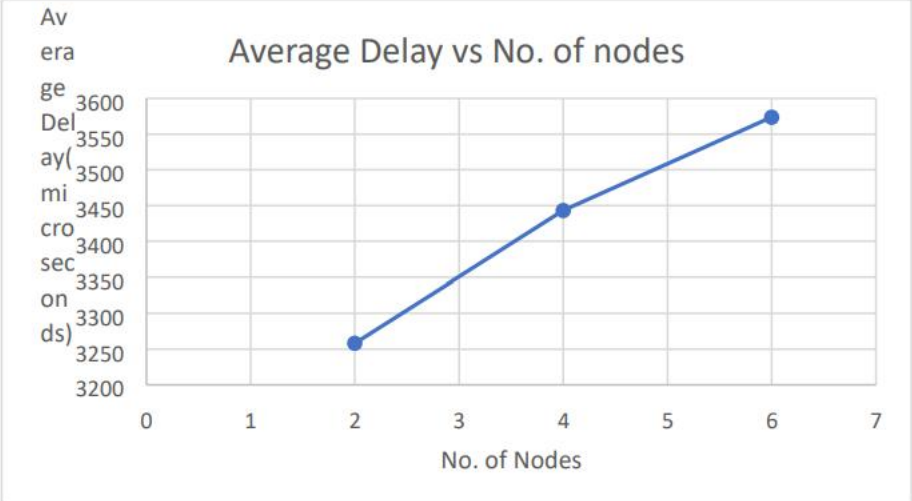
I NFERENCE:

From the graph of CSMA/CD, we get that when BER increases, packets errored also increases. This is because when there is an error in the bits, chance of packets begin correctly decreases.

The performance of CSMA/CD network and minimum bit error rate that can be tolerated without causing excessive packet error is calculated using graph.

CSMA/CA:


Throughput vs No. of Nodes
Average throughput (Mbps)
0.19472
0.1947
0.19468
0.19466
0.19464
0.19462
0.1946
0.19458
0
1
2
3
4
5
6
7
No. of Nodes


Average Delay vs No. of nodes
Average Delay(microseconds)
3600
3550
3500
3450
3400
3350
3300
3250
3200
0
1
2
3
4
5
6
7
No. of Nodes

CSMA/CA:

AIM:

To analyse the performance of a MANET running CSMA/CA in MAC as the node density is increased and to plot

Throughput versus No. of nodes

Delay average versus No. of nodes

SOFTWARE REQUIRED:

NETSIM

THEORY:

Mobile Ad-Hoc Network (MANET) is a self-configuring network of mobile nodes connected by wireless links to form an arbitrary topology without the use of existing infrastructure.

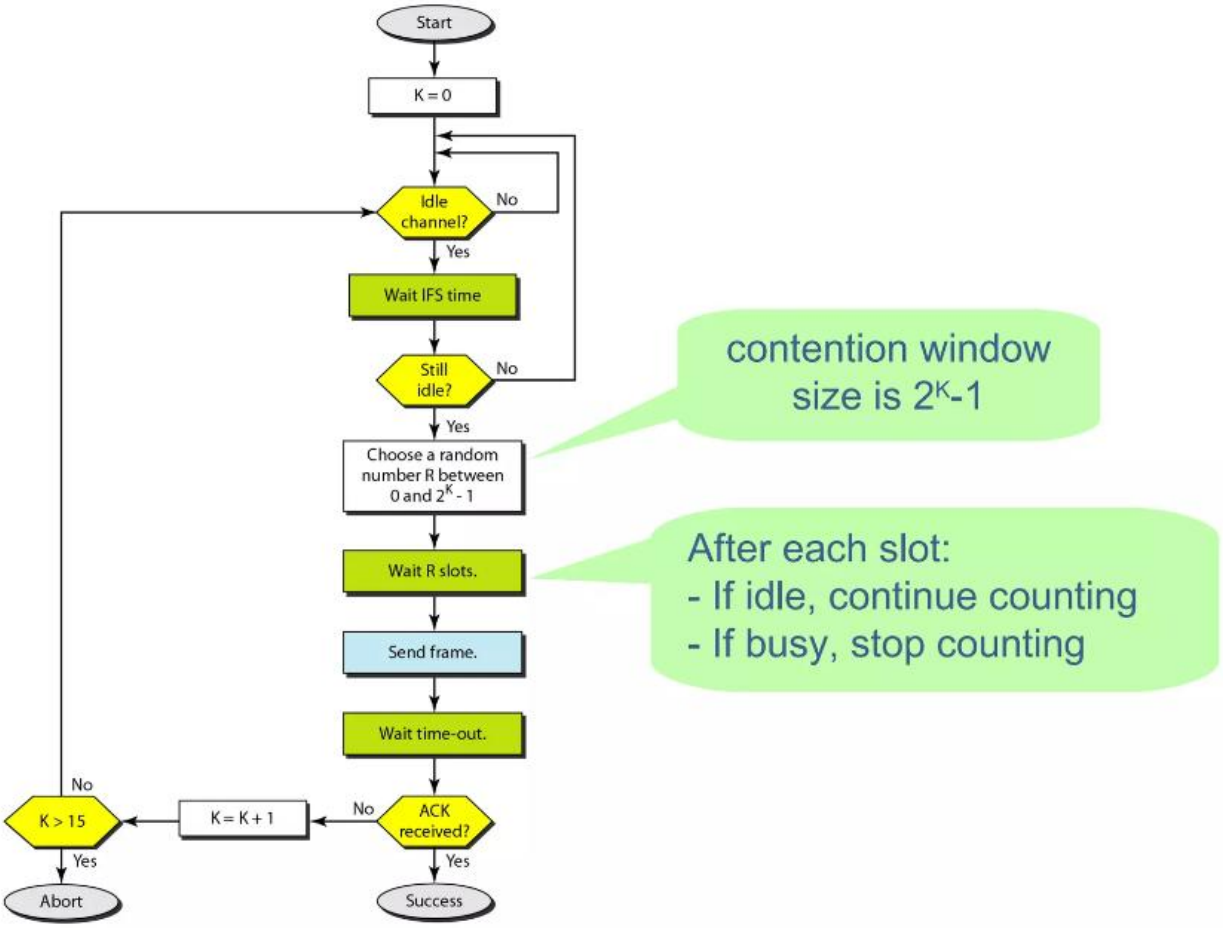
The nodes are free to move randomly. Thus the network's wireless topology may be unpredictable and may change rapidly.

The node density also has an impact on the routing performance. With very sparsely populated network the number of possible connection between any two nodes is very less and hence the performance is poor. It is expected that if the node density is increased the throughput of the network shall increase, but beyond a certain level if density is increased the performance degrades.

PROCEDURE:

- For creating new scenario, we select new legacy window network and then CSMA/CA.
- Then create the network using a wireless nodes. So from now, wireless nodes are increased.
- Correspondingly table below are 2 levels of simulation. Each level should be executed separately with the application properties mentioned.

FLOWCHART


Start
K = 0
Idle channel?
No
Yes
Wait IFS time
Still idle?
No
Yes
Choose a random number R between 0 and 2^K - 1
Wait R slots.
Send frame.
Wait time-out.
ACK received?
No
Yes
K = K + 1
K > 15
No
Yes
Abort
Success
contention window size is 2^K-1
After each slot:
- If idle, continue counting
- If busy, stop counting

Level	Wireless Node	X Ordinate	Y Ordinate
1	1 2	50 100	100 150
2	1 2 3 4	50 100 75 125	100 150 75 125
3	1 2 3 4 5 6	50 100 80 140 110 160	100 150 70 130 40 90

- Disable TCP in all wireless nodes.
- Right click on grid area to modify wireless node properties.

