**ACS College of Engineering, Bangalore**

**Affiliated to VTU, accredited by NAAC with 'A' Grade, Accredited by NBA**

**B.E -CSE-Data Science**

**BCS502-Computer Network manual**

**1.Develop a program to implement a sliding window protocol in the data link layer.**

**Program:(GBN)**

```
import java.util.Random;
import java.util.Scanner;
public class SlidingWindowProtocol {
    // Maximum window size
    private static final int MAX_WINDOW_SIZE = 4;
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();

        System.out.println("Enter the number of frames to be sent:");
        int totalFrames = scanner.nextInt();

        // The sender's window size is MAX_WINDOW_SIZE
        int windowStart = 0;   // This is the base of the window
        int nextFrameToSend = 0;   // This is the next frame to be sent

        while (windowStart < totalFrames) {
            // Sending frames up to the maximum window size
            System.out.println("\nSender: Sending frames in the window...");

            for (int i = nextFrameToSend; i < windowStart + MAX_WINDOW_SIZE && i <
totalFrames; i++) {
```

```java
            System.out.println("Sender: Sending frame " + i);
        }

        // Simulate acknowledgments and loss of frames
        System.out.println("\nReceiver: Receiving frames...");
        for (int i = windowStart; i < windowStart + MAX_WINDOW_SIZE && i < totalFrames;
i++) {
            boolean isCorrupted = random.nextBoolean();  // Randomly decide if the frame is
corrupted or not

            if (isCorrupted) {
                System.out.println("Receiver: Frame " + i + " is corrupted or lost.");
                System.out.println("Receiver: Sending negative acknowledgment (NACK) for frame
" + i);
                nextFrameToSend = i;  // Sender will retransmit from this frame
                break;
            } else {
                System.out.println("Receiver: Successfully received frame " + i);
                System.out.println("Receiver: Sending acknowledgment (ACK) for frame " + i);
                windowStart++;  // Move the sender's window
                nextFrameToSend = windowStart;
            }
        }

        // Simulate waiting for acknowledgments
        System.out.println("\nSender: Waiting for ACK/NACK...");
        try {
            Thread.sleep(1000);  // Sleep to simulate transmission delay
```

```java
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println("\nAll frames have been successfully transmitted!");
        scanner.close();
    }
}
```

**Output:**

```
Enter the number of frames to be sent:
4

Sender: Sending frames in the window...
Sender: Sending frame 0
Sender: Sending frame 1
Sender: Sending frame 2
Sender: Sending frame 3

Receiver: Receiving frames...
Receiver: Successfully received frame 0
Receiver: Sending acknowledgment (ACK) for frame 0
Receiver: Frame 1 is corrupted or lost.
Receiver: Sending negative acknowledgment (NACK) for frame 1

Sender: Waiting for ACK/NACK...

Sender: Sending frames in the window...
Sender: Sending frame 1
```

Sender: Sending frame 2

Sender: Sending frame 3

Receiver: Receiving frames...

Receiver: Successfully received frame 1

Receiver: Sending acknowledgment (ACK) for frame 1

Receiver: Successfully received frame 2

Receiver: Sending acknowledgment (ACK) for frame 2

Receiver: Successfully received frame 3

Receiver: Sending acknowledgment (ACK) for frame 3

Sender: Waiting for ACK/NACK...

All frames have been successfully transmitted!

**2. Develop a program for error detecting code using CRC-CCITT (16- bits).**

**Program:**

```
import java.util.Scanner;
 public class CRC_CCITT {
```

```
    private static final int POLYNOMIAL = 0x1021;
    private static final int CRC_INITIAL = 0xFFFF;
    public static int calculateCRC(byte[] data) {
        int crc = CRC_INITIAL;
         for (byte b : data) {
            crc ^= (b << 8);
            for (int i = 0; i < 8; i++) {
                if ((crc & 0x8000) != 0) {
                    crc = (crc << 1) ^ POLYNOMIAL;
                } else {
                    crc <<= 1;
                }
            }
        }
       return crc & 0xFFFF;
    }
       public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
         System.out.print("Enter message to compute CRC (text): ");
        String message = scanner.nextLine();
         byte[] data = message.getBytes();
         int crc = calculateCRC(data);
         System.out.printf("CRC-CCITT (16-bit) for the input message: %04X\n", crc);
        scanner.close();
    }
}
```

