JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY GURAJDA VIZIANAGARAM

UNIVERSITY COLLEGE OF ENGINEERING VIZIANAGARAM

Department of
INFORMATION TECHNOLOGY

Computer Networks Record

Bachelor of technology
INFORMATION TECHNOLOGY

REGD NO: 20VV1A1263

Certificate

Certified that this is a Bonafide Record of practical work done by

Mr./ Kumari VENKATA SAI YASWANTH BATTU

of II B TECH II SEMESTER class in Computer Networks Laboratories

of JNTUK-UCEV College during the Academic Year 2021-2022

No of experiments done and certified:

Lecturer in-Charge: Mr. W. Anil Head of Department : Dr. G. Jaya Suma

Date:

Index

S.No	Date	Name of the Experiment	Page	Marks	Signature
1		Implement the data link layer framing methods such as character stuffing and bit stuffing.			
2		Write a C program to develop a DNS client server to resolve the given hostname.			
3		Implement on a data set of characters the three CRC polynomials – CRC-12, CRC-16 and CRC-CCIP.			

4		Implement Dijkstra's algorithm to compute the Shortest path in a graph			
5		Take an example subnet graph with weights indicating delay between nodes. Now obtain Routing table art each node using distance vector routing algorithm			
6		Take an example subnet of hosts. Obtain broadcast tree for it.			
7		Write a client-server application for chat using UDP			
8		Write a C program to perform sliding window protocol.			
9		Write a C program to perform Go-Back-N Protocol.			
10		Write a C program to perform Selective-Repeat protocol.			
11		Get the MAC or Physical address of the system using Address Resolution Protocol			
12		Simulate the Implementing Routing Protocols using border gateway protocol(BGP)			
13		Simulate the OPEN SHORTEST PATH FIRST routing protocol based on the cost assigned to the path.			

EXPERIMENT NO - 1

AIM: Implement the data link layer framing methods such as character stuffing and bit stuffing

Description:

Byte stuffing is a mechanism to convert a message formed of a sequence of bytes that may contain reserved values such as frame delimiter, into another byte sequence that does not contain the reserved values.

Bit stuffing is the mechanism of inserting one or more non-information bits into a message to be transmitted, to break up the message sequence, for synchronization purpose.

In character stuffing “E” is known as escape character and “F” as flag bytes.

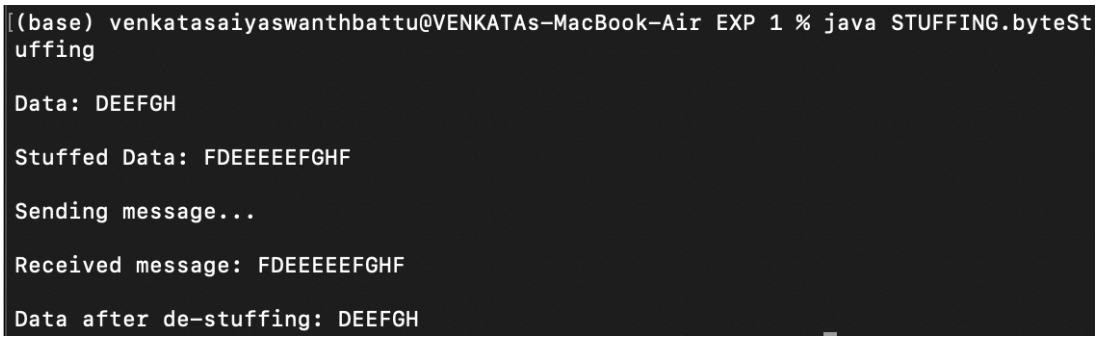
In bit stuffing the pattern “01111110” is considered as flag bytes, essentially determines start and end of data frames.

Program:(byte stuffing)

```
package STUFFING;
import java.util.*;
class byteStuffing {
String std;
// considering 'E' as escape Sequence and 'F' as flag byte
void charStuff(String data){

data = "F"+data+"F";
for(int i = 1; i < data.length()-1; i++){ // exclude first and last flag indexes
if( data.charAt(i) == 'F' || data.charAt(i) == 'E'){
data = data.substring(0, i) + "E" + data.substring(i); i++; // for avoiding pointing to same byte again
}
}
System.out.println("\nStuffed Data: "+data);
System.out.println("\nSending message...");
this.std = data;
}
void charDeStuff(){
System.out.println("\nReceived message: "+this.std);
String data = this.std.substring(1,this.std.length()-1);
for (int i = 0; i < data.length()-1; i++) { // exclude first and last flag indexes if ( data.charAt(i) == 'E') {
String verify = data.substring(i, i+2);
if( verify.equals("EF") || verify.equals("EE")){
data = data.substring(0, i) + data.substring(i+1);
}

}
}
System.out.println("\nData after de-stuffing: "+data); public static void main(String[] args) throws Exception{
byteStuffing bs = new byteStuffing();
System.out.print("\nData: ");
Scanner sc = new Scanner(System.in);
```

```
String data = sc.nextLine();

bs.charStuff(data);
bs.charDeStuff();
}
}
```

Output:

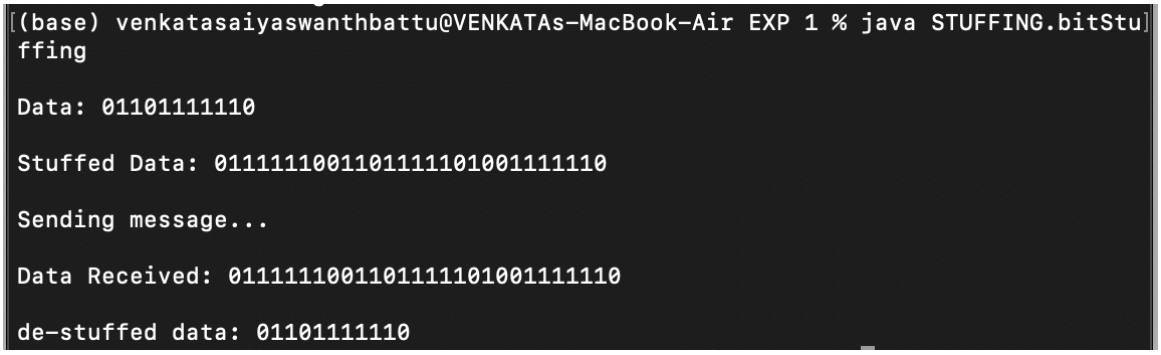


Program: (Bit stuffing)

```
package STUFFING;
import java.util.*;

class bitStuffing{
String std;
void bitStuff(String s){
String flag = "01111110";
for(int i = 0; i <= s.length()-6; i++){
if(s.charAt(i) == '1'){
if( s.substring(i,i+5).equals(“11111")){
s = s.substring(0,i+5) + "0" + s.substring(I+5);
i+=6;
}
}
}
s = flag+s+flag;
System.out.println("\nStuffed Data: "+s);
System.out.println("\nSending message...");
this.std = s;
}
void bitDeStuff(){
System.out.println("\nData Received: "+this.std);
String data = this.std.substring(8,this.std.length()-8);

for(int i = 0; i <= data.length()-6; i++){
if (data.charAt(i) == '1') {
                                        if (data.substring(i, i + 6).equals("111110")) {
data = data.substring(0,i+5) + data.substring(i+6); i+=6;
}
```

```
}
}
System.out.println("\nde-stuffed data: "+data);
}
public static void main(String... args){
// 10000001 is flag byte
Scanner sc = new Scanner(System.in);
System.out.print("\nData: ");
String data = sc.nextLine();

bitStuffing bs = new bitStuffing();
bs.bitStuff(data);
bs.bitDeStuff();
}
}
```

Output:



EXPERIMENT NO - 2

AIM: Write a Java program to develop a DNS client server to resolve the given host name.

Description:

DNS is abbreviated as domain name system.
The server offering the functionality of DNS is termed as DNS server. DNS is the phone book of the internet.
DNS is responsible for finding the correct IP address for these sites.
Browsers then use the addresses to communicate with original servers to access the website information.
In a typical DNS query, without any cachug, there are 4 servers that work together to deliver an IP address; recursive resolvers , deliver an IP address , TLD servers , authoritative servers.

Program: (Client)

```
package DNS;
import java.io.*;
import java.net.*;

public class DNSClient {
public static void main(String... args) throws Exception{
                        //    use socket to send dat gram packets
Socket s = new Socket("localhost", 6363);
OutputStreamWriter os = new OutputStreamWriter(s.getOutputStream()); PrintWriter out = new PrintWriter(os);

System.out.print("\nEnter the Domain Name:");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String domainName = br.readLine();
out.println(domainName);
out.flush();
// for receiving IP address from DNS server
BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream())); String ip = in.readLine();
System.out.println("IP Address of the Domain: "+domainName+" is: "+ip); s.close();
}
}
```

Output:

```
        at DNS.DNSClient.main(DNSClient.java:8)
(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air EXP 2 % java DNS.DNSClient

Enter the Domain Name:www.flipkart.com
IP Address of the Domain: www.flipkart.com is: 163.53.78.110
```

Program: (Server)

```
package DNS;
import java.net.*;
import java.io.*;
class DNSServer {
public static void main(String... args) throws Exception{

String[] domainNames =
{"www.amazon.in","www.apple.com","www.flipkart.com","wwww.google.com"};
String[] ipAddress =
{"52.95.120.67","17.253.144.10","163.53.78.110","173.194.198.139"};

ServerSocket ss = new ServerSocket(6363);
System.out.println("Server is running...");

System.out.println("Waiting for client to connect..."); Socket s = ss.accept();

System.out.println("Client connected...");

BufferedReader br = new BufferedReader(new InputStreamReader(s.getInputStream()));
String domainName = br.readLine();
System.out.println("Domain Name received: " + domainName);

for( int i = 0; i<domainNames.length; i++){
if(domainName.equals(domainNames[i])){
OutputStreamWriter os = new OutputStreamWriter(s.getOutputStream());
PrintWriter out = new PrintWriter(os);
out.println(ipAddress[i]);
out.flush();
break;
}
}
s.close();
ss.close();

}
}
```

Output:

```
Last login: Wed Aug  3 20:19:48 on ttys000
(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air EXP 2 % java DNS.DNSServer
Server is running...
Waiting for client to connect...
Client connected...
Domain Name received: www.flipkart.com
```

EXPERIMENT NO - 3

AIM: Implement on a data set of characters the three CRC polynomials- CRC12, CRC16, CRC-CCIP.

Description:

CRC is abbreviated as Cyclic Redundancy Check.

CRC is used for error detection is data transferred within the data link layer. A generator polynomial is used for generating a CRC bit.

Generated CRC bit appended back of the data and sent to the receiver. Receiver must use the same generating polynomial.

Unless the remainder after modulo 2 division at receiver is all 0's , the data contains errors or corrupted while transmitting.

Program:

```
package CRC;
import java.util.Scanner;
class CRC {

String cdw;
String xor(String a, String b) {
StringBuilder result = new StringBuilder();
for (int i = 1; i < b.length(); i++) {
if (a.charAt(i) == b.charAt(i))
result.append('0');
else
result.append('1');

}
return result.toString();
}

String crc(String message, String key) {
int pick = key.length();
String tmp = message.substring(0, key.length());
String zeroes = "";
while (pick < message.length()) {
if (tmp.charAt(0) == '1'){
tmp = this.xor(key, tmp) + message.charAt(pick);
}
else{
for (int i = 1; i <= key.length(); i++)
zeroes += "0";
tmp = this.xor(zeroes, tmp) + message.charAt(pick);
}

pick++;
}

if (tmp.charAt(0) == '1') {
tmp = this.xor(key, tmp);
```

```
} else {
for (int i = 1; i <= key.length(); i++)
zeroes += "0";
tmp = this.xor(zeroes, tmp);
}
return tmp;

}

void encodedData(String data, String key) {
int l_key = key.length();
String appended_zeroes = "";
for (int i = 1; i < l_key; i++)
appended_zeroes += “0";
String append_data = data + appended_zeroes; String remainder = this.crc(append_data, key); String code_word
    = data + remainder; System.out.println("Code word: " + code_word); System.out.println("Remainder: " +
    remainder); this.cdw = code_word;
}

void reciverSide(String data,String key){
int l_key = key.length();
String remainder = this.crc(data, key);
String appended_zeroes = "";
for (int i = 1; i < l_key; i++)
appended_zeroes += "0";
if( remainder.equals(appended_zeroes ))
{
System.out.println("Data is Correct");
}
else
{
System.out.println("Data is Incorrect");
}
System.out.println("Code word: " + data);
System.out.println("Remainder: " + remainder);
}
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.print("Enter the data: ");
String data = sc.nextLine();
System.out.print("Enter the generator polynomial: “);
String key = sc.nextLine();
System.out.println("----------");
System.out.println("Encoding Sender Side: ");
CRC c = new CRC();
```

```
c.encodedData(data, key);
System.out.println("------------");
System.out.println("Decoding Receiver Side:");
c.reciverSide(c.cdw, key);
}
}
```