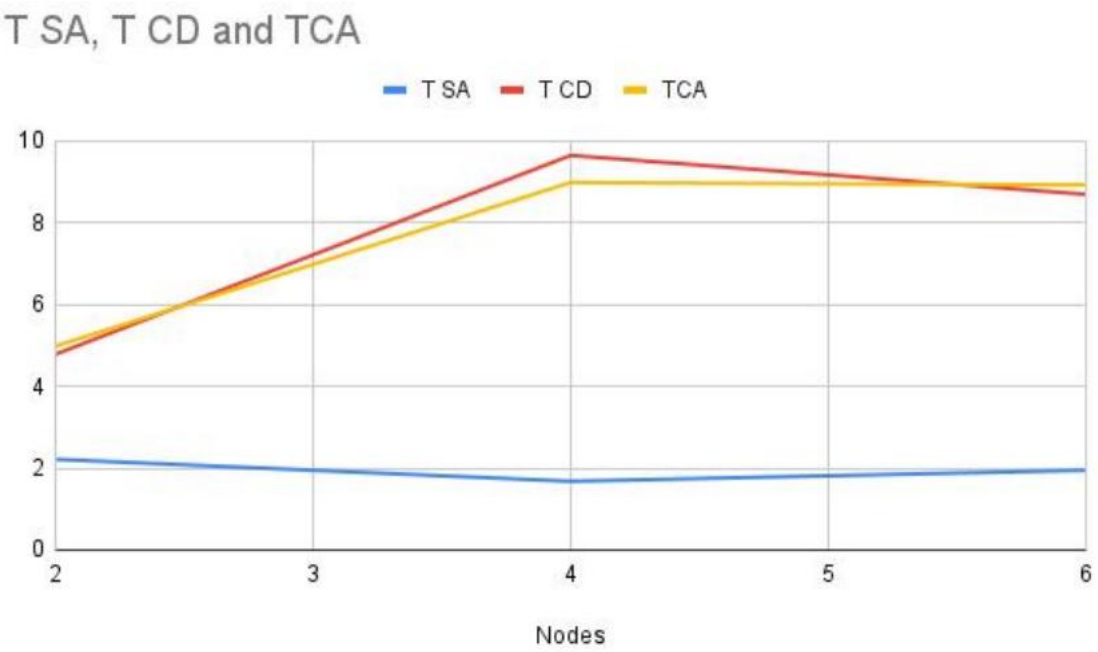
Path loss model → log distance
Path loss component → 2

- Then we have to model the traffic in network using application after selecting the applications dialog box. Set the below mentioned values:

Application type → Custom
Source ID → As per levels source ID is varied as per nodes
Destination ID → Same rule applies for destination ID
Packet size
Distribution → Constant
Value (bytes) → 1472
Inter arrival time
Distribution → Constant
Value (μs) → 60000

- Save the schematic and note down the values. Repeat the same steps for level 2 and level 3with increasing application.

COMPARISON GRAPH:



INFERENCE:

From the graph of average delay versus no. of nodes, when no. of nodes increases, average delay also increases. It takes longer for a packet to reach its destination.

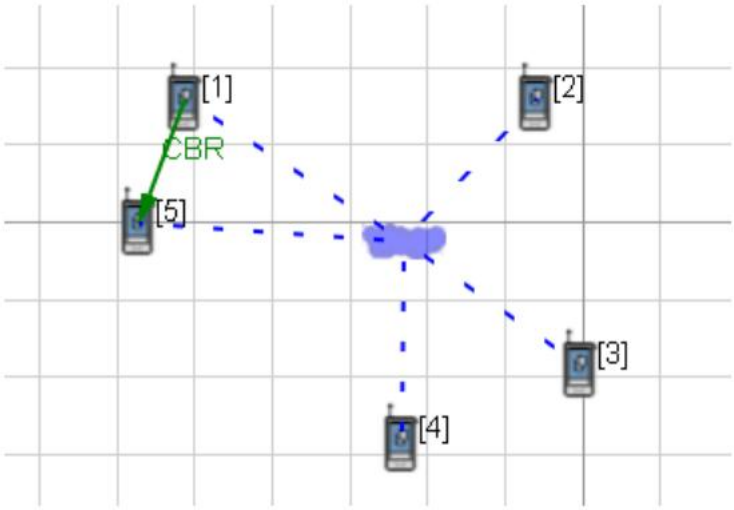
From the graph of average throughput versus no. of nodes, the average throughput increases for well designed network.

RESULT:

MAC protocol namely slotted ALOHA, CSMA/CA, CSMA/CD are implemented and verified using NETSIM.

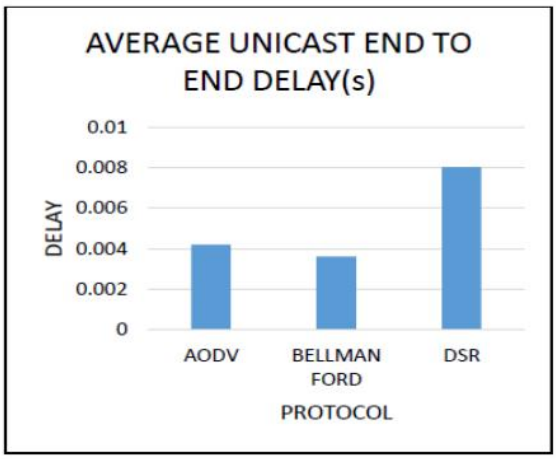
OUTPUTS:

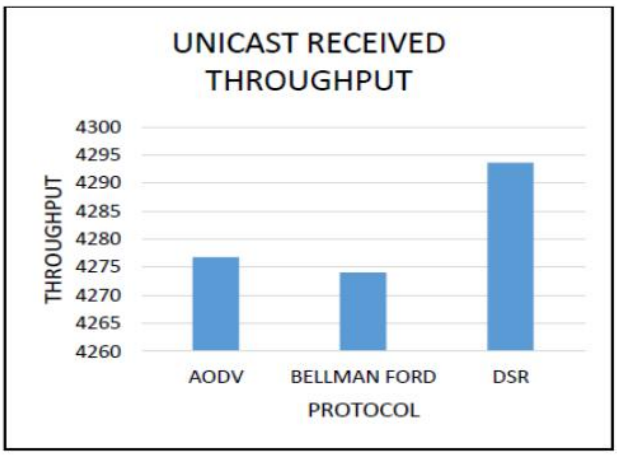
LAYOUT: WITHOUT MOBILITY



TABULATION:

PROTOCOL	AVERAGE UNICAST END TO END DELAY(s)	UNICAST RECEIVED THROUGHPUT
AODV	0.0041981	4276.77
BELLMAN FORD	0.00359559	4274.05
DSR	0.00801132	4293.64





EXP NO: 11 DATE:	Study of Wireless Protocols using QUALNET

AIM:

To study and implement wireless routing protocols using qualnet

- AODV-Ad-hoc on demand distance vector
- DSR-Dynamic Static Routing
- Bellman ford

SOFTWARE REQUIRED:

QUALNET simulator

THEORY:

AODV:AD-HOC ON-DEMAND DISTANCE VECTOR

AODV is a routing protocol for ad-hoc mobile networks with large numbers of mobile nodes.The protocols algorithm creates routes between nodes only when the routes are requested by source nodes giving the network the flexibility to allow nodes to enter and leave the network .Routes remain active only as long as data packets are travelling along the paths from the source to the destination .When the source stops sending packets the path will time out and close.

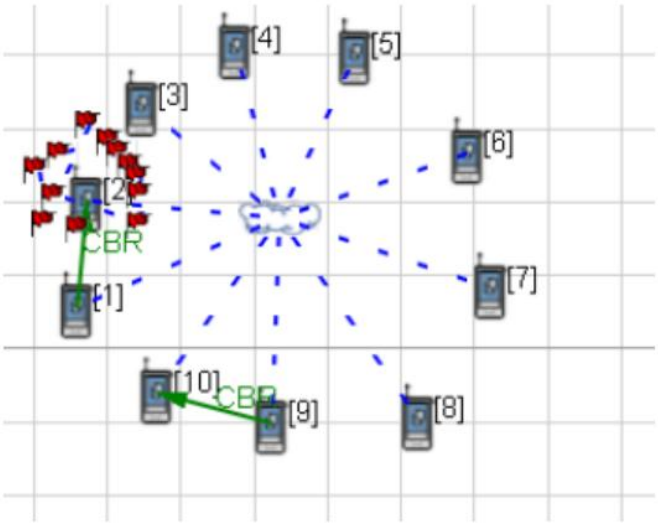
DSR:

Dynamic source routing is a routing protocol for wireless mesh networks. In DSR each source determines the route to be used in transmitting its packets to selected destination. There are two main components called route discovery and route maintenance. Route discovery determines the optimum path for transmission between a given source and destination. Route maintenance ensures that the transmission path remains optimum and loop free as network condition changes, even if this requires changing the route during a transmission.