**OUTPUT:**

Enter message to compute CRC (text): Hello

CRC-CCITT (16-bit) for the input message: DADA

**3. Develop a program to find the shortest path between vertices using the Bellman-Ford and path vector routing algorithm.**

**Program:**

```
import java.util.Arrays;


class Edge {
    int src, dest, weight;


    public Edge(int src, int dest, int weight) {
        this.src = src;
        this.dest = dest;
        this.weight = weight;
    }
}


public class BellmanFord {
    private int V, E;
    private Edge[] edges;


    public BellmanFord(int V, int E) {
        this.V = V;
        this.E = E;
        edges = new Edge[E];
    }


    public void addEdge(int edgeIndex, int src, int dest, int weight) {
```

```java
    edges[edgeIndex] = new Edge(src, dest, weight);
}

public void shortestPath(int src) {
    int[] dist = new int[V];
    Arrays.fill(dist, Integer.MAX_VALUE);
    dist[src] = 0;

    // Relax all edges V - 1 times.
    for (int i = 0; i < V - 1; i++) {
        for (int j = 0; j < E; j++) {
            int u = edges[j].src;
            int v = edges[j].dest;
            int weight = edges[j].weight;
            if (dist[u] != Integer.MAX_VALUE && dist[u] + weight < dist[v]) {
                dist[v] = dist[u] + weight;
            }
        }
    }

    // Check for negative-weight cycles.
    for (int j = 0; j < E; j++) {
        int u = edges[j].src;
        int v = edges[j].dest;
        int weight = edges[j].weight;
        if (dist[u] != Integer.MAX_VALUE && dist[u] + weight < dist[v]) {
            System.out.println("Graph contains a negative-weight cycle.");
```

```java
            return;
        }
    }

    printSolution(dist);
}

private void printSolution(int[] dist) {
    System.out.println("Vertex Distance from Source");
    for (int i = 0; i < V; i++) {
        System.out.println(i + "\t\t" + dist[i]);
    }
}

public static void main(String[] args) {
    int V = 5; // Number of vertices in graph
    int E = 8; // Number of edges in graph

    BellmanFord graph = new BellmanFord(V, E);

    // Define edges: source, destination, and weight
    graph.addEdge(0, 0, 1, -1);
    graph.addEdge(1, 0, 2, 4);
    graph.addEdge(2, 1, 2, 3);
    graph.addEdge(3, 1, 3, 2);
    graph.addEdge(4, 1, 4, 2);
    graph.addEdge(5, 3, 2, 5);
```

```
        graph.addEdge(6, 3, 1, 1);
        graph.addEdge(7, 4, 3, -3);

        // Compute shortest paths from vertex 0
        graph.shortestPath(0);
    }
}
```

**OUTPUT:**

```
Vertex Distance from Source
0       0
1       -1
2       2
3       -2
4       1
```

**4. Develop a program on a datagram socket for client/server to display the messages on client side, typed at the server side.**

**Program:**

**Server Side Program:**

```
import java.io.*;

import java.net.*;

import java.util.Scanner;


public class DatagramServer {

    private static final int PORT = 9876;


    public static void main(String[] args) {

        try (DatagramSocket serverSocket = new DatagramSocket()) {

            Scanner scanner = new Scanner(System.in);

            System.out.println("Server started. Type messages to send to the client.");


            while (true) {

                System.out.print("Server: ");

                String message = scanner.nextLine();


                // Convert message to bytes and send to client

                byte[] buffer = message.getBytes();

                InetAddress clientAddress = InetAddress.getByName("localhost"); // Assuming client
is on localhost
```

```
                DatagramPacket packet = new DatagramPacket(buffer, buffer.length, clientAddress,
PORT);

                serverSocket.send(packet);
                System.out.println("Message sent to client.");
            }
        } catch (IOException e) {
            System.out.println("Server error: " + e.getMessage());
        }
    }
}
```