Output:

```
(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air EXP 3 % java CRC.CRC
Enter the data: 00000001
Enter the generator polynomial: 1101
------------
Encoding Sender Side:
Code word: 00000001101
Remainder: 101
------------
Decoding Receiver Side:
Data is Correct
Code word: 00000001101
Remainder: 000
```

EXPERIMENT NO - 4

AIM: Implement of Dijkstra's algorithm to compute the shortest path in graph.

Description:

Dijkstra's algorithm allows us to find shortest path between any two vertices of a graph.

It differs from the minimum spanning true because the shortest distance between any two vertices might not include all the vertices of the graph.

In computer networks, the Dijkstra's algorithm finds the shortest path from a particular node, called the source node to every other node in a connected graph. It produces a shortest path true with the source node as the root.

Minimizes routing lasts.

Program:

```
package
Dijkstra;
import java.util.*;

class Node implements Comparator<Node>
{
private int v;
private int weight;

Node(int _v, int _w) { v = _v; weight = _w;};

Node(){}

int getV(){ return v; }
int getWeight(){ return weight; }

@Override
public int compare(Node node1, Node node2){
if( node1.weight < node2.weight )
return -1;
if( node1.weight > node2.weight)
return 1;
return 0;
}
}
```

```
class Dijkstra{
// method to find the shortest path
void shortestPath(int s, ArrayList<ArrayList<Node>> adj, int N){
//distace array -- stores shortest distance to every other node fom source
int dist[] = new int[N];
//initialising every at start is at inifinity

for(int i = 0;i < N; i++) dist[i] = 10000000;
dist[s] = 0; // making the distacne from source to itself as 0
//priority queue for storing distance updates nodes
PriorityQueue(int initialCapacity, Comparator<E> comparator) PriorityQueue<Node> pq = new PriorityQueue<Node>(N,
    new Node());

pq.add(new Node(s,0));

while(pq.size()>0){
Node node = pq.poll();

for(Node it: adj.get(node.getV())){
if(dist[node.getV()] + it.getWeight() < dist[it.getV()]){
dist[it.getV()] = dist[node.getV()] + it.getWeight(); pq.add(new Node(it.getV(),dist[it.getV()]));
}
}
}
System.out.print("The distances from source "+s+" are: "); for(int i =0 ; i < N; i++){
System.out.print(dist[i]+ " ");
}
}

public static void main(String... args){
        //  no of nodes in the graph int n = 5;
        //  creating an Adjacency List of each data item of type Node ArrayList<ArrayList<Node>> adj = new
            ArrayList<ArrayList<Node>>();

for(int i = 0 ; i < n; i++)
adj.add(new ArrayList<Node>());

// inserting nodes

adj.get(0).add(new Node(1,2));
adj.get(1).add(new Node(0,2));
adj.get(1).add(new Node(2,4));
adj.get(2).add(new Node(1,4));
adj.get(0).add(new Node(3,1));
adj.get(3).add(new Node(0,1));
adj.get(3).add(new Node(2,3));
adj.get(2).add(new Node(3,3));
adj.get(1).add(new Node(4,5));
adj.get(4).add(new Node(1,5));
adj.get(2).add(new Node(4,1));
```

```
adj.get(4).add(new Node(2,1));


Dijkstra dj = new Dijkstra();

dj.shortestPath(0,adj,n);
}
}
```

Output:

```
(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air EXP 4 % java Dijkstra.Dijkst
ra
The distances from source 0 are: 0 2 4 1 5 %
```

EXPERIMENT NO - 5

AIM: write a client -server application for chat using UDP.

Description:

Data gram socket are java's mechanism for network communication via UDP instead of TCP.

Java provides data gram socket to communicate over UDP instead of TCP. Data gram sockets can be used to both send and receive the packet.

Datagram packet is used for creating packets to send and receive using sockets.

UDP chat employees the User Data gram Protocols, where communication between server and client happens using connectionless protocol.

Program: (client)

```
package
UDP;
import java.io.*;
import java.net.*;

class UDPClient{
static DatagramSocket ds;
static DatagramPacket dp;
static BufferedReader br;
static InetAddress ia;
static byte[] b = new byte[1024];
static int cport = 6363, sport = 6464;

public static void main(String... args) throws Exception{ ds = new DatagramSocket(cport);
dp = new DatagramPacket(b,b.length);
br = new BufferedReader(new InputStreamReader(System.in));
ia = InetAddress.getLocalHost();
System.out.println("Client is Running ... Type'STOP' to Quit");
while(true){
String str = new String(br.readLine());
b = str.getBytes();
if( str.equals("STOP"))
{
System.out.println("Terminated...");
ds.send(new DatagramPacket(b,b.length,ia,sport));
break;
}
ds.send(new DatagramPacket(b, b.length,ia,sport));
ds.receive(dp);
str = new String(dp.getData());
```

```
System.out.println("Server: "+str);
}
}

}
```

Output:

```
(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air EXP 7 % java UDP.UDPClient
Client is Running ... Type'STOP' to Quit
hey
Server: hello
hiii
Server: wonderful
beautifu
Server: hhaharful
harmony
Server: STOParful
STOP
Terminated...
```

