Netaji Subhas University of Technology, Delhi

Practical File

Mobile Computing (INITC17)

Submitted By:-

1. Aryan Doshi 2021UIN2812
2. Anushka Tyagi 2021UIN2813
3. Vipul Bhardwaj 2021UIN2827
4. Divyam Jain 2021UIN3301
5. Shreyas Behl 2021UIN3302
6. Vikas Singh 2021UIN3303

INDEX

PRACTICAL 1

Steps to install MATLAB

Step 1: MATLAB Installation File

Visit the MathWorks website www.mathworks.com and log in using your MathWorks account.

After logging in, navigate to the MATLAB product page and locate the download link for MATLAB R2023b.

Step 2: Run the Installation File

Run the installation file.

Step 3: Log in with Your MathWorks Account

Login using your Id.

Step 4: Accept the License Agreement

Accept the MathWorks Software License.

Step 5: Select Your License

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

Select which MATLAB products you want to install. Select "MATLAB" and "Simulink.

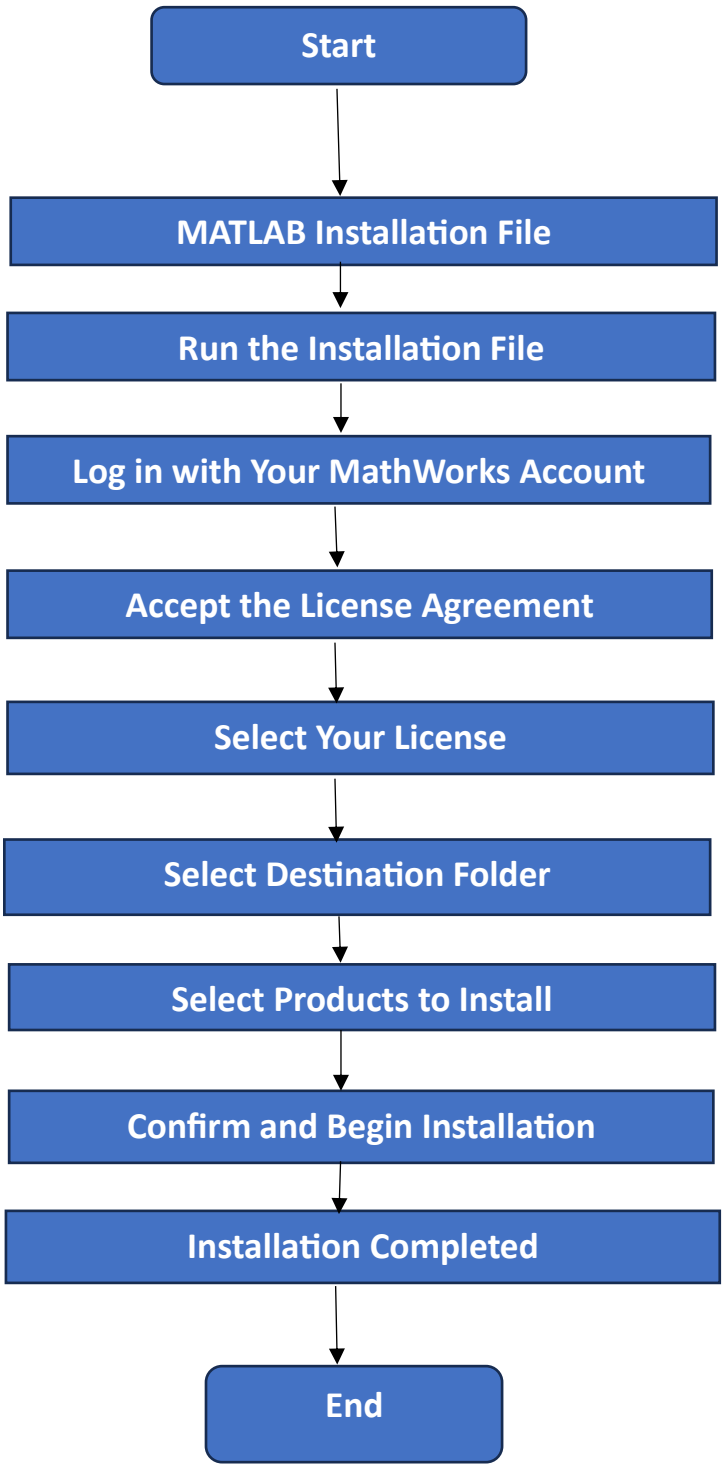
Step 8: Confirm and Begin Installation

Review your selections. If everything is fine, click "Next" to start the installation process.

Step 9: Installation Completed

MATLAB R2023b is now installed on your Windows system.