**Client Side Program:**

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

public class DatagramClient {
    private static final int PORT = 9876;

    public static void main(String[] args) {
        try (DatagramSocket clientSocket = new DatagramSocket(PORT)) {
            System.out.println("Client is listening on port " + PORT);

            byte[] buffer = new byte[1024];

            while (true) {
                // Receive packet from server
                DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
                clientSocket.receive(packet);
```

```
            // Convert byte data to string and display
            String message = new String(packet.getData(), 0, packet.getLength());
            System.out.println("Server: " + message);
          }
        } catch (IOException e) {
          System.out.println("Client error: " + e.getMessage());
        }
    }
}
```

**Output:**

Server started. Type messages to send to the client.

Server: Welcome to CN lab

Message sent to client.

Client is listening on port 9876

Server: Welcome to CN lab

**5. Using TCP/IP sockets, write a client – server program to make the client send the file name and to make the server send back the contents of the requested file if present.**

```
// Server.java

import java.io.*;

import java.net.*;


public class Server {

    public static void main(String[] args) {

        int port = 1234;  // Port number to bind to

        try (ServerSocket serverSocket = new ServerSocket(port)) {

            System.out.println("Server started. Waiting for a client...");


            // Accept the client connection

            Socket socket = serverSocket.accept();

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


            // Input and output streams to read from and write to the client

            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);


            // Receive message from the client

            String clientMessage = in.readLine();
```

```java
            System.out.println("Received from client: " + clientMessage);

            // Send response to the client
            out.println("Hello, Client! I received your message.");

            // Close resources
            in.close();
            out.close();
            socket.close();
            System.out.println("Client disconnected.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
// Client.java
import java.io.*;
import java.net.*;

public class Client {
    public static void main(String[] args) {
        String hostname = "localhost";  // Server hostname or IP address
        int port = 1234;                // Server port number

        try (Socket socket = new Socket(hostname, port)) {
            System.out.println("Connected to the server.");
```

```
            // Output stream to send message to the server
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            // Input stream to read response from the server
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

            // Send message to the server
            out.println("Hello, Server!");

            // Receive and print response from the server
            String serverResponse = in.readLine();
            System.out.println("Received from server: " + serverResponse);

            // Close resources
            in.close();
            out.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

**OUTPUT:**

Server started. Waiting for a client...

Client connected.

Received from client: Hello, Server!

Client disconnected

Connected to the server.

Received from server: Hello, Client! I received your message.

**6. Develop a program for a simple RSA algorithm to encrypt and decrypt the data.**

```
import java.math.BigInteger;
import java.security.SecureRandom;
import java.util.Scanner;

public class RSA {
    private BigInteger n;          // modulus
    private BigInteger e;          // public exponent
    private BigInteger d;          // private exponent
    private int bitLength = 1024;  // bit length of the modulus (key size)

    // Constructor to generate public and private keys
    public RSA() {
        SecureRandom random = new SecureRandom();
        BigInteger p = BigInteger.probablePrime(bitLength / 2, random);
        BigInteger q = BigInteger.probablePrime(bitLength / 2, random);
        n = p.multiply(q);
        BigInteger phi = (p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));
        e = BigInteger.probablePrime(bitLength / 2, random);

        // Ensure e and phi(n) are coprime
        while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0) {
```

```java
            e = e.add(BigInteger.ONE);
        }

        // Calculate the private key d
        d = e.modInverse(phi);
    }

    // Encrypt a message
    public BigInteger encrypt(BigInteger message) {
        return message.modPow(e, n);
    }

    // Decrypt a message
    public BigInteger decrypt(BigInteger encrypted) {
        return encrypted.modPow(d, n);
    }

    // Main method to test the encryption and decryption with user input
    public static void main(String[] args) {
        RSA rsa = new RSA();

        // Getting user input
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a message to encrypt: ");
        String plaintext = scanner.nextLine();
        scanner.close();

        System.out.println("Original Message: " + plaintext);
```

```
        // Convert plaintext to BigInteger
        BigInteger message = new BigInteger(plaintext.getBytes());

        // Encrypt the message
        BigInteger encrypted = rsa.encrypt(message);
        System.out.println("Encrypted Message: " + encrypted);

        // Decrypt the message
        BigInteger decrypted = rsa.decrypt(encrypted);

        // Convert decrypted message back to string
        String decryptedMessage = new String(decrypted.toByteArray());
        System.out.println("Decrypted Message: " + decryptedMessage);
    }
}
```

