CN Lab Internal – 1 (first 5 Programs)

Program – 1 (network Simulation)

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
```

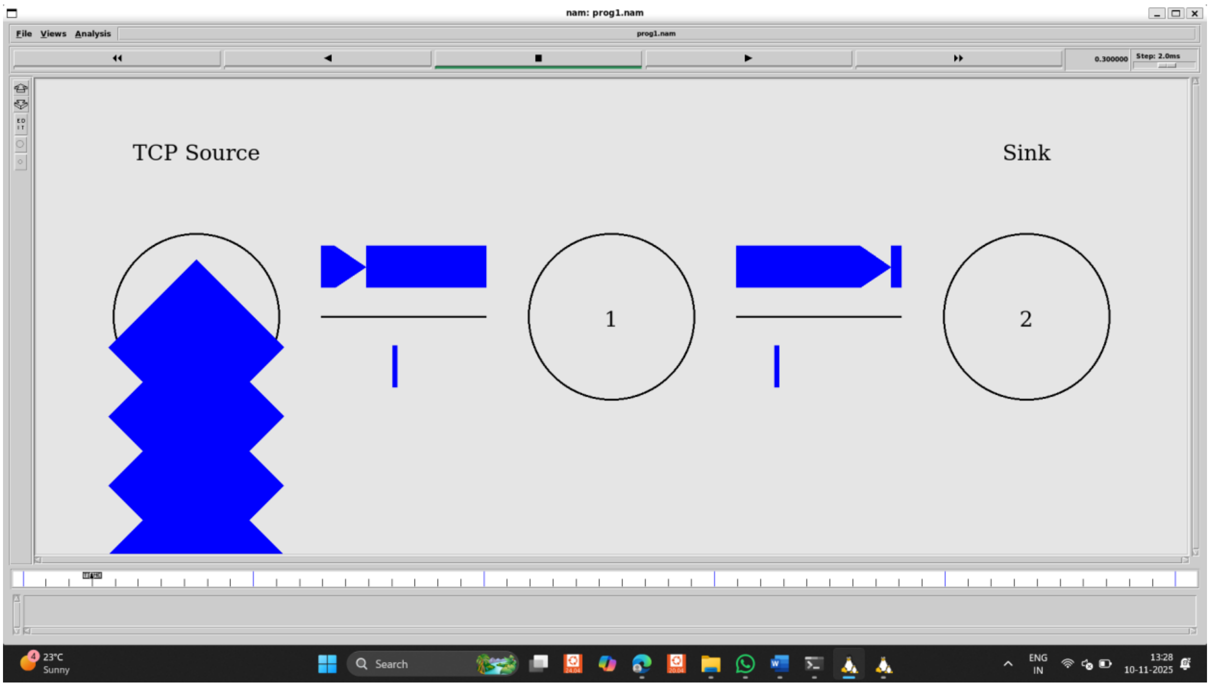
```
#You need to modify the bandwidth to observe the variation in packet drop
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
#Make the Link Orientation
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right
#Set Queue Size
#You can modify 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
```

****

Output : -

```
The number of packet drops is
8
```

Simulation :-



nam: prog1.nam
File
Views
Analysis
prog1.nam
0.300000
Step: 2.0ms
TCP Source
Sink
1
2
23°C
Sunny
Search
ENG
IN
13:28
10-11-2025

****

Program – 2 (network Simulation)

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 prog2.tr w]
$ns trace-all $ntrace
set namfile [open prog2.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 $namfile
#Execute the nam animation file
exec nam prog2.nam &
#Find the number of ping packets dropped
puts "The number of ping packets dropped are "
exec grep "^d" prog2.tr | cut -d " " -f 5 | grep -c "ping" &
exit 0
}
#Create six nodes
for {set i 0} {$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"
```