Program: (SERVER)

```
package
UDP;
import java.io.*;
import java.net.*;

class UDPServer{
static DatagramSocket ds;
static DatagramPacket dp;
static BufferedReader br;
static InetAddress ia;
static byte[] b = new byte[1024];
static int cport = 6363;
static int sport = 6464;

public static void main(String[] args) throws Exception{ ds = new DatagramSocket(sport);
dp = new DatagramPacket(b,b.length);
br = new BufferedReader(new InputStreamReader(System.in));
ia = InetAddress.getLocalHost();
System.out.println("Server is Running .. Type STOP to Quit");
while(true){
String str = new String(br.readLine());
b = str.getBytes();
if( str.equals("STOP")){
System.out.println("Terminated..");
ds.send(new DatagramPacket(b,b.length,ia,cport));
break;
}

ds.send(new DatagramPacket(b,b.length,ia,cport));
ds.receive(dp);
str = new String(dp.getData());
System.out.println("Client: “+str);
}
}
}
```

```
(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air EXP 7 % java UDP.UDPServer
Server is Running .. Type STOP to Quit
hello
Client: hey
wonderful
Client: hiii
hhaha
Client: beautifu
STOP
Terminated..
```

Output:

EXPERIMENT NO - 6

AIM: Take an example of subnet of hosts obtain broadcast free for it.

Description:

A subnet of hosts consists a network, where a limited number of hosts hold connections to communicate with each other.

Broadcast tree is the minimum spanning tree obtained from the graph.

Using minimum spanning tree, communication from a node to any other node happens at the minimal costs.

Either of prims and kruskals algorithm is can be used for obtaining the broadcast tree.

Program:

```
packageMST;
import java.util.*;
class Node {
private int u;
private int v;
private int weight;
Node(int _u, int _v, int _w) {
u = _u;
v = _v;
weight = _w;
}
Node() {
}
int getV() {
return v;
}
int getU() {
return u;
}
int getWeight() {
return weight;
}
}
class SortComparator implements Comparator<Node> {
@Override
public int compare(Node node1, Node node2) {

if (node1.getWeight() < node2.getWeight())
    return -1;
if (node1.getWeight() > node2.getWeight())
    return 1;
return 0;
}
}
class MST {
private int findPar(int u, int parent[]) {
```

```
if (u == parent[u])
return u;
return parent[u] = findPar(parent[u], parent);
}

private void union(int u, int v, int parent[], int rank[]) {
u = findPar(u, parent);
v = findPar(v, parent);
if (rank[u] < rank[v]) {
parent[u] = v;
}
else if (rank[v] < rank[u]) {
parent[v] = u;
}
else {
parent[v] = u;
rank[u]++;
}
}
void KruskalAlgo(ArrayList<Node> adj, int N) { Collections.sort(adj, new SortComparator()); int parent[] = new
    int[N];
int rank[] = new int[N];

for (int i = 0; i < N; i++) {
parent[i] = i;
rank[i] = 0;
}
int costMst = 0;
ArrayList<Node> mst = new ArrayList<Node>();
for (Node it : adj) {
if (findPar(it.getU(), parent) != findPar(it.getV(), parent)) { costMst += it.getWeight();
mst.add(it);
union(it.getU(), it.getV(), parent, rank);
}
}
System.out.println(costMst);
for (Node it : mst) {
System.out.println(it.getU() + " - " + it.getV());
}

}
public static void main(String args[]) {
int n = 5;
ArrayList<Node> adj = new ArrayList<Node>();
adj.add(new Node(0, 1, 2));
adj.add(new Node(0, 3, 6));
adj.add(new Node(1, 3, 8));
adj.add(new Node(1, 2, 3));
adj.add(new Node(1, 4, 5));
adj.add(new Node(2, 4, 7));
MST obj = new MST();
obj.KruskalAlgo(adj, n);
```

```
}
}
```

Output:

```
(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air exp 6 % java MST.MST
16
0 - 1
1 - 2
1 - 4
0 - 3
```

EXPERIMENT NO - 7

AIM: Get the MAC or Physical address of the system using Address Resolution Protocol.

Description:

MAC is abbreviated as Media Access Control.

It is a unique identifier assigned to a network interface controller (NIC) for use a network address in communications within a network segment.

Most of the computer applications use logical address to send/receive messages, however the actual communications happens over physical address.

ARP is abbreviated as Address Resolution protocal, where logical address given is resolved into MAC address of the node.

Program: (Client)

```
package
ARP;
import java.io.BufferedReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.Socket;
import java.io.InputStreamReader;

public class ARPClient {
public static void main(String... args)throws Exception{
Socket s = new Socket("localhost",6363);
// for sending IP to ARP server
OutputStreamWriter os = new OutputStreamWriter(s.getOutputStream());
PrintWriter out = new PrintWriter(os);

System.out.println("Enter the IP Address:");

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
String ip = br.readLine();
out.println(ip);
out.flush();

// for receiving MAC address from ARP server
BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream())); String mac = in.readLine();
System.out.println("MAC Address of the IP: "+ip+" is: "+mac); s.close();
}
}
```

Output:

```
[(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air EXP 10 % java ARP.ARPClient
Enter the IP Address:
165.165.79.1
MAC Address of the IP: 165.165.79.1 is: 8A:BC:E3:FA
```

Program: (Server)

```
package
ARP;
import java.net.ServerSocket;
import java.io.BufferedReader;
import java.net.Socket;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
public class ARPServer {
public static void main(String[] args) throws Exception{ String ipArray[] = {"165.165.80.80","165.165.79.1"}; String
    macArray[] = {"6A:08:AA:C2","8A:BC:E3:FA"};
ServerSocket ss = new ServerSocket(6363);
System.out.println("Server is running...");

System.out.println("Waiting for client to connect…");
 Socket s = ss.accept();

System.out.println("Client connected...");

BufferedReader br = new BufferedReader(new InputStreamReader(s.getInputStream()));
String ip = br.readLine();
System.out.println("IP Address received: "+ip);

                        //   for sending MAC address to client
                    //   Verify if any ip matches with available ipArray for( int i = 0; i<ipArray.length; i++){
if(ip.equals(ipArray[i])){
OutputStreamWriter os = new OutputStreamWriter(s.getOutputStream()); PrintWriter out = new PrintWriter(os);
    out.println(macArray[i]);

out.flush();
break;
}
}
s.close();
ss.close();

}
}
```

Output:

```
(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air EXP 10 % java ARP.ARPServer
Server is running...
Waiting for client to connect...
Client connected...
IP Address received: 165.165.79.1
```

EXPERIMENT NO - 8

AIM: Write a Java program for stop and wait ARQ

Description:

In stop and wait protocol, the sender stops and wait for the acknowledgment for the frame being sent.