Flowchart


Start
MATLAB Installation File
Run the Installation File
Log in with Your MathWorks Account
Accept the License Agreement
Select Your License
Select Destination Folder
Select Products to Install
Confirm and Begin Installation
Installation Completed
End

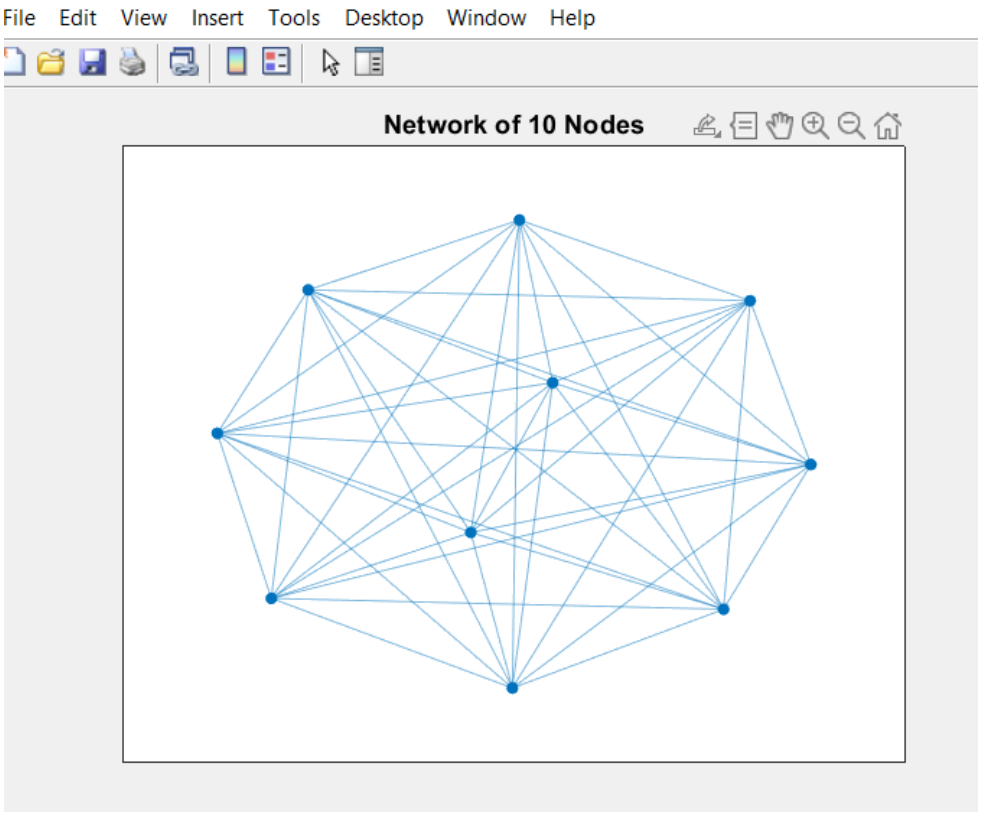
PRACTICAL 2

Program to implement network of 10 Nodes

Code

```
numNodes = 10;
G = graph(rand(numNodes), 'upper', 'omitselfloops');
figure;
plot(G, 'NodeLabel', {});
title('Network of 10 Nodes');
```

Output



PRACTICAL 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.

Network Assumptions

All 10 nodes have equal bandwidth.

All nodes have a constant queue size

Constant packet size, and packet arrival time of 1 ms.

Constants

queue_size = 50;

bandwidth = 16;

packet_size = 2000;

max_packet_size = 1000;

stop_time = 1100;

arrival_time = 1e-3;

propagation_delay = 5e-3;

Data Collection and Analysis

Packet drop ideally occurs only at Node 0, as for all other nodes, inRate equals outRate. The program initializes

arrays to store bandwidth and packets dropped, as well as inRates, outRates, and transmissionDelays. It then

iterates through varying bandwidth values, calculating transmission delays, inRates, outRates, and packet drops.

Results are stored in arrays and printed for analysis.

```
bandwidth_values = [];
packets_dropped = [];
inRates = [];
outRates = [];
transmissionDelays = [];
while bandwidth >= 1
        transmission_delay = (max_packet_size * 8) / (bandwidth * 1e6);
        inRate = packet_size / max_packet_size;
        outRate = 1e-3 / transmission_delay;
        if inRate <= outrate
                PacketsDropped = 0;
        else
                firstDrop = queue_size / (inRate - outRate);
                PacketsDropped = ceil((stop_time - firstDrop) * (inRate - outRate)) + 1;
end
bandwidth_values = [bandwidth_values, bandwidth];
packets_dropped = [packets_dropped, PacketsDropped];
inRates = [inRates, inRate];
outRates = [outRates, outRate];
transmissionDelays = [transmissionDelays, transmission_delay];
fprintf('Bandwidth: %.2f bps\n', bandwidth);
fprintf('InRate: %.2f\n', inRate);
fprintf('OutRate: %.2f\n', outRate);
fprintf('Transmission Delay: %.6f s\n', transmission_delay);
fprintf('Packets Dropped: %d\n\n', PacketsDropped);
bandwidth = bandwidth / 2;
```