**Output:**

Enter a message to encrypt: welcome

Original Message: welcome

Encrypted Message:
2253004347184819206648157902694624309134285236568148600658303085651811882103618404221337980426415643199406765312885922266576087338616845390023113288717762135184247303164500815729854964962939845530069283179945119481413345255927917420894223589792999811252488701824527917780902209479848638074151123376585776653

Decrypted Message: welcome

**7. Develop a program for congestion control using a leaky bucket algorithm.**

```
import java.util.Scanner;
 public class LeakyBucket {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // Input bucket capacity and output rate
        System.out.print("Enter bucket capacity: ");
        int bucketCapacity = scanner.nextInt();

        System.out.print("Enter output rate (packets/sec): ");
        int outputRate = scanner.nextInt();

        // Simulating the process
        System.out.print("Enter the number of seconds to simulate: ");
        int simulationTime = scanner.nextInt();

        int bucketContent = 0; // Current content in the bucket

        for (int second = 1; second <= simulationTime; second++) {
            System.out.print("\nAt second " + second + ", enter number of packets arriving: ");
            int incomingPackets = scanner.nextInt();
```

```java
            // Check for overflow
            if (incomingPackets + bucketContent > bucketCapacity) {
                System.out.println("Overflow! " + (incomingPackets + bucketContent - bucketCapacity) + " packets dropped.");
                bucketContent = bucketCapacity; // Bucket is filled to its capacity
            } else {
                bucketContent += incomingPackets;
            }

            // Transmitting packets
            if (bucketContent > 0) {
                int transmittedPackets = Math.min(outputRate, bucketContent);
                bucketContent -= transmittedPackets;
                System.out.println(transmittedPackets + " packets transmitted.");
            } else {
                System.out.println("No packets to transmit.");
            }
            // Display remaining bucket content
            System.out.println("Bucket content: " + bucketContent + " packets.");
        }
        scanner.close();
    }
}
```

**Output:**

Enter bucket capacity: 10

Enter output rate (packets/sec): 5

Enter the number of seconds to simulate: 5

At second 1, enter number of packets arriving: 8

5 packets transmitted.

Bucket content: 3 packets.

At second 2, enter number of packets arriving: 4

5 packets transmitted.

Bucket content: 2 packets.

At second 3, enter number of packets arriving: 12

Overflow! 7 packets dropped.

5 packets transmitted.