Until unless the acknowledgment is received the sender won't send the next frame. In ARQ, an timer is set at the sender side, after the time out period, the sender

transmits the frame again.

If acknowledgment is received, the sender immediately transmits the next frame. Stop and Wait is an Connection Oriented Protocol.

Program:(Client)

```
package
SAW;
import java.io.*;
import java.net.*;
import java.util.*;
public class SAWClient {
static Socket s;
static BufferedReader br;
static OutputStreamWriter os;
static PrintWriter out;
static int n;

public static void main(String[] args) throws Exception{ s = new Socket("localhost",6363);
Scanner sc = new Scanner(System.in);
System.out.println("Client Started..");
System.out.println("Enter No of Frames: ");

n = Integer.parseInt(sc.nextLine());

int i = 1, frameNo = 0;
os = new OutputStreamWriter(s.getOutputStream()); out = new PrintWriter(os);

out.println(n);
out.flush();
br = new BufferedReader(new InputStreamReader(s.getInputStream()));

while( i <= n){
int cf = frameNo;
System.out.println("Sending frame : "+(cf%2));
```

```
out.println(cf%2);
out.flush();
System.out.println("Recived Ack "+br.readLine()); i++; frameNo++;
}
}
}
```

Output:

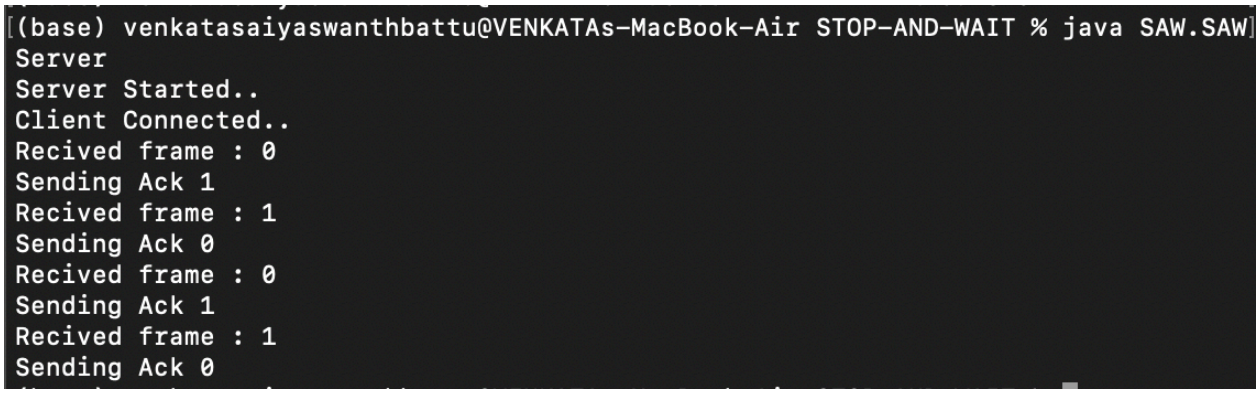
```
(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air STOP-AND-WAIT % java SAW.SAW
Client
Client Started..
Enter No of Frames:
4
Sending frame : 0
Recived Ack 1
Sending frame : 1
Recived Ack 0
Sending frame : 0
Recived Ack 1
Sending frame : 1
Recived Ack 0
```

Program: (Server)

```
package
SAW;
import java.io.*;
import java.net.*;
import java.util.*;

public class SAWServer {
static ServerSocket ss;
static Socket s;
static BufferedReader br;
static OutputStreamWriter os;
static PrintWriter out;

public static void main(String[] args) throws Exception {
                        ss. = new ServerSocket(6363); System.out.println("Server Started..");

s = ss.accept();
System.out.println("Client Connected..");
int i = 1;
os = new OutputStreamWriter(s.getOutputStream()); out = new PrintWriter(os);
br = new BufferedReader(new InputStreamReader(s.getInputStream())); int n = Integer.parseInt(br.readLine());

while (i <= n) {
int cf = Integer.parseInt(br.readLine());
```

```
System.out.println("Recived frame : " + (cf) % 2);
out.println((cf+1)%2);
out.flush();
System.out.println("Sending Ack " + (cf + 1) % 2); i++;

}
}
}
```

Output:



EXPERIMENT NO - 9

AIM: Write a Java program for Go Back N protocol.

Description:

In Go back N, a straight variation to simple stop and wait ARQ happens. Window size at sender is 2^n – 1 and the receiver is 1.

In Go back N, sender keeps a timer after sending all the frames of a window.

If acknowledgement of any frame of the window is lost, entire frames of the window are transmitted by the sender.

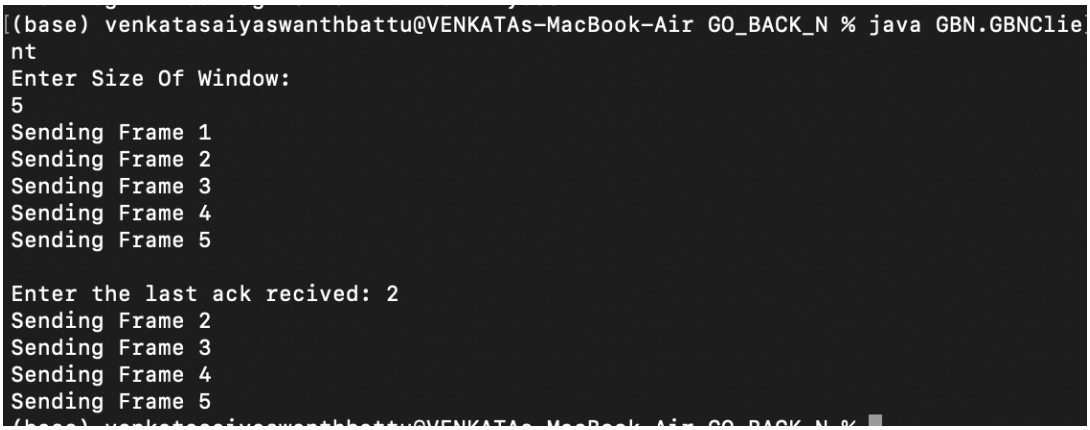
Increased efficiency than stop and wait ARQ.