BELLMAN FORD:

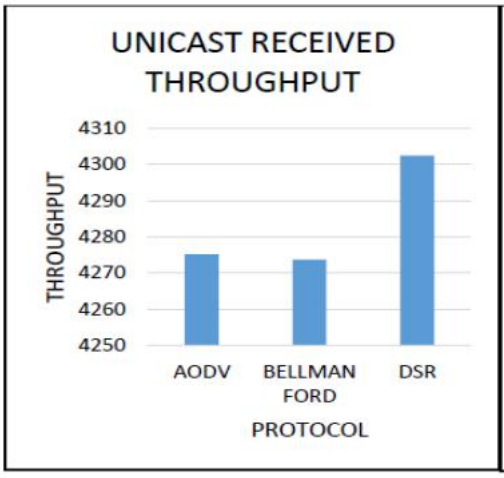
It is used to find the shortest path from one node to all other nodes in a weighed graph. The first step is to initialize the vertices. The algorithm is initially set from the starting vertex to all vertices till infinity. After the initializing step the algorithm starts circulating shortest distance from starting vertex to all other vertices. It is an example of dynamic programming. It follows bottom-up approach

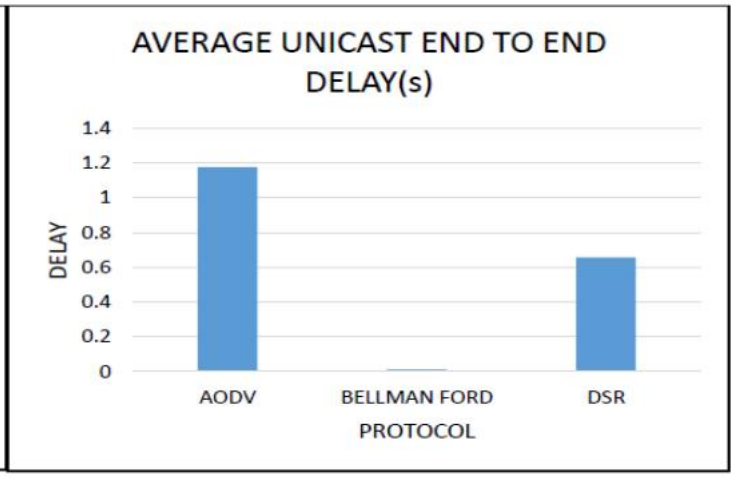
LAYOUT: WITH MOBILITY



TABULATION:

PROTOCOL	AVERAGE UNICAST END TO END DELAY(s)	UNICAST RECEIVED THROUGHPUT
AODV	1.176	4275.19
BELLMAN FORD	0.012	4273.67
DSR	0.656	4302.4





PROCEDURE:

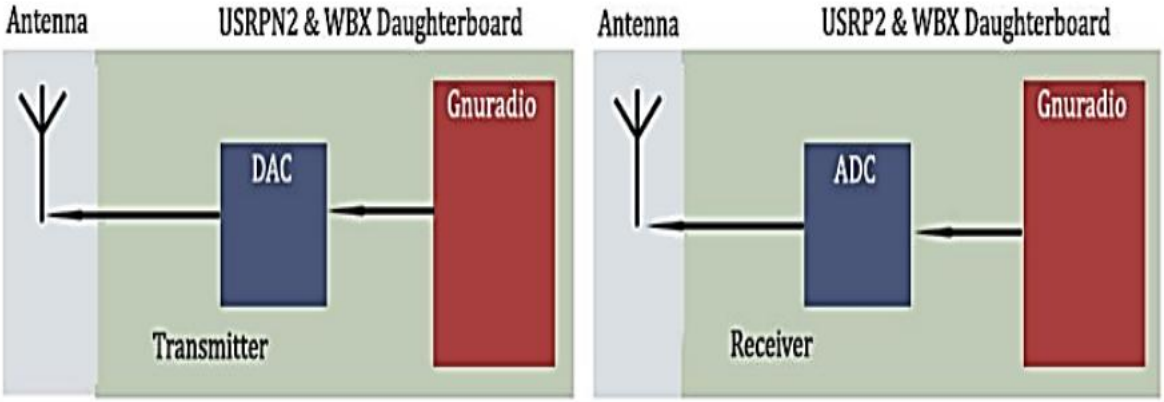
- Open qualnet and set the suitable terrain.
- Place the required nodes and define properties for the nodes and using a subnet.
- In the properties , set the routing protocol to DSR.
- Establish CBR between two nodes.
- Save and run the simulation and there in the analyser window observe the required parameters such as throughput, average end to end delay and average jitter.
- Repeat the steps by varying the node density, (i.e) increase or decrease the number of nodes and tabulate the values.
- The same steps are to be followed for the other routing protocols such as AODV and Bellman Ford .
- The observed metrics are tabulated and plotted for different routing protocols.

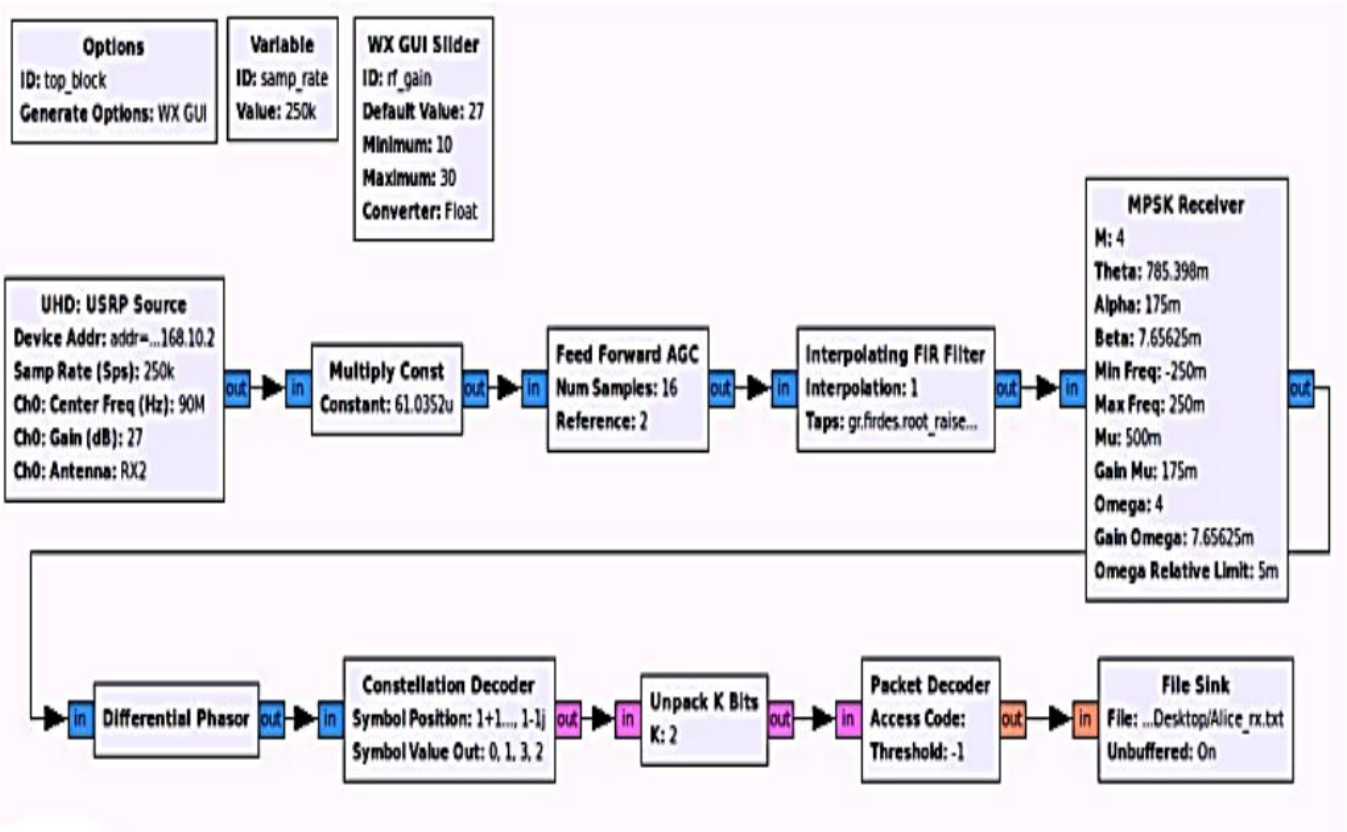
INFERENCE

From the graph, the throughput and delay increase with increase in number of nodes. It is the highest in DSR compared to AODV and Bellman Ford. With mobility as the speed increases, DSR has maximum ed to end delay with AODV has stable delay despite the mobility.

