SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

S.R.M Nagar, Kattankulathur – 603 203

Department of Electronics and Communication Engineering

Laboratory Manual

EC3664– WIRELESS COMMUNICATION AND NETWORKING LABORATORY

Regulation - 2023

Semester/Branch	**:**	**VI semester ECE**
Academic Year	**:**	**2025 - 2026 (EVEN SEMESTER)**
Prepared By	**:**	**Ms.S.Abirami,** *Assistant Professor* / ECE
		Dr. C.Kavitha, *Assistant Professor* / ECE
		Dr. K. Lekha, *Assistant Professor* / ECE

SRM VALLIAMMAI ENGINEERING COLLEGE

(An Autonomous Institution)

SRM Nagar, Kattankulathur – 603 203

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

VISION OF THE INSTITUTE

Educate to excel in social transformation

MISSION OF THE INSTITUTE

- To contribute to the development of human resources in the form of professional engineers and managers of international excellence and competence with high motivation and dynamism, who besides serving as ideal citizen of our country will contribute substantially to the economic development and advancement in their chosen areas of specialization.
- To build the institution with international repute in education in several areas at several levels with specific emphasis to promote higher education and research through strong institute-industry interaction and consultancy.

VISION OF THE DEPARTMENT

To excel in the field of electronics and communication engineering and to develop highly competent technocrats with global intellectual qualities.

MISSION OF THE DEPARTMENT

M1: To train the students with the state-of-art technologies and to develop innovative solutions to cater to the societal needs.

M2: To orient the students towards global career with universal moral values and professional ethics.

PROGRAM OUTCOMES

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOME (PSOs)

1. Ability to solve societal problems by applying knowledge acquired in the core domain.
2. Ability to apply technologies like VLSI, Embedded and Wireless Communications to meet the industrial requirements.

Syllabus

EC3664 WIRELESS COMMUNICATION AND NETWORKING LABORATORY **L T P C 0 0 3 1.5**

OBJECTIVES:

- To understand the Wireless channel characteristics
- To analyse the equalization techniques applied to the wireless channels.
- To explore the multicarrier modulation and multiple access techniques used
- in wireless communication.
- To acquire knowledge on the features of the network layers.
- To implement the protocols in the different layers of network.
- To estimate the performance and QoS of wireless networks.

LIST OF EXPERIMENTS

1. Characterization of Wireless Channels.
2. Simulation of Equalization Techniques for Wireless Channels.
3. Simulation / Implementation of Multicarrier Modulation.
4. Implementation of Space Time Block Codes.
5. Simulation of Adaptive Modulation and Coding.
6. Modelling and simulation of TDMA, FDMA and CDMA for wireless Communication.
7. Simulation of Ethernet Wired LAN (IEEE 802.3).
8. Implementation of Error Detection / Error Correction Techniques.
9. Study the performance of network with CSMA / CA protocol.
10. Implementation of Flow control Algorithms.
11. Simulation and performance evaluation of a network using TCP and UDP.
12. QoS Analysis of Wireless Networks.

TOPIC BEYOND THE SYLLABUS EXPERIMENT

13. Network Topology - Star, Bus, Ring.

Total: 45 Periods

COURSE OUTCOMES:

On completion of the course, the student will be able to

- ❖ Interpret the significance of the protocols used in the network.
- ❖ Illustrate the characteristics of network layers.
- ❖ Discriminate the multicarrier modulation and the multiple access methods.
- ❖ Apply the equalization techniques to the wireless channel.
- ❖ Characterize the behavior of the wireless channels in different scenarios.
- ❖ Assess the QoS of wireless networks.

COURSE OUTCOMES - PROGRAM OUTCOMES MATRIX

CO	PO												PSO	
	1	2	3	4	5	6	7	8	9	10	11	12	1	2
EC3664.1	2	2	-	-	3	-	-	-	2	2	-	2	-	3
EC3664.2	2	-	-	-	3	-	2	-	2	2	-	2	2	3
EC3664.3	2	-	-	-	3	-	-	-	2	2	-	2	2	3
EC3664.4	2	2	3	2	3	-	2	-	2	2	-	2	2	3
EC3664.5	3	-	-	-	3	2	-	-	2	2	-	2	2	3
EC3664.6	2	2	-	2	3	2	2	-	2	2	2	2	2	3
AVG	2	2	3	2	3	2	2	-	2	2	2	2	2	3

LAB REQUIREMENTS FOR A BATCH OF 30 STUDENTS

S.No	Equipment / Software	Required
1	Desktop Computer	30 Nos.
2	MATLAB	15 Users
3	Network Simulator NS3/Glomosim/OPNET/ Packet Tracer	30 Users

EXPT 1 : Characterization of Wireless Channels.

AIM: To design a model of wireless communication systems using Matlab (Two ray channel and Okumura –Hata model).

SOFTWARE REQUIRED:

SYSTEM with MATLAB software.

THEORY:

In wireless communication systems, the transmitted signal undergoes **attenuation, reflection, diffraction, and scattering** before reaching the receiver. To predict signal strength and coverage, **propagation models** are used. These models estimate **path loss**, which is the reduction in signal power with distance. Large-scale propagation models focus on **average signal power** over large distances and are independent of small-scale fading effects.

The received power using the Two-Ray model is given by:

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4}$$

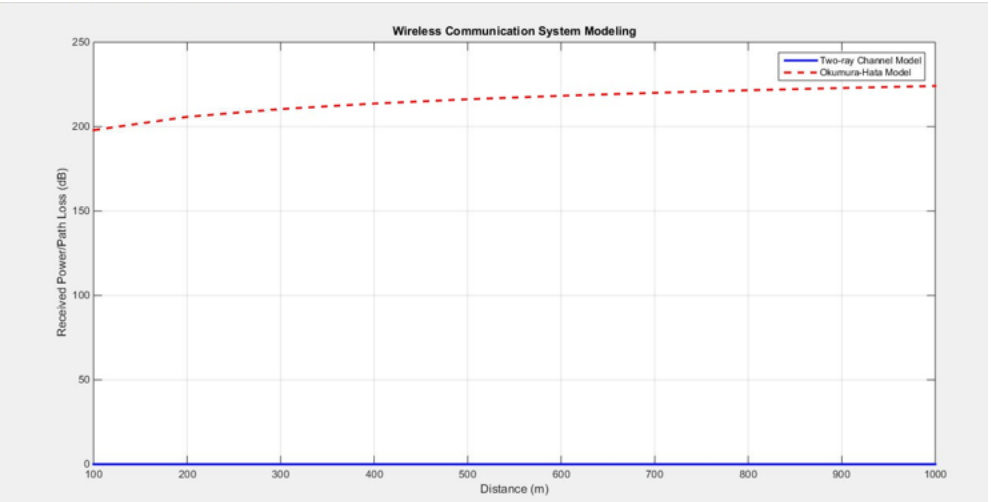
Where:

- P_t= Transmitted power
- G_t, G_r= Transmitter and receiver antenna gains
- h_t, h_r= Heights of transmitter and receiver antennas
- d= Distance between antennas

```
>> % Wireless Communication System Modeling

% Two-ray channel model and Okumura-Hata model

% System Parameters

frequency = 900e6;       % Frequency in Hz transmitter

Height = 50; % Transmitter height in meters

receiverHeight = 10;     %

Receiver height in meters

distance = 100:100:1000; % Distance between transmitter and receiver in meters

% Two-ray Channel Model

Pt = 1;                  % Transmitted power in Watts

Gt = 1;                  % Transmitter antenna gain
```

```
Gr = 1;                % Receiver antenna gain

L = 1;                 % System loss

% Calculate received power using Two-ray channel model

Pr_two_ray = Pt * (Gt * Gr * (transmitterHeight * receiverHeight)^2) ./ (distance.^4 * L);

% Okumura-Hata Model

A = 69.55;             % Model parameter

B = 26.16;             % Model parameter

C = 13.82;             % Model parameter

D = 44.9;              % Model parameter

X = 6.55;              % Model parameter

hb = 30;               % Base station height in meters

% Calculate path loss using Okumura-Hata model

PL_okumura_hata = A + B * log10(distance) + C * log10(frequency/1e6) + D - X * log10(hb);
%

Plotting

figure;

plot(distance, Pr_two_ray, 'b-', 'LineWidth', 2);
hold on;

plot(distance, PL_okumura_hata, 'r--', 'LineWidth', 2);


xlabel('Distance (m)');

ylabel('Received Power/Path Loss (dB)');
legend (‘Two ray channel model’,’Okumura Hata model’);
title(‘wireless communication system modelling’);
grid on;
```

Output:



RESULT : Thus designing a model of wireless communication systems using Matlab(Two ray channel and Okumura –Hata model) is achieved.

VIVA QUESTIONS :

1. What is Okumura-Hata model in wireless communication?
2. What is the difference between Hata model and Okumura model?
3. Is the Hata model used for signal strength prediction?
4. What is wireless channel model?
5. What are the main wireless channels?
6. Which wireless channel is better?

EXPT 2 : Simulation of Equalization Techniques for Wireless Channels.

AIM: . To design a wireless Channel equalization: Zero-Forcing Equalizer (ZFE), MMS Equalizer (MMSEE), Adaptive Equalizer (ADE),Decision Feedback Equalizer (DFE)

SOFTWARE REQUIRED:

SYSTEM with MATLAB software

THEORY:

In wireless communication, the transmitted signal experiences **multipath propagation**, resulting in delayed replicas of the signal reaching the receiver. This causes **Inter Symbol Interference (ISI)**, where symbols overlap and degrade system performance. **Channel equalization** is a signal processing technique used at the receiver to **compensate for channel distortion** and recover the original transmitted symbols.

