Mobile Computing Lab File

INITC17

Submitted to:
Mr. Karan Gupta

Shrey Arora – 2021UIN3366

Mobile Computing Lab File

INITC17

Submitted to:
Mr. Karan Gupta

Ravneet Singh Oberoi – 2021UIN3365

Mobile Computing Lab File

INITC17

Submitted to:
Mr. Karan Gupta

Lakshya Bhardwaj – 2021UIN3335

Mobile Computing Lab File

INITC17

Submitted to:
Mr. Karan Gupta

Aryan Rohela – 2021UIN3361

Mobile Computing Lab File

INITC17

Submitted to:
Mr. Karan Gupta

Rachit Anand – 2021UIN3334

Index

1. Steps to install MatlabR2023b on Windows.
2. Program to Implement a Network of 10 Nodes in MATLAB
3. Implement a point-to-point network consisting of 10 nodes using MATLAB with duplex links between them. Initiate a communication between these nodes. Set the queue size, vary the bandwidth and find the number of packets dropped. Finally plot the graph showing the performance of this network in terms of number of packets dropped with varying bandwidth.
4. Implement FDMA, TDMA & and CDMA using MATLAB and show the results using a graph for 10 users using clustering techniques.
5. Implement GSM using MATLAB.
6. Implement GPRS using MAC layer in MATLAB.
7. Implement LTE in MATLAB.
8. Implement snooping and analyzing the traffic using Wireshark.

Practical 1

AIM- Steps to Install MATLAB R2023b on Windows

PROCEDURE-

Step 1: Obtain the MATLAB Installation File

- Visit the MathWorks website www.mathworks.com and log in using your MathWorks account. If you don't have an account, create one.
- After logging in, navigate to the MATLAB product page and locate the download link for MATLAB R2023b for Windows.

Step 2: Run the Installation File (matlab_R2023b_win64.exe)

- Find the downloaded installation file, typically named "matlab_R2023b_win64.exe," and double-click on it to start the installation process.

Step 3: Log in with Your MathWorks Account

- During the installation, you may be prompted to log in again with your MathWorks account. Use your college email ID associated with MathWorks, or the credentials provided by your college for MATLAB access.

Step 4: Accept the License Agreement

- Read and accept the MathWorks Software License Agreement to continue with the installation. Click the "Next" or "Agree" button to proceed.

Step 5: Select Your License

- In this step, you will be presented with license details. If no further configuration or selection is required, simply review the license details and click "Next" to continue with the installation.

Step 6: Select Destination Folder

- Choose the folder where you want MATLAB to be installed. The default installation path is usually recommended. Click "Next" to move on.

Step 7: Select Products to Install

- In this step, you can select which MATLAB products you want to install. For the initial installation, it's fine to select "MATLAB" and "Simulink." You can always install additional toolboxes later if needed.

Step 8: Confirm and Begin Installation

- Review your selections to ensure they are correct. If everything looks good, click "Next" to start the installation process.

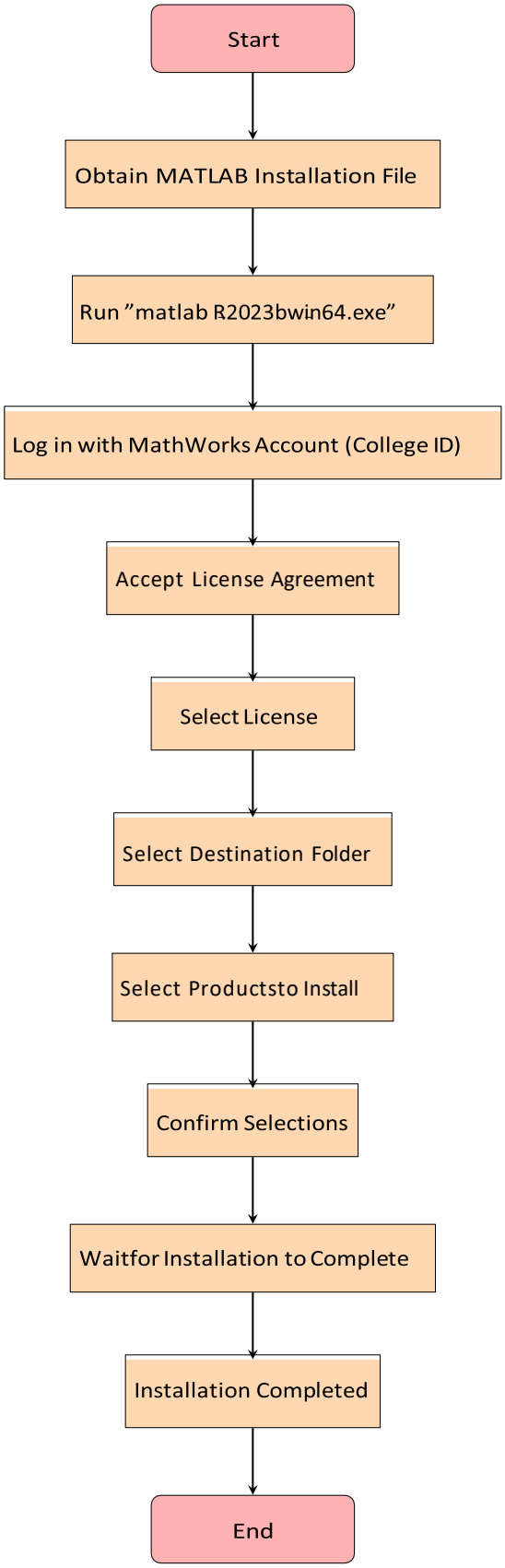
Step 9: Wait for Installation to Complete

- The installation process may take some time, depending on your computer's performance and the selected components. Be patient and wait for the installation to finish.

Step 10: Installation Completed

- Once the installation is complete, you will see a confirmation message. MATLAB R2023b is now installed on your Windows system.

MATLAB INSTALLATION STEPS



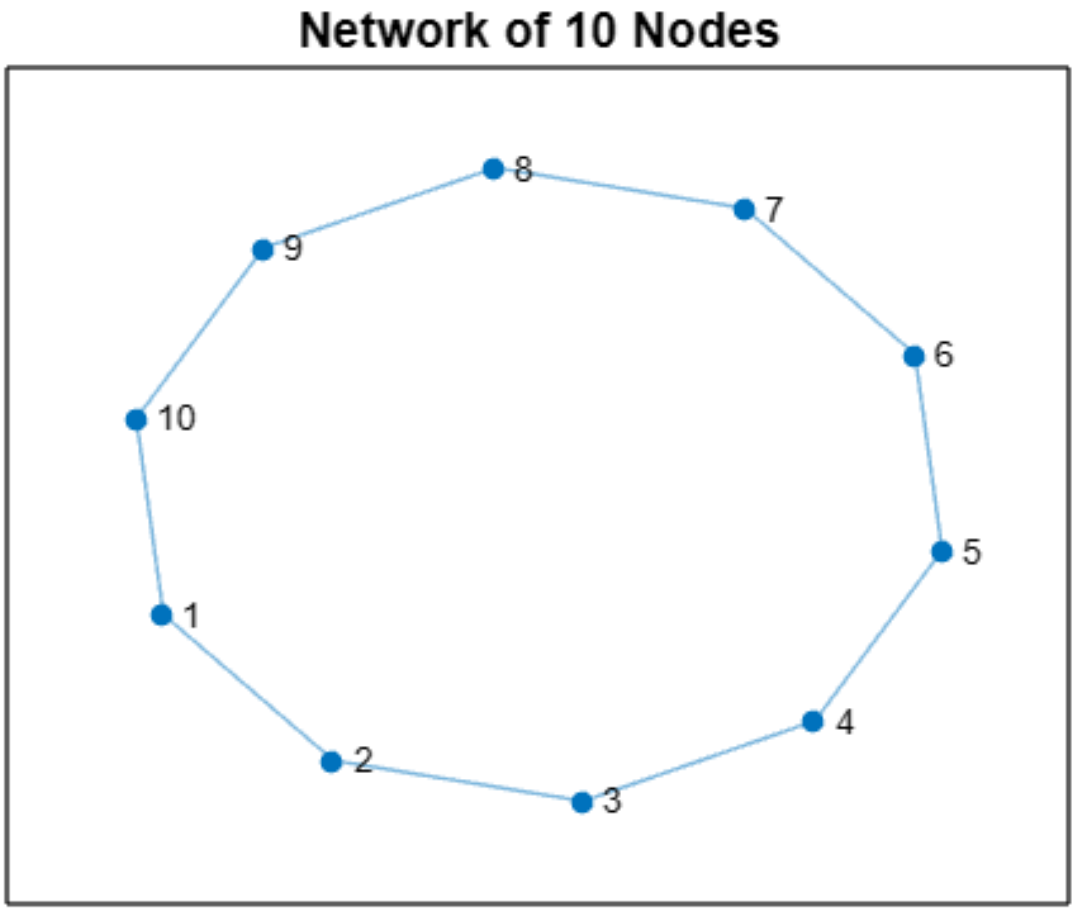
Practical 2

AIM- Program to Implement a Network of 10 Nodes in MATLAB

MATLAB Code:

```
numNodes = 10;
G = graph;
G = addnode(G, numNodes);
for i = 1:numNodes-1
G = addedge(G, i, i+1);
end
G = addedge(G , 1 , numNodes);
h = plot(G, 'Layout', 'force');
title('Network of 10 Nodes');
```

Output:



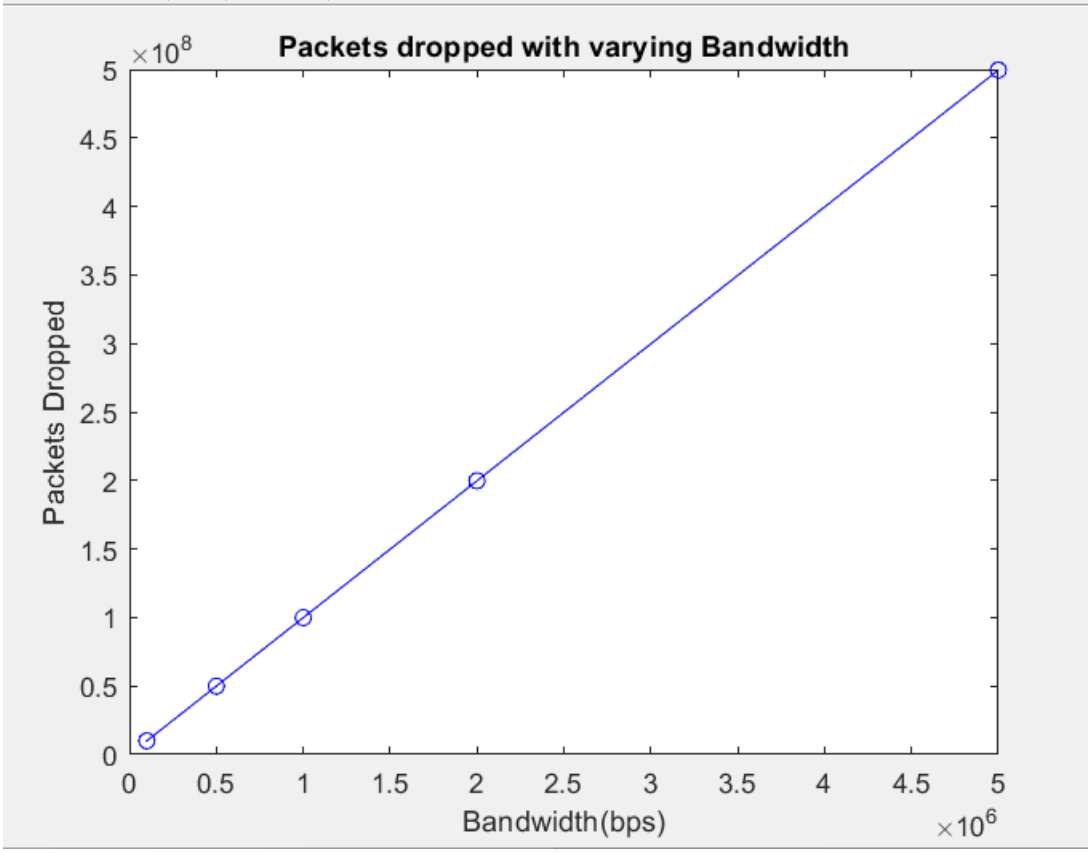
Practical 3

***AIM-*: Implement a point-to-point network consisting of 10 nodes using MATLAB with duplex links between them. Initiate a communication between these nodes.**
Set the queue size, vary the bandwidth and find the number of packets dropped. Finally plot the graph showing the performance of this network in terms of number of packets dropped with varying bandwidth

Matlab Code:

```
>> %Parameters
>> numNodes=10;
>> time=10; %Duration of communication(seconds);
>> queueSize=100;
>> %varying bandwidth values
>> bandwidths=[100000, 500000, 1000000, 2000000, 5000000];
>> %bits per second
>>
>> %transmission delay per packet
>> transmissionTimePerPacket=0.001; %(1 milliseconds)
>>
>> %initialize array to store packet drop value corresponding to each bandwidth value
>> packetsDropped=zeros(1,length(bandwidths));

>> %simulation for different values of bandwidth
>> for b=1:length(bandwidths)
       bandwidth=bandwidths(b);

       %simulation of communication between nodes
       for i=1:numNodes
           %Calculate number of packets based on bandwidth and time considering transimission delay
           numPackets=bandwidth*(time-transmissionTimePerPacket*(numNodes-1));

           %calculate packet drops based on queue size
           if numPackets>queueSize
              packetsDropped(b)=packetsDropped(b) + (numPackets-queueSize);
           end
       end
  end

>> %plotting the graph showing number of packets dropped with varying bandwidth
>> figure;
>> plot(bandwidths,packetsDropped,'bo-');
>> title('Packets dropped with varying Bandwidth');

>> xlabel('Bandwidth(bps)');
>> ylabel('Packets Dropped');
```