Program:(Client)

```
package
GBN;
import java.net.*;
import java.io.*;
import java.util.Scanner;

public class GBNClient {

static Socket s ;
static OutputStreamWriter os;
static PrintWriter out ;
static BufferedReader br;


static void sendFrame(int s,int ws){
int i = s;
while(i<=ws){
System.out.println("Sending Frame " + i);
out.println(i);
out.flush();
i++;
}
}
public static void main(String... args) throws Exception{ s = new Socket("localhost", 6363);
os = new OutputStreamWriter(s.getOutputStream());
out = new PrintWriter(os);
br = new BufferedReader(new InputStreamReader(s.getInputStream())); Scanner sc = new Scanner(System.in);
    System.out.println("Enter Size Of Window: "); int ws = Integer.parseInt(sc.nextLine());

                    //   sending window size to server out.println(ws); out.flush();

sendFrame(1, ws);
                        //   for re transmitting
```

```
System.out.print("\nEnter the last ack recived: "); sendFrame(Integer.parseInt(sc.nextLine()),ws);

}
}
```

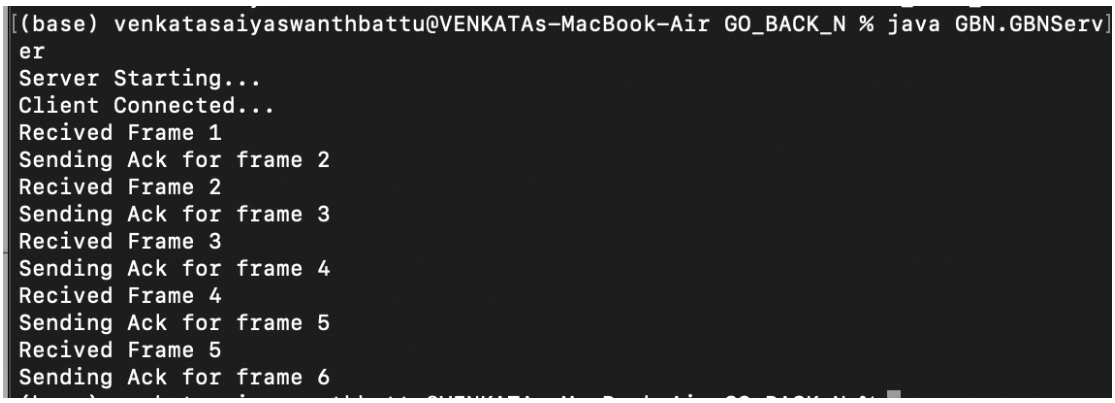
Output:



Program:(Server)

```
package
GBN;
import java.net.*;
import java.io.*;
import java.util.Scanner;

public class GBNServer {

static ServerSocket ss;
static Socket s;
static OutputStreamWriter os ;
static PrintWriter out;
static BufferedReader br;

static void receiveFrame(int ws) throws Exception{ int i = 1;
while(i<= ws){
int fno = Integer.parseInt(br.readLine()); System.out.println("Recived Frame "+fno); System.out.println("Sending
    Ack for frame "+(fno+1)); i++;

}
}
public static void main(String... args) throws Exception{ System.out.println("Server Starting...");
                    ss. = new ServerSocket(6363); s = ss.accept(); System.out.println("Client Connected...");

os = new OutputStreamWriter(s.getOutputStream()); out = new PrintWriter(os);
br = new BufferedReader(new InputStreamReader(s.getInputStream()));

receiveFrame(Integer.parseInt(br.readLine()));
}
}
```

Output:



EXPERIMENT NO - 10

AIM: Write a Java program for Selective repeat Protocol.

Description:

In Selective Repeat, a particular frame which was lost during transmission is only retransmitted.

In this way, instead of transmission of entire window, only particular frames are re transmitted.

Better Bandwidth utilisation compared to set and wait ARQ. Better and efficient transmitted than GBN and SAW AKQ'S. Sender and Receiver maintain window of equal size.

Program: (Client)

```
package
SRP;

import java.net.*;
import java.io.*;
import java.util.Scanner;

public class SRPClient {

static Socket s;
static OutputStreamWriter os;
static PrintWriter out;
static BufferedReader br;

static void sendFrame(int s, int ws) {
int i = s;
while (i <= ws) {
System.out.println("Sending Frame " + i);
out.println(i);
out.flush();
i++;
}
}
static void repeatFrame(int s){
System.out.println("Sending Frame " + s);
out.println(s);
out.flush();
}

public static void main(String... args) throws Exception { s = new Socket("localhost", 6363);
os = new OutputStreamWriter(s.getOutputStream());
out = new PrintWriter(os);
br = new BufferedReader(new InputStreamReader(s.getInputStream()));
```

```
Scanner sc = new Scanner(System.in);
System.out.println("Enter Size Of Window: ");
int ws = Integer.parseInt(sc.nextLine());

// sending window size to server
 out.println(ws);
 out.flush();

sendFrame(1, ws);
// for re transmitting System.out.print("\nEnter the ack lost: “);
repeatFrame(Integer.parseInt(sc.nextLine()));

}
}
```

Output:

```
(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air SELECTIVE REPEAT % java SRP.
SRPClient
Enter Size Of Window:
4
Sending Frame 1
Sending Frame 2
Sending Frame 3
Sending Frame 4

Enter the ack lost: 2
Sending Frame 2
```

Program: (Server)

```
package
SRP;

import java.net.*;
import java.io.*;
import java.util.Scanner;

public class SRPServer {

static ServerSocket ss;
static Socket s;
static OutputStreamWriter os;
static PrintWriter out;
static BufferedReader br;

static void receiveFrame(int ws) throws Exception { int i = 1;
while (i <= ws) {
int fno = Integer.parseInt(br.readLine()); System.out.println("Recived Frame " + fno); System.out.println("Sending
    Ack for frame " + (fno + 1)); i++;

}
}

public static void main(String... args) throws Exception { System.out.println("Server Starting...");
                    ss. = new ServerSocket(6363); s = ss.accept(); System.out.println("Client Connected...");

os = new OutputStreamWriter(s.getOutputStream()); out = new PrintWriter(os);
br = new BufferedReader(new InputStreamReader(s.getInputStream()));

receiveFrame(Integer.parseInt(br.readLine()));
}
}
```

Output:

```
[(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air SELECTIVE REPEAT % java SRP.]
SRPServer
Server Starting...
Client Connected...
Recived Frame 1
Sending Ack for frame 2
Recived Frame 2
Sending Ack for frame 3
Recived Frame 3
Sending Ack for frame 4
Recived Frame 4
Sending Ack for frame 5
```

EXPERIMENT NO - 11

AIM: Take an example subnet graph with weights indicating delay between nodes, now obtain routing table at each node using distance vector algorithm.