CODING :

```
% Zero-Forcing Equalizer (ZFE) MATLAB code

% Define the channel impulse responseh = [0.1 0.3

0.4 0.2];

% Generate random transmitted

symbolsN = 100; % Number of symbols

symbols = randi([0, 1], 1, N);

% Convolve transmitted symbols with the channel impulse

responsereceived_signal = conv(symbols, h);

% Add AWGN (Additive White Gaussian Noise) to the received signalsnr_dB = 20; %

Signal-to-Noise Ratio (SNR) in dB

received_signal = awgn(received_signal, snr_dB, 'measured');

% Zero-Forcing Equalizer

% Define the length of the equalizer

tapL = length(h);

% Initialize the equalizer

taps equalizer_taps =

zeros(1, L);
```

```
% Loop through each received symbol and perform equalizationequalized_symbols
= zeros(1, N); for n = 1:N
  % Extract the received symbols for equalization received_symbols
= received_signal(n:n+L-1);
  % Perform zero-forcing equalization
equalized_symbols(n) = equalizer_taps *
received_symbols';
  % Update the equalizer taps using the least squares
  algorithmerror = symbols(n) -
  equalized_symbols(n);
equalizer_taps = equalizer_taps + error * received_symbols / (received_symbols
*received_symbols'); end
% Print the original symbols and equalized
symbolsdisp('Original Symbols:');
disp(symbols);
disp('Equalized Symbols:');
disp(equalized_symbols);
```

OUTPUT:

```
Original Symbols:
  Columns 1 through 19
    1    1    0    1    1    0    0    1    1    1    0    1    1    0    1    0    0    1    1
  Columns 20 through 38
    1    1    0    1    1    1    1    1    0    1    0    1    0    0    0    0    1    1    0
```

Columns 39 through 57

1 0 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 0 1

Columns 58 through 76

0 1 0 1 0 1 1 1 1 1 0 0 0 1 0 1 0 1 0

Columns 77 through 95

0 0 1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 1 1

Columns 96 through 100

0 1 0 0 0

Equalized Symbols:

Columns 1 through 11

0 1.0242 1.0351 -0.0148 0.8118 0.8423 0.2395 0.3127 0.8637 0.5667 1.0034

Columns 12 through 22

0.3164 0.8987 0.7162 -0.0194 0.8262 0.2108 0.3684 1.3409 0.6328 0.5942 0.7986

Columns 23 through 33

0.3903 1.3034 0.9963 0.6816 0.6242 0.6419 0.1078 0.9584 -0.0282 0.4643 -0.0959

Columns 34 through 44

0.3857 0.3709 0.1746 1.1529 0.6859 -0.3254 0.6316 -0.1321 0.2851 0.6131
0.9881

Columns 45 through 55

0.1328 -0.3112 0.5753 0.4748 1.4226 0.8176 0.5202 0.2300 0.9991 0.4921
-
0.2495

Columns 56 through 66

0.2145 0.5610 1.0497 -0.3251 1.0165 -0.0410 1.1669 0.3767 1.3984 0.8522
0.7683

Columns 67 through 77

0.6932 0.4118 -0.0997 0.1789 0.1747 1.2491 0.0166 1.0660 -0.0451 0.5827
-
0.1786

Columns 78 through 88

0.2406 0.4407 0.5875 -0.0514 0.5994 1.4474 0.8587 0.6711 0.4184 0.5040
1.2422

Columns 89 through 99

0.4668 -0.0972 0.6936 -0.1060 0.7651 1.3313 0.6154 0.7091 0.0191 0.5241
-
0.1900

Column 100

0.0171

---

2. MMSE CODE

```
% Parameters
M = 4;                  % Number of transmitted symbols
N = 1000;               % Number of received symbols
SNRdB = 10;             % Signal-to-Noise Ratio in
dB pilotSymbols = [1 -1 1 -1]; % Known pilot
symbols

% Generate random symbols
transmittedSymbols = randi([0 M-1], 1,
N);

% Modulation
modulatedSymbols = qammod(transmittedSymbols, M);
% Channel
channel = [0.8 -0.4 0.2 -0.1]; % Example channel coefficientschannelOutput =
filter(channel, 1, modulatedSymbols);

% Add noise
SNR = 10^(SNRdB/10);
noiseVar = 1/(2*SNR);
noise = sqrt(noiseVar) * randn(1,
length(channelOutput));receivedSignal = channelOutput + noise;
```

```
% Channel estimation using pilot symbols
pilotIndices = randperm(N, length(pilotSymbols));pilotSignal = receivedSignal(pilotIndices);
estimatedChannel = conv(pilotSignal, conj(pilotSymbols(end:-1:1)));estimatedChannel = estimatedChannel(end-length(channel)+1:end);
% MMSE equalization
equalizerCoefficients = conj(estimated Channel) ./ (abs(estimated Channel).^2 + noiseVar);equalized Symbols = filter(equalizer Coefficients, 1, receivedSignal);
% Demodulation
Demodulated Symbols = qamde mod(equalizedSymbols, M);
% Calculate bit error rate
bitErrors = sum(transmitted Symbols ~= demodulatedSymbols);bitErrorRate = bitErrors / N;
disp(['Bit Error Rate: ' num2str(bitErrorRate)]);
```

output:

Bit Error Rate: 0.787

ADE:

```
% Parameters
channel_length = 10;        % Length of the channel impulse responsesnr_db = 20;
                            % Signal-to-noise ratio in dB
```

```
num_symbols = 1000;    % Number of symbols to
transmitmu = 0.01;% LMS step size
% Generate random symbols
data_symbols = randi([0, 1], 1, num_symbols);
% Modulate symbols (e.g., BPSK
modulation) modulated_symbols = 2 *
data_symbols - 1;
% Create the channel impulse response
channel = (randn(1, channel_length) + 1i * randn(1, channel_length)) / sqrt(2);
% Convolve the modulated symbols with the channel
received_symbols = filter(channel, 1, modulated_symbols);
% Add noise to the
received signal
noise_power = 10^(-
snr_db / 10);
noise = sqrt(noise_power) * (randn(1, length(received_symbols)) + 1i * randn(1,
length(received_symbols)));
received_symbols_noisy = received_symbols + noise;
% Adaptive equalizer using the LMS algorithm
equalizer_length = channel_length; % Set the equalizer length to match the channel
lengthequalizer = zeros(1, equalizer_length);
output_signal = zeros(1, length(received_symbols_noisy));

for i = equalizer_length:length(received_symbols_noisy)
    % Extract the received symbols for the current equalizer window
    received_window = received_symbols_noisy(i:-1:i-
    equalizer_length+1);
```

```
    % Compute the equalizer output output_signal(i) = equalizer
  * received_window.';
    % Compute the error
  error = modulated_symbols(i) - output_signal(i);
    % Update the equalizer coefficients
  equalizer = equalizer + mu * conj(error) * received_window;end
% Demodulate the equalized symbols (decision-
directed) demodulated_symbols = real(output_signal)
> 0;
% Calculate the bit error rate (BER)
ber = sum(data_symbols ~= demodulated_symbols) /
num_symbols;disp(['Bit Error Rate (BER): ', num2str(ber)]);
```

output:

Bit Error Rate (BER): 0.519

RESULT: Thus designing a wireless Channel equalization: Zero-Forcing Equalizer (ZFE), MMS Equalizer(MMSEE), Adaptive Equalizer (ADE),Decision Feedback Equalizer(DFE) has been achieved.

VIVA QUESTIONS :

1. What is the working principle of zero-forcing equalizer?
2. Why is zero forcing used?
3. What is the need of an equalizer?
4. What is the MMSE channel equalization?
5. What are channel equalization methods?
6. What is the function of the equalizer?
7. What is adaptive equalization of channel?

EXPT 3 : SIMULATION / IMPLEMENTATION OF MULTICARRIER MODULATION

AIM: To simulate and implement multicarrier modulation, the OFDM in MATLAB.

SOFTWARE REQUIRED:

SYSTEM with MATLAB software

THEORY

Multicarrier modulation is a technique in which the available bandwidth is divided into multiple **narrowband subcarriers**, and data is transmitted simultaneously over these subcarriers. Each subcarrier is modulated with a low-rate data stream, making the system more robust against **frequency-selective fading** and **Inter Symbol Interference (ISI)**.

OFDM is a special form of multicarrier modulation where subcarriers are **orthogonal** to each other. Orthogonality ensures that subcarriers overlap in frequency without causing **Inter-Carrier Interference (ICI)**.

The subcarrier spacing is chosen as:

$$\Delta f = \frac{1}{T}$$

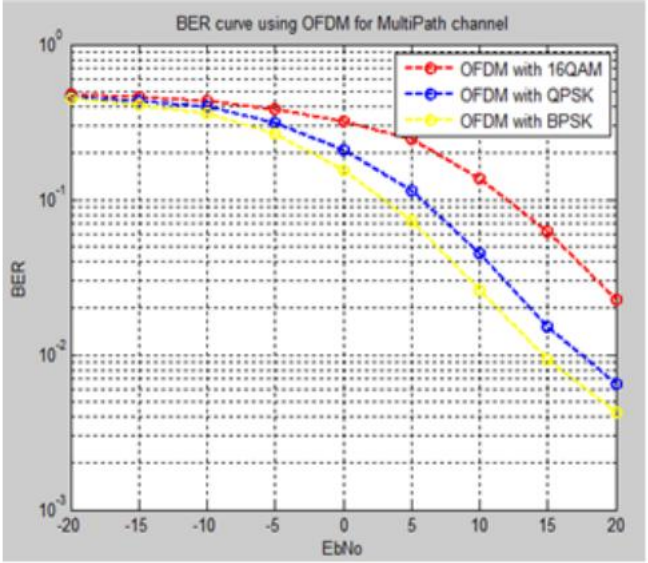
CODING:

```
Nfft = 64; CP = 16; M = 16; % 16-QAM, parameters
numDataCarrs = Nfft - 12; nullIdx = [1:6 59:64]; % Null subcarriers
pilotIdx = [8 22 30 44]; numPilots = length(pilotIdx);
SNRdB = 0:2:20; BER = zeros(size(SNRdB));
for snrIdx = 1:length(SNRdB)
  numSym = 1000; errors = 0; dataTx = randi([0 M-1], numDataCarrs-numPilots, numSym);
  pilots = [1+1j 1-1j -1+1j -1-1j]; % Known pilots
  for sym = 1:numSym
    dataSym = [dataTx(:,sym); pilots(randperm(4))]; % Insert pilots
    X = qammod(dataSym, M, 'UnitAveragePower', true); % Modulate
    tx = ifft(X, Nfft)*sqrt(Nfft); txCP = [tx(end-CP+1:end); tx]; % IFFT + CP
    rx = awgn(txCP, SNRdB(snrIdx), 'measured'); % Channel
    rxSym = rx(CP+1:end); Y = fft(rxSym, Nfft)/sqrt(Nfft); % Demod
    dataRx = qamdemod(Y(setdiff(1:Nfft, [nullIdx' pilotIdx'])), M, 'UnitAveragePower', true);
    errors = errors + sum(dataTx(:,sym) ~= dataRx); end
  BER(snrIdx) = errors / (numSym*(numDataCarrs-numPilots));
end
```

BER Output Plot

```
semilogy(SNRdB, BER, 'b-o', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)');
 ylabel('BER');
title('OFDM BER vs SNR');
```

OUTPUT:



Result:

Thus simulation and implementation of multicarrier modulation OFDM.

VIVA QUESTIONS:

1. What causes CFO and how does it affect OFDM subcarriers?
2. How is OFDM symbol timing detected using cyclic prefix?
3. How does phase noise affect OFDM and how do pilots correct it?
4. How does timing offset affect BER and how is it corrected in MATLAB?
5. Coarse vs. fine frequency synchronization: which is better for multipath

EXPT 4 : IMPLEMENTATION OF SPACE TIME BLOCK CODES.

AIM: To implementation of Space Time Block Codes, the Alamouti implementation in MATLAB.