Output:


Packets dropped with varying Bandwidth
×10^8
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
Packets Dropped
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Bandwidth(bps)
×10^6

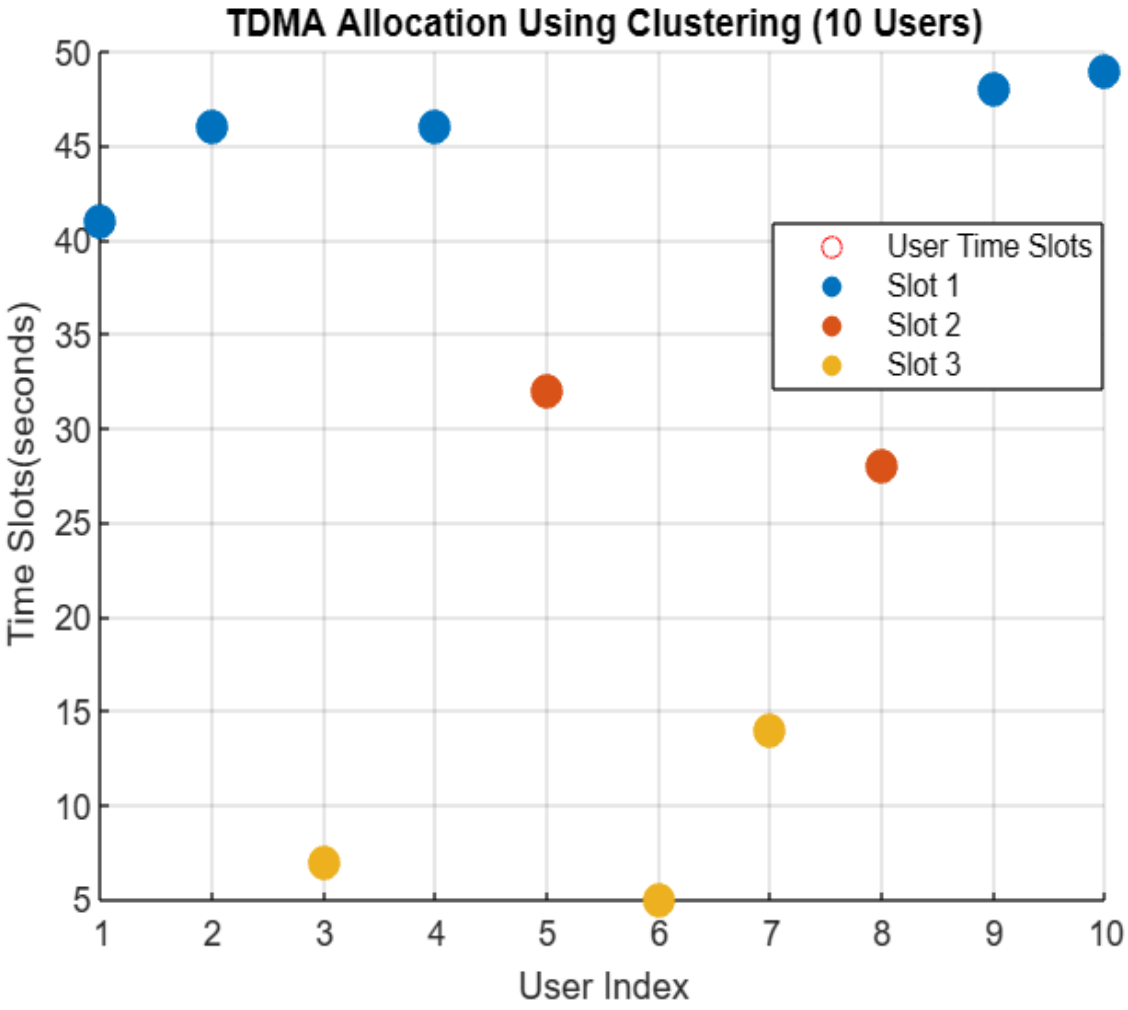
Practical 4

***AIM-*: Implement FDMA, TDMA & and CDMA using MATLAB and show the results using a graph for 10 users using clustering techniques**

- **MATLAB Code for TDMA:**

```
% Simulated user data for 10 users (random time slots users want to access)
numUsers = 10;
totalTimeSlots = 50;  % Total available time slots
userData = randi([1, totalTimeSlots], numUsers, 1);

% Number of TDMA time slots
numSlots = 3;

% Applying K-means clustering to allocate time slots to users
[idx, centers] = kmeans(userData, numSlots);

% Plotting TDMA allocation with colored slots
figure;
scatter(1:numUsers, userData, 'ro', 'DisplayName', 'User Time Slots');
hold on;
grid on;

colors = lines(numSlots);  % Generate colors for each cluster
for i = 1:numSlots
    clusterUsers = find(idx == i);
    scatter(clusterUsers, userData(clusterUsers), 100, colors(i, :), 'filled',
'DisplayName', ['Slot ' num2str(i)]);
end

title('TDMA Allocation Using Clustering (10 Users)');
xlabel('User Index');
ylabel('Time Slots');
legend('Location', 'best');
```

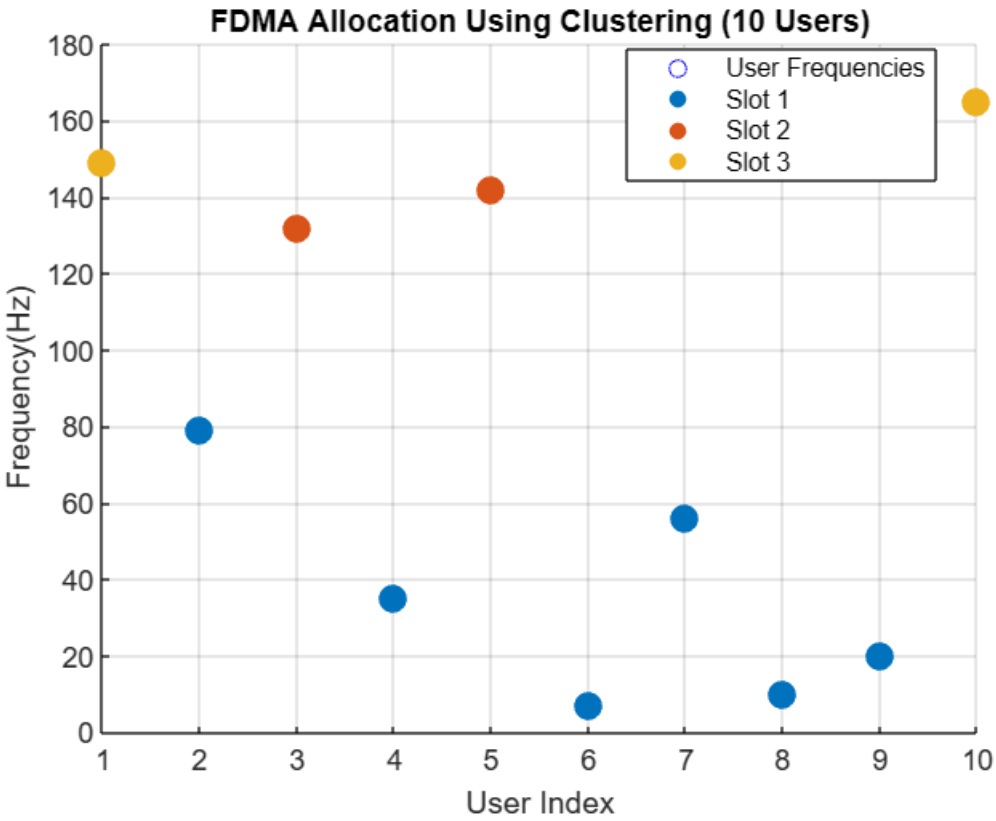
Output:


TDMA Allocation Using Clustering (10 Users)
Time Slots(seconds)
User Index
User Time Slots
Slot 1
Slot 2
Slot 3
50
45
40
35
30
25
20
15
10
5
1
2
3
4
5
6
7
8
9
10

- **MATLAB Code for FDMA:**

```
% Simulated user data for 10 users (random frequencies users want to access)
numUsers = 10;
frequencySpectrum = 200;  % Total available frequency spectrum
userData = randi([1, frequencySpectrum], numUsers, 1);

% Number of FDMA frequency slots
numSlots = 3;

% Applying K-means clustering to allocate frequency bands to users
[idx, centers] = kmeans(userData, numSlots);

% Plotting FDMA allocation with colored clusters
figure;
scatter(1:numUsers, userData, 'bo', 'DisplayName', 'User Frequencies');
hold on;
grid on;
colors = lines(numSlots);  % Generate colors for each cluster
for i = 1:numSlots
    clusterUsers = find(idx == i);
    scatter(clusterUsers, userData(clusterUsers), 100, colors(i, :), 'filled',
'DisplayName', ['Slot ' num2str(i)]);
end

title('FDMA Allocation Using Clustering (10 Users)');
xlabel('User Index');
ylabel('Frequency');
legend;
```

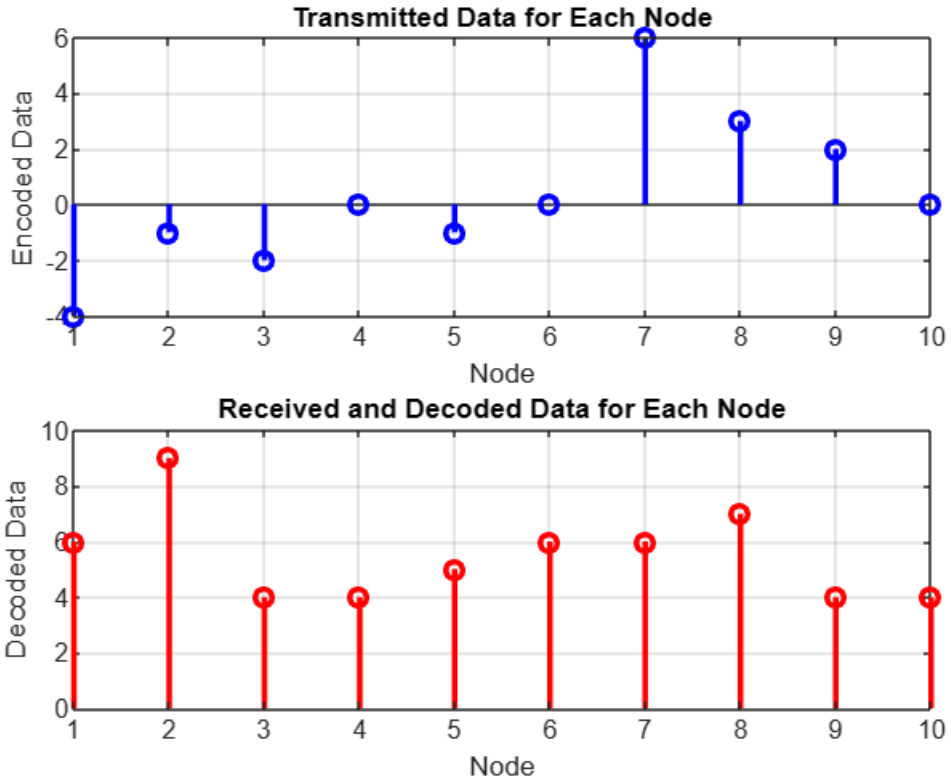
Output:


FDMA Allocation Using Clustering (10 Users)
User Frequencies
Slot 1
Slot 2
Slot 3
Frequency(Hz)
User Index
0
20
40
60
80
100
120
140
160
180
1
2
3
4
5
6
7
8
9
10

- **MATLAB Code for CDMA:**

```
% CDMA Network with 10 Nodes

% Number of nodes in the network
num_nodes = 10;

% Define spreading codes for each node (randomly generated in this example)
orthogonal_code = randi([0, 1], num_nodes, 10) * 2 - 1;

% Generate random data for each node
node_data = randi([0, 1], num_nodes, 10);
disp('the data to be sent is :- ');
disp(node_data) ;

% Transmission (each node multiplies its data with its spreading code)
transmitted_signal = node_data .* orthogonal_code;
disp('the data to be transmitted is :- ');
disp(transmitted_signal);

% Reception and decoding
decoded_data = zeros(num_nodes, 10);
for node = 1:num_nodes
    received_signal = transmitted_signal(node, :).* orthogonal_code(node, :);
    decoded_data(node, :) = received_signal > 0 ;
end
disp('the data received is :- ');
disp(decoded_data);

% Plot the transmitted and received data for each node
figure;
subplot(2, 1, 1);
for node = 1:num_nodes
    stem(node, sum(transmitted_signal(node, :)), 'b', 'LineWidth', 2);
    hold on;
end
title('Transmitted Data for Each Node');
xlabel('Node');
ylabel('Encoded Data');
grid on;

subplot(2, 1, 2);
for node = 1:num_nodes
    stem(node, sum(decoded_data(node, :)), 'r', 'LineWidth', 2);
    hold on;
end
title('Received and Decoded Data for Each Node');
xlabel('Node');
ylabel('Decoded Data');
grid on;
```