Description:

In distance vector routing, each router in the network maintains a routing table.

Initially weights to adjacent nodes are marked, non adjacent nodes are assumed at infinins.

After each updation of distance in a router's routing table, the distance vector is shared to the adjacent nodes in the network.

Using Bellman Ford's algorithm, the minimum shortest distance to reach every node is calculated.

Bellman Ford Algorithm has advantage over Dijkstra's algorithms, as the former also works for the negative weighted edges.

Program:

```
import java.util.*;

class Node{

int dist[];
int from[];

Node(int n){
dist = new int[n];
from = new int[n];
}

}
public class DVRA {
public static void main(String... args){
int distMatrix[][] = new int[20][20];
int n;

System.out.println("No of Nodes: ");
Scanner sc = new Scanner(System.in);
n = Integer.parseInt(sc.nextLine());
Node route[] = new Node[n];
                                //    Initialising objects in the array for( int i = 0; i < n; i++){
route[i] = new Node(n);
}
System.out.println("Enter distance Matrix: ");
for( int i = 0; i < n; i++){ // selectiong each node String[] inputArray = sc.nextLine().split(" “,0);

for(int j = 0; j < n; j++){ // selecting each node in netowrk
 distMatrix[i][j] = Integer.parseInt(inputArray[j]);
```

```
if( i == j){
distMatrix[i][j] = 0;
}
route[i].dist[j] = distMatrix[i][j]; // updating routig table for a particular node
route[i].from[j] = j; // updating from which node the distance is from
}

//implementation of bellman ford
int flag;
do{
flag = 0;
for(int i = 0; i < n; i++){ for( int j = 0; j < n; j++){
for(int k = 0; k < n; k++){
if(route[i].dist[j] > (route[i].dist[k] + route[k].dist[j])){
route[i].dist[j] = route[i].dist[k] + route[k].dist[j];
route[i].from[j] = k;
flag = 1;
}
}
}
}
}while(flag==1){
//printing routing table at each node
for(int i = 0; i< n ; i++){
System.out.println("Router info for router: "+(i+1));
System.out.println("Dest\tNext Hop\tDist”);

 for(int j = 0; j < n; j++){
System.out.println((j+1)+"\t"+(route[i].from[j]+1)+"\t\t"+route[i].dist[j]);
}
}

}
}
```

Output:

```
(base) venkatasaiyaswanthbattu@VENKATAs-MacBook-Air exp 5 % java DVR.DVRA
No of Nodes:
3
Enter distance Matrix:
0 1 5
1 0 2
5 2 0
Router info for router: 1
Dest    Next Hop        Dist
1       1               0
2       2               1
3       2               3
Router info for router: 2
Dest    Next Hop        Dist
1       1               1
2       2               0
3       3               2
Router info for router: 3
Dest    Next Hop        Dist
1       2               3
2       2               2
3       3               0
```

EXPERIMENT NO - 12

AIM: Simulate the implementing Routing Protocols using border gateway protocol(BGP)

Description:

BGP is the postal service of the internet.

When someone submits the data via internet, BGP is responsible for looking at all of the available paths that data could travel and picking the best route, which usually means hopping between autonomous systems.

BGP is the protocol that makes the internet work by enabling data routing.

BGP is the protocol that enables the common communication to happen quickly and efficiently.

Command for:

Router (config-if) #exit

Router (config) # router bgp 100

Router (config-router) # network 192.168.1.0.

Router(config-router) # network 192.168.2.2.

Remote-as 200.

Router(config-router) #neighbor 192.168.3.11

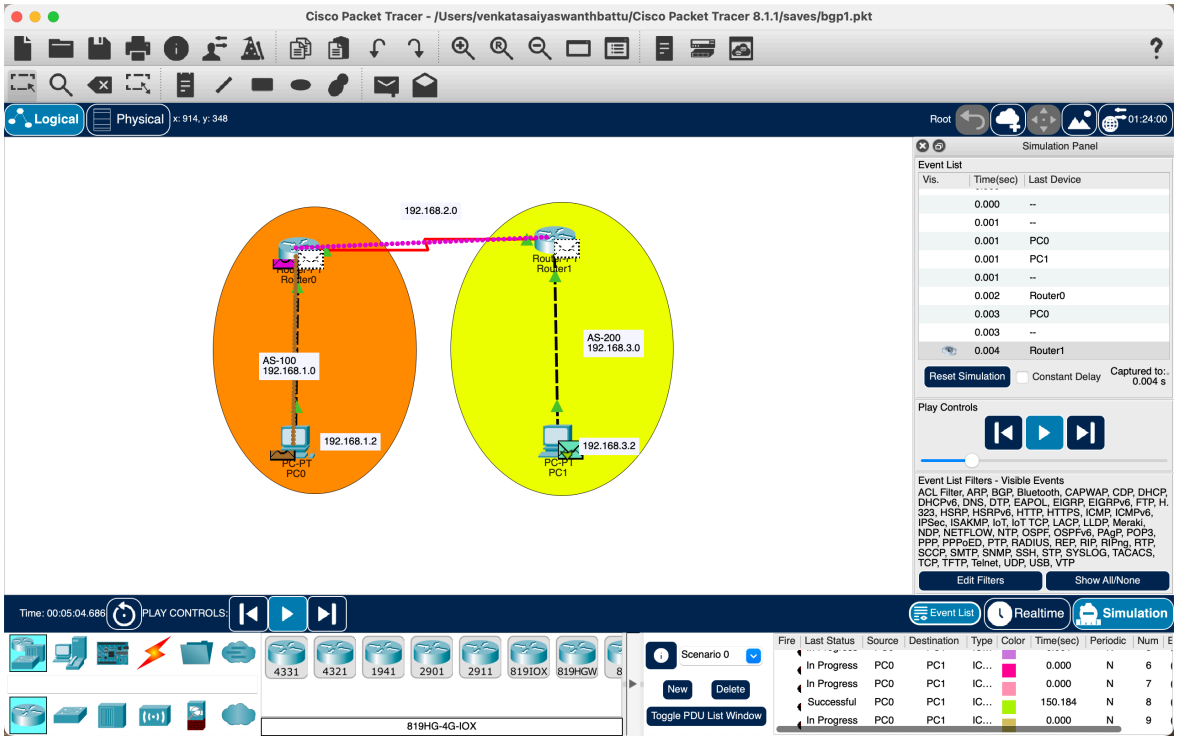
Remote-as 200.