RESULT:

The performance evaluation of routing protocols Bellman Ford, AODV and DSR are studied by varying node density using Qualnet





Expanded system in GNU Radio Companion

EXP NO: 12 DATE:	Study of SDR based transceiver using USRP and GNU radio

AIM:

To study about software defined ratio: SDR based transceiver of digital communication system using universal software radio peripheral: USRP family of products.

To simulate OFDM and to study its characteristics.

SOFTWARE REQUIRED:

GNU Radio

HARDWARE REQUIRED:

USRP kit

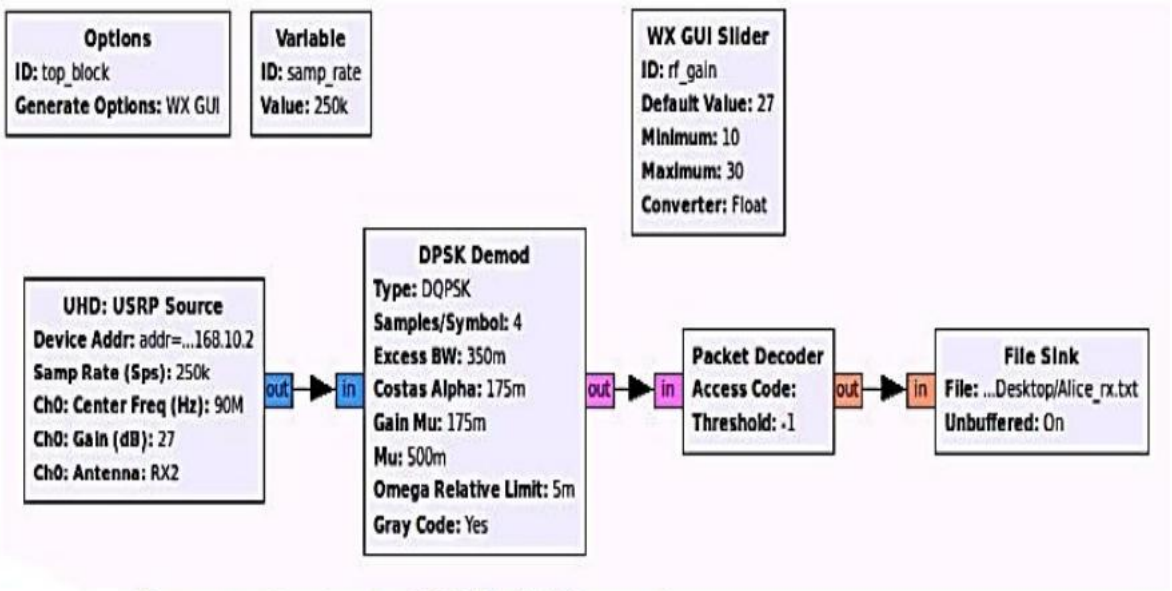
THEORY:

GNU Radio:

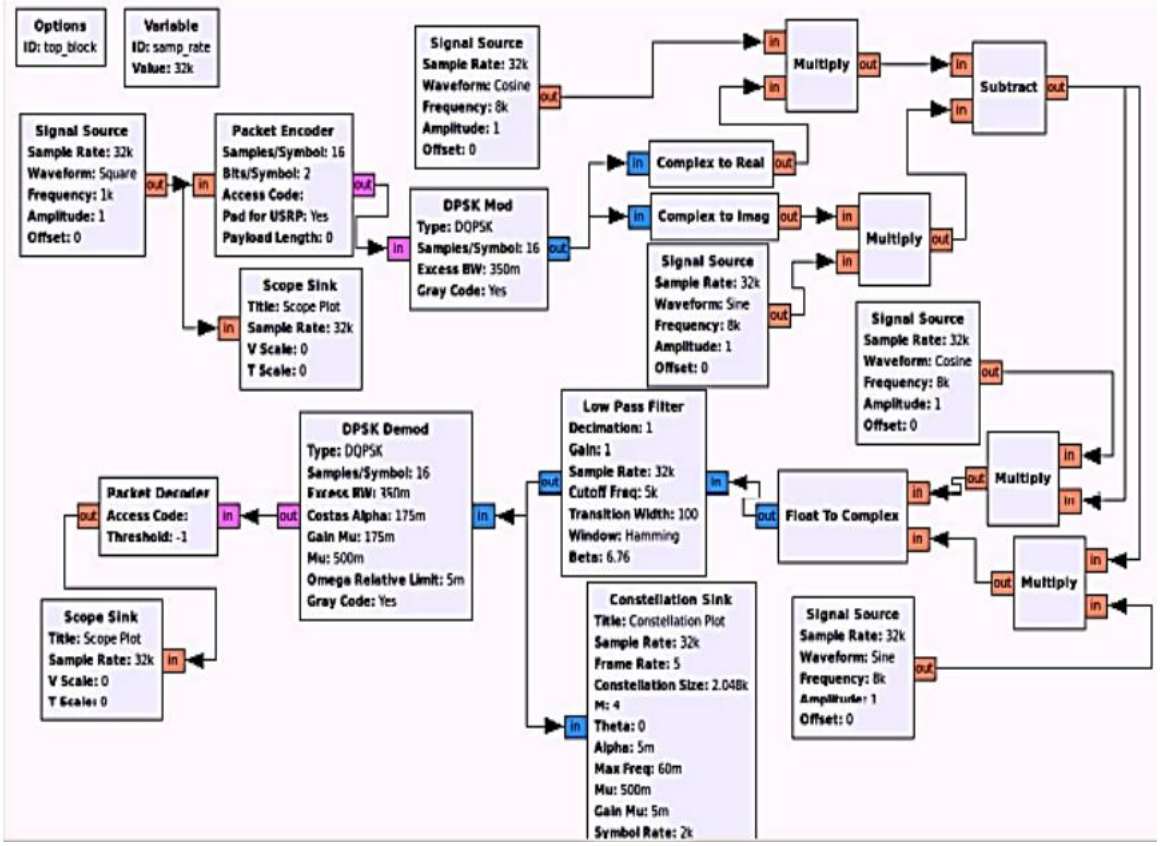
GNU Radio is a framework that enables users to design, simulate, and deploy highly capable real-world radio systems. It is a highly modular, "flowgraph"-oriented framework that comes with a comprehensive library of processing blocks that can be readily combined to make complex signal processing applications. GNU Radio has been used for a huge array of real-world radio applications, including audio processing, mobile communications, tracking satellites, radar systems, GSM networks, Digital Radio Mondiale, and much more - all in computer software. It is, by itself, not a solution to talk to any specific hardware. Nor does it provide outof the-box applications for specific radio communications standards (e.g., 802.11, ZigBee,LTE, etc.,), but it can be (and has been) used to develop implementations of basically any band-limited communication standard. GNU Radio is a framework dedicated to writing signal processing applications for commodity computers. GNU Radio wraps functionality in easy-to-use reusable blocks, offers excellent scalability, provides an extensive library of standard algorithms, and is heavily optimized for a large variety of common platforms.

USRP kit:

USRP Software Defined Radio device provides a software defined RF architecture to design, prototype and deploy wireless systems with custom signal processing. USRP Software Defined Radio Devices also include options with an onboard FPGA—an SDR game changer that provides wireless communications designers an affordable SDR with unprecedented performance for developing



Compressed system in GNU Radio Companion



next generation 5G wireless communication systems. USRP Software Defined Radio Devices have an optional onboard GPS that you can use to synchronize multiple software defined radios and enable advanced application possibilities such as distributed multi-channel synchronized systems.