Output:




```
the data to be sent is :-
     0     0     0     0     0     0     1     1     0     0
     1     0     1     0     1     1     1     0     0     1
     1     0     0     0     0     1     0     0     1     0
     0     1     0     1     1     0     1     0     0     1
     1     1     0     0     0     0     1     0     0     1
     0     0     0     0     1     0     0     1     1     0
     0     0     0     0     0     0     1     0     1     0
     0     0     1     1     0     1     1     0     0     0
     0     1     1     1     1     1     1     0     0     0
     0     0     1     1     1     1     1     0     0     0

the data to be transmitted is :-
     0     0     0     0     0     0    -1     1     0     0
    -1     0     1     0    -1    -1    -1     0     0    -1
    -1     0     0     0     0    -1     0     0     1     0
     0     1     0     1     1     0    -1     0     0    -1
     1    -1     0     0     0     0     1     0     0    -1
     0     0     0     0     1     0     0    -1     1     0
     0     0     0     0     0     0     1     0     1     0
     0     0     1    -1     0     1    -1     0     0     0
     0    -1    -1    -1     1    -1     1     0     0     0
     0     0    -1     1    -1    -1     1     0     0     0

the data received is :-
     0     0     0     0     0     0     1     1     0     0
     1     0     1     0     1     1     1     0     0     1
     1     0     0     0     0     1     0     0     1     0
     0     1     0     1     1     0     1     0     0     1
     1     1     0     0     0     0     1     0     0     1
     0     0     0     0     1     0     0     1     1     0
     0     0     0     0     0     0     1     0     1     0
     0     0     1     1     0     1     1     0     0     0
     0     1     1     1     1     1     1     0     0     0
     0     0     1     1     1     1     1     0     0     0
```

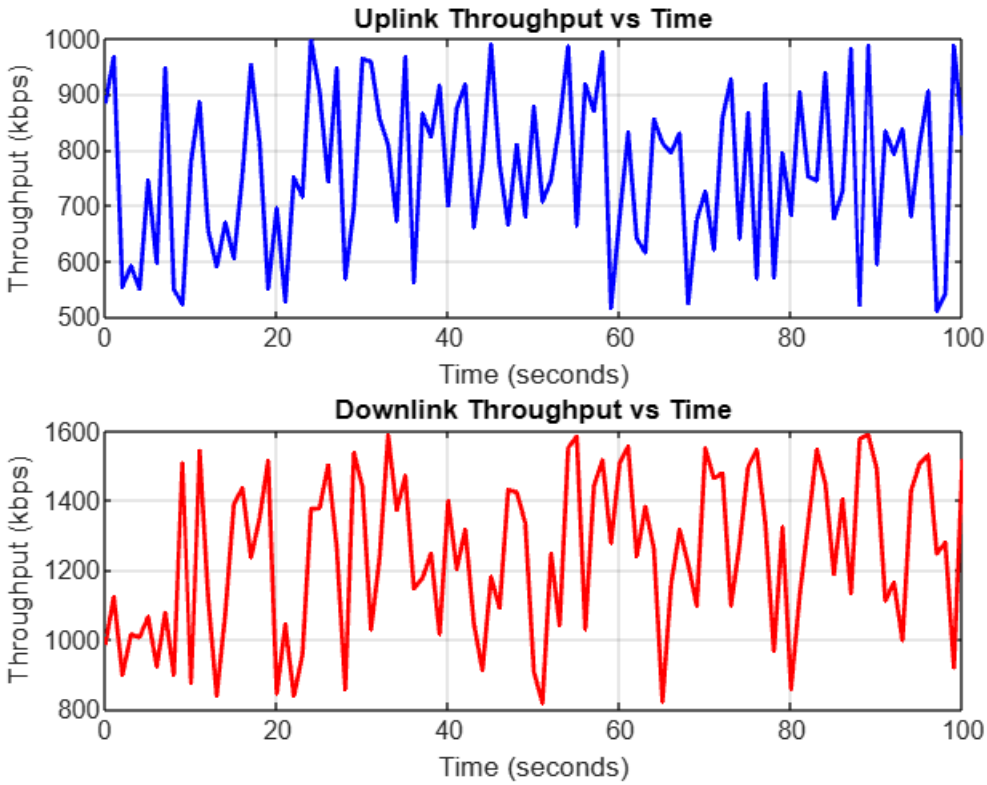
Practical 5

***AIM*- Implement GSM using MATLAB**

- **MATLAB Code for (Throughput vs Time):**

```
% Parameters
total_time = 100; % Total simulation time (in seconds)
time_interval = 1; % Time interval for calculation (in seconds)

% Simulation
time = 0:time_interval:total_time;
num_intervals = numel(time);

% Generating example throughput values (random in this case)
uplink_throughput = rand(1, num_intervals)*500 + 500; % Random values between
500 and 1000 kbps
downlink_throughput = rand(1, num_intervals)*800 + 800; % Random values between
800 and 1600 kbps

% Creating subplots for uplink and downlink
figure;

% Subplot for uplink throughput
subplot(2, 1, 1);
plot(time, uplink_throughput, 'b', 'LineWidth', 1.5);
title('Uplink Throughput vs Time');
xlabel('Time (seconds)');
ylabel('Throughput (kbps)');
grid on;

% Subplot for downlink throughput
subplot(2, 1, 2);
plot(time, downlink_throughput, 'r', 'LineWidth', 1.5);
title('Downlink Throughput vs Time');
xlabel('Time (seconds)');
ylabel('Throughput (kbps)');
grid on;
```

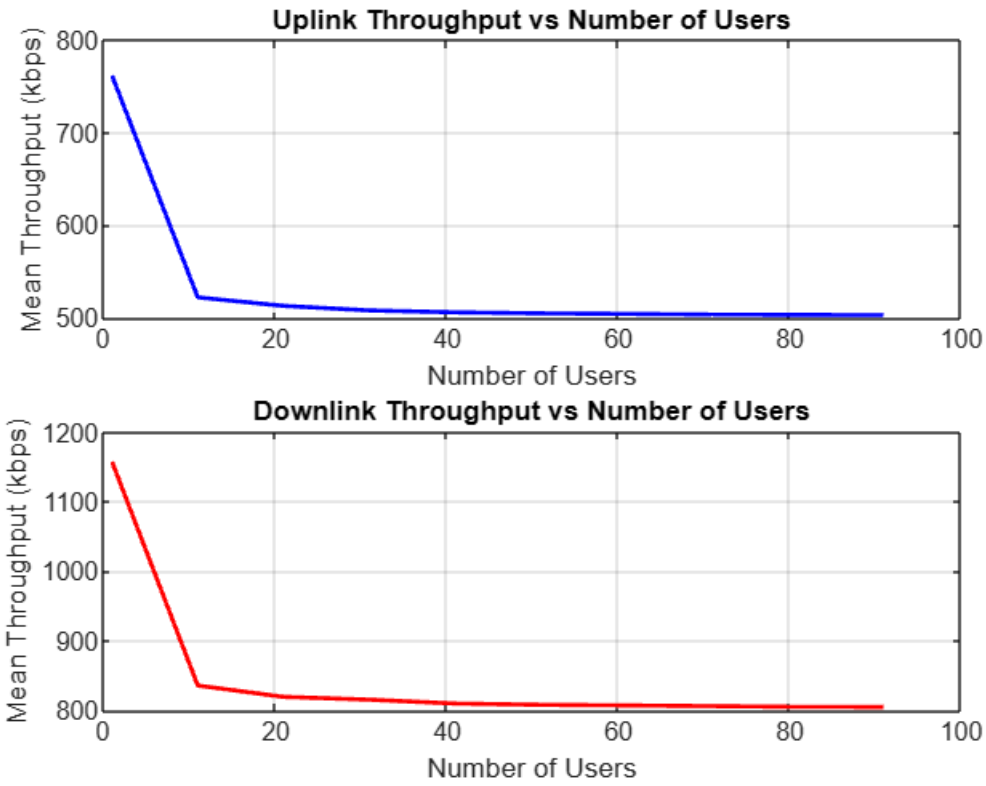
Output:


Uplink Throughput vs Time
Throughput (kbps)
1000
900
800
700
600
500
0
20
40
60
80
100
Time (seconds)
Downlink Throughput vs Time
Throughput (kbps)
1600
1400
1200
1000
800
0
20
40
60
80
100
Time (seconds)

- **MATLAB Code for GSM (Throughput vs Number Of Users):**

```
total_time = 100; % Total simulation time (in seconds)
time_interval = 1; % Time interval for calculation (in seconds)
num_users = 1:10:100; % Varying number of users

% Pre-allocating arrays for throughput
uplink_throughput = zeros(length(num_users), total_time);
downlink_throughput = zeros(length(num_users), total_time);

% Simulating throughput for different numbers of users
for i = 1:length(num_users)
    % Generating example throughput values (decreasing with more users for
demonstration purposes)
    uplink_throughput(i, :) = rand(1, total_time) * (500 / num_users(i)) + 500;
    downlink_throughput(i, :) = rand(1, total_time) * (800 / num_users(i)) + 800;
end

% Plotting the change in throughput with the number of users in separate subplots
figure;

% Subplot for uplink throughput
subplot(2, 1, 1);
plot(num_users, mean(uplink_throughput, 2), 'b', 'LineWidth', 1.5);
title('Uplink Throughput vs Number of Users');
xlabel('Number of Users');
ylabel('Mean Throughput (kbps)');
grid on;

% Subplot for downlink throughput
subplot(2, 1, 2);
plot(num_users, mean(downlink_throughput, 2), 'r', 'LineWidth', 1.5);
title('Downlink Throughput vs Number of Users');
xlabel('Number of Users');
ylabel('Mean Throughput (kbps)');
grid on;
```

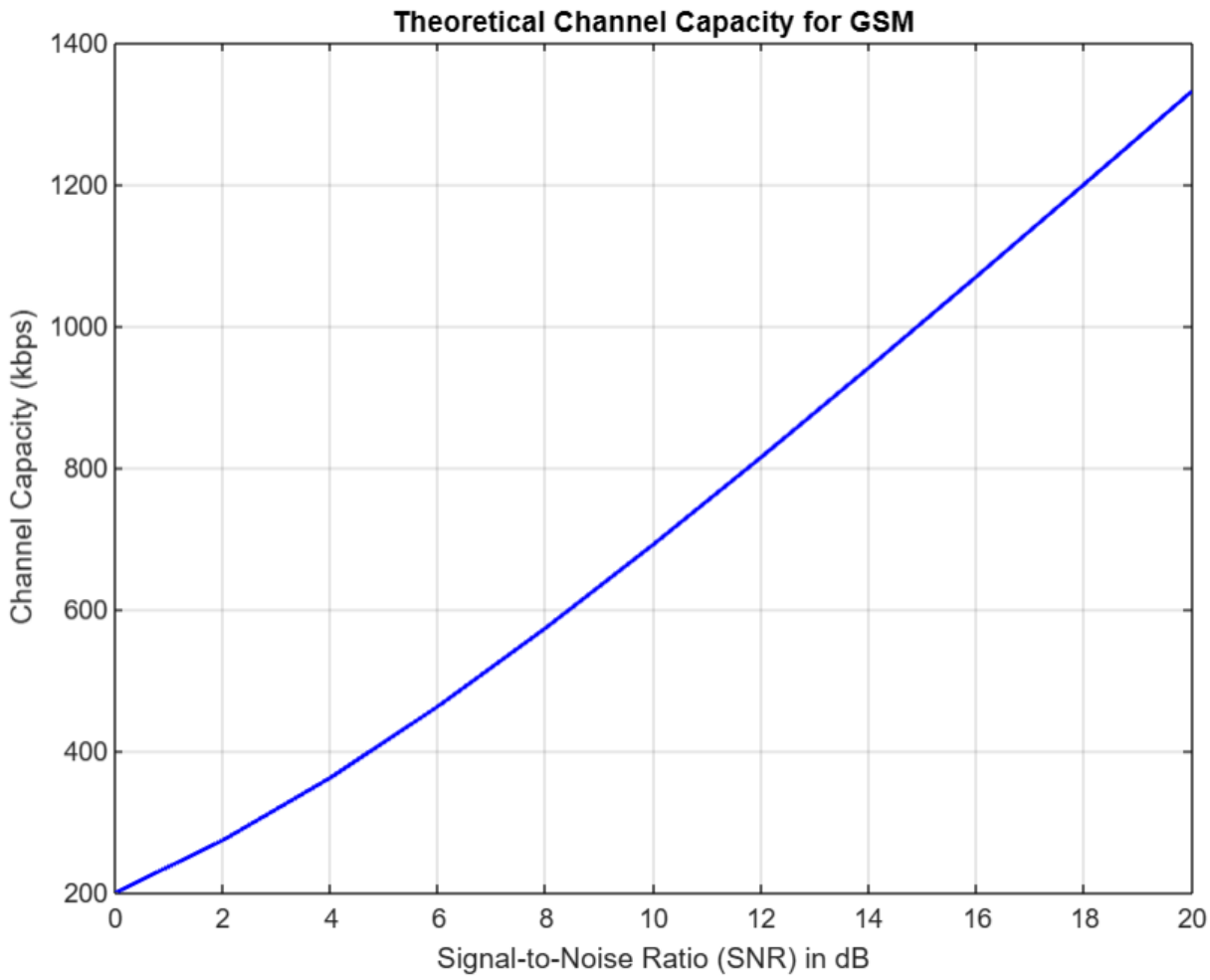
Output:


Uplink Throughput vs Number of Users
Mean Throughput (kbps)
800
700
600
500
0
20
40
60
80
100
Number of Users
Downlink Throughput vs Number of Users
Mean Throughput (kbps)
1200
1100
1000
900
800
0
20
40
60
80
100
Number of Users