SOFTWARE REQUIRED:

SYSTEM with MATLAB software

THEORY:

Space Time Block Codes are transmit diversity techniques that transmit multiple copies of data symbols across different antennas and time slots to improve reliability without increasing transmit power or bandwidth.

STBCs provide:

- Diversity gain
- Simple decoding
- Improved BER performance

The Alamouti scheme is the **simplest and most widely used STBC**, offering **full diversity** with a **code rate of 1**.

It transmits two symbols over **two transmit antennas** in **two time slots**.

Alamouti Encoder Function

Define the encoder for two symbols s_1, s_2:

```
function stbc_tx = alamoutiEnc(s1, s2)
   stbc_tx = [s1, -conj(s2); s2, conj(s1)];  % 2x2 matrix [web:13]
end
```

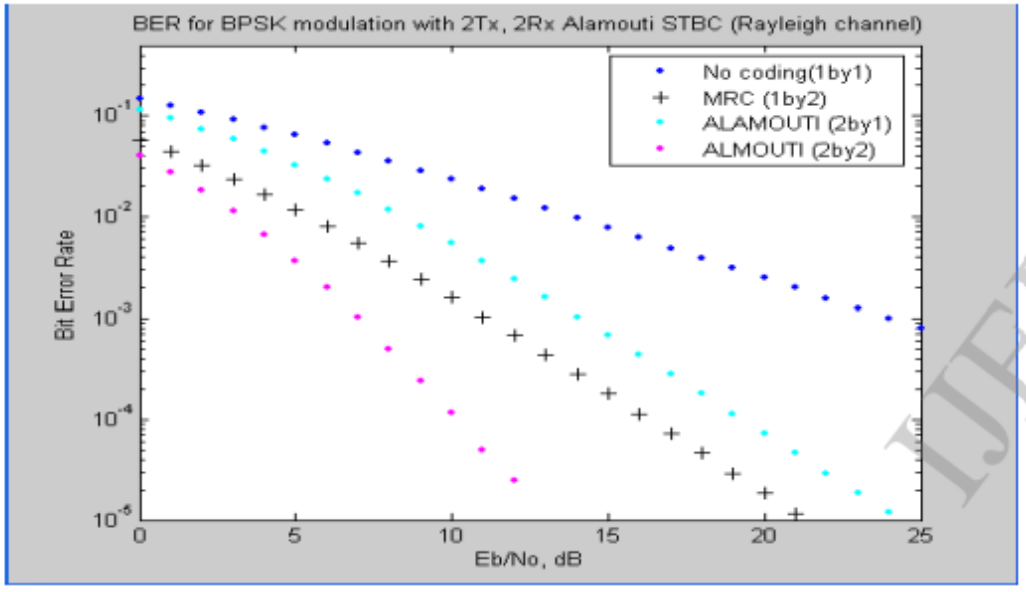
Complete BER Simulation Script

```
N = 1e5;  % Symbols per SNR point
SNR_dB = 0:2:20; BER = zeros(size(SNR_dB));
modOrder = 2;  % BPSK
for snr_idx = 1:length(SNR_dB)
   snr_lin = 10^(SNR_dB(snr_idx)/10);
   numErr = 0; numBits = 0
   while numBits < N*modOrder
```

```
% Generate BPSK symbols (0/1 -> +/-1)
data = randi([0 1], 2, 1);
s = 1 - 2*data;  % s1, s2

% Encode
X = [s(1), -conj(s(2)); s(2), conj(s(1))];
X = X / sqrt(2);  % Normalize

% Channel: 2 Tx, 1 Rx, Rayleigh
H = (randn(1,2) + 1j*randn(1,2)) / sqrt(2);  % h1, h2
noiseVar = 1 / snr_lin;
n1 = sqrt(noiseVar/2) * (randn + 1j*randn);
n2 = sqrt(noiseVar/2) * (randn + 1j*randn);

% Received signals
r1 = H(1)*X(1,1) + H(2)*X(2,1) + n1;
r2 = H(1)*X(1,2) + H(2)*X(2,2) + n2;
R = [r1; r2];

% Decode: Combine
s1_hat = conj(H(1))*R(1) + H(2)*conj(R(2));
s2_hat = conj(H(2))*R(1) - H(1)*conj(R(2));

% Detect
data_hat1 = (real(s1_hat) < 0);
data_hat2 = (real(s2_hat) < 0);
```

```
        data_hat = [data_hat1; data_hat2];
        numErr = numErr + sum(data ~= data_hat);
        numBits = numBits + length(data);
    end
    BER(snr_idx) = numErr / numBits;
end
semilogy(SNR_dB, BER, 'b-o'); grid on; xlabel('SNR (dB)'); ylabel('BER');
```

OUTPUT:



Result :

To implementation of Space Time Block Codes, the Alamouti implementation in MATLAB.

VIVA QUESTIONS:

1. What is the Alamouti STBC matrix for symbols s_1, s_2, and how do you normalize it in MATLAB?
2. How does the Alamouti receiver combine signals to recover s_1, and how is this done in MATLAB?
3. Why does Alamouti STBC give better (2nd-order) diversity than single-antenna SISO in Rayleigh fading?
4. How do you generate Rayleigh fading channel coefficients in MATLAB, and what does "perfect CSI" mean?
5. How does ML detection work for BPSK/QPSK after STBC combining (e.g., using real(s_hat) < 0)?

EXPT 5. SIMULATION OF ADAPTIVE MODULATION AND CODING

AIM: To simulate Adaptive modulation and coding using MATLAB software

SOFTWARE REQUIRED : System with MATLAB software.

THEORY:

Modern wireless communication systems operate in time-varying and unpredictable channel conditions due to fading, interference, and noise. To maintain reliable communication while maximizing data rate, systems use Adaptive Modulation and Coding (AMC).
AMC dynamically adjusts:

- Modulation order
- Coding rate

based on the instantaneous channel quality, usually measured in terms of Signal-to-Noise Ratio (SNR).

CODING:

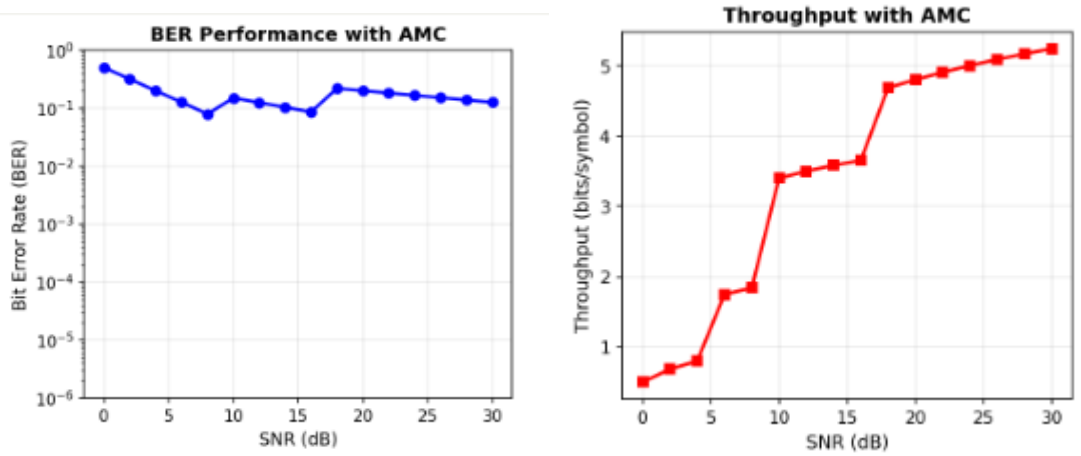
```
% Adaptive Modulation and Coding (AMC) Simulation
% This code demonstrates AMC with SNR-based modulation scheme selection

clear all;
close all;
clc;

%% Parameters
numBits = 10000;              % Number of bits to transmit
SNR_dB = 0:2:30;              % SNR range in dB
modulationSchemes = {'BPSK', 'QPSK', '16QAM', '64QAM'};

% SNR Thresholds for modulation switching (in dB)
SNR_threshold = [0, 5, 10, 18];

% Initialize results
BER = zeros(1, length(SNR_dB));
throughput = zeros(1, length(SNR_dB));
selectedModulation = cell(1, length(SNR_dB));

%% Main Simulation Loop
for idx = 1:length(SNR_dB)
    snr = SNR_dB(idx);
```

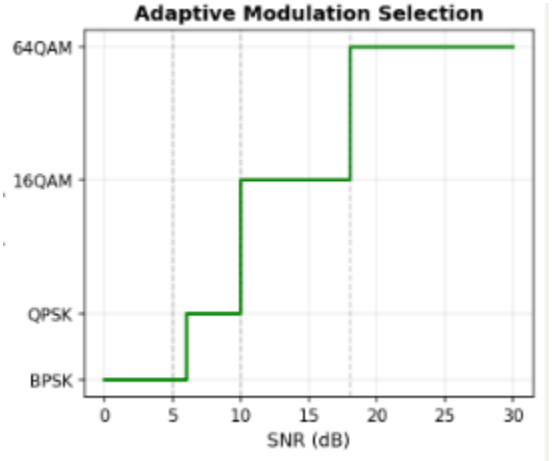
```
% Adaptive Modulation Scheme Selection based on SNR
if snr < 5
    modScheme = 'BPSK';
    M = 2;              % Modulation order
    k = 1;              % Bits per symbol
elseif snr >= 5 && snr < 10
    modScheme = 'QPSK';
    M = 4;
    k = 2;
elseif snr >= 10 && snr < 18
    modScheme = '16QAM';
    M = 16;
    k = 4;
else
    modScheme = '64QAM';
    M = 64;
    k = 6;
end

selectedModulation{idx} = modScheme;

% Generate random binary data
txBits = randi([0 1], numBits, 1);

% Modulation
if strcmp(modScheme, 'BPSK')
    txSymbols = pskmod(txBits, M);
elseif strcmp(modScheme, 'QPSK')
    txSymbols = pskmod(txBits, M, pi/4);
else
    txSymbols = qammod(txBits, M, 'InputType', 'bit', 'UnitAveragePower', true);
end

% AWGN Channel
rxSymbols = awgn(txSymbols, snr, 'measured');

% Demodulation
if strcmp(modScheme, 'BPSK')
    rxBits = pskdemod(rxSymbols, M);
elseif strcmp(modScheme, 'QPSK')
    rxBits = pskdemod(rxSymbols, M, pi/4);
else
    rxBits = qamdemod(rxSymbols, M, 'OutputType', 'bit', 'UnitAveragePower', true);
end

% Calculate BER
[numErrors, BER(idx)] = biterr(txBits, rxBits);

% Calculate throughput (bits per symbol)
```

```
    throughput(idx) = k * (1 - BER(idx));

    % Display progress
    fprintf('SNR = %d dB | Modulation: %s | BER: %.4e\n', snr, modScheme, BER(idx));
end

%% Plot Results
figure('Position', [100, 100, 1200, 400]);

% Plot 1: BER vs SNR
subplot(1,3,1);
semilogy(SNR_dB, BER, 'b-o', 'LineWidth', 2, 'MarkerSize', 6);
grid on;
xlabel('SNR (dB)');
ylabel('Bit Error Rate (BER)');
title('BER Performance with AMC');
ylim([1e-6 1]);

% Plot 2: Throughput vs SNR
subplot(1,3,2);
plot(SNR_dB, throughput, 'r-s', 'LineWidth', 2, 'MarkerSize', 6);
grid on;
xlabel('SNR (dB)');
ylabel('Throughput (bits/symbol)');
title('Throughput with AMC');

% Plot 3: Modulation Scheme Selection
subplot(1,3,3);
modulation_numeric = zeros(1, length(SNR_dB));
for i = 1:length(SNR_dB)
    if strcmp(selectedModulation{i}, 'BPSK')
        modulation_numeric(i) = 1;
    elseif strcmp(selectedModulation{i}, 'QPSK')
        modulation_numeric(i) = 2;
    elseif strcmp(selectedModulation{i}, '16QAM')
        modulation_numeric(i) = 4;
    else
        modulation_numeric(i) = 6;
    end
end
stairs(SNR_dB, modulation_numeric, 'g-', 'LineWidth', 2);
grid on;
xlabel('SNR (dB)');
ylabel('Bits per Symbol');
title('Adaptive Modulation Selection');
yticks([1 2 4 6]);
yticklabels({'BPSK', 'QPSK', '16QAM', '64QAM'});

%% Display Summary
fprintf('\n=== AMC Simulation Summary ===\n');
```

```
fprintf('Total SNR points simulated: %d\n', length(SNR_dB));
fprintf('Modulation schemes used: BPSK, QPSK, 16-QAM, 64-QAM\n');
fprintf('Maximum throughput achieved: %.2f bits/symbol at SNR = %d dB\n', ...
    max(throughput), SNR_dB(find(throughput == max(throughput), 1)));
```