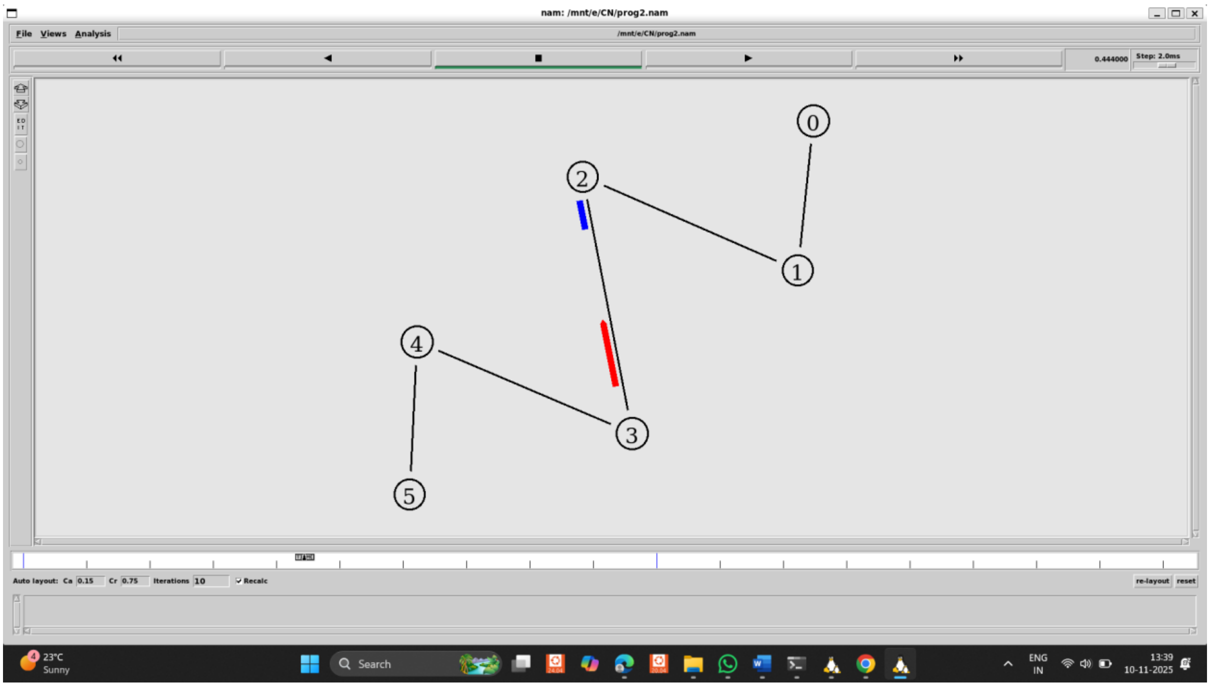
```
$ns at 1.6 "$p1 send"
$ns at 1.8 "Finish"
#Run the Simulation
$ns run
```

****

Output :-

```
node 0 received ping answer from 5 with round trip time 151.2
ms
node 0 received ping answer from 5 with round trip time 301.4
ms
node 5 received ping answer from 0 with round trip time 155.4
ms
The number of ping packets dropped are
3
```

Simulation : -



****

Program – 3 (network Simulation)

