1. CDMA

clc; clear;

c1 = [1, 1, 1, 1]; c2 = [1, -1, 1, -1]; c3 = [1, 1, -1, -1]; c4 = [1, -1, -1, 1];

disp('Enter the data bits:'); d1 = input('Enter D1: '); d2 = input('Enter D2: '); d3 = input('Enter D3: '); d4 = input('Enter D4: ');

r1 = c1 .\* d1; r2 = c2 .\* d2; r3 = c3 .\* d3; r4 = c4 .\* d4;

resultant\_channel = r1 + r2 + r3 + r4; disp('Resultant Channel:'); disp(resultant\_channel);

channel = input('Enter the station to listen for C1=1, C2=2, C3=3, C4=4: ');

if channel == 1 then rc = c1;

elseif channel == 2 then rc = c2;

elseif channel == 3 then rc = c3;

elseif channel == 4 then rc = c4;

end

inner\_product = resultant\_channel .\* rc; disp('Inner Product:'); disp(inner\_product);

data = sum(inner\_product) / length(inner\_product); disp('Data bit that was sent:');  
disp(data);

**2.BER.**

clc;

clear;

N = 10^6; // number of bits or symbols

// Transmitter

ip = rand(1, N) > 0.5; // generating 0,1 with equal probability

s = 2\*ip - 1; // BPSK modulation 0 -> -1; 1 -> 0

Eb\_N0\_dB = (-3:35); // multiple Eb/N0 values

for ii = 1:length(Eb\_N0\_dB)

n = (1/sqrt(2)) \* (rand(1, N) + %i\*rand(1, N)); // white Gaussian noise, 0dB variance

h = (1/sqrt(2)) \* (rand(1, N) + %i\*rand(1, N)); // Rayleigh channel

// Channel and noise Noise addition

y = h.\*s + 10^(-Eb\_N0\_dB(ii)/20)\*n;

// equalization

yHat = y./h;

// receiver - hard decision decoding

ipHat = real(yHat) > 0;

// counting the errors

nErr(ii) = sum(ip ~= ipHat);

end

simBer = nErr/N; // simulated ber

theoryBerAWGN = 0.5\*erfc(sqrt(10.^(Eb\_N0\_dB/10))); // theoretical ber

EbN0Lin = 10.^(Eb\_N0\_dB/10);

theoryBer = 0.5.\*(1-sqrt(EbN0Lin./(EbN0Lin+1)));

// plot

figure; // open a new figure window

plot(Eb\_N0\_dB, theoryBerAWGN, 'cd-', 'LineWidth', 2);

plot(Eb\_N0\_dB, theoryBer, 'r--', 'LineWidth', 2);

plot(Eb\_N0\_dB, simBer, 'bs-', 'LineWidth', 2);

xlabel('Eb/No, dB');

ylabel('Bit Error Rate');

legend('AWGN-Theory', 'Rayleigh-Theory', 'Rayleigh-Simulation');

title('BER for BPSK modulation in Rayleigh channel');

// Save figure to the PC

saveas(gcf, 'BER\_plot.png'); // Save the current figure as a PNG file

**3.DHCP**

**2811** Router

2950T Switch(x2)

3 comp and 1 laptop

Connection- (copper straight through cable- fast ethernet 0)

S1-r 0/0

S2-r 0/1

Router- CLI

1. n
2. en
3. conf t
4. int fa0/0
5. ip address 192.168.0.1 255.255.255.0
6. no shutdown
7. do write memory
8. ip dhcp pool net1
9. network 192.168.0.1 255.255.255.0
10. exit
11. int fa0/1
12. ip address 192.168.1.1 255.255.255.0
13. no shutdown
14. do write memory
15. ip dhcp pool net2
16. network 192.168.1.1 255.255.255.0
17. exit

Click laptop -> desktop-> ip config->Click DHCP (repeat for all 4)

**TCP Client Server**

**//Client**

import socket

IP = socket.gethostbyname(socket.gethostname())

PORT = 4456

ADDR = (IP, PORT)

FORMAT = "utf-8"

SIZE = 1024

def main():

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect(ADDR)

file = open("Data/yt.txt", "r")

data = file.read()

client.send("yt.txt".encode(FORMAT))

msg = client.recv(SIZE).decode(FORMAT)

print(f"[SERVER]: {msg}")

client.send(data.encode(FORMAT))

msg = client.recv(SIZE).decode(FORMAT)

print(f"[SERVER]: {msg}")

file.close()

client.close()

if \_name\_ == "\_main\_":

main()

**Make folder in VSCODE named Data then make file name yt.txt and add some text.**

1. **Server terminal python server.py**
2. **Client terminal- python client.py**

**//Server**

import socket

IP = socket.gethostbyname(socket.gethostname())

PORT = 4456

ADDR = (IP, PORT)

SIZE = 1024

FORMAT = "utf-8"

def main():

print("[STARTING] Server is starting.")

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.bind(ADDR)

server.listen()

print("[LISTENING] Server is listening.")

while True:

conn, addr = server.accept()

print(f"[NEW CONNECTION] {addr} connected.")

filename = conn.recv(SIZE).decode(FORMAT)

print(f"[RECV] Receiving the filename: {filename}")

conn.send("Filename received.".encode(FORMAT))

with open(filename, "w") as file:

while True:

data = conn.recv(SIZE).decode(FORMAT)

if not data:

break

file.write(data)

conn.send("Data received.".encode(FORMAT))

print(f"[FILE RECEIVED] File {filename} received from {addr}.")

conn.close()

print(f"[DISCONNECTED] {addr} disconnected.")

if \_\_name\_\_ == "\_\_main\_\_":

main()