- **MATLAB Code for GSM(SNR vs Shannon's capacity):**

```
% Bandwidth for GSM in Hz (200 kHz)
bandwidth = 200e3;

% Varying Signal-to-Noise Ratio (SNR)
SNR_dB = 0:2:20; % Vary SNR from 0 to 20 dB
SNR = 10.^(SNR_dB/10); % Convert SNR from dB to linear scale

% Calculate Shannon Capacity for varying SNR
capacity = bandwidth * log2(1 + SNR);

% Plotting the Shannon Capacity vs. SNR
figure;
plot(SNR_dB, capacity / 1000, 'b', 'LineWidth', 1.5);
title('Theoretical Channel Capacity for GSM');
xlabel('Signal-to-Noise Ratio (SNR) in dB');
ylabel('Channel Capacity (kbps)');
grid on;
```

Output:

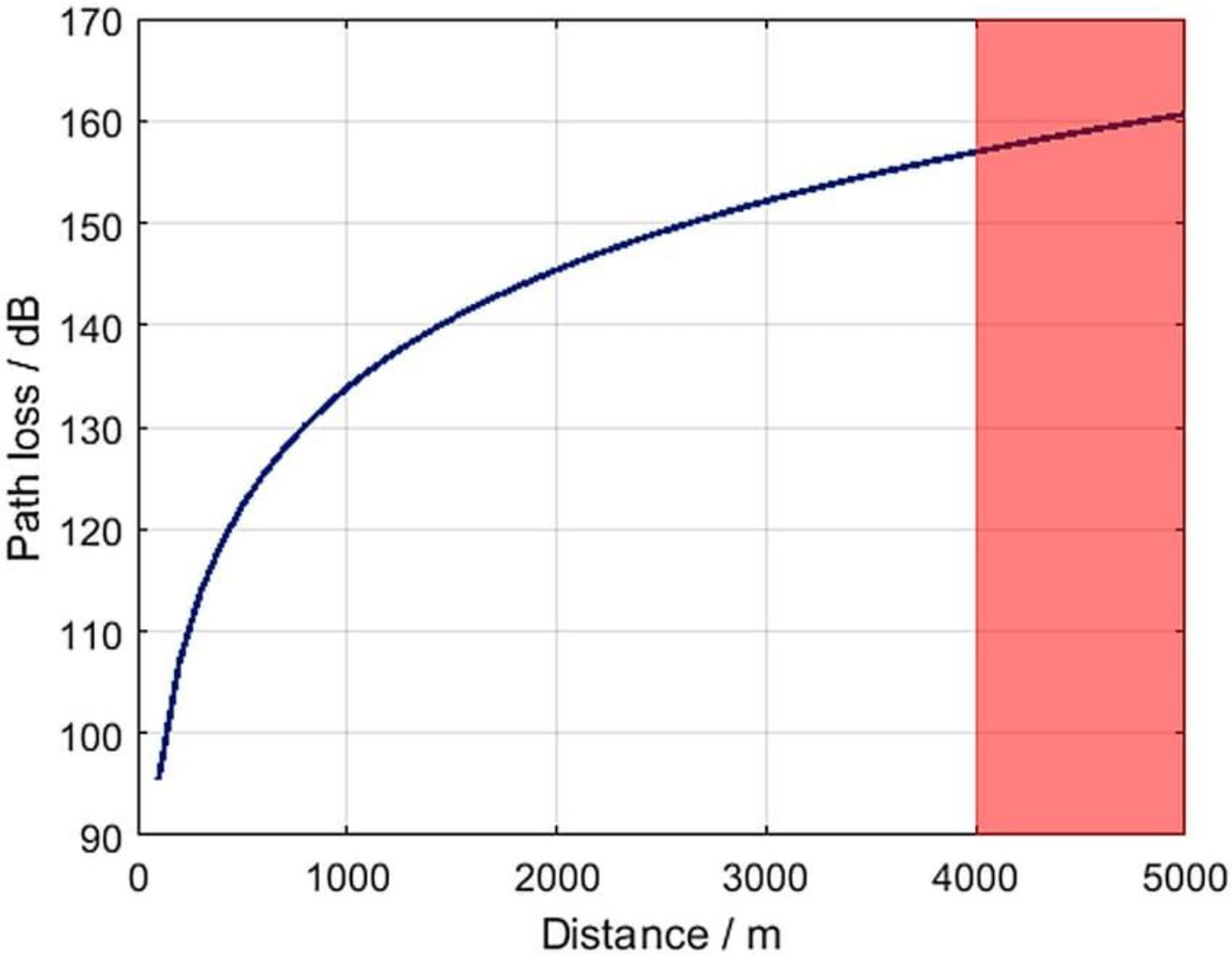


- **MATLAB Code for GSM Pathloss Model (Okumura_hata):**

```
% okumura_hata.m
function path_loss = okumura_hata(distance, frequency)
    % Okumura-Hata parameters for urban environments
    C = 69.55; % constant term
    ahm = 26.16; % constant term
    if frequency >= 150 && frequency <= 1500
        % For frequency in the range of 150 MHz to 1500 MHz
        hm = 1.1 * log10(frequency) - 0.7;
    else
        % Outside the range
        hm = 1.1 * log10(frequency) - 0.7;
    end

    path_loss = C + 26.16 * log10(frequency) - 13.82 * log10(hm) - ahm *
log10(distance);
end
```

```
% Your main script or function

% Define parameters
frequency = 900; % Frequency of the signal in MHz
distances = 1:0.1:10; % Varying distances from 1 km to 10 km

% Calculate path loss for varying distances using the okumura_hata function
path_loss_values = okumura_hata(distances, frequency);

% Plotting the Path Loss vs. Distance
figure;
plot(distances, path_loss_values, 'b', 'LineWidth', 1.5);
title('Path Loss Model for GSM (Okumura-Hata)');
xlabel('Distance (km)');
ylabel('Path Loss (dB)');
```

Output:


170
160
150
140
130
120
110
100
90
Path loss / dB
0
1000
2000
3000
4000
5000
Distance / m

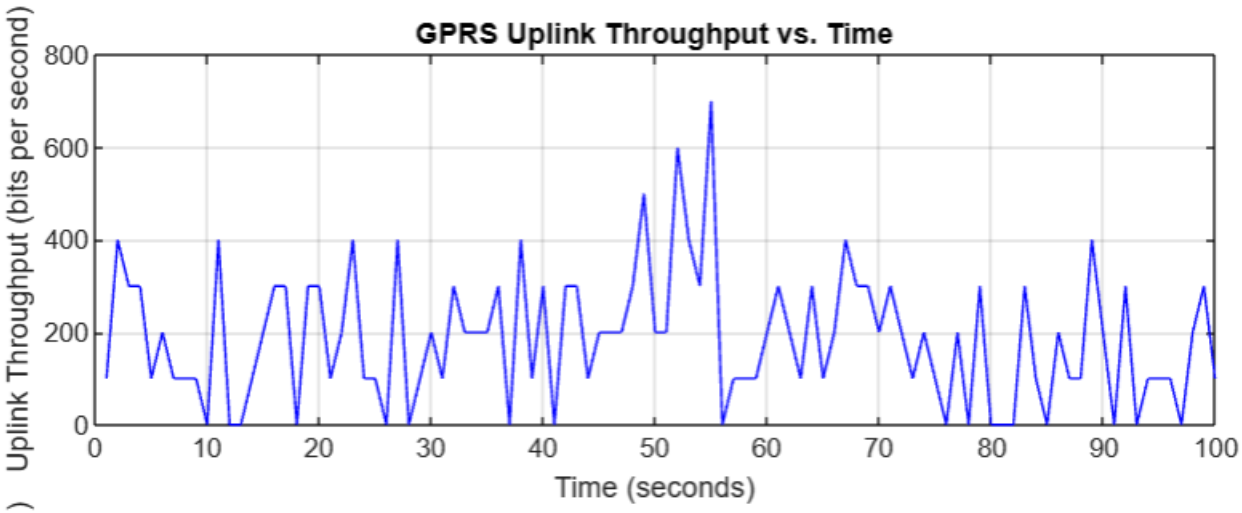
Practical 6

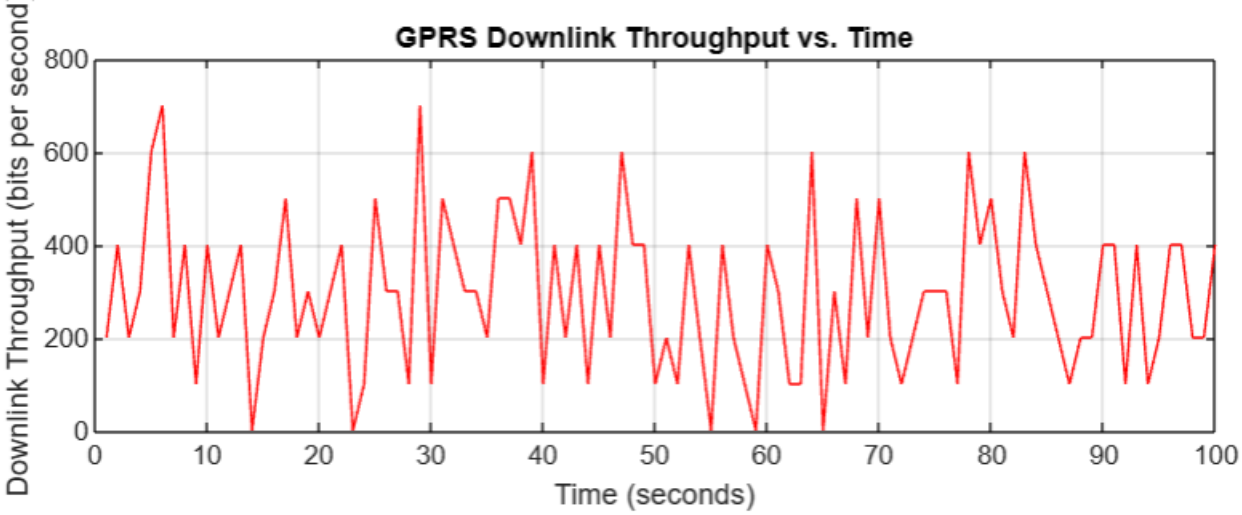
AIM- Implement GPRS using MATLAB

- **MATLAB Code for (Throughput vs Time):**

```
% Simulation parameters
simulation_duration = 100;  % Duration of the simulation in seconds
packet_rate_uplink = 2;     % Uplink packets per second
packet_rate_downlink = 3;   % Downlink packets per second
packet_size = 100;          % Packet size in bits

% Initialize variables
time = 0;
throughput_uplink = zeros(1, simulation_duration);
throughput_downlink = zeros(1, simulation_duration);

% Simulation loop
for t = 1:simulation_duration
    % Generate random uplink and downlink packets
    num_packets_uplink = poissrnd(packet_rate_uplink);
    num_packets_downlink = poissrnd(packet_rate_downlink);

    total_bits_transferred_uplink = num_packets_uplink * packet_size;
    total_bits_transferred_downlink = num_packets_downlink * packet_size;

    % Update uplink and downlink throughputs
    throughput_uplink(t) = total_bits_transferred_uplink;
    throughput_downlink(t) = total_bits_transferred_downlink;

    % Update time
    time = time + 1;
end

% Plot uplink throughput vs. time
figure;
subplot(2, 1, 1);
plot(1:simulation_duration, throughput_uplink, 'b-');
xlabel('Time (seconds)');
ylabel('Uplink Throughput (bits per second)');
title('GPRS Uplink Throughput vs. Time');
grid on;

% Plot downlink throughput vs. time
subplot(2, 1, 2);
plot(1:simulation_duration, throughput_downlink, 'r-');
xlabel('Time (seconds)');
ylabel('Downlink Throughput (bits per second)');
title('GPRS Downlink Throughput vs. Time');
grid on;
```

Output:

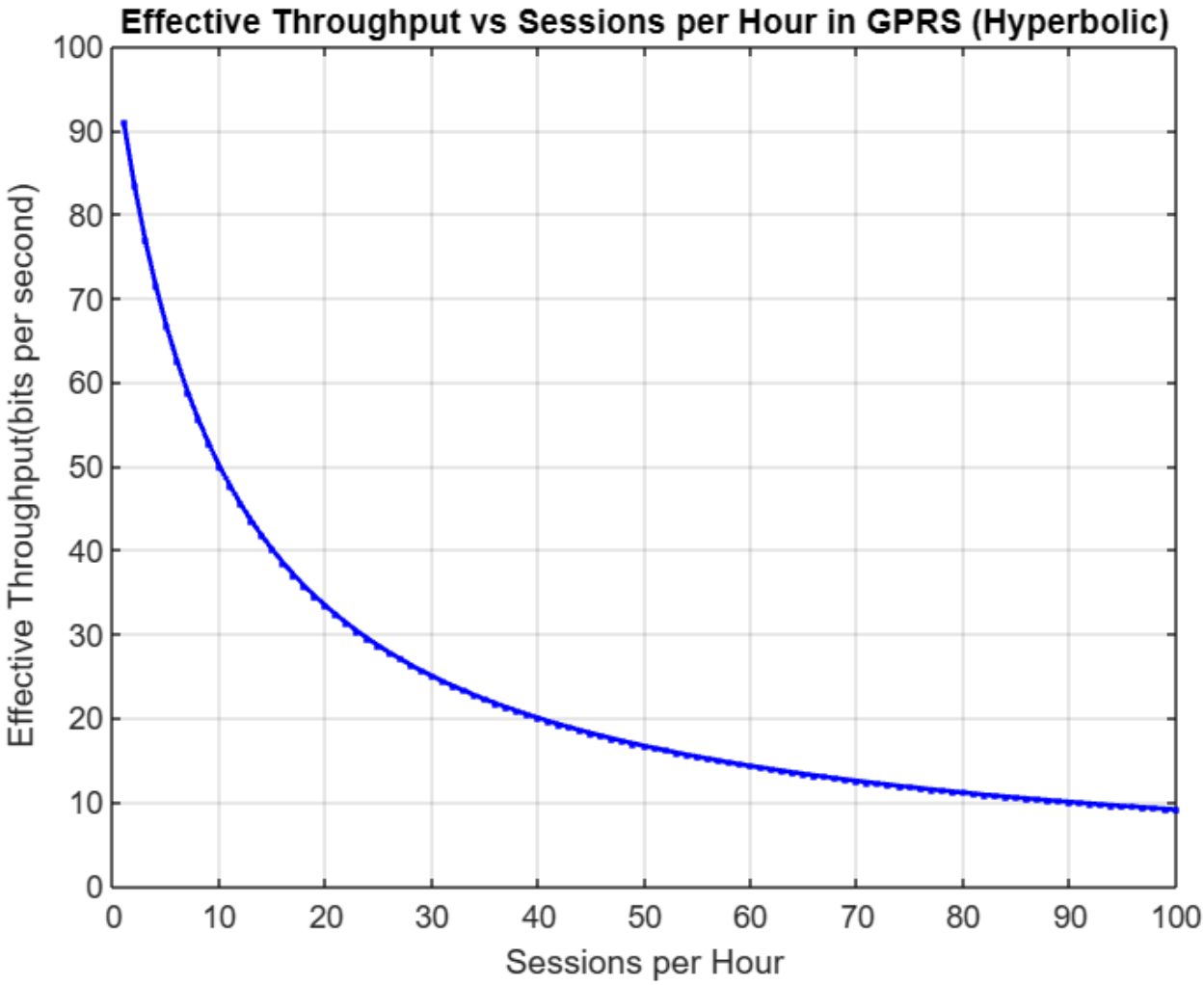

GPRS Uplink Throughput vs. Time
Uplink Throughput (bits per second)
Time (seconds)
0
200
400
600
800
0
10
20
30
40
50
60
70
80
90
100


GPRS Downlink Throughput vs. Time
Downlink Throughput (bits per second)
Time (seconds)
0
200
400
600
800
0
10
20
30
40
50
60
70
80
90
100

- **MATLAB Code for GPRS (Throughput vs Number Of Users):**

```
% Constants for simulation
total_time = 3600; % Total simulation time (in seconds) - 1 hour
packet_duration = 1; % Duration of each packet transmission (in seconds)
max_sessions_per_hour = 100; % Maximum sessions per hour to simulate

% Initialize variables
sessions_per_hour = 1:max_sessions_per_hour; % Varying number of sessions per hour
throughput = zeros(1, max_sessions_per_hour); % Initialize array for throughput

for i = 1:max_sessions_per_hour
    % Simulate effective throughput for each session count

    num_sessions = sessions_per_hour(i);
    packets_per_session = total_time / num_sessions / packet_duration;
    packets_per_session = round(packets_per_session);

    % Simulate decreasing throughput values for each session (hyperbolic relationship)
    initial_throughput = 100; % Initial throughput value
    session_throughputs = initial_throughput / (1 + 0.1 * num_sessions); % Hyperbolic
model

    % Store the effective throughput value for each session
    throughput(i) = session_throughputs;
end

% Plot effective throughput against sessions per hour (with a hyperbolic trend)
figure;
plot(sessions_per_hour, throughput, 'b.-', 'LineWidth', 1.5);
title('Effective Throughput vs Sessions per Hour in GPRS (Hyperbolic)');
xlabel('Sessions per Hour');
ylabel('Effective Throughput(bits per second)');
grid on;
```

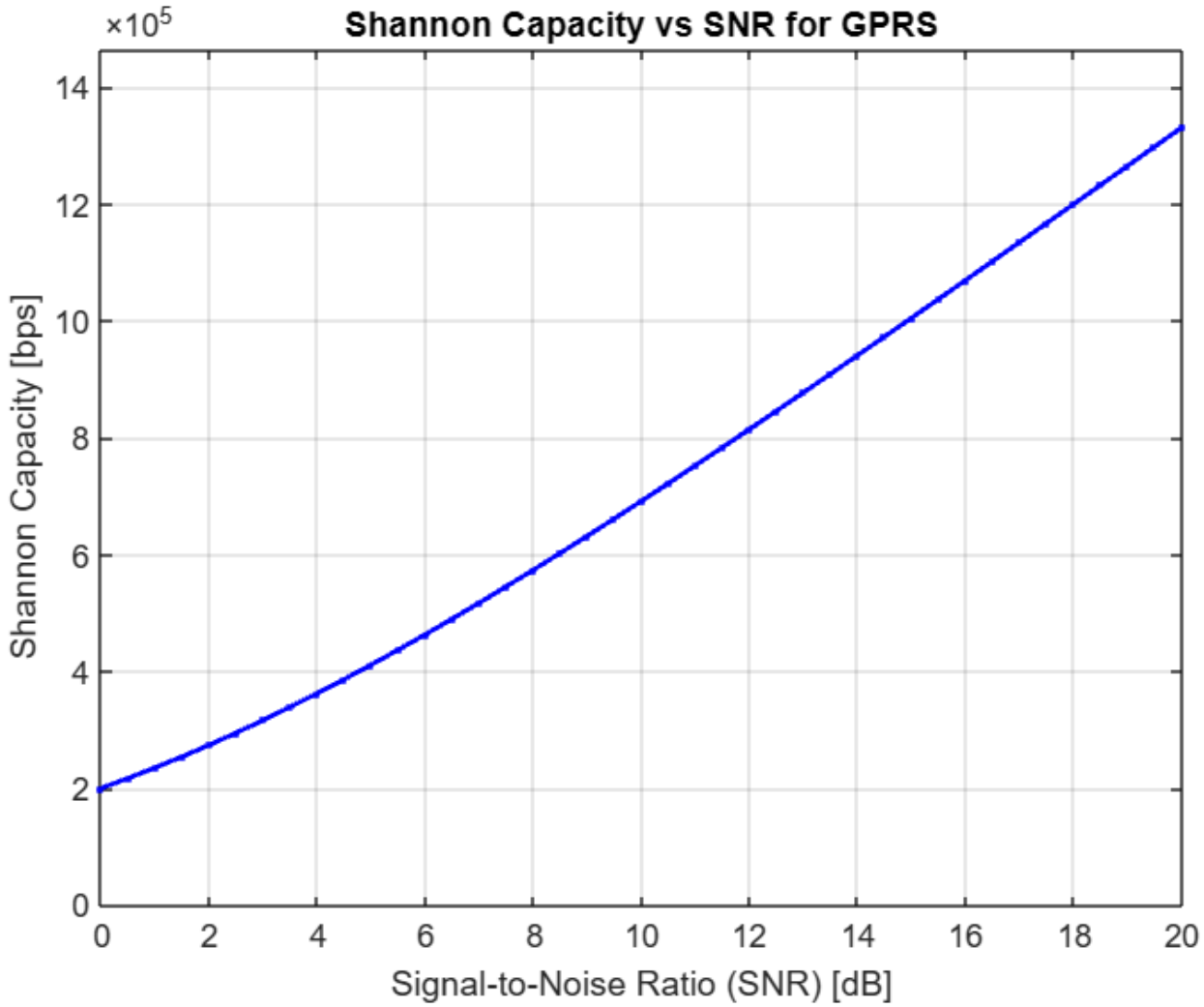
Output:


Effective Throughput vs Sessions per Hour in GPRS (Hyperbolic)
Effective Throughput(bits per second)
Sessions per Hour
0
10
20
30
40
50
60
70
80
90
100

- **MATLAB Code for GPRS (SNR vs Shannon capacity):**

```
% Given bandwidth for GPRS (as an example)
bandwidth_GPRS = 200000; % Bandwidth in Hz (200 kHz)

% Range of Signal-to-Noise Ratio (SNR) values in dB
SNR_dB = 0:0.5:20; % From 0 dB to 20 dB in steps of 0.5 dB

% Calculate Shannon capacity using the formula
shannon_capacity = bandwidth_GPRS * log2(1 + 10.^(SNR_dB / 10));

% Plot SNR vs Shannon capacity
figure;
plot(SNR_dB, shannon_capacity, 'b.-', 'LineWidth', 1.5);
title('Shannon Capacity vs SNR for GPRS');
xlabel('Signal-to-Noise Ratio (SNR) [dB]');
ylabel('Shannon Capacity [bps]');
grid on;

% Set custom axis limits for x and y axes
xlim([0, 20]); % Adjust the x-axis limits from 0 to 20 dB
ylim([0, max(shannon_capacity) * 1.1]); % Adjust the y-axis limits (10% more than the
maximum capacity)
```

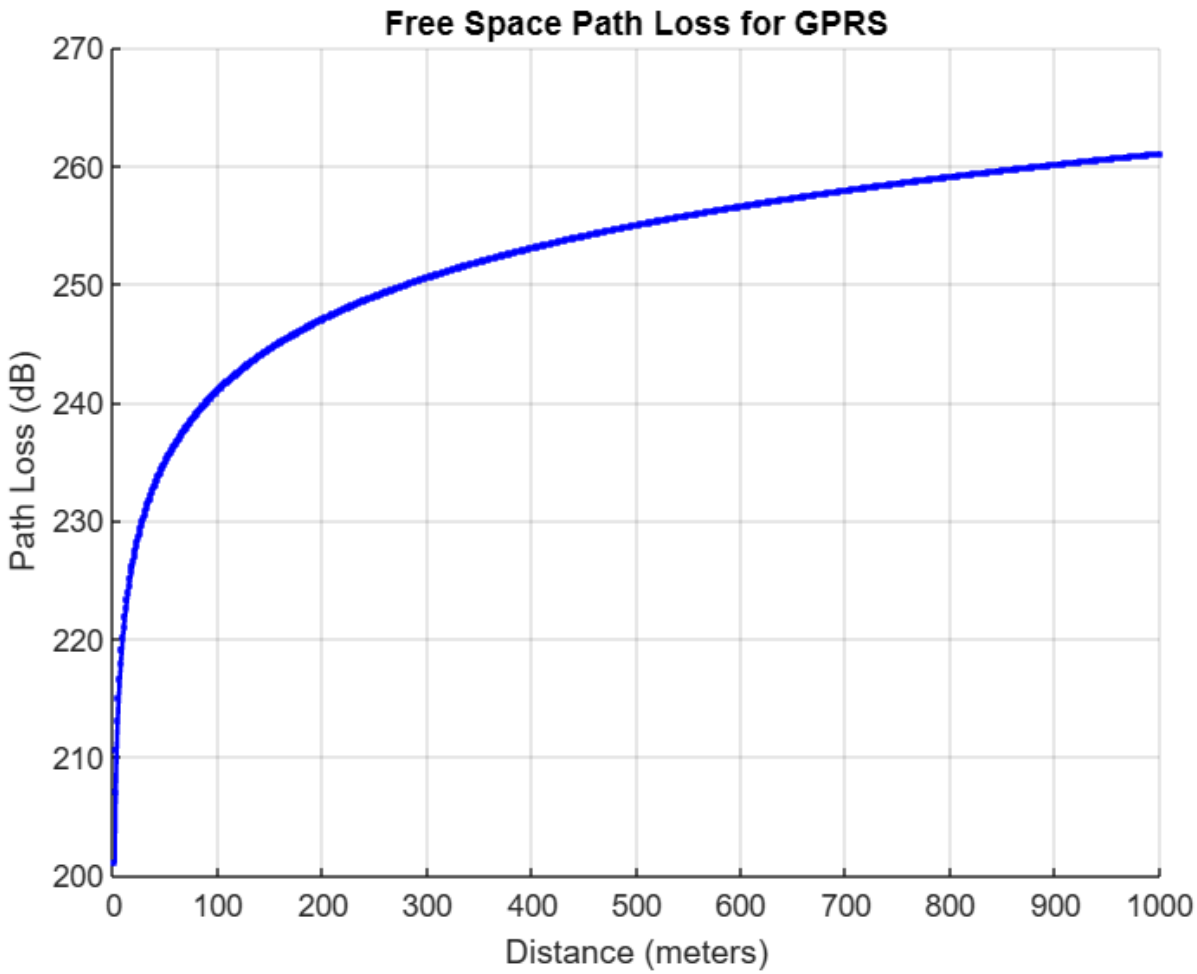
Output:


Shannon Capacity vs SNR for GPRS
×10^5
Shannon Capacity [bps]
Signal-to-Noise Ratio (SNR) [dB]
0
2
4
6
8
10
12
14
16
18
20

- **MATLAB Code for GPRS (Free Space Path Loss Model):**

```
% Constants for the path loss model
frequency_GPRS = 900 * 10^6; % GPRS frequency in Hz (e.g., 900 MHz)
d_ref = 1; % Reference distance in meters
antenna_gain = 0; % Antenna gains in dB (assuming no gain for simplicity)

% Range of distances to simulate in meters
distance = 1:1000; % Distance range from 1 to 1000 meters

% Calculate Free Space Path Loss (FSPL) in dB
fspl_dB = 20 * log10(distance / d_ref) + 20 * log10(frequency_GPRS) + 20 * log10(4 *
pi) - antenna_gain;

% Plot Path Loss vs Distance
figure;
hold on;
grid on;
plot(distance, fspl_dB, 'b.-', 'LineWidth', 1.5);
title('Free Space Path Loss for GPRS');
xlabel('Distance (meters)');
ylabel('Path Loss (dB)');
```