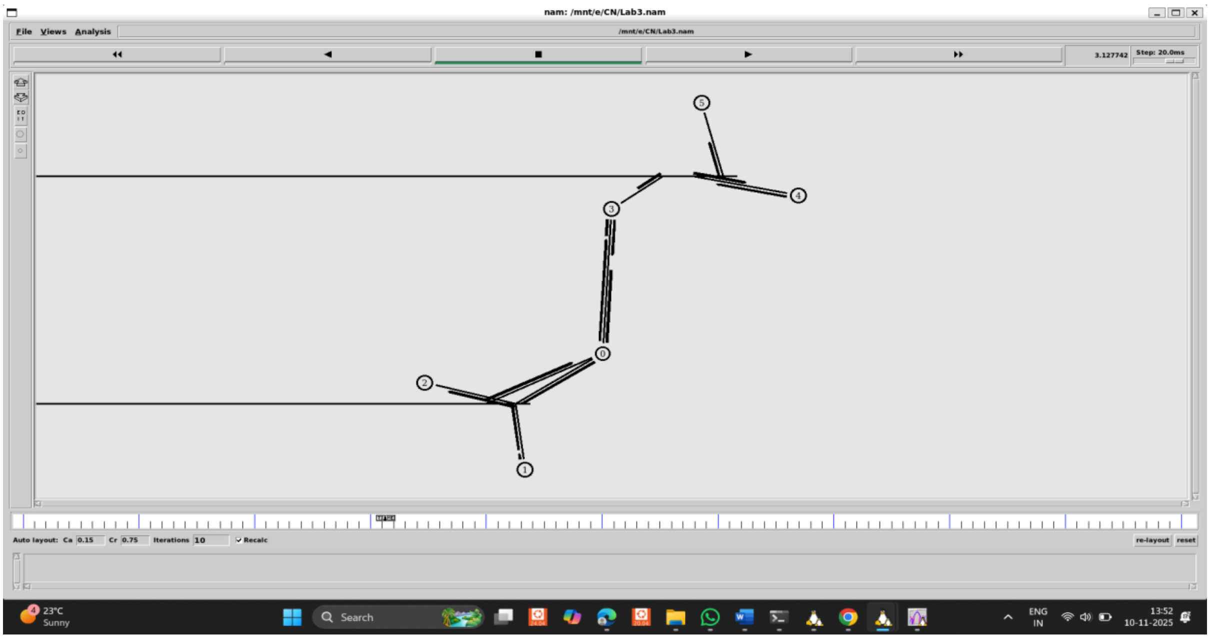
Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.

```
set ns [new Simulator]
$ns color 1 Red
$ns color 2 Blue
set na [open Lab3.nam w]
$ns namtrace-all $na
set nt [open Lab3.tr w]
$ns trace-all $nt
set ng1 [open tcp1.xg w]
set ng2 [open tcp2.xg w]
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$ns make-lan "$n0 $n1 $n2" 1Mb 10ms LL Queue/DropTail Mac/802_3
$ns make-lan "$n3 $n4 $n5" 2Mb 10ms LL Queue/DropTail Mac/802_3
$ns duplex-link $n0 $n3 1Mb 10ms DropTail
set tcp1 [new Agent/TCP]
set tcp2 [new Agent/TCP]
set cbr1 [new Application/Traffic/CBR]
set cbr2 [new Application/Traffic/CBR]
$ns attach-agent $n4 $tcp1
$cbr1 attach-agent $tcp1
$ns attach-agent $n1 $tcp2
$cbr2 attach-agent $tcp2
set sink1 [new Agent/TCPSink]
set sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $sink1
$ns attach-agent $n5 $sink2
$ns connect $tcp1 $sink1
$ns connect $tcp2 $sink2
proc End {} {
global ns na nt
$ns flush-trace
close $na
close $nt
exec nam Lab3.nam &
exec xgraph tcp1.xg tcp2.xg &
exit 0
}
proc Draw {Agent File} {
global ns
set Cong [$Agent set cwnd_]
set Cong [$Agent set cwnd_]
set Time [$ns now]
puts $File "$Time $Cong"
$ns at [expr $Time+0.01] "Draw $Agent $File"
}
$ns at 0.0 "$cbr1 start"
$ns at 0.7 "$cbr2 start"
$ns at 0.0 "Draw $tcp1 $ng1"
$ns at 0.0 "Draw $tcp2 $ng2"
$ns at 10.0 "End"
$ns run
```