OUTPUT:





RESULT

Thus , the adaptive modulation and coding is simulated using MATLAB

VIVA QUESTIONS

1. What is Adaptive Modulation and Coding (AMC) and why is it needed in wireless communication systems?
2. Explain the SNR threshold-based modulation selection strategy used in your MATLAB simulation.
3. What are the key advantages and challenges of implementing AMC in practical wireless systems?
4. How does throughput vary with SNR in AMC, and what is the relationship between BER and throughput?
5. What is the role of channel state information (CSI) feedback in AMC, and what happens if feedback is delayed or inaccurate?

EXPT 6 : Modelling and simulation of TDMA, FDMA and CDMA for wireless Communication.

AIM: To model and simulate TDMA, FDMA and CDMA for wireless communication

SOFTWARE REQUIRED:

SYSTEM with MATLAB software

THEORY:

In wireless communication systems, multiple users share a limited available bandwidth. Multiple access techniques allow several users to transmit information simultaneously without significant interference.

The most widely used multiple access techniques are:

- TDMA
- FDMA
- CDMA

In TDMA, all users share the **same frequency band**, but each user transmits in a **different time slot** within a repeating frame. Only one user transmits at any given time.

If the frame duration is T_f and the number of users is N:

$$T_{slot} = \frac{T_f}{N}$$

In FDMA, the total available bandwidth is divided into non-overlapping frequency channels, and each user is assigned a unique frequency band for the entire duration of communication.

All users transmit simultaneously.

In CDMA, all users transmit **simultaneously over the same frequency band** and at the same time, but each user is assigned a **unique spreading code**.

The receiver uses the corresponding code to extract the desired user's signal.

CODING:

1. TDMA

```
%function TDMA_simulation()

num_users = 3;     % Number of users
num_time_slots = 10;    % Number of time slots in the
simulation slot_duration = 1;    % Time duration of each
time slot (in seconds)

% Generate random data for each
user user_data = cell(1, num_users);
for user = 1:num_users
  user_data{user}     =     randi([0,     1],     1,
num_time_slots); end

% Simulation of TDMA
for          time_slot          =
```

```
    1:num_time_slots for user
    = 1:num_users
      if user_data{user}(time_slot) == 1
        fprintf('Time Slot %d: User %d is transmitting.\n',
      time_slot, user); else
        fprintf('Time Slot %d: User %d is idle.\n',
      time_slot, user);
        end
    end
    pause(slot_duration)
end

 end
```

OUTPUT

Time Slot 1: User 1 is idle.
Time Slot 1: User 2 is idle.
Time Slot 1: User 3 is transmitting.
Time Slot 2: User 1 is transmitting.
Time Slot 2: User 2 is idle.
Time Slot 2: User 3 is idle.
Time Slot 3: User 1 is idle.
Time Slot 3: User 2 is transmitting. Time Slot 3: User 3 is idle.

2. FDMA:

%function FDMA_simulation()

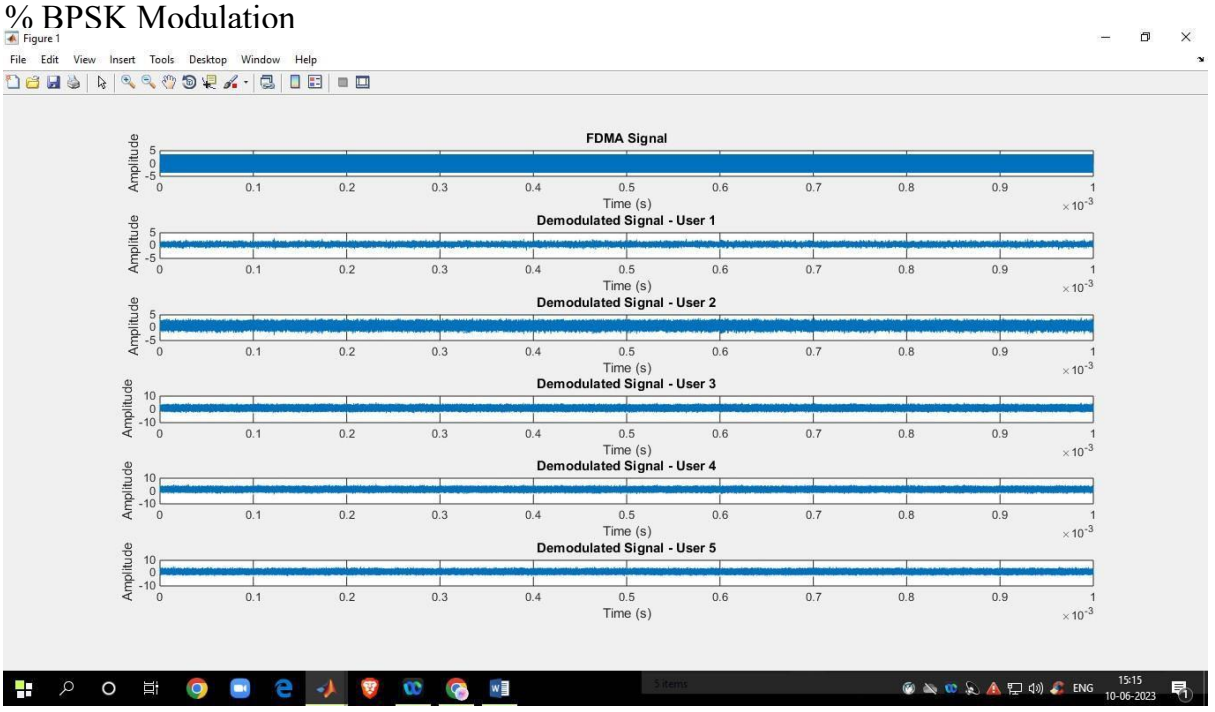
```
  num_users = 3;      % Number of users
  num_time_slots = 10;    % Number of time slots in the
  simulation num_frequency_bands = 5; % Number of
  frequency bands

  % Generate random data for each
  user user_data = cell(1, num_users);
  for user = 1:num_users
    user_data{user}    =    randi([0,    1],    num_frequency_bands,
  num_time_slots); end

  % Simulation of FDMA
  for time_slot = 1:num_time_slots
    for frequency_band = 1:num_frequency_bands
      fprintf('Time Slot %d: Frequency Band %d -> ', time_slot,
      frequency_band); for user = 1:num_users
```

```
            if user_data{user}(frequency_band, time_slot) == 1
               fprintf('User %d ', user);
            end
            end
          fprintf('is
       transmitting.\n'); end
    end

end
```

OUTPUT

Time Slot 1: Frequency Band 1 -> User 1 User 2 is transmitting. Time Slot 1: Frequency Band 2 -> User 1 is transmitting.
Time Slot 1: Frequency Band 3 -> User 3 is transmitting.
Time Slot 1: Frequency Band 4 -> User 1 User 3 is transmitting.
Time Slot 1: Frequency Band 5 -> User 1 User 2 is transmitting. Time Slot 2: Frequency Band 1 -> User 1 is transmitting.
Time Slot 2: Frequency Band 2 -> User 2 is transmitting. Time Slot 2: Frequency Band 3 -> User 2 is transmitting.
Time Slot 2: Frequency Band 4 -> User 1 User 3 is transmitting. Time Slot 2: Frequency Band 5 -> User 2 is transmitting.

% BPSK Modulation



3.CDMA:

% CDMA Simulation Parameters

numBits = 1000; % Number of bits per user

chipRate = 1e6; % Chip rate (chips per second)

snr = 10; % Signal-to-Noise Ratio (dB)

% Generate random data bits for User 1 and User 2

user1Bits = randi([0, 1], 1, numBits);

user2Bits = randi([0, 1], 1, numBits);

user1Symbols = 2 * user1Bits - 1; % Map 0s to -1 and

1s to 1 user2Symbols = 2 * user2Bits - 1;

% Chip-level Spreading (using a simple chip sequence)

chipSequence = [1, -1, 1, 1, -1, 1, -1, -1]; % Chip sequence for

spreading user1SpreadSymbols = kron(user1Symbols,

chipSequence); user2SpreadSymbols = kron(user2Symbols,

```
chipSequence);

% Add AWGN (Additive White Gaussian

Noise)noiseVar = 10^(-snr/10);  %        Noise

variance

user1NoisySymbols = user1SpreadSymbols + sqrt(noiseVar/2) *
randn(1, length(user1SpreadSymbols));

user2NoisySymbols = user2SpreadSymbols + sqrt(noiseVar/2) *
randn(1, length(user2SpreadSymbols));

% Matched Filtering (correlation with chip sequence)

user1FilteredSymbols = filter(fliplr(chipSequence), 1,
user1NoisySymbols);user2FilteredSymbols

= filter(fliplr(chipSequence), 1, user2NoisySymbols);

% Symbol Detection (using correlation with chip sequence) user1DetectedBits =

user1FilteredSymbols(1:length(user1Symbols)) > 0;user2DetectedBits =

user2FilteredSymbols(1:length(user2Symbols)) > 0;

% Bit Error Rate (BER) Calculation

berUser1 = sum(user1DetectedBits ~= user1Bits) /

numBits;berUser2 = sum(user2DetectedBits ~= user2Bits) /

numBits;

% Display results

disp(['User 1 BER: ',

num2str(berUser1)]);disp(['User 2 BER: ',

num2str(berUser2)]);
```

Output:

User 1 BER: 0.523

User 2 BER: 0.535

Result : Thus modeling and simulation of TDMA, FDMA and CDMA for wireless communication has been achieved.