Output:


Free Space Path Loss for GPRS
Path Loss (dB)
270
260
250
240
230
220
210
200
0
100
200
300
400
500
600
700
800
900
1000
Distance (meters)

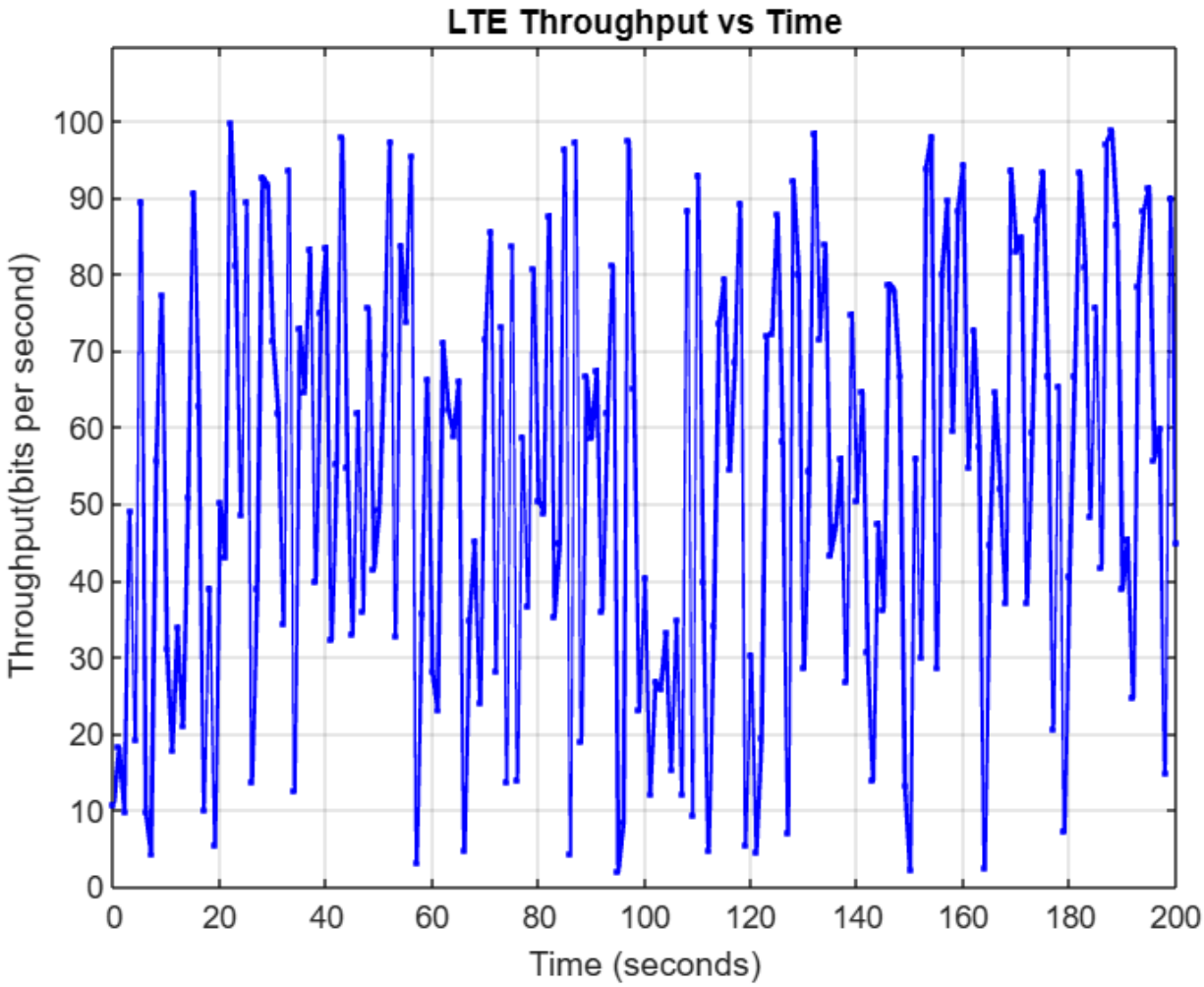
Practical 7

***AIM*-Implement LTE using MATLAB**

•MATLAB Code for LTE(Throughput vs Time):

```
% Constants for simulation
total_time = 200; % Total simulation time (in seconds) - 1 hour
packet_duration = 1; % Duration of each packet transmission (in seconds)

% Initialize variables
time = 0:packet_duration:total_time; % Time vector
num_packets = length(time); % Number of packets

% Simulate random throughput values for LTE over time
throughput = rand(1, num_packets) * 100; % Generating random throughput values (scaled
by 100 for example purposes)

% Plot throughput vs time for LTE
figure;
plot(time, throughput, 'b.-', 'LineWidth', 1.5);
title('LTE Throughput vs Time');
xlabel('Time (seconds)');
ylabel('Throughput(bits per second)');
grid on;
```

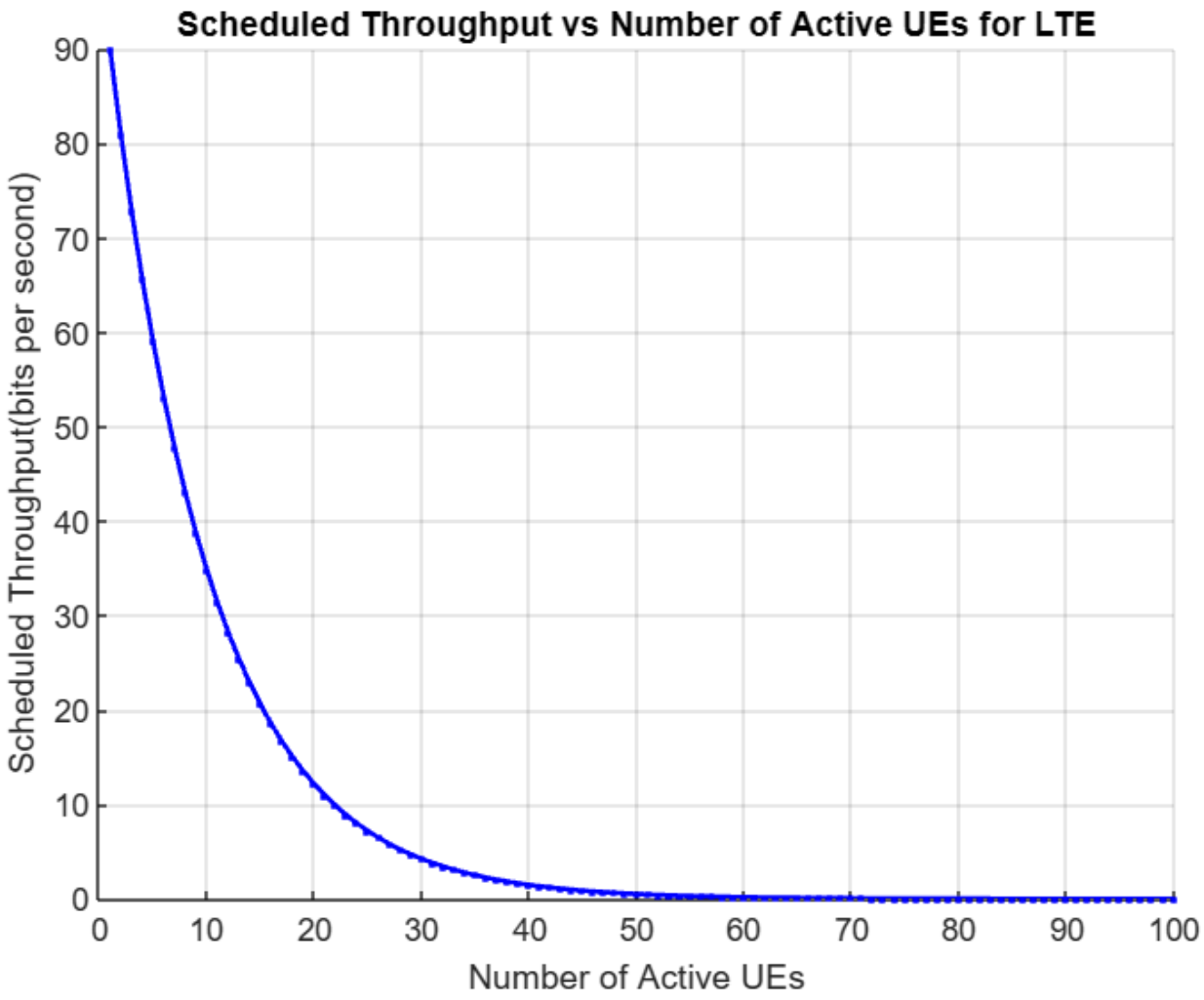
Output:


LTE Throughput vs Time
Throughput(bits per second)
Time (seconds)
0
10
20
30
40
50
60
70
80
90
100
0
20
40
60
80
100
120
140
160
180
200

• MATLAB Code for LTE(Throughput vs Number Of Users):

```
% Constants for simulation
total_time = 3600; % Total simulation time (in seconds) - 1 hour
packet_duration = 1; % Duration of each packet transmission (in seconds)
max_UEs = 100; % Maximum number of UEs to simulate

% Initialize variables
num_UEs = 1:max_UEs; % Varying number of active UEs
scheduled_throughput = zeros(1, max_UEs); % Initialize array for scheduled throughput

for i = 1:max_UEs
    % Simulate scheduled throughput for each number of UEs

    % For this example, assume scheduled throughput decreases with more UEs
    num_active_UEs = num_UEs(i);

    % Simulate decreasing throughput values for each number of UEs
    initial_throughput = 100; % Initial throughput value
    decreasing_factor = 0.9; % Factor for decreasing throughput
    scheduled_throughput(i) = initial_throughput * decreasing_factor^num_active_UEs;
end

% Plot scheduled throughput against number of active UEs
figure;
hold on;
grid on;
plot(num_UEs, scheduled_throughput, 'b.-', 'LineWidth', 1.5);
title('Scheduled Throughput vs Number of Active UEs for LTE');
xlabel('Number of Active UEs');
ylabel('Scheduled Throughput(bits per second)');
```

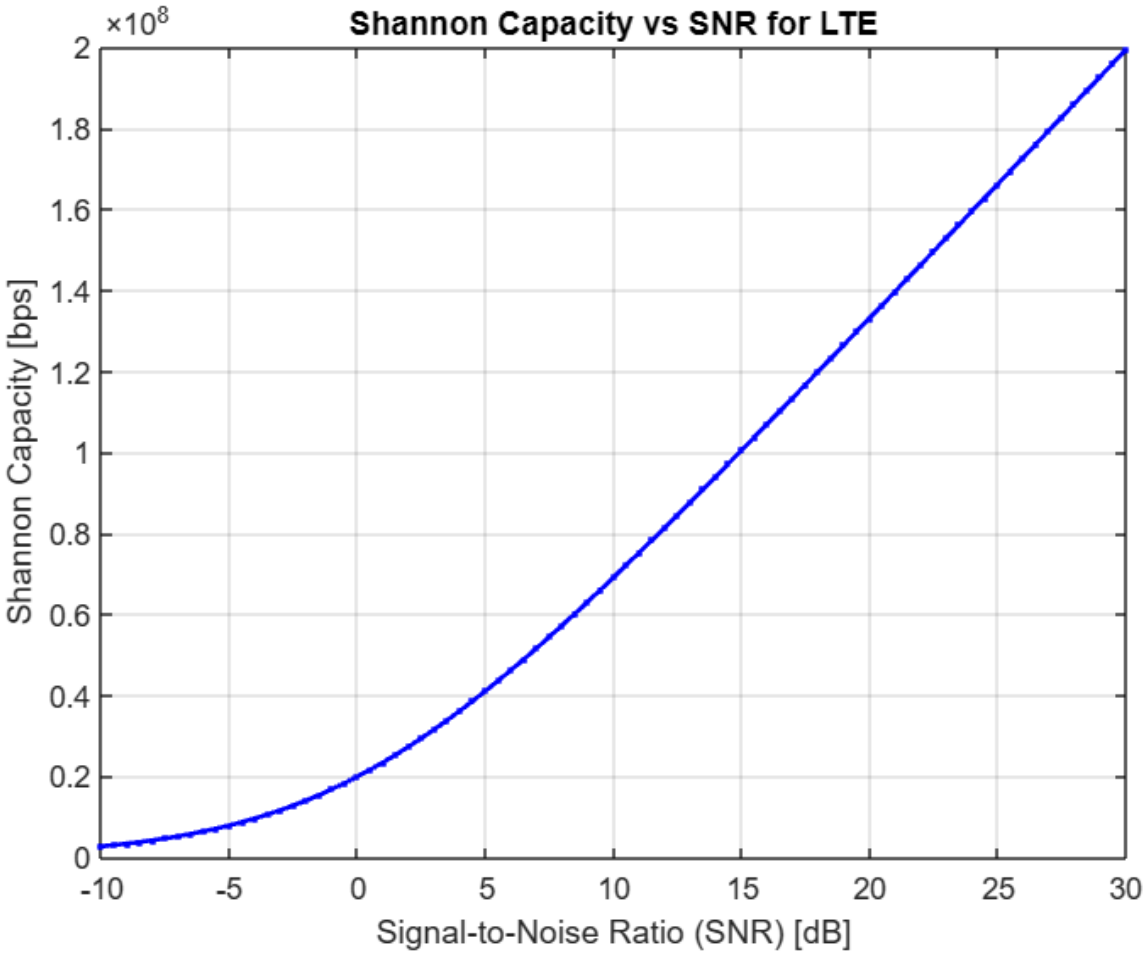
Output:


Scheduled Throughput vs Number of Active UEs for LTE
Scheduled Throughput(bits per second)
Number of Active UEs
0
10
20
30
40
50
60
70
80
90
100

• MATLAB Code for LTE(SNR vs Shannon capacity):

```
% Constants for LTE
bandwidth_LTE = 20 * 10^6; % LTE bandwidth in Hz (e.g., 20 MHz)

% Range of Signal-to-Noise Ratio (SNR) values in dB
SNR_dB = -10:0.5:30; % From -10 dB to 30 dB in steps of 0.5 dB

% Calculate Shannon capacity using the formula
shannon_capacity = bandwidth_LTE * log2(1 + 10.^(SNR_dB / 10));

% Plot SNR vs Shannon capacity for LTE
figure;
plot(SNR_dB, shannon_capacity, 'b.-', 'LineWidth', 1.5);
title('Shannon Capacity vs SNR for LTE');
xlabel('Signal-to-Noise Ratio (SNR) [dB]');
ylabel('Shannon Capacity [bps]');
grid on;
```

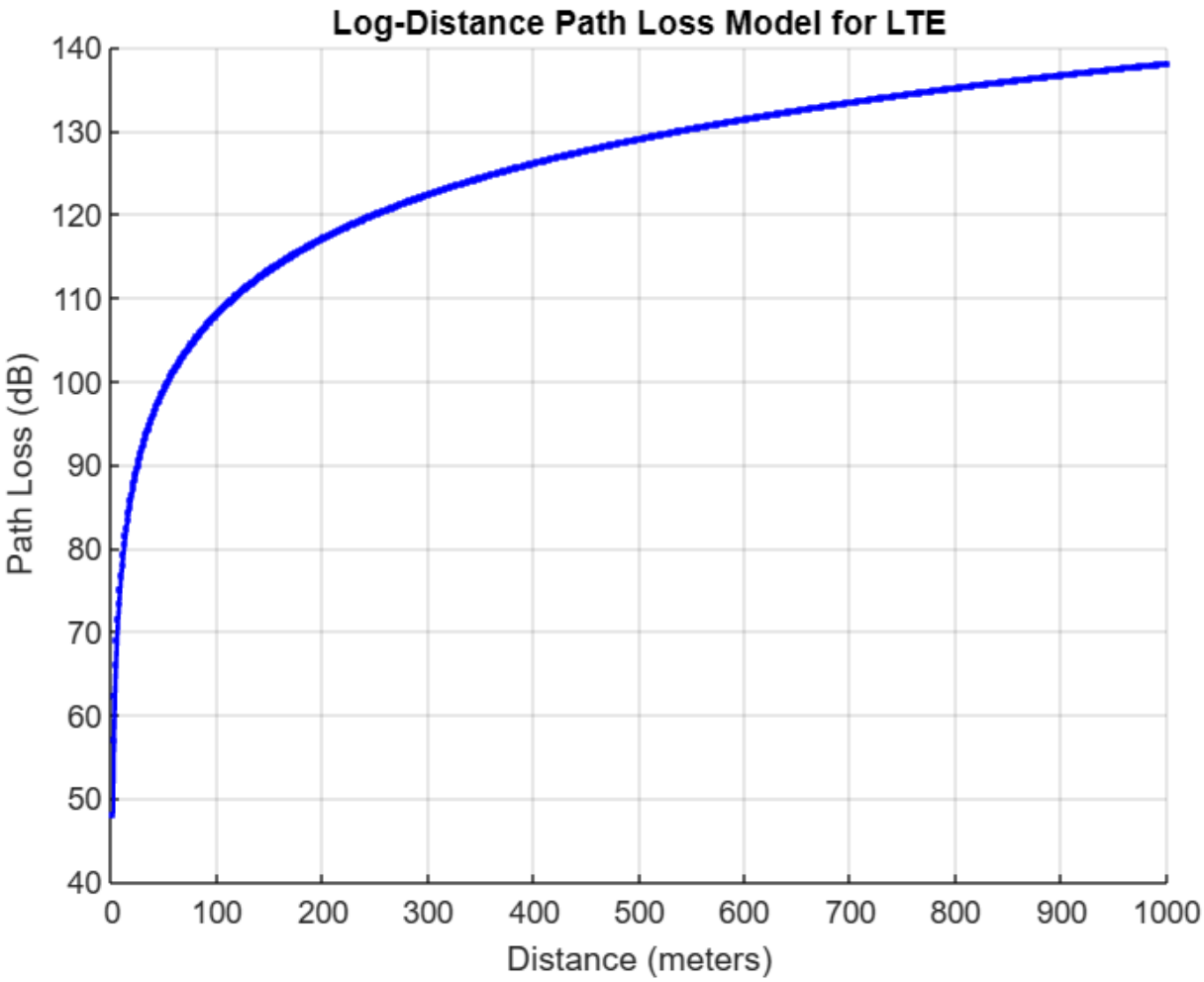
Output:


Shannon Capacity vs SNR for LTE
×10^8
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Shannon Capacity [bps]
-10
-5
0
5
10
15
20
25
30
Signal-to-Noise Ratio (SNR) [dB]

• MATLAB Code for LTE(Log-Distance Path Loss model):

```
% Constants for Log-Distance Path Loss Model
d0 = 1; % Reference distance in meters
PL_d0 = 50; % Path Loss at reference distance (example value in dB)
n = 3; % Path loss exponent (example value)
X_f = normrnd(0, 4); % Log-Normal Shadowing (random factor with mean 0 and std
deviation 4)

% Range of distances to simulate in meters
distance = 1:1000; % Distance range from 1 to 1000 meters

% Calculate Log-Distance Path Loss (PL) in dB
PL = PL_d0 + 10 * n * log10(distance / d0) + X_f;

% Plot Log-Distance Path Loss vs Distance for LTE
figure;
hold on;
grid on;
plot(distance, PL, 'b.-', 'LineWidth', 1.5);
title('Log-Distance Path Loss Model for LTE');
xlabel('Distance (meters)');
ylabel('Path Loss (dB)');
```

Output:


Log-Distance Path Loss Model for LTE
Path Loss (dB)
Distance (meters)
140
130
120
110
100
90
80
70
60
50
40
0
100
200
300
400
500
600
700
800
900
1000

***Practical-8*:**

***AIM*-**IMPLEMENT SNOOPING AND ANALYSING THE TRAFFIC USING WIRESHARK

PROCEDURE-

Getting Wireshark:-

In order to run Wireshark, you will need to have access to a computer that supports both Wireshark and the libpcap or WinPCap packet capture library. The libpcap software will be installed for you, if it is not installed within your operating system, when you install Wireshark.

See http://www.wireshark.org/download.html for a list of supported operating systems and download sites.

Download and install the Wireshark software:-

- Go to http://www.wireshark.org/download.html and download and install the Wireshark binary for your computer.
- Download the Wireshark user guide. The Wireshark FAQ has several helpful hints and interesting tidbits of information, particularly if you have trouble installing or running Wireshark.

Running Wireshark:-

When you run the Wireshark program, the Wireshark graphical user interface shown in Figure 2 will de displayed. Initially, no data will be displayed in the various windows.

The Wireshark interface has five major components:-

1. The command menus are standard pulldown menus located at the top of the window. Of interest to us now are the File and Capture menus. The File menu allows you to save captured packet data or open a file containing previously captured packet data, and exit the Wireshark application. The Capture menu allows you to begin packet capture.
2. The packet-listing window displays a one-line summary for each packet captured, including the packet number (assigned by Wireshark; this is not a packet number contained in any protocol's header), the time at which the packet was captured, the packet's source and destination addresses, the protocol type, and protocol-specific information contained in the packet. The packet listing can be sorted according to any of these categories by clicking on a column name. The protocol type field lists the highest-level protocol that sent or received this packet, i.e., the protocol that is the source or ultimate sink for this packet.
3. The packet-header details window provides details about the packet selected (highlighted) in the packet listing window. (To select a packet in the packet listing window, place the cursor over the packet's one-line summary in the packet listing window and click with the left mouse button.). These details include information about the Ethernet frame (assuming the packet was sent/received over an Ethernet interface) and IP datagram that contains this packet. The amount of Ethernet and IP-layer detail displayed can be expanded or minimized by clicking on the plus minus boxes to the left of the Ethernet frame or IP datagram line in the packet details window. If the packet has been carried over TCP or UDP, TCP or UDP details will also be displayed, which can

similarly be expanded or minimized. Finally, details about the highest-level protocol that sent or received this packet are also provided.

4. The packet-contents window displays the entire contents of the captured frame, in both ASCII and hexadecimal format.
5. Towards the top of the Wireshark graphical user interface, is the packet display filter field, into which a protocol name or other information can be entered in order to filter the information displayed in the packet-listing window (and hence the packet-header and packet-contents windows). In the example below, we'll use the packet-display filter field to have Wireshark hide (not display) packets except those that correspond to HTTP messages.