Data Visualization

The program plots a graph showing the relationship between bandwidth and packets dropped. Text annotations

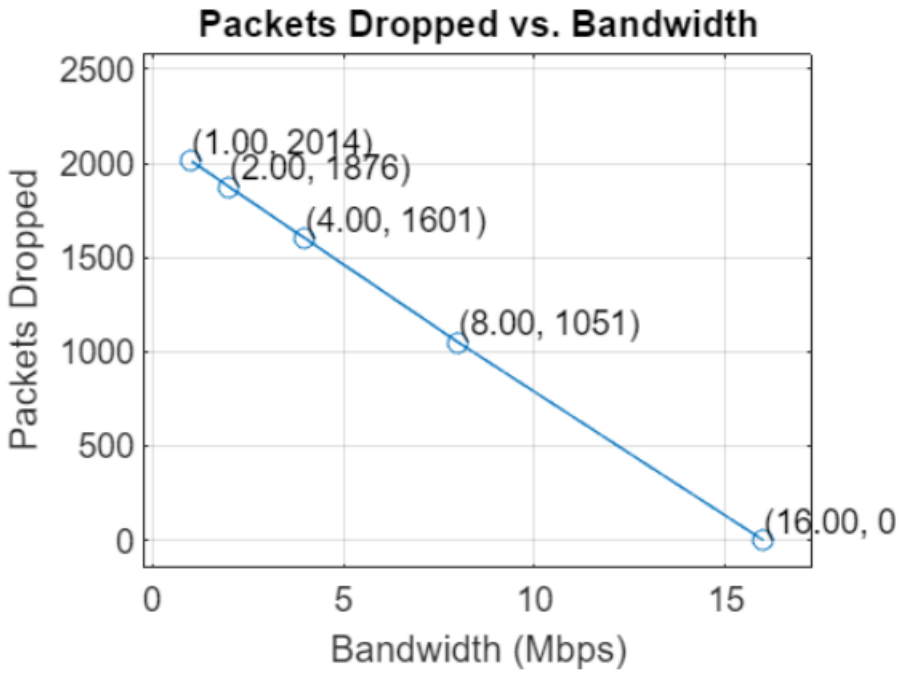
are added to the graph to display the values of bandwidth and packets dropped for each data point.

```
plot(bandwidth_values, packets_dropped, '-o');
xlabel('Bandwidth (Mbps)');
ylabel('Packets Dropped');
title('Packets Dropped vs. Bandwidth');
grid on;
for i = 1:length(bandwidth_values)
        x = bandwidth_values(i);
        y = packets_dropped(i);
        text(x, y, sprintf('(%.2f, %d)', x, y), 'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'left')
end
```

Output

```
>> pract2
Bandwidth: 16.00 bps
InRate: 2.00
OutRate: 2.00
Transmission Delay: 0.000500 s
Packets Dropped: 0
```

```
Bandwidth: 8.00 bps
InRate: 2.00
OutRate: 1.00
Transmission Delay: 0.001000 s
Packets Dropped: 1051

Bandwidth: 4.00 bps
InRate: 2.00
OutRate: 0.50
Transmission Delay: 0.002000 s
Packets Dropped: 1601

Bandwidth: 2.00 bps
InRate: 2.00
OutRate: 0.25
Transmission Delay: 0.004000 s
Packets Dropped: 1876

Bandwidth: 1.00 bps
InRate: 2.00
OutRate: 0.12
Transmission Delay: 0.008000 s
Packets Dropped: 2014
```

