SIR M. VISVESVARAYA INSTITUTE OF TECHNOLOGY
Bengaluru - 562 157
Approved by AICTE | Affiliated to VTU, Belagavi | Accredited by NAAC
DEPARTMENT OF ARTIFICIAL INTELLIGENCE & MACHINE LEARNING

Academic Year 2024 – 2025 (ODD Semester)

LAB MANUAL

COMPUTER NETWORK LABORATORY

BCS502

V Semester

Prepared By:

Dr S Ambareesh

Professor

Dept. of AI&ML

Vision and Mission of the Institute

Vision

- To be a centre of excellence in technical and management education concurrently focusing on disciplined and integrated development of personality through quality education, sports, cultural and co-curricular activities.
- To promote transformation of students into better human beings, responsible citizens and competent professionals to serve as a valuable resource for industry, work environment and society.

Mission

- To impart quality technical education, provide state-of-art facilities, achieve high quality in teaching-learning & research and encourage extra & co-curricular activities.
- To stimulate in students a spirit of inquiry and desire to gain knowledge and skills to meet the changing needs that can enrich their lives.
- To provide opportunity and resources for developing skills for employability and entrepreneurship, nurturing leadership qualities, imbibing professional ethics and societal commitment.
- To create an ambiance and nurture conducive environment for dedicated and quality staff to upgrade their knowledge & skills and disseminate the same to students on a sustainable long term basis.
- To facilitate effective interaction with the industries, alumni and research institutions.

Vision and Mission of the Department

Vision

- To become a globally recognized institution that excels in providing quality education and fostering the development of highly skilled professionals in the field of Artificial Intelligence and Machine Learning.

Mission

- To provide a comprehensive and rigorous academic curriculum that equips students with the knowledge, Skills and Expertise required excelling in realm of Artificial Intelligence.
- To fostering a culture of innovation, critical thinking and ethical practices, while promoting inter-disciplinary collaboration and research.

- To empower our students to become leaders and contribute to the advancement of Artificial Intelligence on a global scale with the help of dedicated faculty, state-of-the-art facilities and industry partnerships.

PROGRAM EDUCATIONAL OBJECTIVES (PEOS)

PEO1: Apply practical engineering skills to pursue ethical and productive careers in the Artificial Intelligence and machine learning domain, as well as in related industries and organizations.

PEO2: Possess a foundation that enables and promotes qualified individuals to pursue advanced technical and professional degrees.

PEO3: Demonstrate adequate breadth to successfully transition into various professional areas, such as business, and management.

PROGRAM SPECIFIC OUTCOMES (PSOs)

PSO1: Apply tools and techniques in Artificial Intelligence and Machine Learning to address multidisciplinary problems.

PSO2: Expertise in identifying, Analysing , Designing and Developing Systems by applying principles and concepts of Devops , MangoDB, Project Management with GIT, Block-Chain Technology and MLOps.

PROGRAM OUTCOMES (POs)

1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental consideration.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one’s own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

SYLLABUS

PART - A

Description:

For the experiments below modify the topology and parameters set for the experiment and take multiple rounds of reading and analyze the results available in log files. Plot necessary graphs and conclude. Use NS2/NS3.

1. 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.
2. 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.
3. Implement an Ethernet LAN using "n" nodes and set multiple traffic nodes and plot congestion window for different source / destination.

PART B

Implement the following in Java:

4. Develop a program for error detecting code using CRC-CCITT (16- bits).
5. Develop a program to implement a sliding window protocol in the data link layer.
6. Develop a program to find the shortest path between vertices using Bellman-Ford and path vector routing algorithm.
7. 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.
8. Develop a program on datagram socket for client/server to display the messages on client side, typed at the server side.
9. Develop a program for a simple RSA algorithm to encrypt and decrypt the data.
10. Develop a program for congestion control using leaky bucket algorithm.