****

Output : -

Simulation –



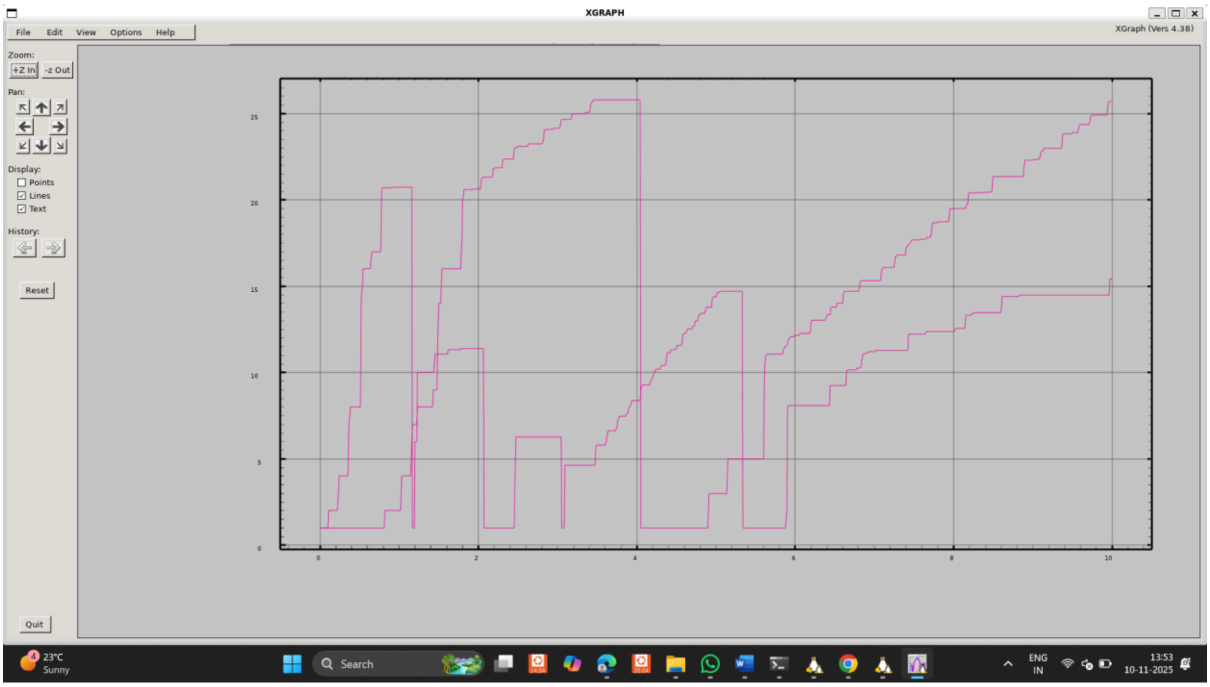
nam: /mnt/e/CN/Lab3.nam

/mnt/e/CN/Lab3.nam

3.127742 Step: 20.0ms

Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ☑ Recalc

Xgraph –


XGRAPH
File
Edit
View
Options
Help
XGraph (Vers 4.38)
Zoom:
+Z In
-z Out
Pan:
Display:
□ Points
☑ Lines
☑ Text
History:
Reset
Quit
25
20
15
10
5
0
0
2
4
6
8
10
23°C
Sunny
Search
ENG
IN
13:53
10-11-2025

****

Program – 4 (Java)

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

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

class crc {
  public static void main(String args[]) throws IOException {

    BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));
    int[] data;
    int[] div;
    int[] divisor;
    int[] rem;
    int[] crc;
    int data_bits, divisor_bits, tot_length;

    System.out.println("Enter number of data bits : ");
    data_bits = Integer.parseInt(br.readLine());
    data = new int[data_bits];

    System.out.println("Enter data bits : ");
    for (int i = 0; i < data_bits; i++)
      data[i] = Integer.parseInt(br.readLine());

    System.out.println("Enter number of bits in divisor : ");
    divisor_bits = Integer.parseInt(br.readLine());
    divisor = new int[divisor_bits];

    System.out.println("Enter Divisor bits : ");
    for (int i = 0; i < divisor_bits; i++)
      divisor[i] = Integer.parseInt(br.readLine());

    tot_length = data_bits + divisor_bits - 1;
    div = new int[tot_length];
    rem = new int[tot_length];
    crc = new int[tot_length];
```

```
for (int i = 0; i < data.length; i++)
   div[i] = data[i];