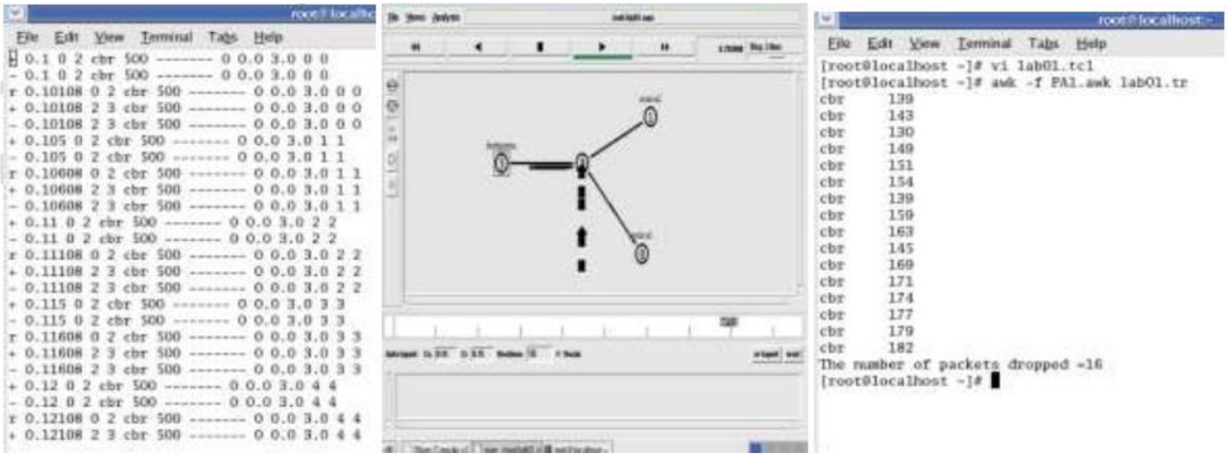
Bucket content: 2 packets.

**8. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth, and find the number of packets dropped.**

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

#Open Trace file and NAM file set ntrace [open prog1.tr w]
$ns trace-all $ntrace
set namfile [open prog1.nam w]
$ns namtrace-all $namfile

#Finish Procedure proc Finish {} {
global ns ntrace namfile

#Dump all the trace data and close the files
$ns flush-trace close $ntrace close $namfile

#Execute the nam animation file exec nam prog1.nam &

#Show the number of packets dropped
exec echo "The number of packet drops is " & exec grep -c "^d" prog1.tr &
```

```
exit 0

}

#Create 3 nodes set n0 [$ns node] set n1 [$ns node] set n2 [$ns node]

#Label the nodes
$n0 label "TCP Source"
$n2 label "Sink"

#Set the color

$ns color 1 blue

#Create Links between nodes
#You need to modify the bandwidth to observe the variation in packet drop
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail

#Make the Link Orientation
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right

#Set Queue Size
#You can modify the queue length as well to observe the variation in packet drop
$ns queue-limit $n0 $n1 10
$ns queue-limit $n1 $n2 10
```

```
#Set up a Transport layer connection. set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $n2 $sink0
$ns connect $tcp0 $sink0

#Set up an Application layer Traffic
set cbr0 [new Application/Traffic/CBR]
$cbr0 set type_ CBR
$cbr0 set packetSize_ 100
$cbr0 set rate_ 1Mb
$cbr0 set random_ false
$cbr0 attach-agent $tcp0

$tcp0 set class_ 1

#Schedule Events
$ns at 0.0 "$cbr0 start"
$ns at 5.0 "Finish"