The USRP Software Defined Radio Device is a reconfigurable RF device that includes a combination of host based processors, FPGAs, and RF front ends. The USRP Software Defined Radio Device include options that range from lower cost options with fixed FPGA personalities to high end radios with a large, open FPGAs and wide instantaneous bandwidth. These devices can be used for applications such as multiple input, multiple output (MIMO) and LTE/WiFi testbeds, SIGINT, and radar systems.

OFDM:

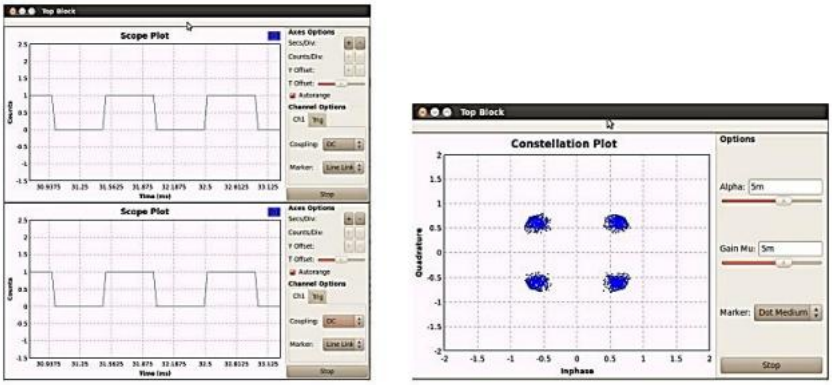
- Orthogonal Frequency Division Multiplexing (OFDM) is a digital multi-carrier modulation scheme that extends the concept of single subcarrier modulation by using multiple subcarriers within the same single channel. Rather than transmit a high-rate stream of data with a single subcarrier, OFDM makes use of a large number of closely spaced orthogonal subcarriers that are transmitted in parallel.

- Each subcarrier is modulated with a conventional digital modulation scheme (such as QPSK, 16QAM, etc.) at low symbol rate. However, the combination of many subcarriers enables data rates similar to conventional single-carrier modulation schemes within equivalent bandwidths.

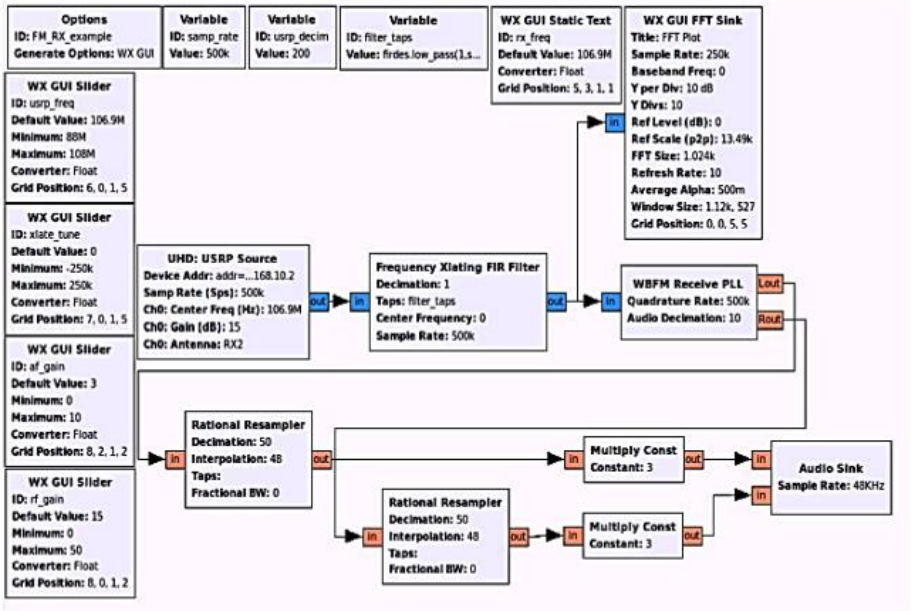
- The OFDM signal can be described as a set of closely spaced FDM subcarriers. In the frequency domain, each transmitted subcarrier results in a sinc function spectrum with side lobes that produce overlapping spectra between subcarriers. This results in subcarrier interference except at orthogonally spaced frequencies.

- At orthogonal frequencies, the individual peaks of subcarriers all line up with the nulls of the other subcarriers. This overlap of spectral energy does not interfere with the system's ability to recover the original signal. The receiver multiplies (i.e., correlates) the incoming signal by the known set of sinusoids to recover the original set of bits sent.

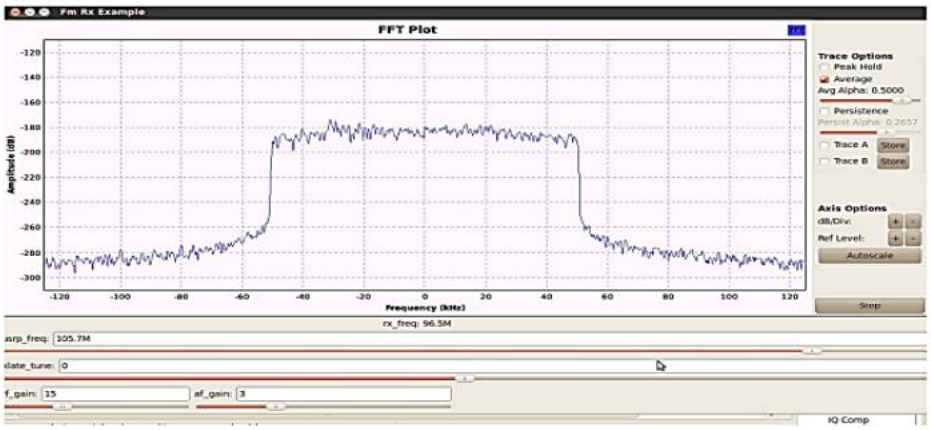
- The use of orthogonal subcarriers allows more subcarriers per bandwidth resulting in an increase in spectral efficiency. In a perfect OFDM signal, Orthogonality prevents interference between overlapping carriers.



Left: Transmitted signal and Received signal. Right: The constellation plot of receiver



FM Radio Rx System

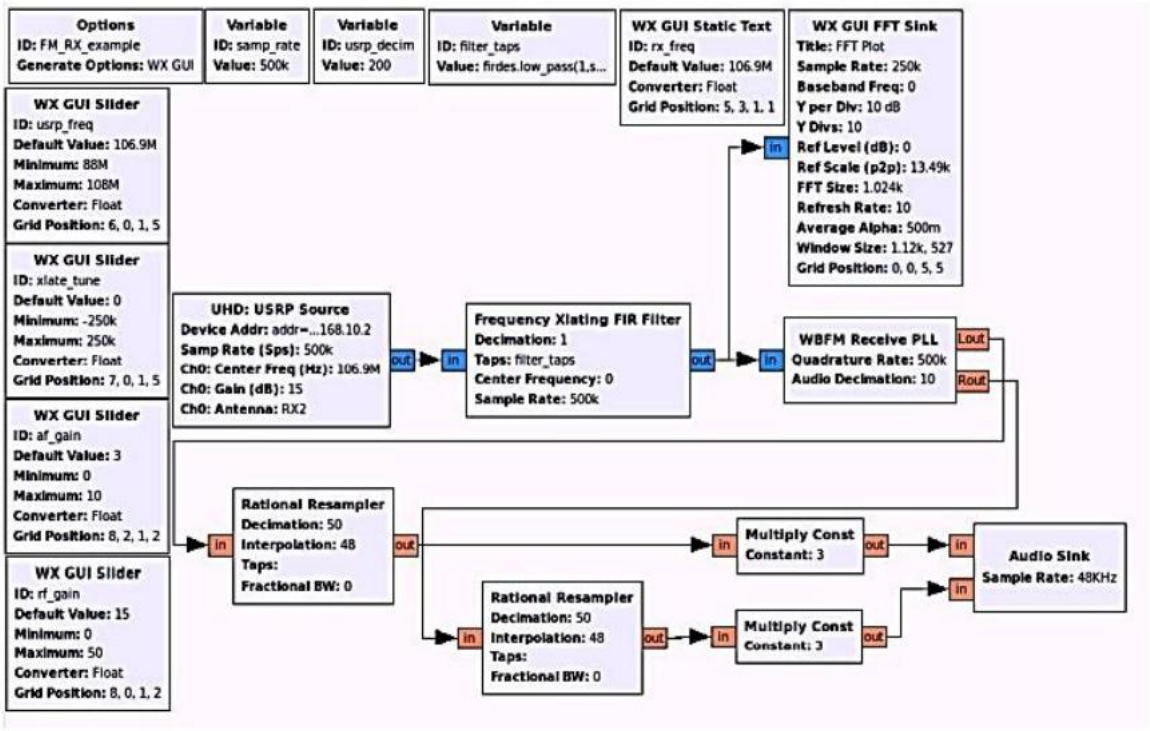


Received Spectrum of FM Receiver

PROCEDURE:

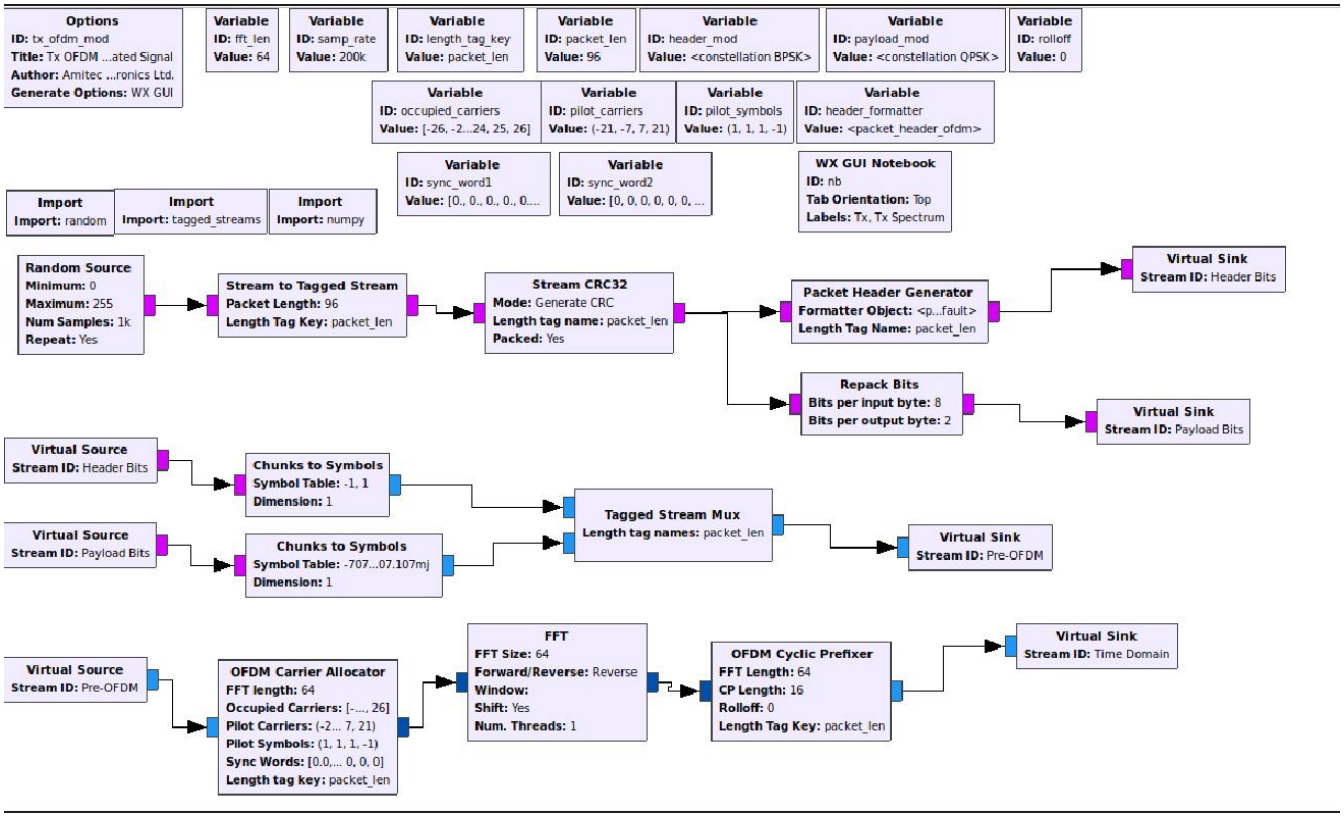
GNU Radio Companion (GRC) is a graphical user interface that allows you to build GNU radio flow graphs.

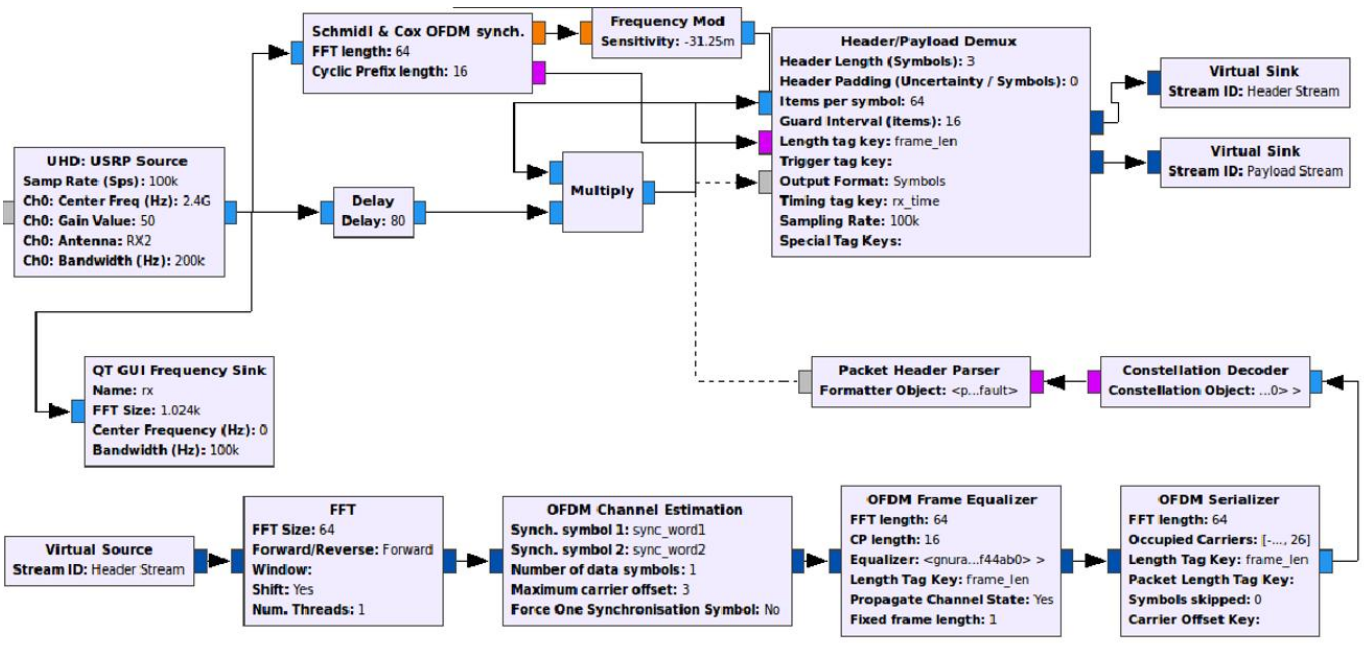
1. Open a terminal window using: Applications > Accessories > Terminal. At the prompt type: grc
2. An untitled GRC window should open.
3. Double click on the Options block. This block sets some general parameters for the flow graph. Leave the ID as top_block. Type in a project title (such as Tutorial 1) and author. Set Generate Options to WX GUI, Run to Autostart, and Realtime Scheduling to Off. Then close the properties window. The other block that is present is the Variable block. It is used to set the sample rate.
4. On the right side of the window is a list of the blocks that are available. By expanding any of the categories (click on the triangle to the left) you can see the blocks available.
5. Open the Sources category and double click on the Signal Source. Note that a signal source block will now appear in the main window. Double click on the block and the properties window will open. In order to view this wave, we need one of the graphical sinks. Expand the Graphical Sink category and double click on the Scope Sink. It should appear in the main window. Double click on the block and change the Type to Float. Leave the other Parameters at their default values and close the properties window.
6. In order to observe the operation of this simple system we must generate the flow graph and then execute it. Click first on the “Generate the flow graph” icon. Click next on the “Execute the flow graph” icon to view the graph.
7. Open a file browser in Ubuntu (Places > Home Folder). Go to the directory that contains the GRC file that you have been working on. If you are unsure as to where this is, the path to this file is shown in the bottom portion of the GRC window. In addition to saving a “.grc” file with your flow graph, note that there is also a file titled “top_block.py”. Double click on this block. You will be given the option to Run or Display this file. Select Display. This is the Python file that is generated by GRC. It is the file that is being run when you execute the flow graph. You can modify this file and run it from the terminal window. This allows you to use features that are not included in GRC. Keep in mind that every time you run your flow graph in GRC, it will overwrite the Python script that is generated. So, if you make changes directly in the Python Script that you want to keep, save it under another name.