Viva Questions:

1. What is the basic principle of TDMA?
2. What is TDMA used for?
3. Why GSM is called TDMA?
4. What is the basic principle of FDMA?
5. What is the frequency range of FDMA?
6. What are the applications of FDMA?
7. What is the principle of CDMA?
8. What technology is used in CDMA?
9. What is the noise power of a CDMA system?

Experiment 7
Simulation of Ethernet Wired LAN (IEEE 802.3)

Aim

To simulate an **Ethernet wired LAN based on IEEE 802.3** using NS-2 and analyze packet transmission between nodes.

Software Required

- NS-2.35
- Linux OS (Ubuntu)
- NAM (Network Animator)

Theory

IEEE 802.3 defines the standards for **Ethernet LANs**, which use:

- **CSMA/CD** for medium access
- **Frame-based transmission**
- Shared or switched wired medium

In NS-2, Ethernet LANs are simulated using the **make-lan** command, which internally models IEEE 802.3 behavior.

Network Topology

- Three nodes connected in a **wired Ethernet LAN**
- One TCP source and one UDP source

NS-2 TCL Code

```
# Create Simulator
set ns [new Simulator]

# Open trace and NAM files
set tracefile [open ethernet.tr w]
set namfile [open ethernet.nam w]
$ns trace-all $tracefile
$ns namtrace-all $namfile

# Finish procedure
proc finish {} {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam ethernet.nam &
   exit 0
}

# Create nodes
```

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

# Create Ethernet LAN (IEEE 802.3)
$ns make-lan "$n0 $n1 $n2" 10Mb 10ms LL Queue/DropTail MAC/802_3

# TCP Connection
set tcp [new Agent/TCP]
$tcp set packetSize_ 1000
$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink

set ftp [new Application/FTP]
$ftp attach-agent $tcp

# UDP Connection
set udp [new Agent/UDP]
$ns attach-agent $n2 $udp

set null [new Agent/Null]
$ns attach-agent $n1 $null
$ns connect $udp $null

set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 512
$cbr set interval_ 0.02
$cbr attach-agent $udp

# Schedule events
$ns at 0.5 "$ftp start"
$ns at 1.0 "$cbr start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
$ns at 5.0 "finish"

# Run simulation
$ns run
```

Simulation Output

NAM Output

- Ethernet LAN displayed with shared medium
- TCP packets show acknowledgements
- UDP packets transmitted without reliability

- Packet drops may be observed due to collisions

Trace File Output (Sample)

```
+ 0.52 0 1 tcp 1000 ------- 0 0.0 1.0 1
r 0.54 1 0 ack 40 ------- 0 0.0 1.0 1
+ 1.01 2 1 cbr 512 ------- 0 0.0 2.0 1
d 1.03 1 2 cbr 512 ------- 0 0.0 2.0 1
```

Result

An Ethernet wired LAN based on **IEEE 802.3** was successfully simulated using NS-2, and packet transmission between nodes was observed.

EXPT 8: Implementation of Error Detection / Error Correction Techniques.

AIM:

To develop a program to study and implement the error correction and error detection using *CRC-CCITT (16bit).*

SOFTWARE REQUIRED:

SYSTEM with MATLAB software

ALGORITHM:

1. Multiply M(x) by highest power in G(x). i.e. Add So much zeros to M(x).
2. Divide the result by G(x). The remainder = C(x).
3. If: x div y gives remainder c that means: x = n y + c Hence (x-c) = n y (x-c) div y gives remainder 0Here (x-c) = (x+c) Hence (x+c) div y gives remainder 0
4. Transmit: T(x) = M(x) + C(x)
5. Receiver end: Receive T(x). Divide by G(x), should have remainder 0.

THEORY :

Whenever digital data is stored or interfaced, data corruption might occur. Since the beginning of computer science, developers have been thinking of ways to deal with this type of problem. For serial data they came up with the solution to attach a parity bit to each sent byte. This simple detection mechanism works if an odd number of bits in a byte changes, but an even number of false bits in one byte will not be detected by the parity check. To overcome this problem developers have searched for mathematical sound mechanisms to detect multiple false bits. The **CRC** calculation or *cyclic redundancy check* was the result of this. Nowadays CRC calculations are used in all types of communications. All packets sent over a network connection are checked with a CRC.

The idea behind CRC calculation is to look at the data as one large binary number. This number is divided by a certain value and the remainder of the calculation is called the CRC. Dividing in the CRC calculation at first looks to cost a lot of computing power, but it can be performed very quickly if we use a method similar to the one learned at school. Example: (Write in the left side of the observation/Record) We will as an example calculate the

```
                1 0 1 = 5
           -------------
1 0 0 1 1 / 1 1 0 1 1 0 1
            1 0 0 1 1 | |
            --------- | |
              1 0 0 0 0 |
              0 0 0 0 0 |
              --------- |
              1 0 0 0 0 1
                1 0 0 1 1
                ---------
                  1 1 1 0 = 14 = remainder
```

remainder for the character 'm'—which is 1101101 in binary notation—by dividing it by 19 or 10011. Please note that 19 is an odd number. This is necessary as we will see further on. Please refer to your schoolbooks as the binary calculation method here is not very different

from the decimal method you learned when you were young. It might only look a little bit strange. Also notations differ between countries, but the method is similar

Basic mathematical process:

The message bits are appended with *c* zero bits; this *augmented message* is the dividend

- A predetermined *c+1*-bit binary sequence, called the *generator polynomial*, is the divisor
- The checksum is the *c*-bit remainder that results from the division operation

Table 1 lists some of the most commonly used generator polynomials for 16- and 32-bit CRCs. Remember that the width of the divisor is always one bit wider than the remainder. So, for example, you'd use a 17-bit generator polynomial whenever a 16-bit checksum is required.

	CRC-CCITT	CRC-16	CRC-32
Checksum Width	16 bits	16 bits	32 bits
Generator Polynomial	1000100000010001	1100000000000101	10000010011000001000111011010111

Table 1. International Standard CRC Polynomials

Error detection with CRC

Consider a message represented by the polynomial M(x)

Consider a *generating polynomial* G(x) This is used to generate a CRC = C(x) to be appended to M(x).Note this G(x) is prime.

Steps:

1. Multiply M(x) by highest power in G(x). i.e. Add So much zeros to M(x).
2. Divide the result by G(x). The remainder = C(x). Special case: This won't work if bitstring =all zeros. We don't allow such an M(x).But M(x) bitstring = 1 will work, for example. Can divide 1101into 1000.
3. If: x div y gives remainder c that means: x = n y + c Hence (x-c) = n y (x-c) div y gives remainder 0Here (x-c) = (x+c) Hence (x+c) div y gives remainder 0
4. Transmit: T(x) = M(x) + C(x)
5. Receiver end: Receive T(x). Divide by G(x), should have remainder 0.

Note if G(x) has order n - highest power is xn, then G(x) will cover (n+1) bits and the *remainder* **will cover n bits. i.e. Add n bits (Zeros) to message.**

Source Code:

```
#include<stdio.h>
int a[100],b[100],i,j,len,k,count=0;
//Generator Polynomial:g(x)=x^16+x^12+x^5+1
int gp[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1,};
int main()
{
void div(); system("clear");
printf("\nEnter the length of Data
Frame :");scanf("%d",&len);
printf("\nEnter the
```

```
    Message                    :");
    for(i=0;i<len;i++)
    scanf("%d",&a[i]);
//Append r(16) degree Zeros to Msg bits
    for(i=0;i<16;i
    ++)
    a[len++]=0;
//Xr.M(x) (ie. Msg+16 Zeros)
    for(i=0;i<len;i++)
    b[i]=a[i];
//No of times to be divided ie. Msg Length
    k=len-16;div();
    for(i=0;i<len;i++)
    b[i]=b[i]^a[i];       //MOD       2
    Substraction  printf("\nData   to
    be       transmitted       :       ");
    for(i=0;i<len;i++)
    printf("%2d",b[i]);
    printf("\n\nEnter    the    Reveived
    Data : ");for(i=0;i<len;i++)
    scanf("%d",&a
    [i]);          div();
    for(i=0;i<len;i+
    +) if(a[i]!=0)
    {
    printf("\nERROR   in   Recived
    Data");return 0;
    }
    printf("\nData Recived is ERROR FREE");
    }
    void div()
    {
    for(i=0;i<k;i++)
    {
    if(a[i]==gp[0])
    {
    for(j=i;j<17+i;j++)
    a[j]=a[j]^gp[count
    ++];
    }
    count=0;
    }
    }
```

Output:

```
    [root@localhost   ]#   cc
    prg1.c   [root@localhost
    ]# ./a.out
```

Enter the length of Data
Frame :4 Enter the Message
:1 0 1 1
Data to be transmitted : 1 0 1 1 1 0 1 1 0 0 0 1 0 1 1 0 1 0 1 1
Enter the Reveived Data : 1 0 1 1 1 0 1 1 0 0 0 0 0 1 1 0
1 0 1 1ERROR in Recived Data
Reminder is : 0000000100000000

RESULT: Thus the error correction and error detection using *CRC-CCITT (16bit)* was implemented.

INTRODUCTION TO NETWORK SIMULATOR 2

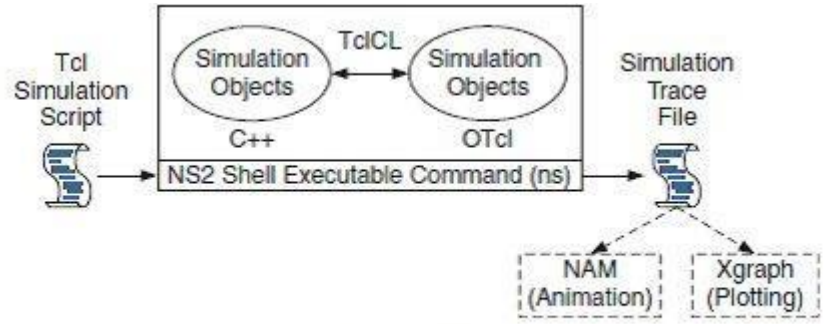
NS2 stands for Network Simulator Version 2. It is an open-source event-driven simulator designed specifically for research in computer communication networks.

Features of NS2

1. It is a discrete event simulator for networking research.
2. It provides substantial support to simulate bunch of protocols like TCP, FTP, UDP, https and DSR.
3. It simulates wired and wireless network.
4. It is primarily Unix based.
5. Uses TCL as its scripting language.
6. Otcl: Object oriented support
7. Tclcl: C++ and otcl linkage
8. Discrete event scheduler

Basic Architecture

NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation objects, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events. The C++ and the OTcl are linked together using TclCL



Basic architecture of NS.

- Nam (**Network Animator**) is an animation tool to graphically represent the network andpacket traces.
- **UDP** (User Datagram Protocol) is an alternative communications protocol to TransmissionControl Protocol (TCP) used primarily for establishing low-latency and loss-tolerating connections between applications on the internet.

Some basic Otcl script syntax :

- Basic Command :

```
set a 8

set b [expr $a/8]
```

In the first line, the variable **a** is assigned the value 8. In the second line, the result of the command [expr $a/8], which equals 1, is then used as an argument to another command, which in turn assigns a value to the variable **b**. The "$" sign is used to obtain a value contained in a variable and square brackets are an indication of a command substitution.

- Define new procedures with *proc* command

```
proc      factorial
    fact    {    if
    {$fact <= 1}
    {
       return 1
    }
expr $fact * [factorial [expr $fact-1]]
}
```

- To open a file for reading :

```
set testfile [open hello.dat r]
```

Similarly, **put** command is used to write data into

```
the fileset testfile [open hello.dat w]
puts $testfile "hello1"
```

- To call sub processes within another process, **exec** is used, which creates a sub process and waits for it to complete.