#Run the Simulation
$ns run
```



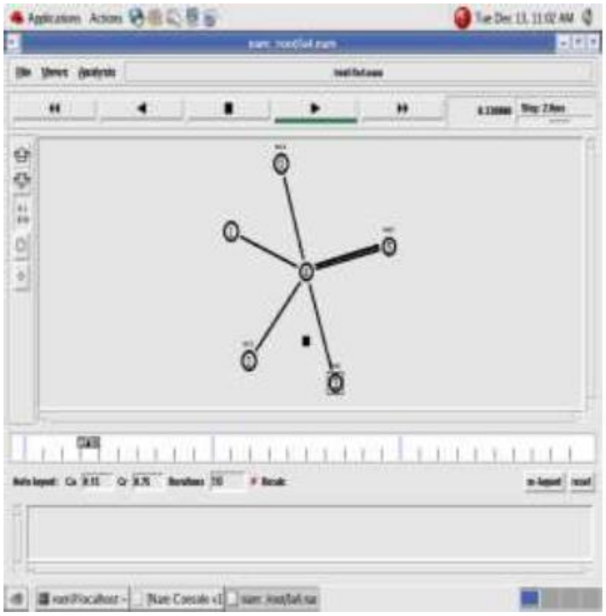

**9.Implement transmission of ping messages/trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.**

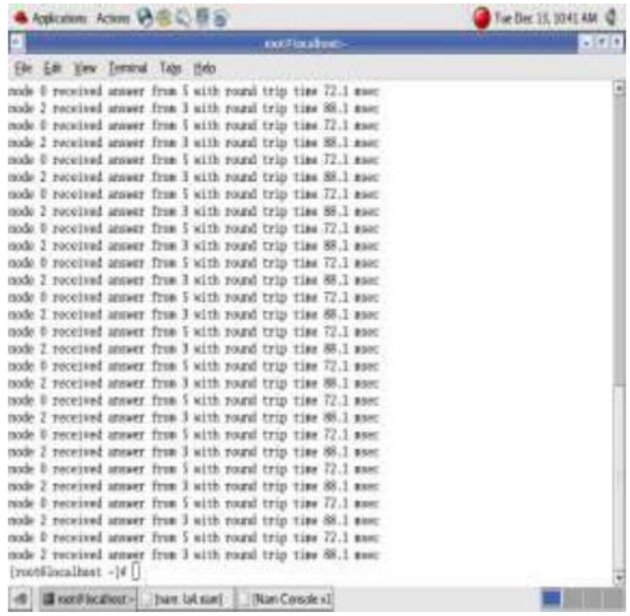
```
#Create Simulator
set ns [new Simulator]

#Use colors to differentiate the traffic
$ns color 1 Blue
$ns color 2 Red

#Open trace and NAM trace file set ntrace [open prog3.tr w]
$ns trace-all $ntrace
```

```
set namfile [open prog3.nam w]

$ns namtrace-all $namfile


#Finish Procedure proc Finish {} {

global ns ntrace namfile


#Dump all trace data and close the file

$ns flush-trace close $ntrace close $namfile


#Execute the nam animation file exec nam prog3.nam &


#Find the number of ping packets dropped

puts "The number of ping packets dropped are "

exec grep "^d" prog3.tr | cut -d " " -f 5 | grep -c "ping" & exit 0

}


#Create six nodes

for {set i 0} {$i < 6} {incr i} {

set n($i) [$ns node]

}

#Connect the nodes

for {set j 0} {$j < 5} {incr j} {

$ns duplex-link $n($j) $n([expr ($j+1)]) 0.1Mb 10ms DropTail

}


#Define the recv function for the class 'Agent/Ping'

Agent/Ping instproc recv {from rtt} {

$self instvar node_
```

```
puts "node [$node_ id] received ping answer from $from with round trip time $rtt
ms"
}

#Create two ping agents and attach them to n(0) and n(5)
set p0 [new Agent/Ping]
$p0 set class_ 1
$ns attach-agent $n(0) $p0

set p1 [new Agent/Ping]
$p1 set class_ 1
$ns attach-agent $n(5) $p1
$ns connect $p0 $p1

#Set queue size and monitor the queue
#Queue size is set to 2 to observe the drop in ping packets
$ns queue-limit $n(2) $n(3) 2
$ns duplex-link-op $n(2) $n(3) queuePos 0.5

#Create Congestion
#Generate a Huge CBR traffic between n(2) and n(4)
set tcp0 [new Agent/TCP]
$tcp0 set class_ 2
$ns attach-agent $n(2) $tcp0 set sink0 [new Agent/TCPSink]
$ns attach-agent $n(4) $sink0
$ns connect $tcp0 $sink0

#Apply CBR traffic over TCP
```

```
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set rate_ 1Mb
$cbr0 attach-agent $tcp0

#Schedule events
$ns at 0.2 "$p0 send"
$ns at 0.4 "$p1 send"
$ns at 0.4 "$cbr0 start"
$ns at 0.8 "$p0 send"
$ns at 1.0 "$p1 send"
$ns at 1.2 "$cbr0 stop"
$ns at 1.4 "$p0 send"
$ns at 1.6 "$p1 send"
$ns at 1.8 "Finish"