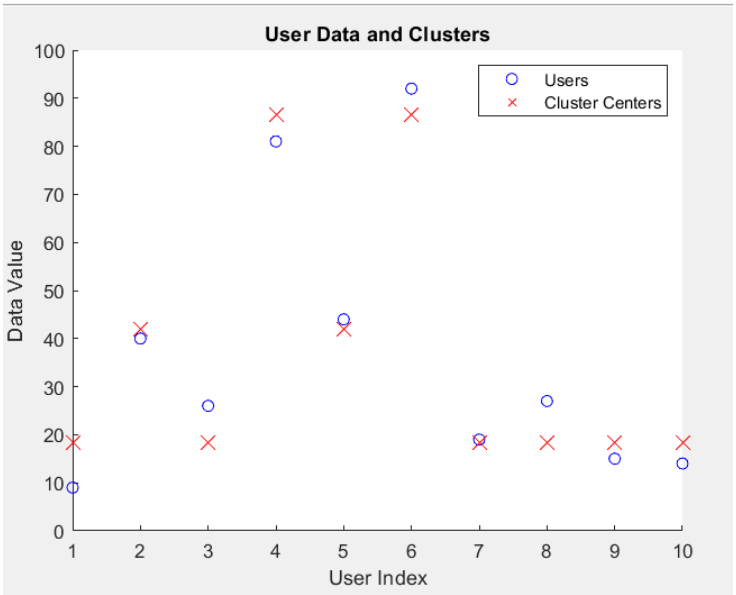


PRACTICAL 4

Implementation of FDMA, TDMA and CDMA using MATLAB

Code

```
numUsers = 10;
numClusters = 3;
userData = randi([1, 100], numUsers, 1);
[idx, centers] = kmeans(userData, numClusters);
figure;
scatter(1:numUsers, userData, 'bo', 'DisplayName', 'Users');
hold on;
scatter(1:numUsers, centers(idx), 100, 'rx', 'DisplayName', 'Cluster Centers');
title('User Data and Clusters');
xlabel('User Index');
ylabel('Data Value');
legend;
```



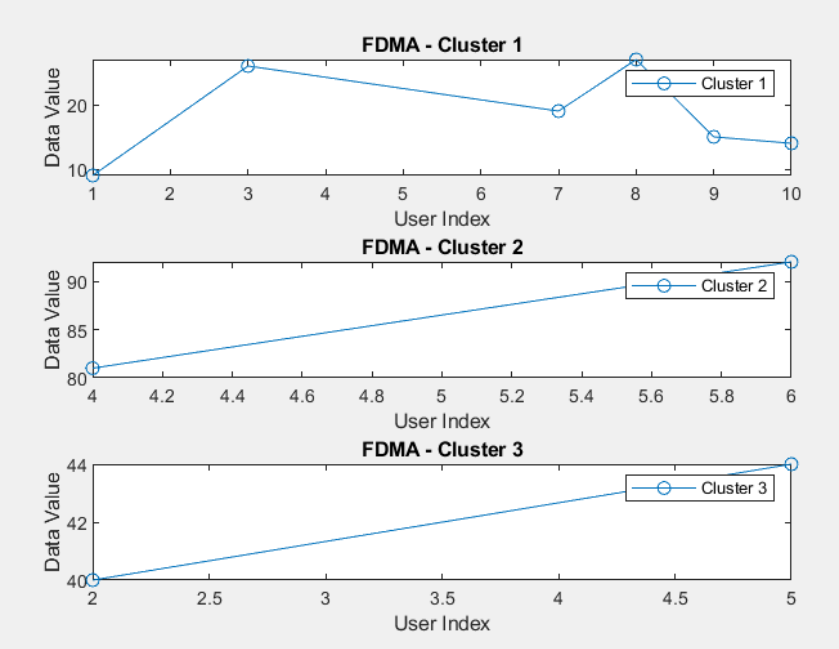
Frequency Division Multiple Access (FDMA)

Code

```
figure;
for i = 1:numClusters
    subplot(numClusters, 1, i);
    usersInCluster = find(idx == i);
    plot(usersInCluster, userData(usersInCluster), 'o-', 'DisplayName', sprintf('Cluster %d', i));
    title(sprintf('FDMA - Cluster %d', i));
    xlabel('User Index');
    ylabel('Data Value');
    legend;
 end
```

```
>> % Frequency Division Multiple Access (FDMA)
>> figure;
>> for i = 1:numClusters
       subplot(numClusters, 1, i);
       usersInCluster = find(idx == i);
       plot(usersInCluster, userData(usersInCluster), 'o-', 'DisplayName', sprintf('Cluster %d', i));
       title(sprintf('FDMA - Cluster %d', i));
       xlabel('User Index');
       ylabel('Data Value');
       legend;
   end
>>
```

Output



Time Division Multiple Access (TDMA)

Code

figure;

for i = 1:numClusters

    subplot(numClusters, 1, i);

    usersInCluster = find(idx == i);

    timeSlots = 1:length(usersInCluster);

plot(timeSlots, userData(usersInCluster), 'o-', 'DisplayName', sprintf('Cluster %d', i));

title(sprintf('TDMA - Cluster %d', i));

xlabel('Time Slot');