CIE for the practical component of the IPCC

i. 15 marks for the conduction of the experiment and preparation of laboratory record, and 10 marks for the test to be conducted after the completion of all the laboratory sessions.

ii. On completion of every experiment/program in the laboratory, the students shall be evaluated including viva-voce and marks shall be awarded on the same day.

iii. The CIE marks awarded in the case of the Practical component shall be based on the continuous evaluation of the laboratory report. Each experiment report can be evaluated for 10 marks. Marks of all experiments' write-ups are added and scaled down to 15 marks.

iv. The laboratory test (duration 02/03 hours) after completion of all the experiments shall be conducted for 50 marks and scaled down to 10 marks.

v. Scaled-down marks of write-up evaluations and tests added will be CIE marks for the laboratory component of IPCC for 25 marks.

vi. The student has to secure 40% of 25 marks to qualify in the CIE of the practical component of the IPCC.

Introduction to Network Simulators

- Network simulators implemented in software are valuable tools for researchers to develop, test, and diagnose network protocols. Simulation is economical because it can carry out experiments without the actual hardware. It is flexible because it can, for example, simulate a link with any bandwidth and propagation delay. Simulation results are easier to analyze than experimental results because important information at critical points can be easily logged to help researchers diagnose network protocols.
- Network simulators, however, have their limitations. A complete network simulator needs to simulate networking devices (e.g., hosts and routers) and application programs that generate network traffic. It also needs to provide network utility programs to configure, monitor, and gather statistics about a simulated network. Therefore, developing a complete network simulator is a large effort.
- Due to limited development resources, traditional network simulators usually have the following drawbacks:
- Simulation results are not as convincing as those produced by real hardware and software equipment. In order to constrain their complexity and development cost, most network simulators usually can only simulate real-life network protocol implementations with limited details, and this may lead to incorrect results.
- These simulators are not extensible in the sense that they lack the standard UNIX POSIX application programming interface (API). As such, existing or to-be-developed real-life application programs cannot run normally to generate traffic for a simulated network.
- Instead, they must be rewritten to use the internal API provided by the simulator (if there is any) and be compiled with the simulator to form a single, big, and complex program.
- To overcome these problems, Wang invented a kernel re-entering simulation methodology and used it to implement the Harvard network simulator. Later on, Wang further improved the methodology and used it to design and implement the NCTUns network simulator and emulator.

- Different types of simulators, Some of the different types of simulators are as follows:-
    i. MIT's NETSIM
    ii. NIST
    iii. CPSIM
    iv. INSANE
    v. NEST
    vi. REAL
    vii. NS
    viii. OPNET
    ix. NCTUns
    x. P
    xi. GNS3
    xii. WireShark

Table of contents

PART A

1. 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.

TCL Program:

lab1.tcl

```
set ns [new Simulator]
set nf [open lab1.nam w]
$ns namtrace-all $nf
set tf [open lab1.tr w]
$ns trace-all $tf
proc finish { } {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam lab1.nam &
exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
$ns duplex-link $n0 $n2 200Mb 10ms DropTail
$ns duplex-link $n1 $n2 100Mb 5ms DropTail
$ns duplex-link $n2 $n3 1Mb 1000ms DropTail
$ns queue-limit $n0 $n2 10
$ns queue-limit $n1 $n2 10
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
```

```
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
set udp2 [new Agent/UDP]
$ns attach-agent $n2 $udp2
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
set null0 [new Agent/Null]
$ns attach-agent $n3 $null0
$ns connect $udp0 $null0
$ns connect $udp1 $null0
$ns at 0.1 "$cbr0 start"
$ns at 0.2 "$cbr1 start"
$ns at 1.0 "finish"
$ns run
```

AWK Program:

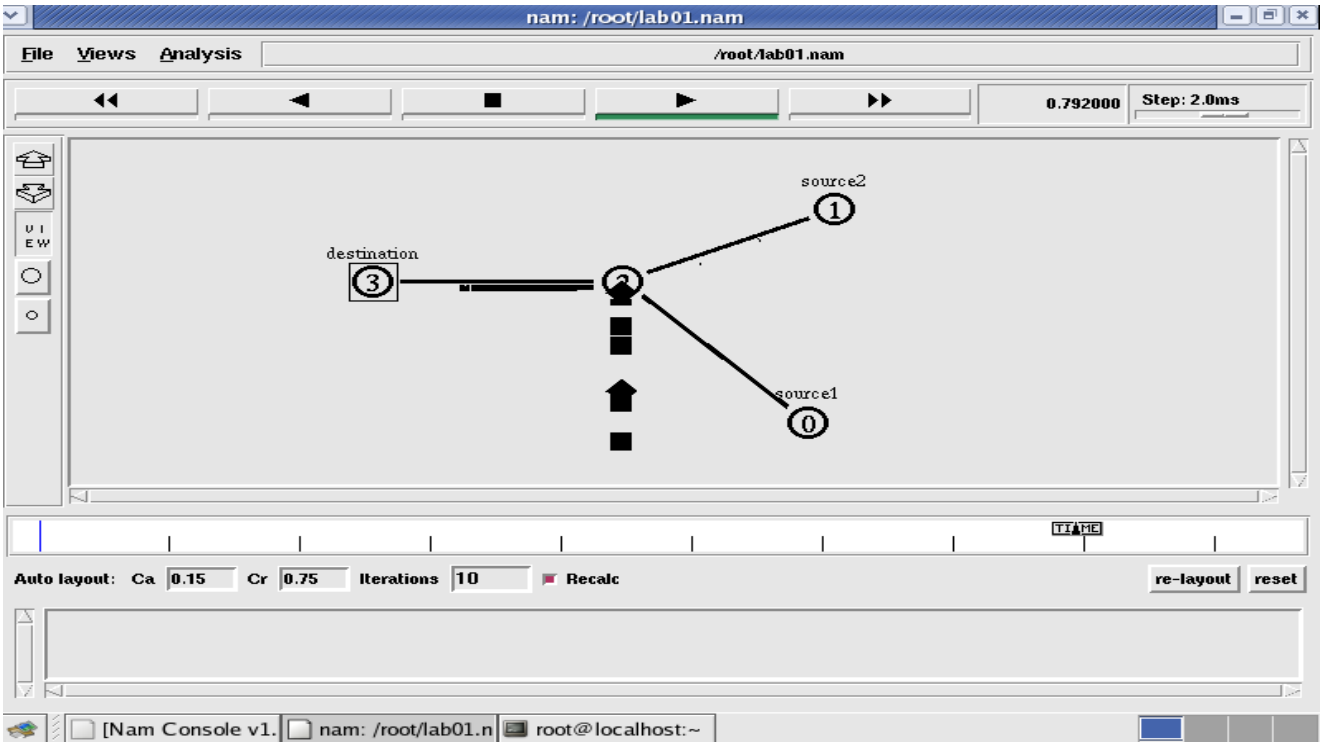
lab1.awk

```
BEGIN {  c=0;
}
{
If ($1= ="d")
{
c++;
printf("%s\t%s\n",$5,$11);
}
}
END{
printf("The number of packets dropped =%d\n",c);
}
```

OUTPUT:

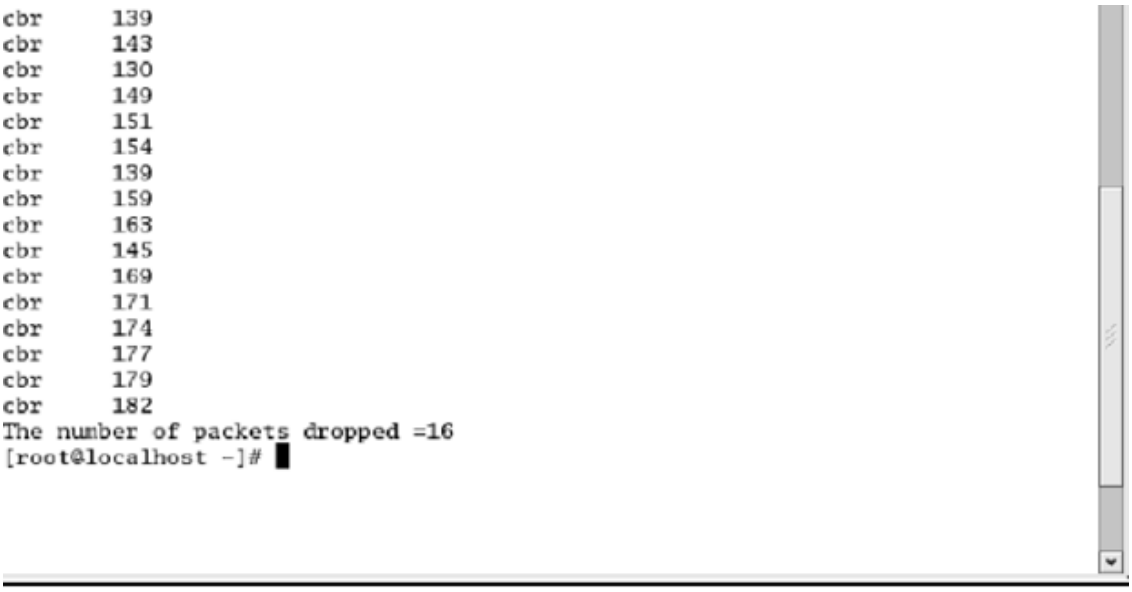
[root@localhost ~]# gedit lab1.tcl

[root@localhost ~]# ns lab1.tcl



[root@localhost ~]# gedit lab1.awk

[root@localhost ~]# awk –f lab1.awk lab1.tr



2. 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.

TCL Program:

lab2.tcl

```
set ns [ new Simulator ]
set nf [ open lab2.nam w ]
$ns namtrace-all $nf
set tf [ open lab2.tr w ]
$ns trace-all $tf

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n4 shape box

$ns duplex-link $n0 $n4 1005Mb 1ms DropTail
$ns duplex-link $n1 $n4 50Mb 1ms DropTail
$ns duplex-link $n2 $n4 2000Mb 1ms DropTail
$ns duplex-link $n3 $n4 200Mb 1ms DropTail
$ns duplex-link $n4 $n5 1Mb 1ms DropTail

set p1 [new Agent/Ping]
$ns attach-agent $n0 $p1
$p1 set packetSize_ 50000
$p1 set interval_ 0.0001


set p2 [new Agent/Ping]
$ns attach-agent $n1 $p2
```

```
set p3 [new Agent/Ping]
$ns attach-agent $n2 $p3
$p3 set packetSize_30000
$p3 set interval_ 0.00001

set p4 [new Agent/Ping]
$ns attach-agent $n3 $p4

set p5 [new Agent/Ping]
$ns attach-agent $n5 $p5

$ns queue-limit $n0 $n4 5
$ns queue-limit $n2 $n4 3b
$ns queue-limit $n4 $n5 2

Agent/Ping instproc recv {from rtt} {
$self instvar node_
puts "node [$node_ id] received answer from $from with round trip time $rtt msec"
}

$ns connect $p1 $p5
$ns connect $p3 $p4

proc finish { } {
global ns nf tf
$ns flush-trace
close $nf
close $tf
exec nam lab2.nam &
exit 0
}
```

```
$ns at 0.1 "$p1 send"
$ns at 0.2 "$p1 send"
$ns at 0.3 "$p1 send"
$ns at 0.4 "$p1 send"
$ns at 0.5 "$p1 send"
$ns at 0.6 "$p1 send"
$ns at 0.7 "$p1 send"
$ns at 0.8 "$p1 send"
$ns at 0.9 "$p1 send"
$ns at 1.0 "$p1 send"
$ns at 1.1 "$p1 send"
$ns at 1.2 "$p1 send"
$ns at 1.3 "$p1 send"
$ns at 1.4 "$p1 send"
$ns at 1.5 "$p1 send"
$ns at 1.6 "$p1 send"
$ns at 1.7 "$p1 send"
$ns at 1.8 "$p1 send"
$ns at 1.9 "$p1 send"
$ns at 2.0 "$p1 send"
$ns at 2.1 "$p1 send"
$ns at 2.2 "$p1 send"
$ns at 2.3 "$p1 send"
$ns at 2.4 "$p1 send"
$ns at 2.5 "$p1 send"
$ns at 2.6 "$p1 send"
$ns at 2.7 "$p1 send"
$ns at 2.8 "$p1 send"
$ns at 2.9 "$p1 send"
$ns at 0.1 "$p3 send"
$ns at 0.2 "$p3 send"
```

```
$ns at 0.3 "$p3 send"
$ns at 0.4 "$p3 send"
$ns at 0.5 "$p3 send"
$ns at 0.6 "$p3 send"
$ns at 0.7 "$p3 send"
$ns at 0.8 "$p3 send"
$ns at 0.9 "$p3 send"
$ns at 1.0 "$p3 send"
$ns at 1.1 "$p3 send"
$ns at 1.2 "$p3 send"
$ns at 1.3 "$p3 send"
$ns at 1.4 "$p3 send"
$ns at 1.5 "$p3 send"
$ns at 1.6 "$p3 send"
$ns at 1.7 "$p3 send"
$ns at 1.8 "$p3 send"
$ns at 1.9 "$p3 send"
$ns at 2.0 "$p3 send"
$ns at 2.1 "$p3 send"
$ns at 2.2 "$p3 send"
$ns at 2.3 "$p3 send"
$ns at 2.4 "$p3 send"
$ns at 2.5 "$p3 send"
$ns at 2.6 "$p3 send"
$ns at 2.7 "$p3 send"
$ns at 2.8 "$p3 send"
$ns at 2.9 "$p3 send"
$ns at 3.0 "finish"
$ns run
```