```
exec rm $testfile
```

- To be able to run a simulation scenario, a network topology must first be created. In ns2, the topology consists of a collection of nodes and links.

```
set ns [new Simulator]
```

- The simulator object has member functions which enables to create the nodes and define the links between them. The class simulator contains all the basic functions. Since ns was defined to handle the Simulator object, the command $ns is used for using the functions belonging to the simulator class.
- In the network topology nodes can be added in the following

```
manner :set n0 [$ns node]
set n1 [$ns node]
```

- Traffic agents (TCP, UDP etc.) and traffic sources (FTP, CBR etc.) must be set up if the node is not a router. It enables to create CBR traffic source using UDP as transport protocol or an FTP traffic source using TCP as a transport protocol.

CBR traffic source using UDP :

```
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
```

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packet_size_ 512
```

FTP traffic source using TCP :

```
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$tcp0 set packet_size_ 512
```

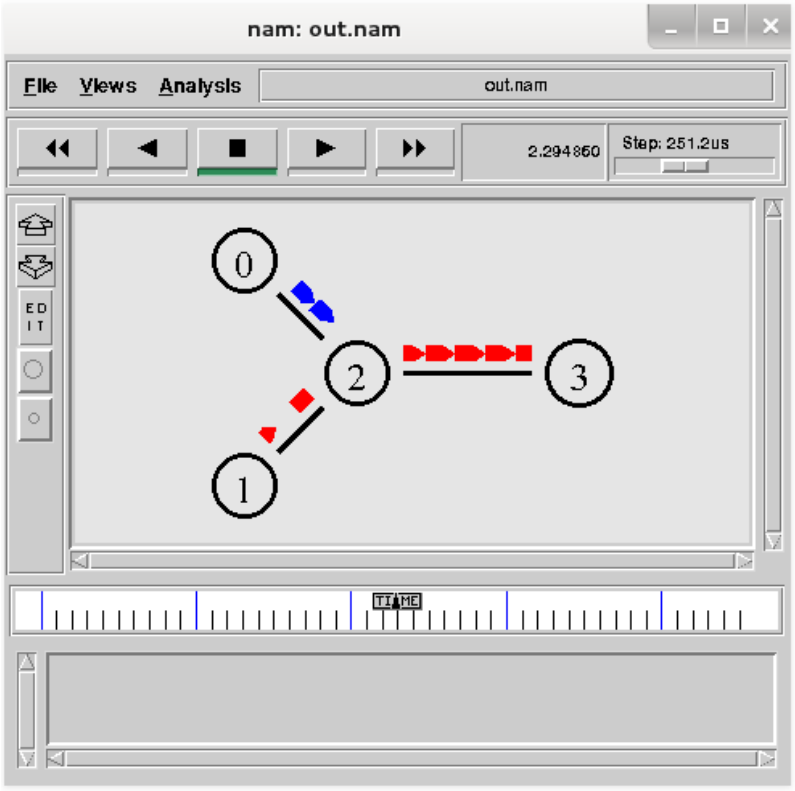
Below is the implementation of creating links between the source and destination using both ftpand tcp :

```
# Create a simulator
object set ns [new
Simulator]

# Define different colors
# for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

# Open the NAM trace
file set nf [open out.nam
w]
$ns namtrace-all $nf

# Define a 'finish'
procedureproc finish {}
{
  global ns nf
  $ns flush-trace

  # Close the NAM trace
  fileclose $nf

  # Execute NAM on the
  trace file exec nam out.nam
  &
```

```
   exit 0
}

# Create four
nodes set n0
[$ns node] set
n1 [$ns node]
set n2 [$ns
node]
set n3 [$ns node]
# Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

# Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

# Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

# Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5

# Setup a TCP
connection set tcp
[new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

# Setup a FTP over TCP
connection set ftp [new
Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
```

```
# Setup a UDP
connection set udp
[new Agent/UDP]
$ns attach-agent $n1
$udp set null [new
Agent/Null]

$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

# Setup a CBR over UDP
connection set cbr [new
Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false

# Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

# Detach tcp and sink
agents # (not really
necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"
# Call the finish procedure
after # 5 seconds of
simulation time
$ns at 5.0 "finish"

# Print CBR packet size and interval
puts "CBR packet size = [$cbr set
packet_size_]" puts "CBR interval = [$cbr
set interval_]"

# Run the simulation
$ns run
```

Output :


nam: out.nam
File
Views
Analysis
out.nam
2.294860
Step: 251.2us
EDIT
0
1
2
3
TIME

EXPT:9 Study the performance of network with CSMA / CA protocol.

AIM : To create a scenario and study the performance of network with CSMA protocol

THEORY:

Carrier Sense Multiple Access (CSMA) is a network protocol that listens to or senses network signals on the carrier/medium before transmitting any data. CSMA is implemented in Ethernet networks with more than one computer or network device attached to it. CSMA is part of the Media Access Control (MAC) protocol. CSMA works on the principle that only one device can transmit signals on the network, otherwise a collision will occur resulting in the loss of data packets or frames. CSMA works when a device needs to initiate or transfer data over the network. Before transferring, each CSMA must check or listen to the network for any other transmissions that may be in progress. If it senses a transmission, the device will wait for it to end. Once the transmission is completed, the waiting device can transmit its data/signals. However, if multiple devices access it simultaneously and a collision occurs, they both have to wait for a specific time before reinitiating the transmission process.

CSMA protocol was developed to overcome the problem found in ALOHA i.e. to minimize the chances of collision, so as to improve the performance. CSMA protocol is based on the principle of 'carrier sense'. The station senses the carrier or channel before transmitting a frame. It means the station checks the state of channel, whether it is idle or busy.

Even though devices attempt to sense whether the network is in use, there is a good chance that two stations will attempt to access it at the same time. On large networks, the transmission time between one end of the cable and another is enough that one station may access the cable even though another has already just accessed it.

The chances of collision still exist because of propagation delay. The frame transmitted by one station takes some time to reach other stations. In the meantime, other stations may sense the channel to be idleand transmit their frames. This results in the collision.

CSMA with collision detection:

CSMA/CD is used to improve CSMA performance by terminating transmission as soon as a collision is detected, thus shortening the time required before a retry can be attempted\

CSMA with Collision detection:

CSMA/CA collision avoidance is used to improve the performance of CSMA. If the transmission medium is sensed busy before transmission, then the transmission is deferred for a random interval. Thisrandom interval reduces the likelihood that two or more nodes waiting to transmit simultaneously begin transmission upon termination of the detected transmission, thus reducing the incidence of collision

ALGORITHM:

Step 1 : Start.

Step 2 : Create a simulator object.

Step 3 : Configure the simulator to use Link state routing.

Step 4 : Open the nam trace file.
Step 5 : Open the output file.
Step 6 : Define the finish procedure.
Step 7 : Close the trace file.
Step 8 : Create the required nodes.
Step 10 : Create links between the nodes.
Step 11 : Create a UDP agent to attach the node. **Step 12 :** Create a CBR traffic router and attach. **Step 13 :** Create a Null agent to the traffic sink. **Step 14 :** Connect the traffic source to the sink. **Step 15 :** Schedule the events for CBR agent.
Step 16 : Stop.

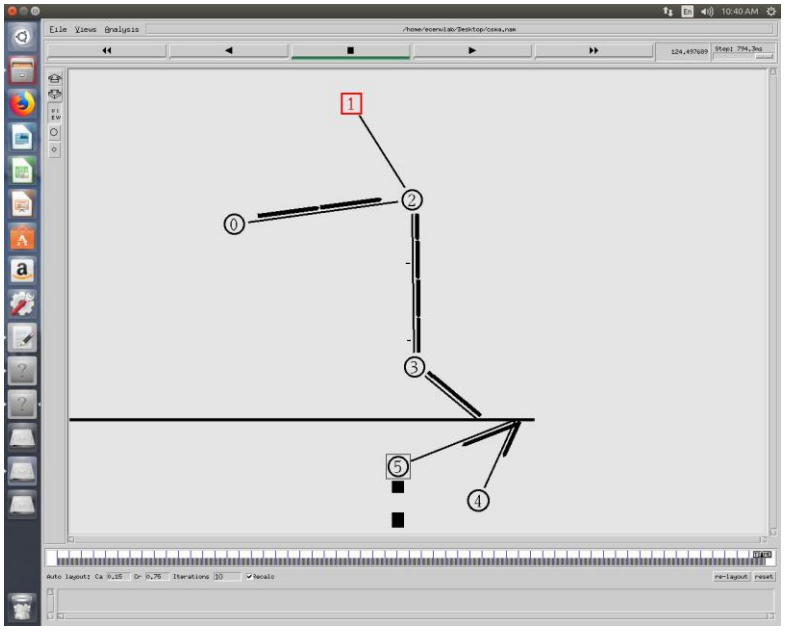
CSMA Protocol

```
set ns [new Simulator]

#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the
Trace files set
file1 [open
out.tr w]
set winfile [open WinFile w]
$ns trace-all $file1

#Open the NAM
trace file set file2
[open out.nam w]
$ns namtrace-all $file2

#Define a 'finish'
procedure proc
finish {} {
    global ns file1 file2
    $ns
    flush-
    trace
    close
    $file1
    close
    $file2
    exec nam
    out.nam & exit
```

```
    0
}

#Create
six   nodes
set n0 [$ns
node]   set
n1      [$ns
node]   set
n2      [$ns
node]   set
n3      [$ns
node]   set
n4      [$ns
node]   set
n5      [$ns
node]

$n1 color red
$n1 shape box

$n5 color red
$n5 shape box
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail




set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail

MAC/Csma/Cd Channel]#Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 80
$tcp set packetSize_ 5

#Setup a FTP over TCP
connection set ftp [new
Application/FTP]
$ftp attach-agent $tcp
```

```
$ftp set type_ FTP

#Setup a UDP
connection set udp
[new Agent/UDP]
$ns attach-agent
$n1 $udp set null
[new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP
connection set cbr [new
Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 100
$cbr set rate_ 0.01mb
$cbr set random_ false

$ns at 0.1 "$cbr start"
$ns at 2.0 "$ftp start"
$ns at 3.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
$ns at 5.0 "finish"
$ns run
```

OUTPUT:



RESULT:

Thus the study on CSMA protocol is carried out and implemented using NS2

EXPT: 10 Implementation of Flow control Algorithms.