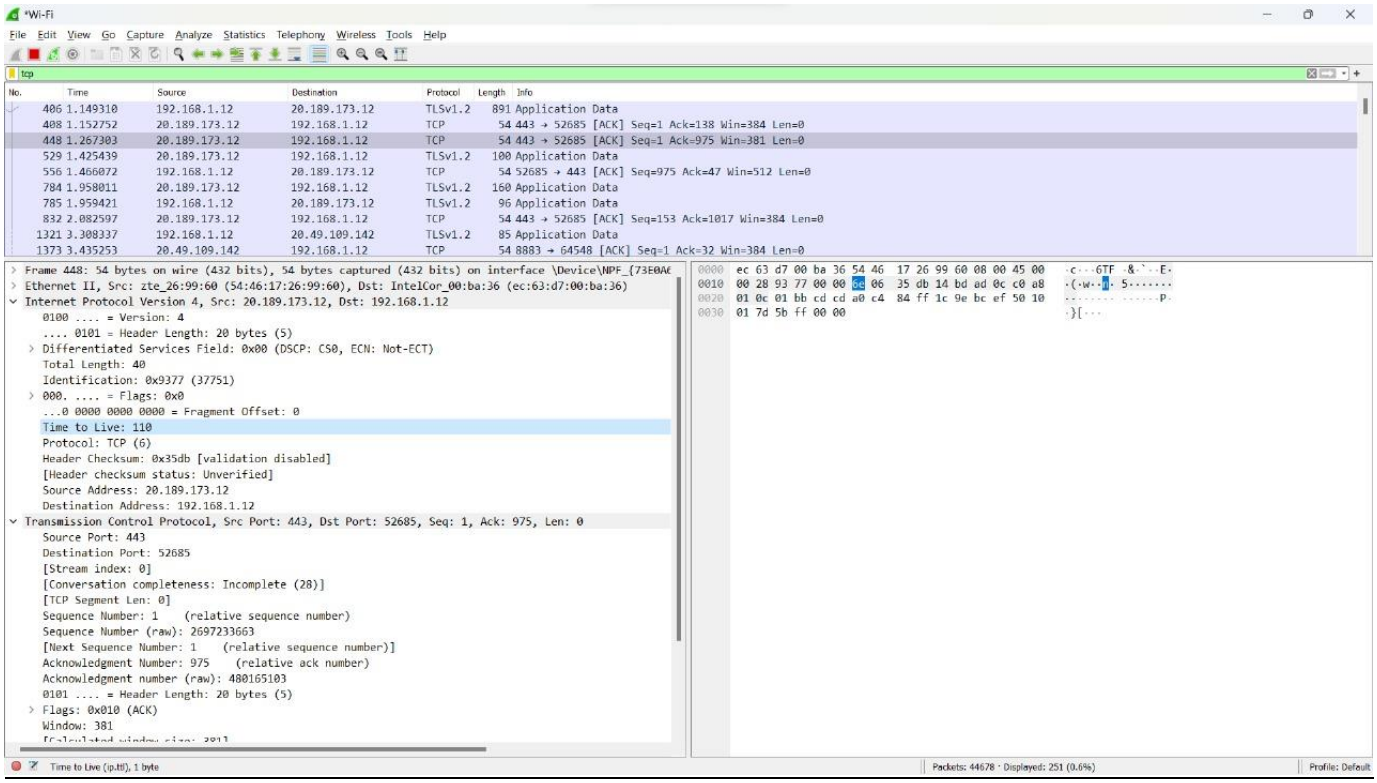
Taking Wireshark for a Test Run:-

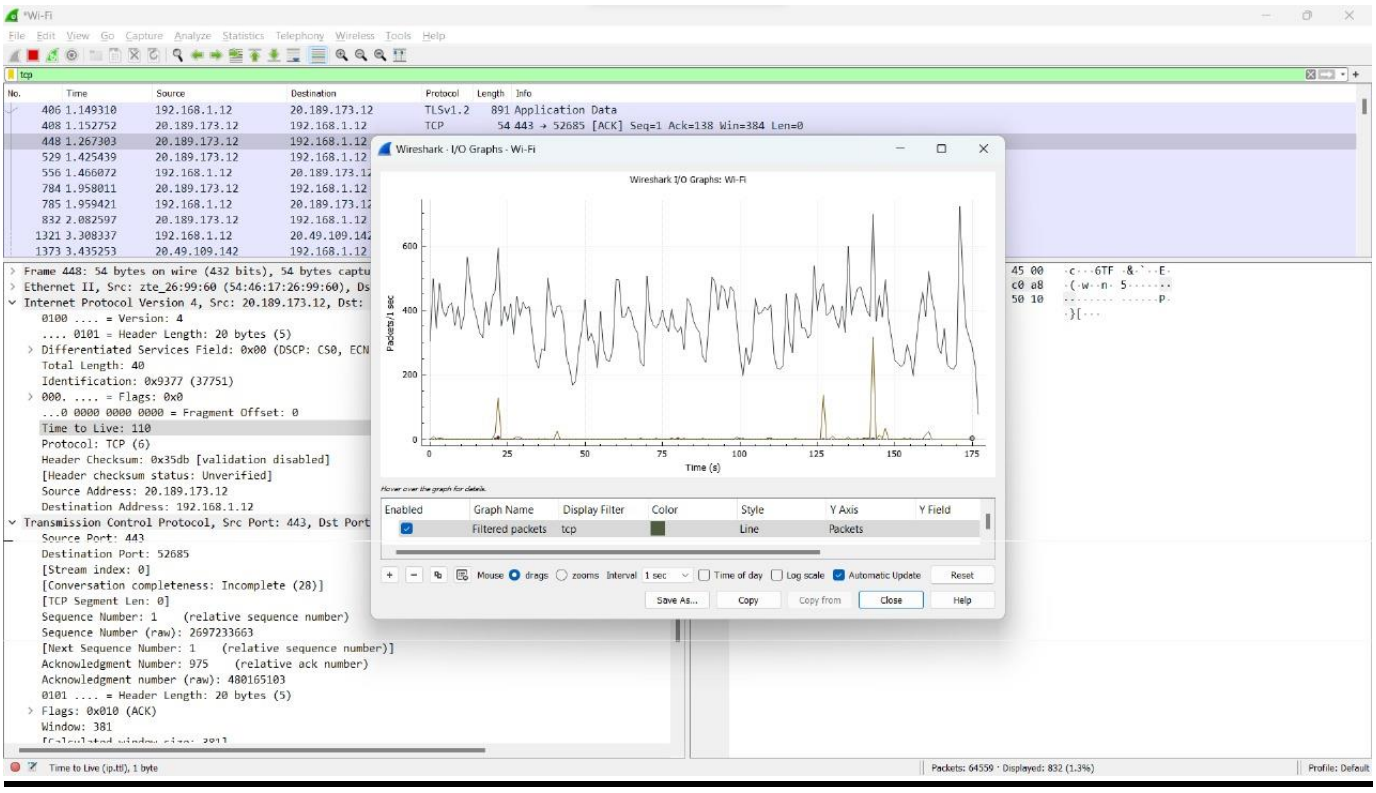
The best way to learn about any new piece of software is to try it out! We'll assume that your computer is connected to the Internet via a wired Ethernet interface. Do the following:

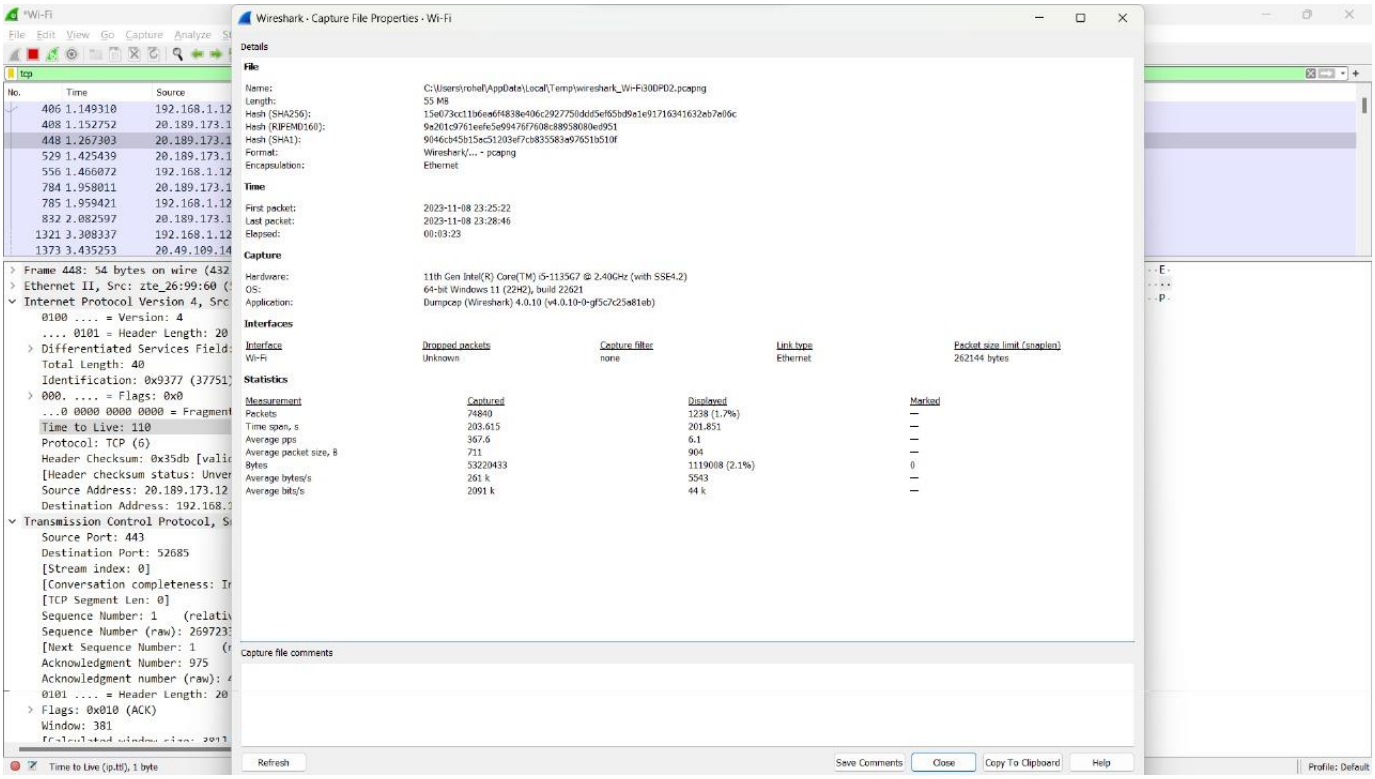
1. Start up your favorite web browser, which will display your selected homepage.

2. Start up the Wireshark software. You will initially see a window similar to that shown in Figure 2, except that no packet data will be displayed in the packetlisting, packet-header, or packet-contents window, since Wireshark has not yet begun capturing packets.

3. To begin packet capture, select the Capture pull down menu and select Options. This will cause the "Wireshark: Capture Options" window to be displayed, as shown in Figure 3.


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp
No. Time Source Destination Protocol Length Info
406 1.149310 192.168.1.12 20.189.173.12 TLSv1.2 891 Application Data
408 1.152752 20.189.173.12 192.168.1.12 TCP 54 443 → 52685 [ACK] Seq=1 Ack=138 Win=384 Len=0
448 1.267303 20.189.173.12 192.168.1.12 TCP 54 443 → 52685 [ACK] Seq=1 Ack=975 Win=381 Len=0
529 1.425439 20.189.173.12 192.168.1.12 TLSv1.2 100 Application Data
556 1.466072 192.168.1.12 20.189.173.12 TCP 54 52685 → 443 [ACK] Seq=975 Ack=47 Win=512 Len=0
784 1.958011 20.189.173.12 192.168.1.12 TLSv1.2 160 Application Data
785 1.959421 192.168.1.12 20.189.173.12 TLSv1.2 96 Application Data
832 2.082597 20.189.173.12 192.168.1.12 TCP 54 443 → 52685 [ACK] Seq=153 Ack=1017 Win=384 Len=0
1321 3.308337 192.168.1.12 20.49.109.142 TLSv1.2 85 Application Data
1373 3.435253 20.49.109.142 192.168.1.12 TCP 54 8883 → 64548 [ACK] Seq=1 Ack=32 Win=384 Len=0
Frame 448: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface \Device\NPF_{73E0AC
Ethernet II, Src: zte_26:99:60 (54:46:17:26:99:60), Dst: IntelCor_00:ba:36 (ec:63:d7:00:ba:36)
Internet Protocol Version 4, Src: 20.189.173.12, Dst: 192.168.1.12
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 40
Identification: 0x9377 (37751)
000. .... = Flags: 0x0
...0 0000 0000 0000 = Fragment Offset: 0
Time to Live: 110
Protocol: TCP (6)
Header Checksum: 0x35db [validation disabled]
[Header checksum status: Unverified]
Source Address: 20.189.173.12
Destination Address: 192.168.1.12
Transmission Control Protocol, Src Port: 443, Dst Port: 52685, Seq: 1, Ack: 975, Len: 0
Source Port: 443
Destination Port: 52685
[Stream index: 0]
[Conversation completeness: Incomplete (28)]
[TCP Segment Len: 0]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 2697233663
[Next Sequence Number: 1 (relative sequence number)]
Acknowledgment Number: 975 (relative ack number)
Acknowledgment number (raw): 480165103
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 381
0000 ec 63 d7 00 ba 36 54 46 17 26 99 60 08 00 45 00
0010 00 28 93 77 00 00 6e 06 35 db 14 bd ad 0c c0 a8
0020 01 0c 01 bb cd cd a0 c4 84 ff 1c 9e bc ef 50 10
0030 01 7d 5b ff 00 00
Time to Live (ip.ttl), 1 byte
Packets: 44678 · Displayed: 251 (0.6%)
Profile: Default


*Wi-Fi
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp
No. Time Source Destination Protocol Length Info
406 1.149310 192.168.1.12 20.189.173.12 TLSv1.2 891 Application Data
408 1.152752 20.189.173.12 192.168.1.12 TCP 54 443 → 52685 [ACK] Seq=1 Ack=138 Win=384 Len=0
448 1.267303 20.189.173.12 192.168.1.12
529 1.425439 20.189.173.12 192.168.1.12
556 1.466072 192.168.1.12 20.189.173.12
784 1.958011 20.189.173.12 192.168.1.12
785 1.959421 192.168.1.12 20.189.173.12
832 2.082597 20.189.173.12 192.168.1.12
1321 3.308337 192.168.1.12 20.49.109.142
1373 3.435253 20.49.109.142 192.168.1.12
Wireshark · I/O Graphs · Wi-Fi
Wireshark I/O Graphs: Wi-Fi
Packets/1 sec
Time (s)
Enabled Graph Name Display Filter Color Style Y Axis Y Field
Filtered packets tcp Line Packets
Mouse drags zooms Interval 1 sec Time of day Log scale Automatic Update Reset
Save As... Copy Copy from Close Help
Time to Live (ip.ttl), 1 byte
Packets: 64559 · Displayed: 832 (1.3%)
Profile: Default


*Wi-Fi
File Edit View Go Capture Analyze
tcp
No. Time Source
406 1.149310 192.168.1.12
408 1.152752 20.189.173.1
448 1.267303 20.189.173.1
529 1.425439 20.189.173.1
556 1.466072 192.168.1.12
784 1.958011 20.189.173.1
785 1.959421 192.168.1.12
832 2.082597 20.189.173.1
1321 3.308337 192.168.1.12
1373 3.435253 20.49.109.14
Frame 448: 54 bytes on wire (432
Ethernet II, Src: zte_26:99:60 (
Internet Protocol Version 4, Src
0100 .... = Version: 4
.... 0101 = Header Length: 20
Differentiated Services Field:
Total Length: 40
Identification: 0x9377 (37751)
000. .... = Flags: 0x0
...0 0000 0000 0000 = Fragment
Time to Live: 110
Protocol: TCP (6)
Header Checksum: 0x35db [valid
[Header checksum status: Unver
Source Address: 20.189.173.12
Destination Address: 192.168.1
Transmission Control Protocol, S
Source Port: 443
Destination Port: 52685
[Stream index: 0]
[Conversation completeness: In
[TCP Segment Len: 0]
Sequence Number: 1 (relativ
Sequence Number (raw): 269723
[Next Sequence Number: 1 (
Acknowledgment Number: 975
Acknowledgment number (raw): 4
0101 .... = Header Length: 20
Flags: 0x010 (ACK)
Window: 381
Time to Live (ip.ttl), 1 byte
Profile: Default
Wireshark · Capture File Properties · Wi-Fi
Details
File
Name: C:\Users\rohel\AppData\Local\Temp\wireshark_Wi-Fi30DPD2.pcapng
Length: 55 MB
Hash (SHA256): 15e073cc11b6ea6f4838e406c2927750ddd5ef65bd9a1e91716341632eb7e06c
Hash (RIPEMD160): 9a201c9761eefe5e69476f7608c88958080ed951
Hash (SHA1): 9046cb45b15ac51203ef7cb835583a97651b510f
Format: Wireshark/... - pcapng
Encapsulation: Ethernet
Time
First packet: 2023-11-08 23:25:22
Last packet: 2023-11-08 23:28:46
Elapsed: 00:03:23
Capture
Hardware: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz (with SSE4.2)
OS: 64-bit Windows 11 (22H2), build 22621
Application: Dumpcap (Wireshark) 4.0.10 (v4.0.10-0-gf5c7c25a81eb)
Interfaces
Interface Dropped packets Capture filter Link type Packet size limit (snaplen)
Wi-Fi Unknown none Ethernet 262144 bytes
Statistics
Measurement Captured Displayed Marked
Packets 74840 1238 (1.7%) —
Time span, s 203.615 201.851 —
Average pps 367.6 6.1 —
Average packet size, B 711 904 —
Bytes 53220433 1119008 (2.1%) 0
Average bytes/s 261 k 5543 —
Average bits/s 2091 k 44 k —
Capture file comments
Refresh
Save Comments
Close
Copy To Clipboard
Help