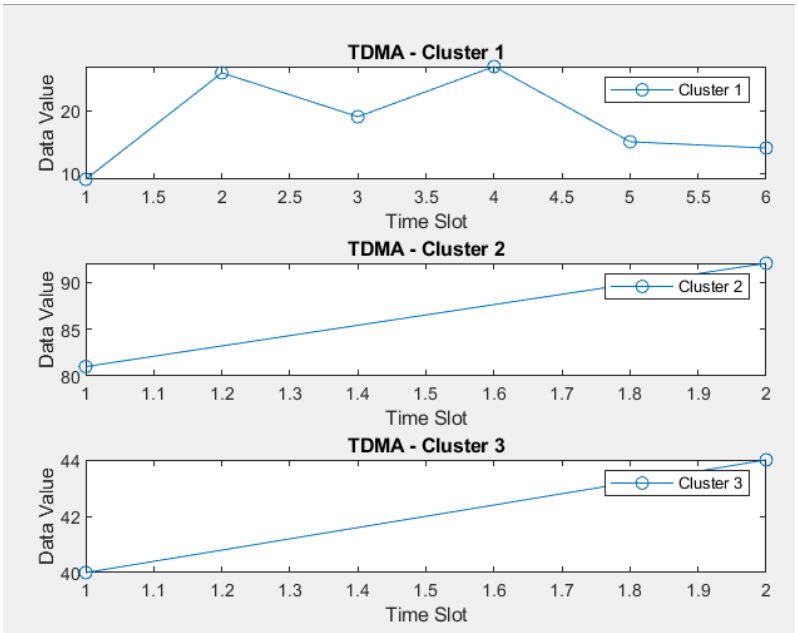
ylabel('Data Value');

legend;

end

```
>> % Time Division Multiple Access (TDMA)
>> figure;
>> for i = 1:numClusters
       subplot(numClusters, 1, i);
       usersInCluster = find(idx == i);
       timeSlots = 1:length(usersInCluster);
       plot(timeSlots, userData(usersInCluster), 'o-', 'DisplayName', sprintf('Cluster %d', i));

       title(sprintf('TDMA - Cluster %d', i));
       xlabel('Time Slot');
       ylabel('Data Value');
       legend;
   end
```

Output



Code Division Multiple Access (CDMA)

Code

figure;

spreadCodes = randi([0, 1], numUsers, numUsers);

cdmaData = spreadCodes * userData;

plot(1:numUsers, cdmaData, 'o-', 'DisplayName', 'CDMA');

title('Code Division Multiple Access (CDMA)');

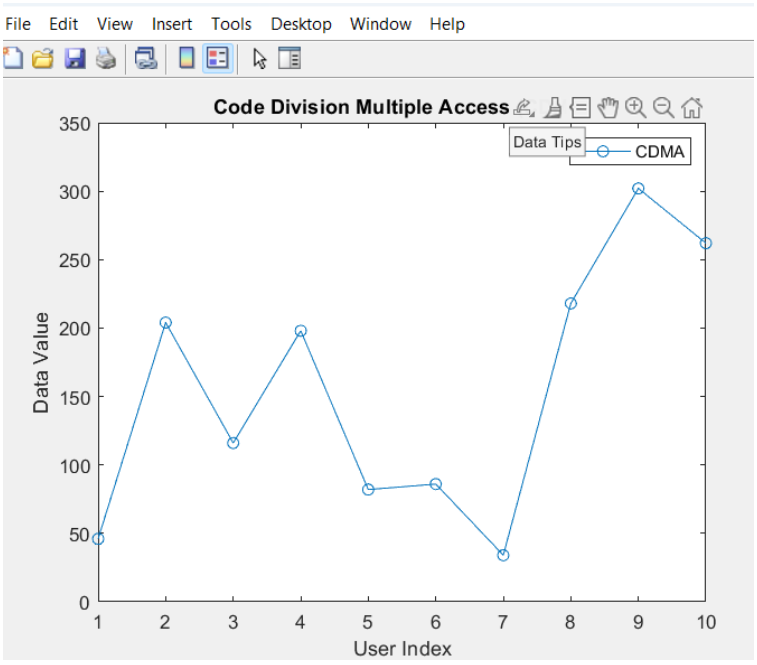
xlabel('User Index');

ylabel('Data Value');

legend;

```
>> % Code Division Multiple Access (CDMA)
>> figure;
>> spreadCodes = randi([0, 1], numUsers, numUsers);
>> cdmaData = spreadCodes * userData;
>> plot(1:numUsers, cdmaData, 'o-', 'DisplayName', 'CDMA');
>> title('Code Division Multiple Access (CDMA)');
>> xlabel('User Index');
>> ylabel('Data Value');
>> legend;
```