AIM:

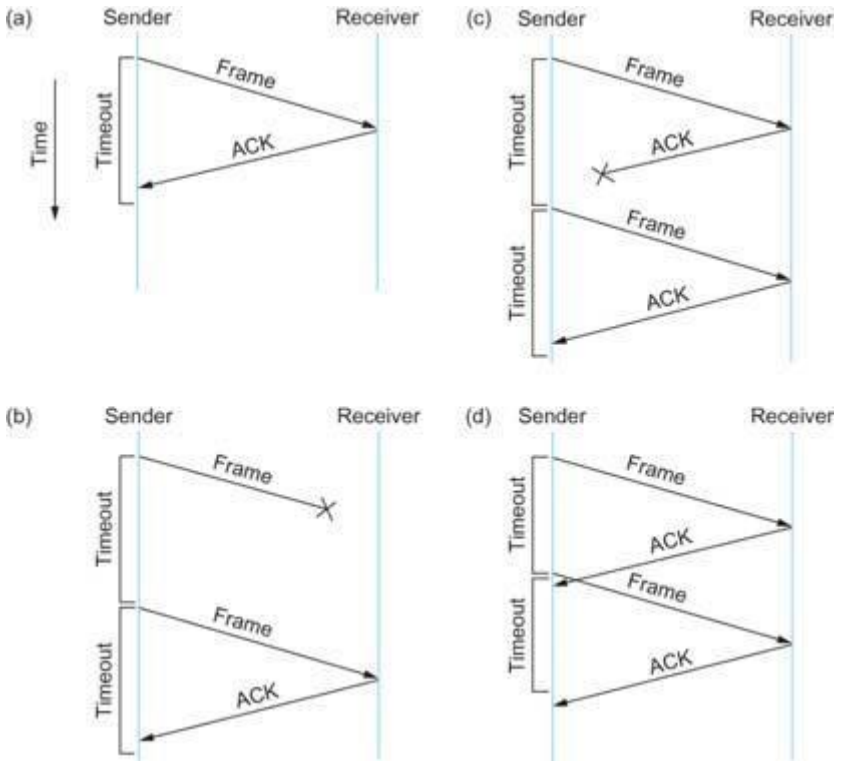
To write a C program to demonstrate the working of simple Stop and Wait Protocol.

THEORY:

The **Stop-and-Wait protocol** uses both flow and error control. In this protocol, the sender sends one frame at a time and waits for an acknowledgment before sending the next one. To detect corrupted frames, we need to add a CRC to each data frame. When a frame arrives at the receiver site, it is checked. If its CRC is incorrect, the frame is corrupted and silently discarded. The silence of the receiver is a signal for the sender that a frame was either corrupted or lost.

Every time the sender sends a frame, it starts a timer. If an acknowledgment arrives before the timer expires, the timer is stopped and the sender sends the next frame (if it has one to send). If the timer expires, the sender resends the previous frame, assuming that the frame was either lost or corrupted. This means that the sender needs to keep a copy of the frame until its acknowledgment arrives. When the corresponding acknowledgment arrives, the sender discards the copy and sends the next frame if it is ready. Only one frame and one acknowledgment can be in the channels at any time.

The advantages are Limited buffer size, sooner Errors detection and prevents one station occupying medium for long periods.



ALGORITHM:

(i) *Start Program*

1. Start
2. Declare the required variables and file pointers.
3. Get the choice from the user if he needs a Selective Retransmission or Go-Back-N protocol.
4. Select "1" for Sending the data and "2" for the end of transmission
5. For sending the data open a text file in write mode and write the data that has to be sent.
6. Once written close the file.
7. Check the ack.txt file in which the acknowledgement from the receiver is stored.
8. If the acknowledgment is positive, then send the data to the receiver.
9. If all the data are sent, then select Option "2" to end the transmission.

(ii) *Stop Program*

1. Start
2. Declare the required variables and file pointers.
3. For receiving the data, open a data.txt text file in read mode and read the data that was sent bythe receiver.
4. Open the ack.txt file in write mode and write as "Yes" if received correctly.
5. Else, write as 'No" in the ack.txt file.
6. Close the file.

Sender Module

```
#include<stdio.h>  #include<stdlib.h> #include<conio.h> #include<math.h>  #include<string.h>
void main()
{
int ch,strt=1;
char
input[20],ack[
4];
FILE*in,*ak;
clrscr();
printf("\nStop          and          Wait
Protocol\n");
printf("\n1.Send\n2.End                Of
Transmission\n");strcpy(ack,"yes");
while(1)
{
if(strt|=1)
printf("\nEnter       your
choice...");
scanf("%d",&ch);
switch(ch)
{
case 1:
if (strt|=1)
{
```

```
ak=fopen("ack.txt","r");
fscanf(ak,"%
s",ack);
fclose(ak);
}
if(((strcmp(ack,"yes"))==0)||(strt==1))
{
in=fopen("data.txt",
"w");
printf("\nEnter    the
Data...");
scanf("%s",input);
fprintf(in,"%s",i
nput);
fclose(in);
printf("\nData
Sent\n");strt=0;
}
else
{exit(0);}break; case 2:
{exit(0);break;}
}

getch();
}
}
```

Receiver Module

```
#include<st
dio.h>
#include<st
dlib.h>
#include<st
ring.h>
#include<c
onio.h>
void main()
{
int
one,choice,frst=
1;              char
output[20],akstr[
4];
FILE*out,*ak1;
clrscr();while(1)
{
if(frst==0)
```

```
{
out=fopen("data.tx
t","r");
fscanf(out,"%s",ou
tput);
printf("\nReceived
data:");
printf("%s",output
);
}
fclose(out);
printf("\n1.Yes\n2.No");
printf("\nDo you want to Acknowledge the
Data?\t");scanf("%d",&choice);
frst=0; if(choice==1)
  {
  strcpy(akstr,"yes");
  }
else
  {
  strcpy(akstr,"no");
  }
ak1=fopen("ack.tx
t","w");
fprintf(ak1,"%s",a
kstr); fclose(ak1);
   if(choice==2)

   break;

}
}
```

RESULT: Thus the working of simple error control functionality was implemented using the simple Stop and wait protocol in C language.

EXPT :11 Simulation and performance evaluation of a network using TCP and UDP.

Aim

To study and compare the performance of TCP and UDP protocols in a network using Cisco Packet Tracer simulation.

Requirements

1. Software:
o Cisco Packet Tracer (Network Simulator)
2. Hardware:
o PCs, switches, and routers (simulated in Cisco Packet Tracer)
3. Knowledge:
o Basics of TCP and UDP protocols
o Configuration of IP addresses and devices in a network.

Theory

TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) are two primary transport layer protocols.

1. TCP (Transmission Control Protocol):
o A connection-oriented protocol that ensures reliable data delivery.
o Uses mechanisms like error checking, retransmission, and acknowledgment (ACK).
o Best suited for applications requiring high reliability (e.g., file transfer, emails).

2. UDP (User Datagram Protocol):
o A connectionless protocol offering faster data transfer but without reliability guarantees.
o Suitable for time-sensitive applications like video streaming or gaming.

Comparison:
TCP ensures reliable data transfer with higher latency, while UDP offers faster but less reliable data transfer. This simulation demonstrates these differences.

Observations

Parameter	TCP	UDP
Reliability	Ensures reliable delivery	No reliability mechanism
Packet Delivery	Retransmits lost packets	Drops lost packets
Latency	Higher due to ACKs and retransmissions	Lower latency
Applications	File Transfer, Web Browsing	Video Streaming, Gaming

```
# Create Simulator Object
set ns [new Simulator]

# Define trace files
set tracefile [open out.tr w]
set namfile [open out.nam w]
$ns trace-all $tracefile
$ns namtrace-all $namfile

# Finish procedure
proc finish {} {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam out.nam &
   exit 0
}

# Create nodes
set n0 [$ns node]   ;# TCP Source
set n1 [$ns node]   ;# Router
set n2 [$ns node]   ;# TCP Destination
set n3 [$ns node]   ;# UDP Source
set n4 [$ns node]   ;# UDP Destination

# Create links
$ns duplex-link $n0 $n1 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n3 $n1 2Mb 10ms DropTail
$ns duplex-link $n1 $n4 2Mb 10ms DropTail
```

```
# TCP configuration
set tcp [new Agent/TCP]
$tcp set packetSize_ 1000
$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $n2 $sink
$ns connect $tcp $sink

set ftp [new Application/FTP]
$ftp attach-agent $tcp

# UDP configuration
set udp [new Agent/UDP]
$ns attach-agent $n3 $udp

set null [new Agent/Null]
$ns attach-agent $n4 $null
$ns connect $udp $null

set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 512
$cbr set interval_ 0.01
$cbr attach-agent $udp

# Start and Stop simulation
$ns at 0.5 "$ftp start"
$ns at 1.0 "$cbr start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
$ns at 5.0 "finish"

# Run simulation
$ns run
```

Result

The performance of TCP and UDP protocols was successfully studied and analyzed using Cisco Packet Tracer. The observations highlight the trade-offs between reliability and speed in TCP and UDP.

EXPT:12 QoS Analysis of Wireless Networks

Aim

To analyze the **Quality of Service (QoS)** parameters of a wireless network such as **throughput, delay, packet loss, and jitter** using **NS-2 / Cisco Packet Tracer**.

Software / Tools Required

- **NS-2/**Ubuntu/Linux/**Cisco Packet Tracer**
- Computer System

Theory

Quality of Service (QoS):

QoS refers to the ability of a network to provide **better service to selected traffic** by controlling network resources.

Key QoS Parameters:

- **Throughput:** Rate of successful data delivery
- **End-to-End Delay:** Time taken for a packet to reach destination
- **Packet Loss:** Number of packets dropped
- **Jitter:** Variation in packet delay

QoS in Wireless Networks:

Wireless networks suffer from:

- Limited bandwidth
- Interference
- Mobility

  Hence QoS analysis is essential for **real-time applications** such as VoIP and video streaming.