Router(config-router) #exit.

Similarly, configure neighbours to the gate way of autonomous system 200(as-200).

After configuring, try passing message between the host computes , the message gets passed successfully.

Output:


Cisco Packet Tracer - /Users/venkatasaiyaswanthbattu/Cisco Packet Tracer 8.1.1/saves/bgp1.pkt
Logical
Physical
Root
Simulation Panel
Event List
Vis.
Time(sec)
Last Device
0.000 --
0.001 --
0.001 PC0
0.001 PC1
0.001 --
0.002 Router0
0.003 PC0
0.003 --
0.004 Router1
Reset Simulation
Constant Delay
Play Controls
Event List Filters - Visible Events
Edit Filters
Show All/None
192.168.2.0
AS-100
192.168.1.0
AS-200
192.168.3.0
192.168.1.2
192.168.3.2
Router0
Router1
PC0
PC1
Time: 00:05:04.686
PLAY CONTROLS:
Event List
Realtime
Simulation
4331
4321
1941
2901
2911
819IOX
819HGW
819HG-4G-IOX
Scenario 0
New
Delete
Toggle PDU List Window
Fire
Last Status
Source
Destination
Type
Color
Time(sec)
Periodic
Num
In Progress PC0 PC1 IC... 0.000 N 6
In Progress PC0 PC1 IC... 0.000 N 7
Successful PC0 PC1 IC... 150.184 N 8
In Progress PC0 PC1 IC... 0.000 N 9

EXPERIMENT NO - 13

AIM: Simulate the OPEN SHORTEST PATH FIRST routing protocol based on the cost assigned to the path.

Description:

Open Shortest Path First (OSPF) is a link-state routing protocol that was developed for IP networks and is based on the Shortest Path First (SPF).

Example of link state routing protocols include Open Shortest Path First (OSPF) and intermediate system to intermediate system (IS-IS).

The link state protocol is performed by every switching node in the protocol (network). OSPF operates between a single autonomous system.

Commands for:

At router 0.

Router (config-if) #exit.

Router (config) # router osff 1

Router (config-route) #network 192.168.1.0 0.0.0.0.255 area 0

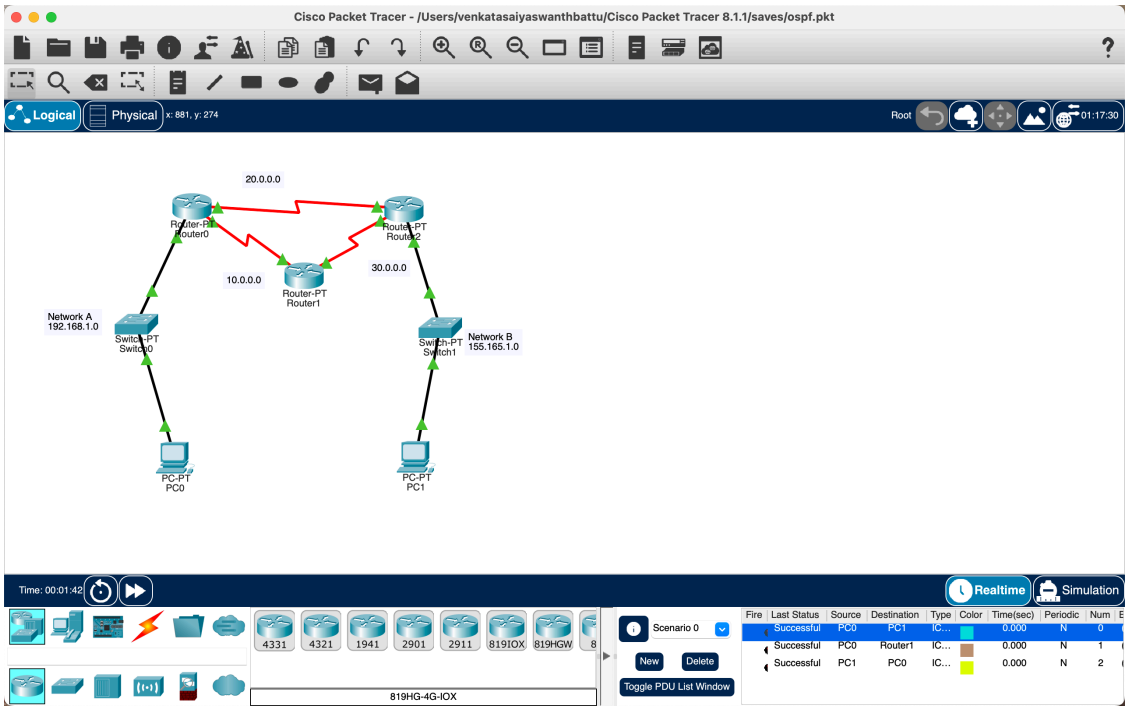
Router (config-route) #network 10.0.0.0 0.255.255.255 area 0

Router (config-route) #network 20.0.0.0 0.255.255.255 area 0

Router (config-route) #exit.

Similarly, configure to router2,router 1 with suitable marking in the commands .

Output:



NOTE:

- **Cisco Packet Tracer v8.8.1 is used for making the packets available at the repository**
- **All class files are curated into respective packages for each experiment uniquely.**
- **Use command Java -d filename.java for compiling & use java package_name.filename to run.**
- **Source Code for the record is available at:** https://github.com/s-1-n-t-h/COMPUTER-NETWORKS-LAB

Thank You