Output



PRACTICAL 5

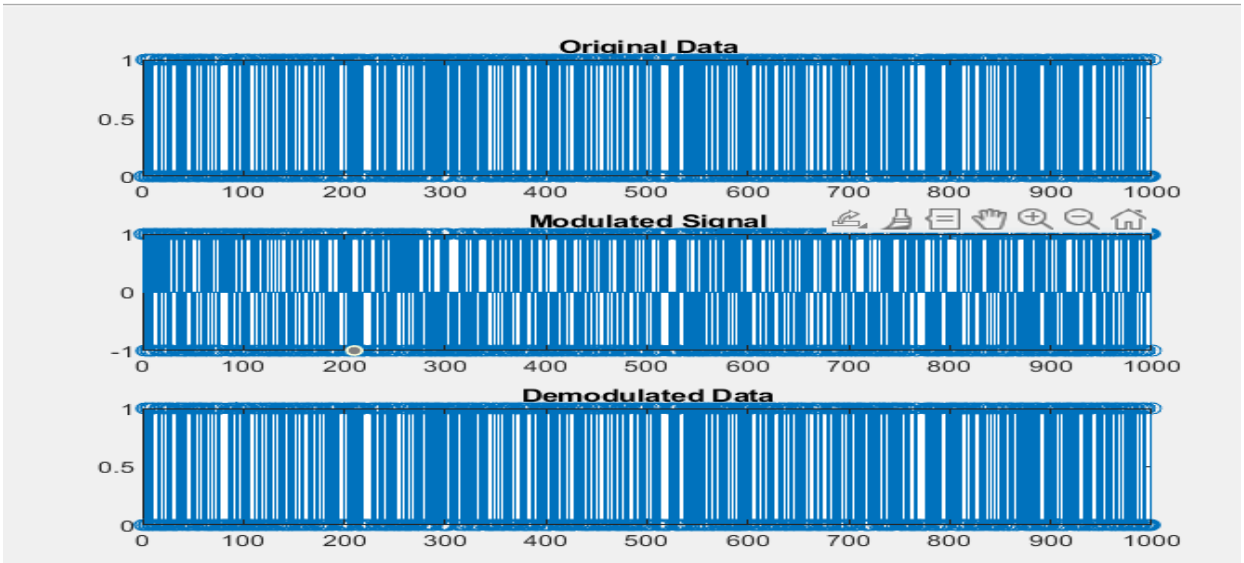
Implement GSM using MATLAB

CODE

```
numBits = 1000;

EbNo = 10; % Energy per bit to noise power spectral density ratio (dB)

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

modulatedSignal = pskmod(data, 2);

noisySignal = awgn(modulatedSignal, EbNo, 'measured');

demodulatedSignal = pskdemod(noisySignal, 2);

ber = biterr(data, demodulatedSignal) / numBits;

disp(['Bit Error Rate (BER): ' num2str(ber)]);

figure;


subplot(3, 1, 1);

stem(data);

title('Original Data');


subplot(3, 1, 2);

stem(modulatedSignal);

title('Modulated Signal');


subplot(3, 1, 3);
```

```
stem(demodulatedSignal);
title('Demodulated Data');
```