#Run the Simulation
$ns run
```




**10. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion**

**window for different source / destination.**

#Create Simulator

set ns [new Simulator]

#Use colors to differentiate the traffics

$ns color 1 Blue

$ns color 2 Red

#Open trace and NAM trace file set ntrace [open prog5.tr w]

$ns trace-all $ntrace

set namfile [open prog5.nam w]

$ns namtrace-all $namfile

#Use some flat file to create congestion graph windows set winFile0 [open WinFile0 w]

set winFile1 [open WinFile1 w]

#Finish Procedure proc Finish {} {

#Dump all trace data and Close the files global ns ntrace namfile

$ns flush-trace close $ntrace close $namfile

#Execute the NAM animation file exec nam prog5.nam &

#Plot the Congestion Window graph using xgraph exec xgraph WinFile0 WinFile1 &

exit 0

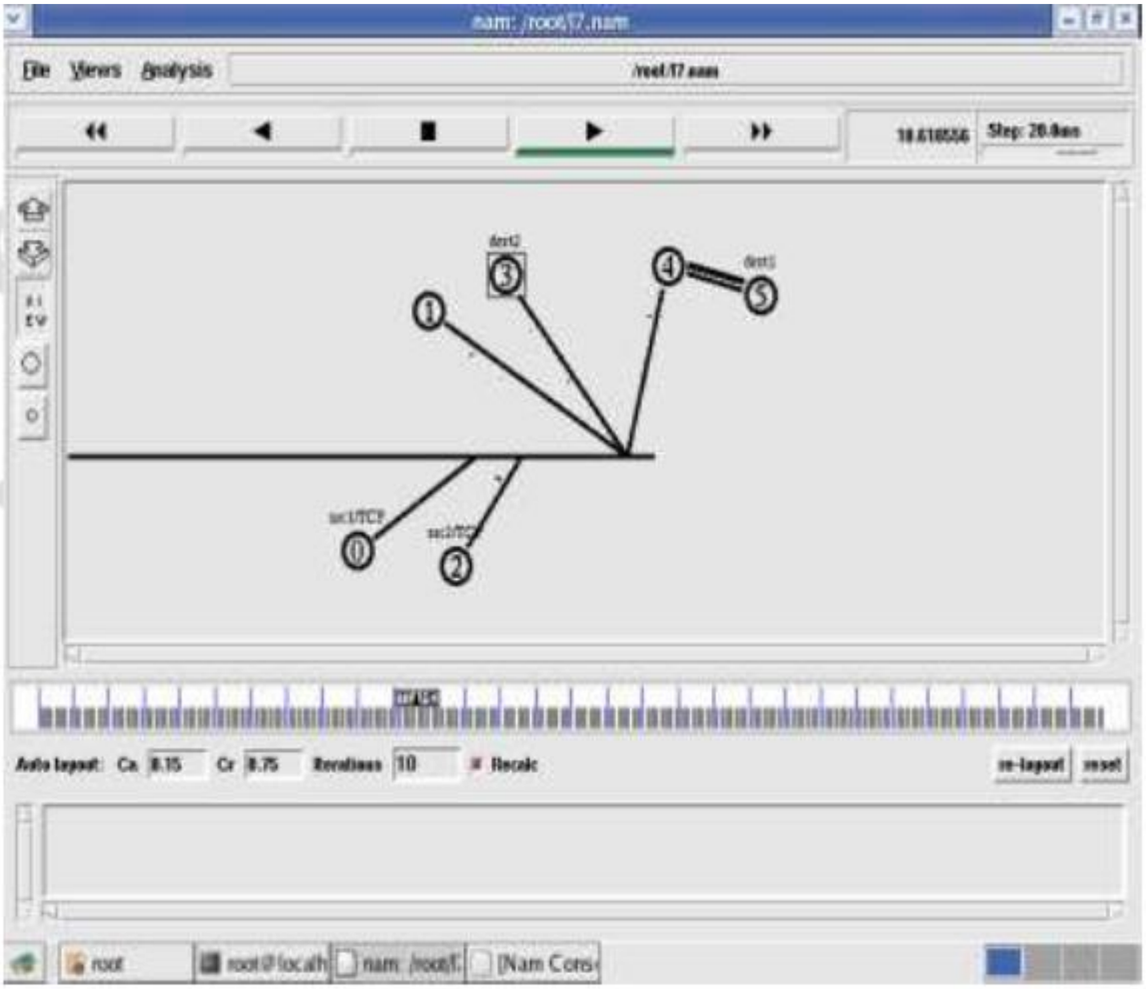
}

#Plot Window Procedure