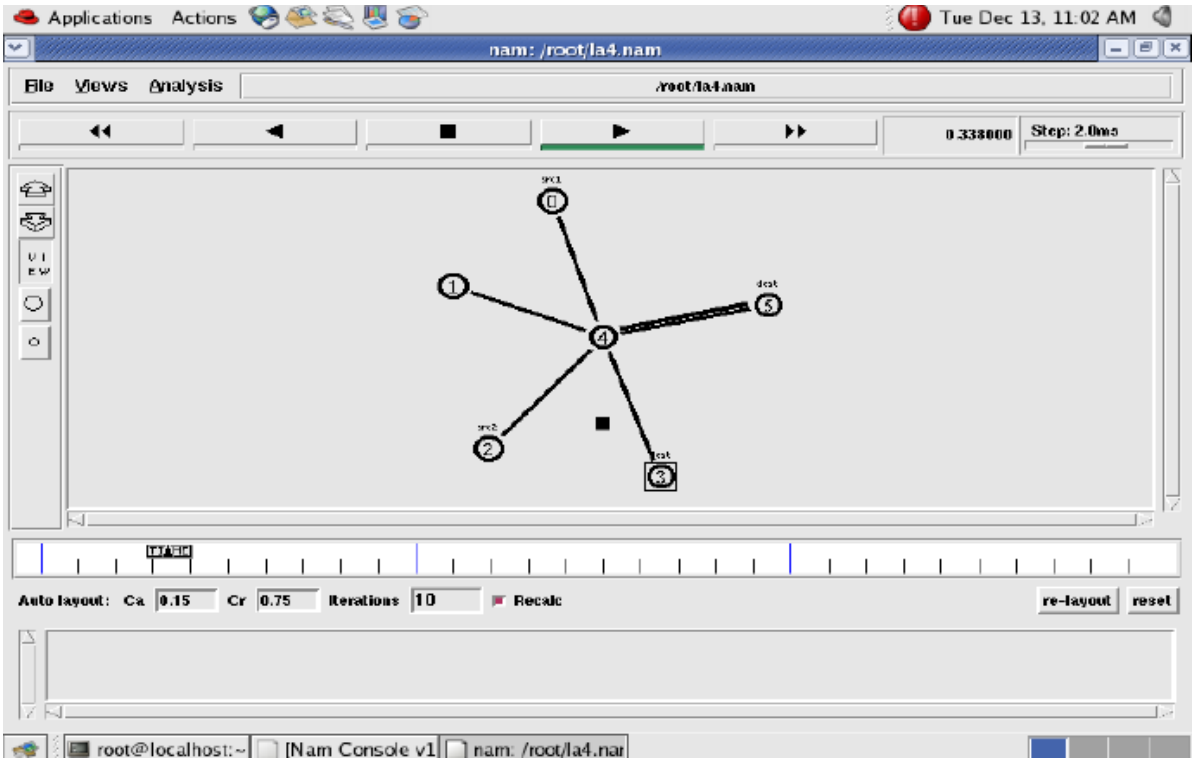
AWK Program:

lab2.awk

```
BEGIN{
drop=0;
}
{
if($1=="d" )
{
drop++;
}
} END{
printf("Total number of %s packets dropped due to congestion =%d\n",$5,drop);
}
```