OUTPUT



PRACTICAL 6

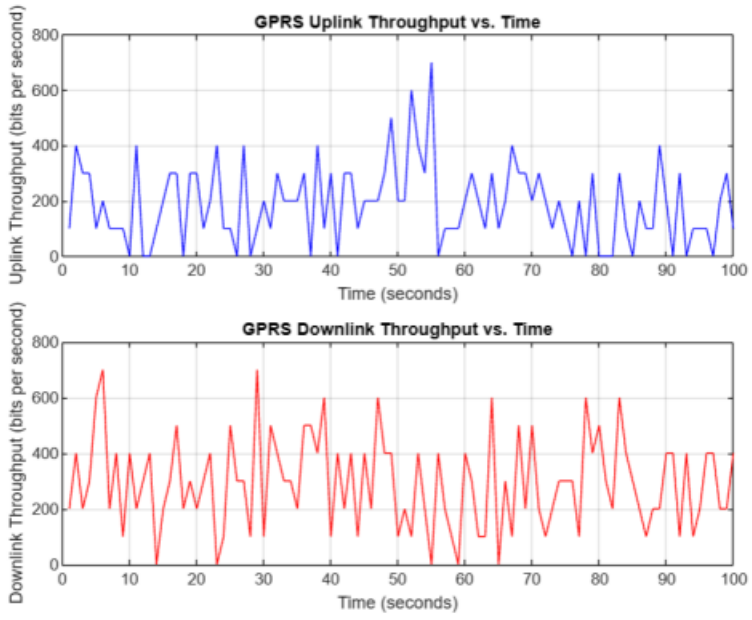
MATLAB Code for GPRS using MAC Layer

CODE

```
% 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



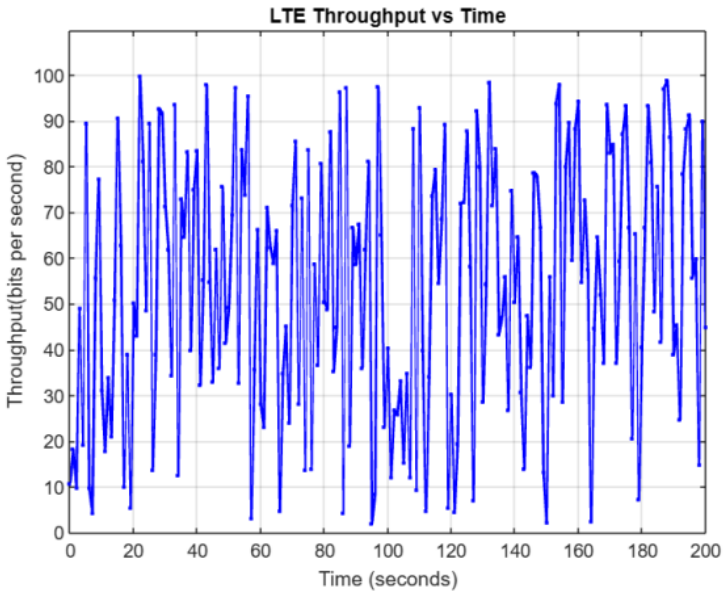
Practical 7

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

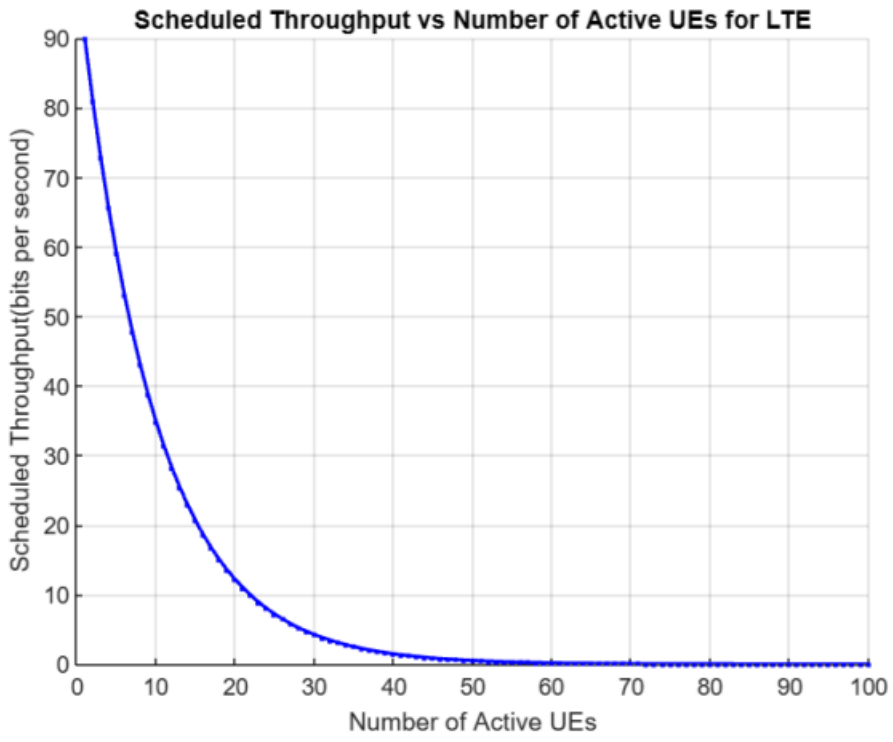


MATLAB Code for LTE(Throughput vs Number Of Users):

```
% 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



Practical-8:

AIM-IMPLEMENT SNOOPING AND ANALYSING THE TRAFFIC USING WIRESHARK

Launch Wireshark:

Start Wireshark with administrative privileges, especially if you're using Windows or need to capture traffic on certain interfaces.

Select the Network Interface:

In Wireshark, select the network interface you want to capture traffic from. This could be your Wi-Fi or Ethernet connection.

Start Capturing:

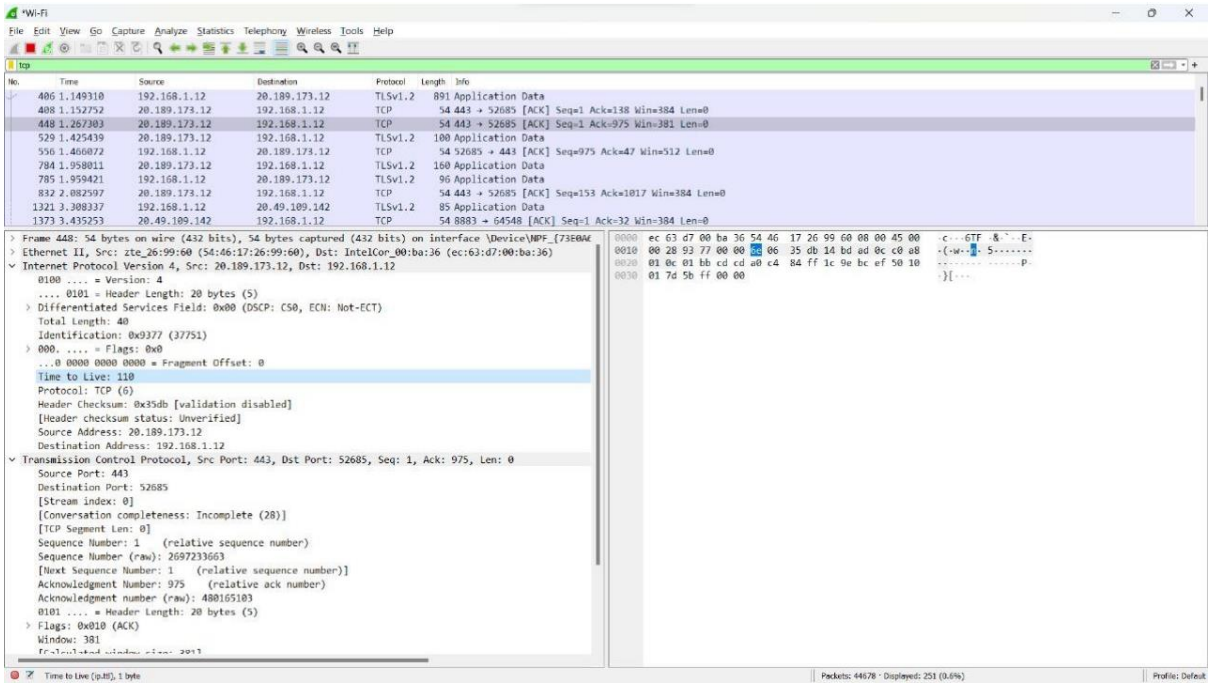
Click on the "Start" or "Capture" button to begin capturing network traffic.