```
proc PlotWindow {tcpSource file} { global ns

set time 0.1

set now [$ns now]


set cwnd [$tcpSource set cwnd_] puts $file "$now $cwnd"

$ns at [expr $now+$time] "PlotWindow $tcpSource $file"

}


#Create 6 nodes

for {set i 0} {$i<6} {incr i} { set n($i) [$ns node]

}


#Create duplex links between the nodes

$ns duplex-link $n(0) $n(2) 2Mb 10ms DropTail

$ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail

$ns duplex-link $n(2) $n(3) 0.6Mb 100ms DropTail


#Nodes n(3) , n(4) and n(5) are considered in a LAN

set lan [$ns newLan "$n(3) $n(4) $n(5)" 0.5Mb 40ms LL Queue/DropTail MAC/802_3 Channel]


#Orientation to the nodes

$ns duplex-link-op $n(0) $n(2) orient right-down

$ns duplex-link-op $n(1) $n(2) orient right-up

$ns duplex-link-op $n(2) $n(3) orient right


#Setup queue between n(2) and n(3) and monitor the queue

$ns queue-limit $n(2) $n(3) 20

$ns duplex-link-op $n(2) $n(3) queuePos 0.5
```

```
#Set error model on link n(2) to n(3) set loss_module [new ErrorModel]
$loss_module ranvar [new RandomVariable/Uniform]
$loss_module drop-target [new Agent/Null]
$ns lossmodel $loss_module $n(2) $n(3)

#Set up the TCP connection between n(0) and n(4) set tcp0 [new Agent/TCP/Newreno]
$tcp0 set fid_ 1
$tcp0 set window_ 8000
$tcp0 set packetSize_ 552
$ns attach-agent $n(0) $tcp0
set sink0 [new Agent/TCPSink/DelAck]
$ns attach-agent $n(4) $sink0
$ns connect $tcp0 $sink0



#Apply FTP Application over TCP set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ftp0 set type_ FTP

#Set up another TCP connection between n(5) and n(1) set tcp1 [new Agent/TCP/Newreno]
$tcp1 set fid_ 2
$tcp1 set window_ 8000
$tcp1 set packetSize_ 552
$ns attach-agent $n(5) $tcp1
set sink1 [new Agent/TCPSink/DelAck]
```

```
$ns attach-agent $n(1) $sink1

$ns connect $tcp1 $sink1


#Apply FTP application over TCP set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ftp1 set type_ FTP


#Schedule Events

$ns at 0.1 "$ftp0 start"

$ns at 0.1 "PlotWindow $tcp0 $winFile0"

$ns at 0.5 "$ftp1 start"

$ns at 0.5 "PlotWindow $tcp1 $winFile1"

$ns at 25.0 "$ftp0 stop"

$ns at 25.1 "$ftp1 stop"

$ns at 25.2 "Finish"


#Run the simulation

$ns run
```


nam: /root/17.nam
File
Views
Analysis
/root/17.nam
Step: 20.0ms
Auto layout:
Ca
Cr
Iterations
10
Recalc
re-layout
reset
root
[Nam Cons