QoS Analysis using NS-2

Network Topology

- Wireless nodes
- One TCP flow
- One UDP flow
- IEEE 802.11 MAC

NS-2 TCL Code

```
# Create Simulator
set ns [new Simulator]

# Trace files
set tracefile [open qos.tr w]
set namfile [open qos.nam w]
$ns trace-all $tracefile
$ns namtrace-all-wireless $namfile 500 500

# Finish procedure
proc finish {} {
   global ns tracefile namfile
```

```
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam qos.nam &
    exit 0
}

# Wireless parameters
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(ifqlen) 50
set val(nn) 4
set val(rp) DSDV

# Topography
set topo [new Topography]
$topo load_flatgrid 500 500

create-god $val(nn)

# Node configuration
$ns node-config \
    -adhocRouting $val(rp) \
    -llType $val(ll) \
    -macType $val(mac) \
    -ifqType $val(ifq) \
    -ifqLen $val(ifqlen) \
    -antType $val(ant) \
    -propType $val(prop) \
    -phyType $val(netif) \
    -channelType $val(chan) \
    -topoInstance $topo \
    -agentTrace ON \
    -routerTrace ON

# Create nodes
for {set i 0} {$i < $val(nn)} {incr i} {
    set node($i) [$ns node]
    $node($i) set X_ [expr 100*$i]
    $node($i) set Y_ 100
    $node($i) set Z_ 0
}
```

```
# TCP flow
set tcp [new Agent/TCP]
$ns attach-agent $node(0) $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $node(2) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp

# UDP flow
set udp [new Agent/UDP]
$ns attach-agent $node(1) $udp
set null [new Agent/Null]
$ns attach-agent $node(3) $null
$ns connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr set packetSize_ 512
$cbr set interval_ 0.02
$cbr attach-agent $udp

# Schedule events
$ns at 0.5 "$ftp start"
$ns at 1.0 "$cbr start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"
$ns at 5.0 "finish"

# Run
$ns run
```

Sample Output (NS-2)

QoS Parameter	TCP	UDP
Throughput	Moderate	High
Delay	High	Low
Packet Loss	Very Low	High
Jitter	Low	High

Result

The QoS parameters of the wireless network were analyzed successfully. TCP provided reliable transmission with higher delay, while UDP achieved lower delay with packet loss.

EXPT:13 **Network Topology – Star, Bus, Ring, Mesh**

AIM:

To build and simulate a network under different topologies like Star,bus, mesh and ring, andobserve the performance parameters.

REQUIREMENTS:

Operating System	:	Windows NT/2000/XP or
LINUXProgramming Tool	:	Network Simulator (NS2)

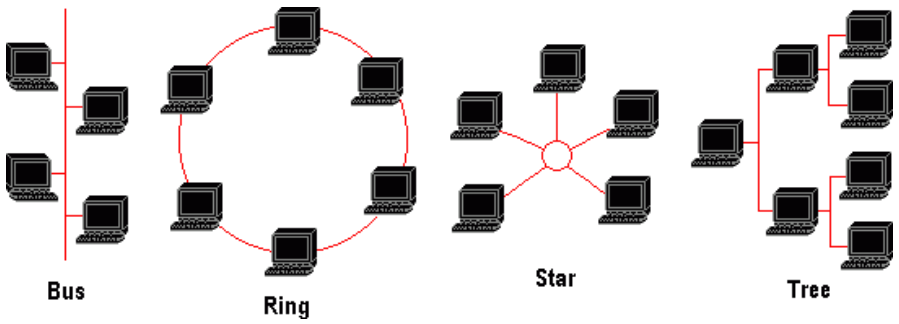
THEORY:

Network topology refers to the physical or logical layout of a network. It defines the way different nodes are placed and interconnected with each other. Alternately, network topology may describe howthe data is transferred between these nodes.

There are two types of network topologies: physical and logical. Physical topology emphasizes the physical layout of the connected devices and nodes, while the logical topology focuses on the patternof data transfer between network nodes.

The physical and logical network topologies of a network do not necessarily have to be identical. However, both physical and network topologies can be categorized into five basic models:

- **Bus Topology**: All the devices/nodes are connected sequentially to the same backbone or transmission line. This is a simple, low-cost topology, but its single point of failure presents arisk.
- **Star Topology**: All the nodes in the network are connected to a central device like a hub or switch via cables. Failure of individual nodes or cables does not necessarily create downtime inthe network but the failure of a central device can. This topology is the most preferred and popular model.
- **Ring Topology**: All network devices are connected sequentially to a backbone as in bus topology except that the backbone ends at the starting node, forming a ring. Ring topology shares many of bus topology's disadvantages so its use is limited to networks that demand highthroughput.
- **Tree Topology**: A root node is connected to two or more sub-level nodes, which themselves are connected hierarchically to sub-level nodes. Physically, the tree topology is similar to bus and star topologies; the network backbone may have a bus topology, while the low-level nodesconnect using star topology.
- **Mesh Topology**: The topology in each node is directly connected to some or all the other nodes present in the network. This redundancy makes the network highly fault tolerant but theescalated costs may limit this topology to highly critical networks.



PROCEDURE:

1. Open a terminal and type the TCL file in the vim editor with the command "vifilename.tcl" and save it.
2. Run the saved file with the command "ns filename.tcl".
3. If any errors, edit them in the vim editor and rerun it again.
4. The output is displayed in the nam console.

STAR TOPOLOGY

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n0 shape square
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 10ms DropTail
$ns duplex-link $n0 $n4 1Mb 10ms DropTail
$ns duplex-link $n0 $n5 1Mb 10ms DropTail
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
$ns connect $tcp0 $sink0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
```

```
$ns run
```

BUS TOPOLOGY

```
#Create a simulator object
set ns [new Simulator]

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the trace file
    close $nf
    #Executenam on the trace file
    exec nam out.nam &
    exit 0
}
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set lan0 [$ns newLan "$n0 $n1 $n2 $n3 $n4 $n5" 0.5Mb 80ms LL Queue/DropTail MAC/Csma/Cd
Channel]

#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0

# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 5
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
```

```
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

RING TOPOLOGY

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the trace file
    close $nf
    #Executenam on the trace file
    exec nam out.nam &
    exit 0
}
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n4 $n5 1Mb 10ms DropTail
$ns duplex-link $n5 $n0 1Mb 10ms DropTail

#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0
# Create a CBR traffic source and attach it to tcp0
```

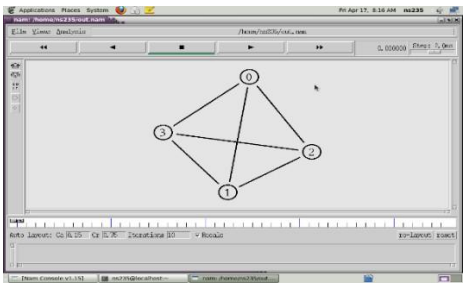
```
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Run the simulation
$ns run
```

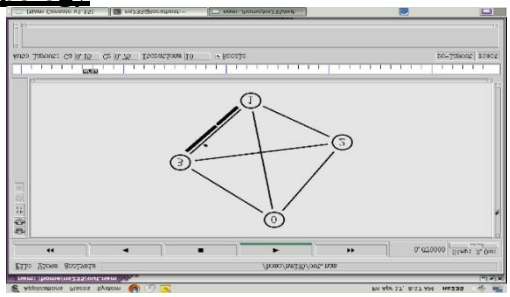
MESH TOPOLOGY

```
#Create a simulator object
set ns [new Simulator]

#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
    global ns nf
    $ns flush-trace
    #Close the trace file
    close $nf
    #Executenam on the trace file
    exec nam out.nam &
    exit 0
}
#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n0 $n3 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n3 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail

#Create a TCP agent and attach it to node n0
set tcp0 [new Agent/TCP]
```

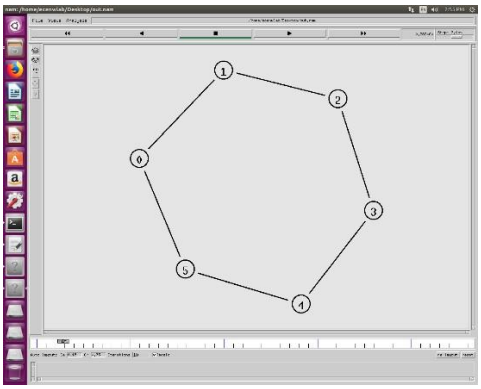
```
$tcp0 set class_ 1
$ns attach-agent $n1 $tcp0
#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3
set sink0 [new Agent/TCPSink]
$ns attach-agent $n3 $sink0
#Connect the traffic sources with the traffic sink
$ns connect $tcp0 $sink0

# Create a CBR traffic source and attach it to tcp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.01
$cbr0 attach-agent $tcp0
#Schedule events for the CBR agents
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"
#Run the simulation
$ns run
```

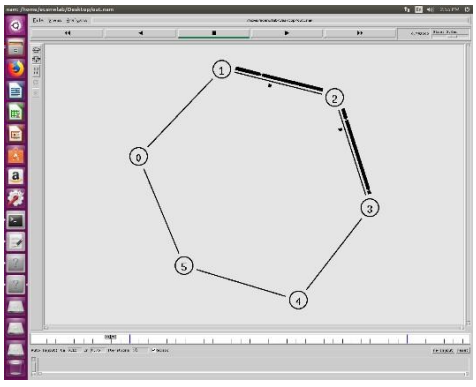
Mesh Topology



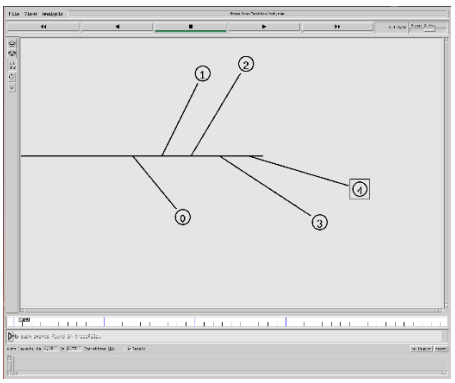


Ring Topology

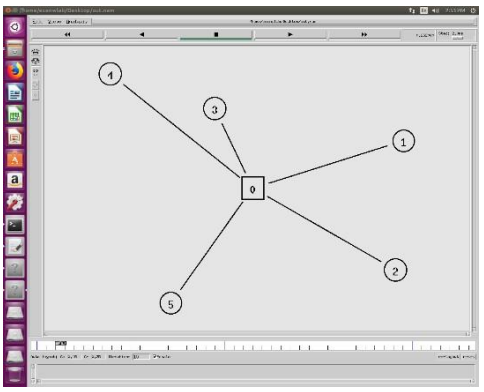




Bus Topology



Star Topology



RESULT:

Thus the network topologies like star, mesh, bus and ring have been implemented using network simulator.