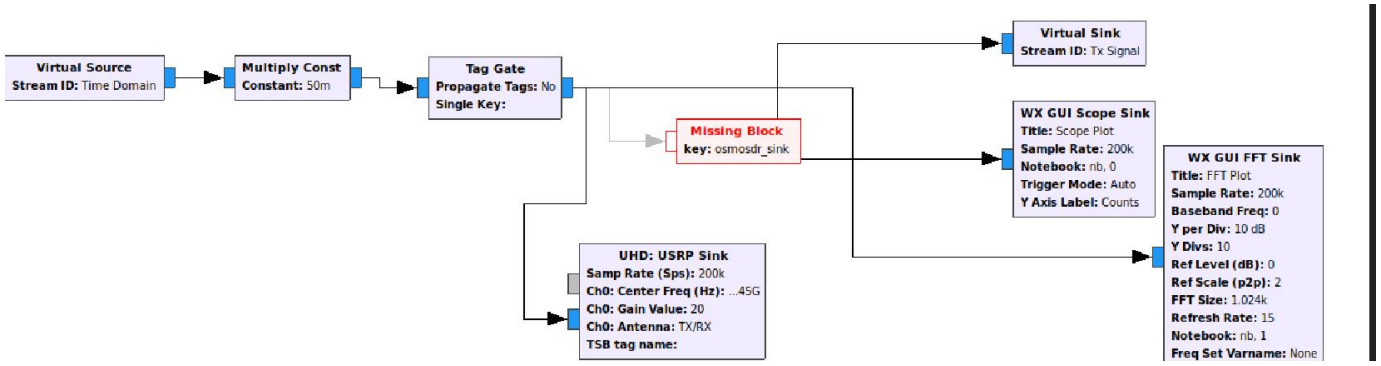


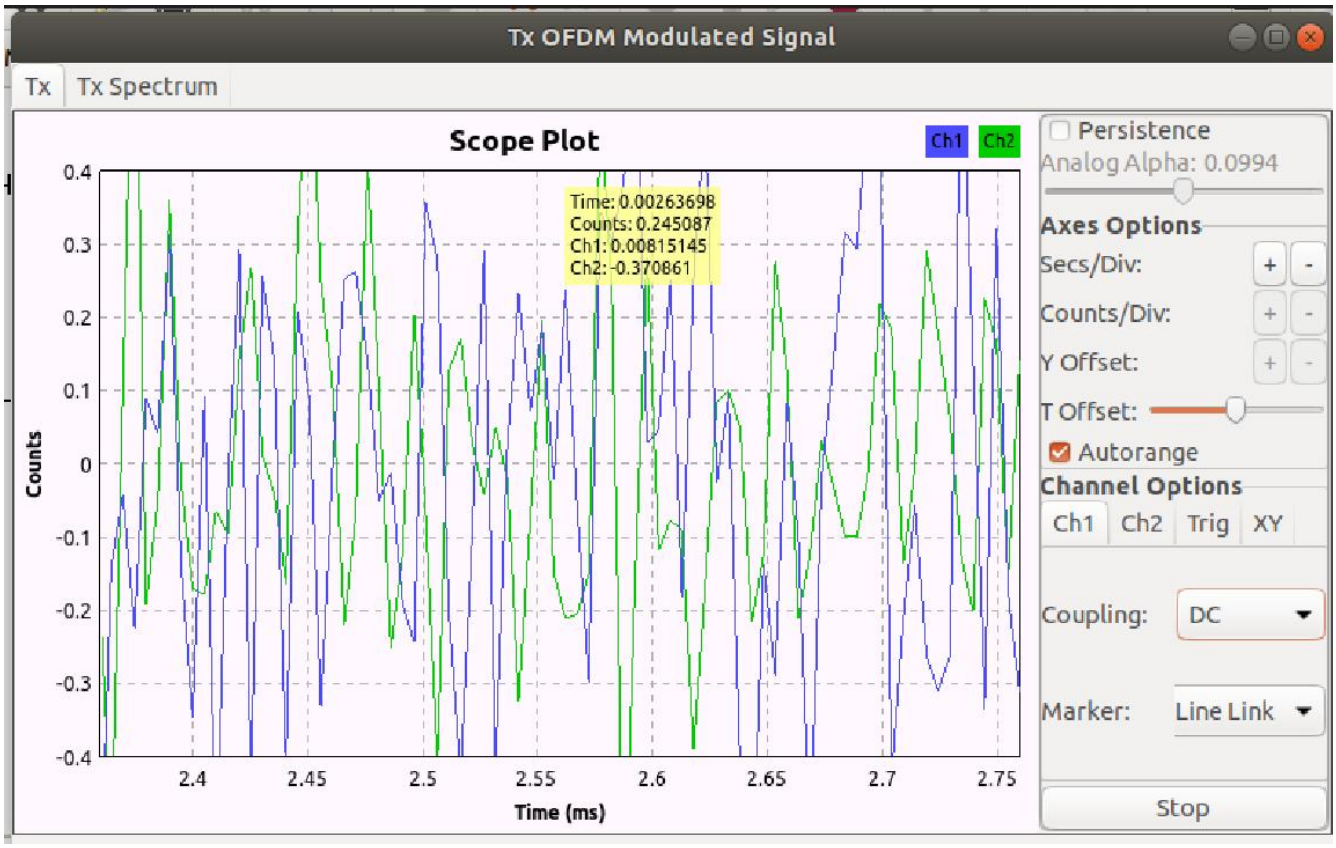
- Complete flowgraph with appropriate connections, data types, and parameters

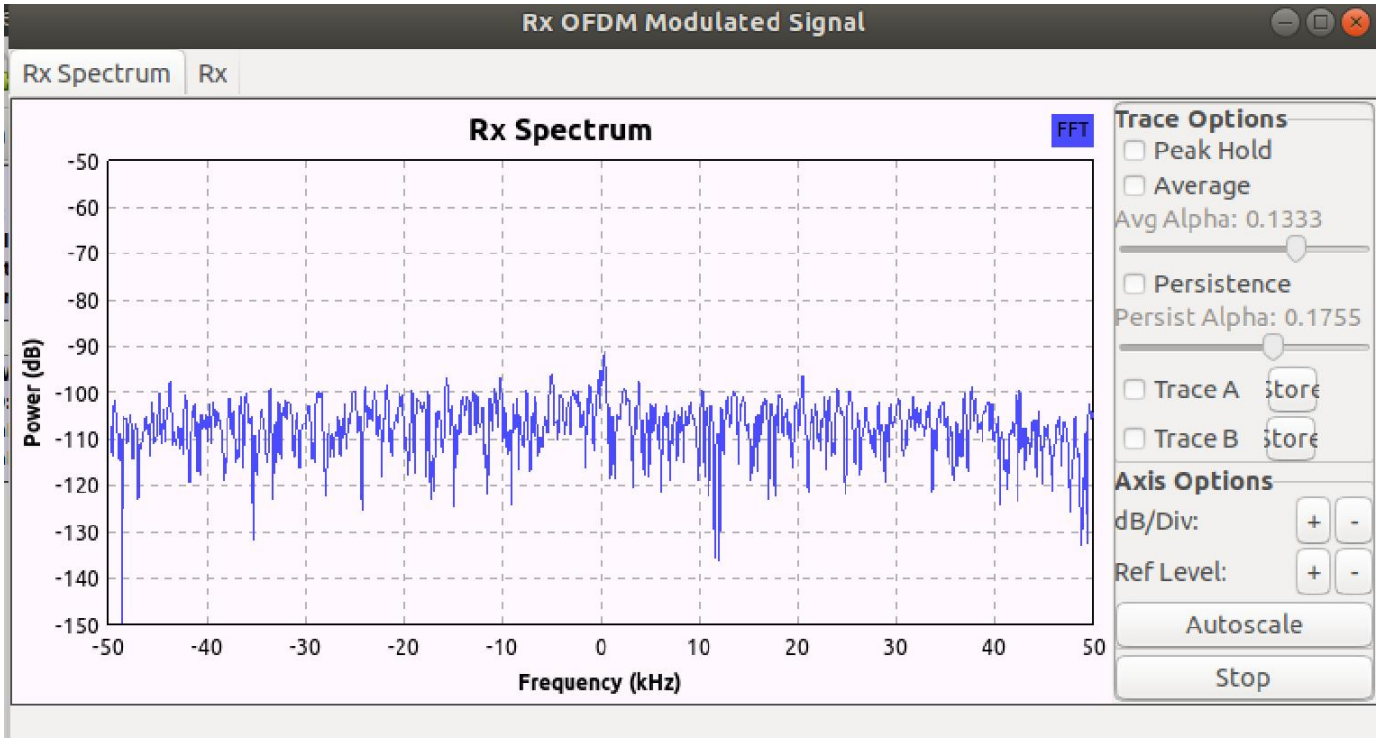
OFDM SYSTEM:


Options
ID: tx_ofdm_mod
Title: Tx OFDM ...ated Signal
Author: Amitec ...ronics Ltd.
Generate Options: WX GUI
Variable
ID: fft_len
Value: 64
Variable
ID: samp_rate
Value: 200k
Variable
ID: length_tag_key
Value: packet_len
Variable
ID: packet_len
Value: 96
Variable
ID: header_mod
Value: <constellation BPSK>
Variable
ID: payload_mod
Value: <constellation QPSK>
Variable
ID: rolloff
Value: 0
Variable
ID: occupied_carriers
Value: [-26, -2...24, 25, 26]
Variable
ID: pilot_carriers
Value: (-21, -7, 7, 21)
Variable
ID: pilot_symbols
Value: (1, 1, 1, -1)
Variable
ID: header_formatter
Value: <packet_header_ofdm>
Variable
ID: sync_word1
Value: [0., 0., 0., 0., 0....
Variable
ID: sync_word2
Value: [0, 0, 0, 0, 0, 0, ...
WX GUI Notebook
ID: nb
Tab Orientation: Top
Labels: Tx, Tx Spectrum
Import
Import: random
Import
Import: tagged_streams
Import
Import: numpy
Random Source
Minimum: 0
Maximum: 255
Num Samples: 1k
Repeat: Yes
Stream to Tagged Stream
Packet Length: 96
Length Tag Key: packet_len
Stream CRC32
Mode: Generate CRC
Length tag name: packet_len
Packed: Yes
Packet Header Generator
Formatter Object: <p...fault>
Length Tag Name: packet_len
Virtual Sink
Stream ID: Header Bits
Repack Bits
Bits per input byte: 8
Bits per output byte: 2
Virtual Sink
Stream ID: Payload Bits
Virtual Source
Stream ID: Header Bits
Chunks to Symbols
Symbol Table: -1, 1
Dimension: 1
Virtual Source
Stream ID: Payload Bits
Chunks to Symbols
Symbol Table: -707...07.107mj
Dimension: 1
Tagged Stream Mux
Length tag names: packet_len
Virtual Sink
Stream ID: Pre-OFDM
Virtual Source
Stream ID: Pre-OFDM
OFDM Carrier Allocator
FFT length: 64
Occupied Carriers: [-..., 26]
Pilot Carriers: (-2... 7, 21)
Pilot Symbols: (1, 1, 1, -1)
Sync Words: [0.0,... 0, 0, 0]
Length tag key: packet_len
FFT
FFT Size: 64
Forward/Reverse: Reverse
Window:
Shift: Yes
Num. Threads: 1
OFDM Cyclic Prefixer
FFT Length: 64
CP Length: 16
Rolloff: 0
Length Tag Key: packet_len
Virtual Sink
Stream ID: Time Domain


Schmidl & Cox OFDM synch.
FFT length: 64
Cyclic Prefix length: 16
Frequency Mod
Sensitivity: -31.25m
Header/Payload Demux
Header Length (Symbols): 3
Header Padding (Uncertainty / Symbols): 0
Items per symbol: 64
Guard Interval (items): 16
Length tag key: frame_len
Trigger tag key:
Output Format: Symbols
Timing tag key: rx_time
Sampling Rate: 100k
Special Tag Keys:
Virtual Sink
Stream ID: Header Stream
Virtual Sink
Stream ID: Payload Stream
UHD: USRP Source
Samp Rate (Sps): 100k
Ch0: Center Freq (Hz): 2.4G
Ch0: Gain Value: 50
Ch0: Antenna: RX2
Ch0: Bandwidth (Hz): 200k
Delay
Delay: 80
Multiply
QT GUI Frequency Sink
Name: rx
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 100k
Packet Header Parser
Formatter Object: <p...fault>
Constellation Decoder
Constellation Object: ...0> >
Virtual Source
Stream ID: Header Stream
FFT
FFT Size: 64
Forward/Reverse: Forward
Window:
Shift: Yes
Num. Threads: 1
OFDM Channel Estimation
Synch. symbol 1: sync_word1
Synch. symbol 2: sync_word2
Number of data symbols: 1
Maximum carrier offset: 3
Force One Synchronisation Symbol: No
OFDM Frame Equalizer
FFT length: 64
CP length: 16
Equalizer: <gnura...f44ab0> >
Length Tag Key: frame_len
Propagate Channel State: Yes
Fixed frame length: 1
OFDM Serializer
FFT length: 64
Occupied Carriers: [-..., 26]
Length Tag Key: frame_len
Packet Length Tag Key:
Symbols skipped: 0
Carrier Offset Key:


Virtual Source
Stream ID: Time Domain
Multiply Const
Constant: 50m
Tag Gate
Propagate Tags: No
Single Key:
Virtual Sink
Stream ID: Tx Signal
Missing Block
key: osmosdr_sink
WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 200k
Notebook: nb, 0
Trigger Mode: Auto
Y Axis Label: Counts
WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 200k
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 2
FFT Size: 1.024k
Refresh Rate: 15
Notebook: nb, 1
Freq Set Varname: None
UHD: USRP Sink
Samp Rate (Sps): 200k
Ch0: Center Freq (Hz): ...45G
Ch0: Gain Value: 20
Ch0: Antenna: TX/RX
TSB tag name:


Tx OFDM Modulated Signal
Tx
Tx Spectrum
Scope Plot
Ch1
Ch2
Time: 0.00263698
Counts: 0.245087
Ch1: 0.00815145
Ch2: -0.370861
Counts
Time (ms)
Persistence
Analog Alpha: 0.0994
Axes Options
Secs/Div:
Counts/Div:
Y Offset:
T Offset:
Autorange
Channel Options
Ch1
Ch2
Trig
XY
Coupling:
DC
Marker:
Line Link
Stop


Rx OFDM Modulated Signal
Rx Spectrum
Rx
Rx Spectrum
FFT
Power (dB)
Frequency (kHz)
Trace Options
Peak Hold
Average
Avg Alpha: 0.1333
Persistence
Persist Alpha: 0.1755
Trace A
Trace B
Axis Options
dB/Div:
Ref Level:
Autoscale
Stop

INFERENCE:

A software defined window offers flexibility to deliver the highly reconfigurable system requirements. This approach allows a different type of communication system requirements such as standard, protocol or signal processing method, to be deployed by using the same set of hardware and software such as USRP and GNU radio respectively. However, the realization of SDR concept is inherently limited by analog components of hardware being used.

RESULT and CONCLUSION:

Thus, we have studies the SDR based transceiver system using USRP kit and GNU radio