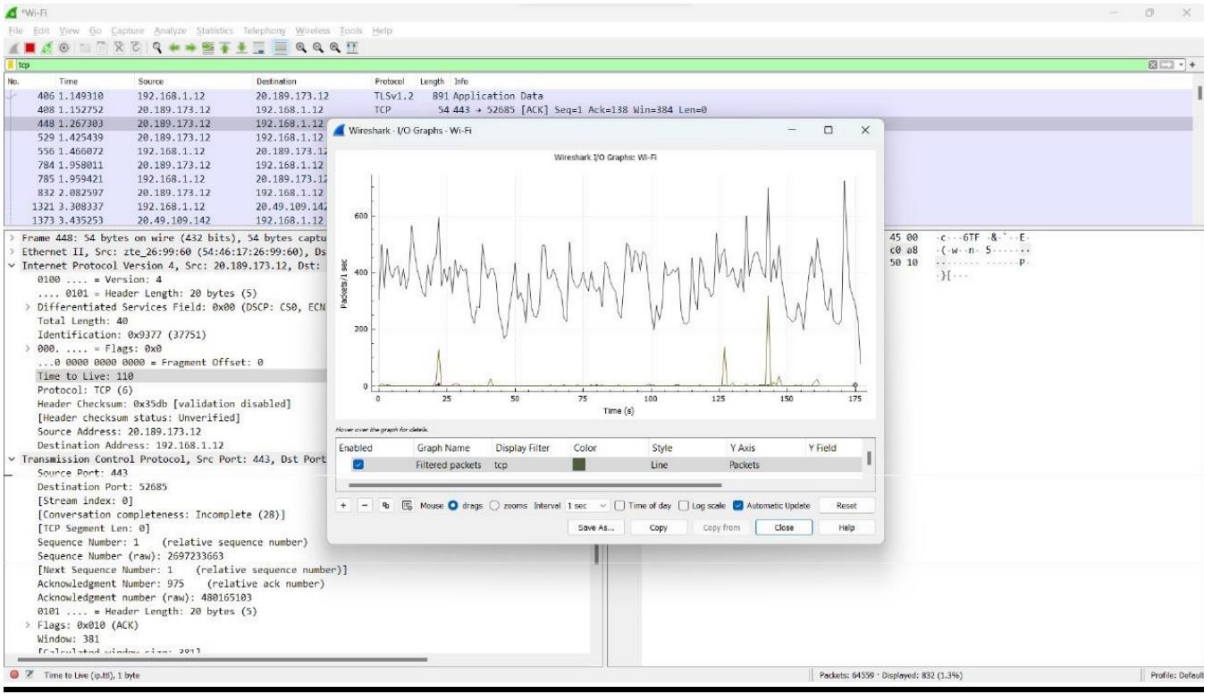
Stop Capturing:

Once you've captured the desired traffic, click the "Stop" or "Capture" button to stop the capture.

Analyze the Traffic:

You can now analyze the captured traffic. You can look at individual requests and responses, examine the headers, see the content, and much more


*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_{73E0AA
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
Frame 448: 54 bytes on wire (432 bits), 54 bytes captu
Ethernet II, Src: zte_26:99:60 (54:46:17:26:99:60), Ds
Internet Protocol Version 4, Src: 20.189.173.12, Dst:
Transmission Control Protocol, Src Port: 443, Dst Port
Time to Live (ip.ttl), 1 byte
Packets: 64559 · Displayed: 832 (1.3%)
Profile: Default