System.out.print("Dividend (after appending 0's) are : ");
for (int j = 0; j < div.length; j++)
   System.out.print(div[j]);
System.out.println();

for (int j = 0; j < div.length; j++) {
   rem[j] = div[j];
}

rem = divide(div, divisor, rem);
for (int j = 0; j < div.length; j++) {
   crc[j] = (div[j] ^ rem[j]);
}

System.out.println();
System.out.println("CRC code : ");
for (int i = 0; i < crc.length; i++)
   System.out.print(crc[i]);
System.out.println();

System.out.println("Enter CRC code of " + tot_length + " bits : ");
for (int i = 0; i < crc.length; i++)
   crc[i] = Integer.parseInt(br.readLine());

for (int j = 0; j < crc.length; j++) {
   rem[j] = crc[j];
}

rem = divide(crc, divisor, rem);
for (int j = 0; j < rem.length; j++) {
   if (rem[j] != 0) {
      System.out.println("Error");
      break;
   }
   if (j == rem.length - 1)
      System.out.println("No Error");
```

```
        }
    }

    static int[] divide(int div[], int divisor[], int rem[]) {
        int cur = 0;
        while (true) {
            for (int j = 0; j < divisor.length; j++)
                rem[cur + j] = (rem[cur + j] ^ divisor[j]);
            while (rem[cur] == 0 && cur != rem.length - 1)
                cur++;

            if ((rem.length - cur) < divisor.length)
                break;
        }
        return rem;
    }
}
```

****

Output :- (error)

```
Enter number of data bits :
8
Enter data bits :
1
1
0
0
1
0
0
1
Enter number of bits in divisor :
4
Enter Divisor bits :
1
0
0
1
```

```
Dividend (after appending 0's) are : 11001001000

CRC code :
11001001011
Enter CRC code of 11 bits :
1
1
0
0
1
0
0
1
0
1
0
Error
```

Output :- (No error)

```
Enter number of data bits :
8
Enter data bits :
1
1
0
0
1
0
0
1
Enter number of bits in divisor :
4
Enter Divisor bits :
1
0
0
1
Dividend (after appending 0's) are : 11001001000
```

```
CRC code :
11001001011
Enter CRC code of 11 bits :
1
1
0
0
1
0
0
1
0
1
1
No Error

****************************************************************
```

Program – 5 (Java)

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

Sender : -

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketTimeoutException;
import java.util.Random;
class Sender {
    public static final int WINDOW_SIZE = 4;
    private static final int TOTAL_FRAMES = 10;
    private static final Random random = new Random();

    public static void main(String[] args) throws IOException,
InterruptedException {
        DatagramSocket socket = new DatagramSocket();
        InetAddress receiverAddress = InetAddress.getLocalHost();

        int frameNumber = 0;
        int nextAck = 0;
        boolean[] ackReceived = new boolean[TOTAL_FRAMES];

        while (nextAck < TOTAL_FRAMES) {
            // Send frames within the window size
            for (int j = 0; j < WINDOW_SIZE && (frameNumber < TOTAL_FRAMES);
j++) {
                if (!ackReceived[frameNumber]) {
                    String message = "Frame " + frameNumber;
                    byte[] buffer = message.getBytes();

                    DatagramPacket packet = new DatagramPacket(buffer,
buffer.length, receiverAddress, 9876);
                    socket.send(packet);
```

```
          System.out.println("Sent: " + message);
          frameNumber++;
        }
      }

      // Receive acknowledgments
      socket.setSoTimeout(2000); // 2 seconds timeout for ACKs
      try {
        for (int j = nextAck; j < frameNumber; j++) {
          if (!ackReceived[j]) {
            byte[] ackBuffer = new byte[1024];
            DatagramPacket ackPacket = new DatagramPacket(ackBuffer,
ackBuffer.length);
            socket.receive(ackPacket);

            String ackMessage = new String(ackPacket.getData()).trim();
            int ackNum = Integer.parseInt(ackMessage.split(" ")[1]);

            if (ackNum == j) {
              System.out.println("Received ACK for Frame " + ackNum);
              ackReceived[j] = true;

              if (j == nextAck) {
                while (nextAck < TOTAL_FRAMES && ackReceived[nextAck])
{
                  nextAck++;
                }
              }
            }
          }
        }
      } catch (SocketTimeoutException e) {
        System.out.println("Timeout: Resending frames from " + nextAck);
        frameNumber = nextAck; // Resend frames from the next
acknowledgment needed
      }
    }

    System.out.println("All frames sent successfully.");
    socket.close();
```

```
  }
}
```

****

Receiver :-

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.util.Random;
class Receiver {
  private static final int TOTAL_FRAMES = 10;
  private static final Random random = new Random();

  public static void main(String[] args) throws IOException {
    DatagramSocket socket = new DatagramSocket(9876);
    int expectedFrame = 0;

    while (true) {
      byte[] buffer = new byte[1024];
      DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
      socket.receive(packet);

      String message = new String(packet.getData()).trim();
      int frameNumber = Integer.parseInt(message.split(" ")[1]);

      // Randomly decide whether to acknowledge the frame to simulate
packet loss
      if (random.nextInt(10) < 8) { // 80% chance to ACK
        if (frameNumber == expectedFrame) {
          System.out.println("Received: " + message);
          expectedFrame++;
        } else {
          System.out.println("Received out-of-order frame: " +
frameNumber);
        }

        String ackMessage = "ACK " + frameNumber;
```

```
            byte[] ackBuffer = ackMessage.getBytes();

            DatagramPacket ackPacket = new DatagramPacket(
                  ackBuffer, ackBuffer.length, packet.getAddress(),
packet.getPort());
            socket.send(ackPacket);

            System.out.println("Sent: " + ackMessage);
         } else {
            System.out.println("Simulated loss for Frame " + frameNumber);
         }

         if (frameNumber == TOTAL_FRAMES - 1) {
            break;
         }
      }

      socket.close();
   }
}
```

****

Output :-

Sender Console –

```
Sent: Frame 0
Sent: Frame 1
Sent: Frame 2
Sent: Frame 3
Timeout: Resending frames from 0
Sent: Frame 0
Sent: Frame 1
Sent: Frame 2
Sent: Frame 3
Timeout: Resending frames from 0
Sent: Frame 0
Sent: Frame 1
```

```
Sent: Frame 2
Sent: Frame 3
Timeout: Resending frames from 0
Sent: Frame 0
Sent: Frame 1
Sent: Frame 2
Sent: Frame 3
Timeout: Resending frames from 0
Sent: Frame 0
Sent: Frame 1
Sent: Frame 2
Sent: Frame 3
Received ACK for Frame 0
Received ACK for Frame 1
Timeout: Resending frames from 2
Sent: Frame 2
Sent: Frame 3
Sent: Frame 4
Sent: Frame 5
Received ACK for Frame 2
Received ACK for Frame 3
Received ACK for Frame 4
Timeout: Resending frames from 5
Sent: Frame 5
Sent: Frame 6
Sent: Frame 7
Sent: Frame 8
Received ACK for Frame 5
Received ACK for Frame 6
Received ACK for Frame 7
Received ACK for Frame 8
Sent: Frame 9
Received ACK for Frame 9
All frames sent successfully.
```

Receiver Console –

```
Simulated loss for Frame 0
Simulated loss for Frame 1
Received out-of-order frame: 2
Sent: ACK 2
Received out-of-order frame: 3
Sent: ACK 3
Received: Frame 0
Sent: ACK 0
Received: Frame 1
Sent: ACK 1
Simulated loss for Frame 2
Received out-of-order frame: 3
Sent: ACK 3
Received: Frame 2
Sent: ACK 2
Received: Frame 3
Sent: ACK 3
Received: Frame 4
Sent: ACK 4
Simulated loss for Frame 5
Received: Frame 5
Sent: ACK 5
Received: Frame 6
Sent: ACK 6
Received: Frame 7
Sent: ACK 7
Received: Frame 8
Sent: ACK 8
Received: Frame 9
Sent: ACK 9
```