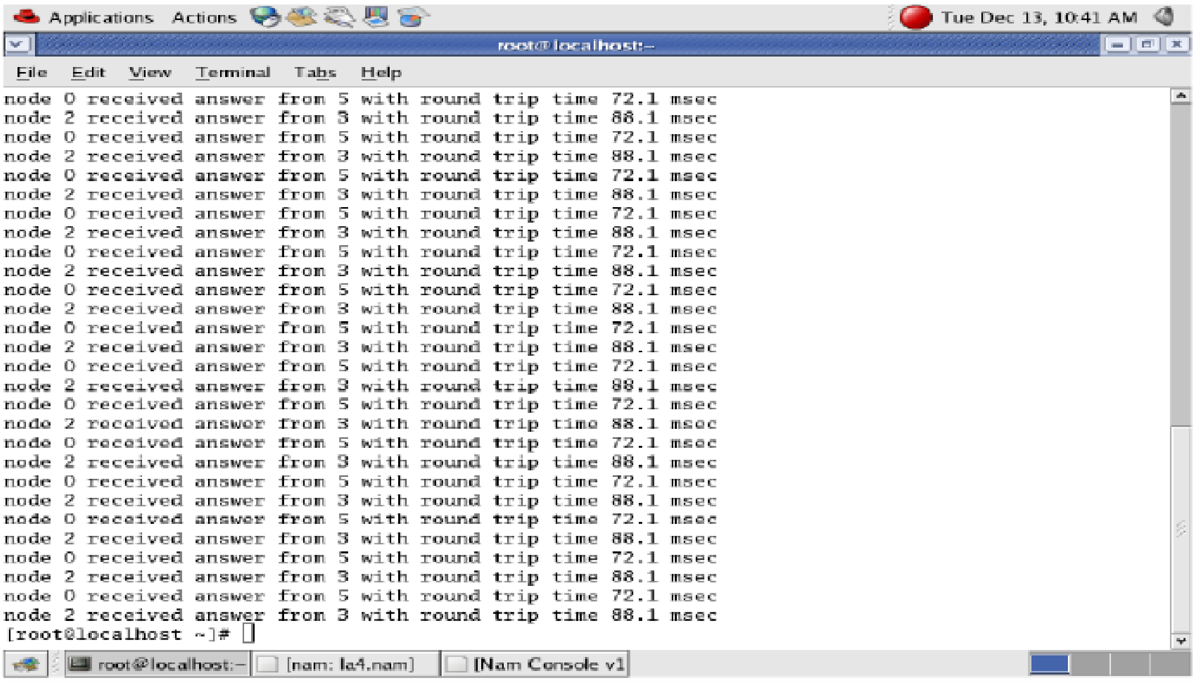
OUTPUT:

[root@localhost ~]# ns lab2.tcl



[root@localhost ~]# awk –f lab2.awk lab2.tr

Total number of ping packets dropped due to congestion = 20



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

TCL Program:

lab3.tcl

```
set ns [new Simulator]

set tf [open lab3.tr w]
$ns trace-all $tf


set nf [open lab3.nam w]
$ns namtrace-all $nf


set n0 [$ns node]
$n0 color "magenta"
$n0 label "src1"

set n1 [$ns node]

set n2 [$ns node]
$n2 color "magenta"
$n2 label "src2"

set n3 [$ns node]
$n3 color "blue"
$n3 label "dest2"

set n4 [$ns node]

set n5 [$ns node]
$n5 color "blue"
$n5 label "dest1"
```

```
LanRouter set debug_ 0
$ns make-lan "$n0 $n1 $n2 $n3 $n4" 100Mb 100ms LL Queue/DropTail Mac/802_3
$ns duplex-link $n4 $n5 1Mb 1ms DropTail

set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 500
$ftp0 set interval_ 0.0001

set sink5 [new Agent/TCPSink]
$ns attach-agent $n5 $sink5

$ns connect $tcp0 $sink5

set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set packetSize_ 600
$ftp2 set interval_ 0.001

set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3
$ns connect $tcp2 $sink3

set file1 [open file1.tr w]
```

```
$tcp0 attach $file1

set file2 [open file2.tr w]
$tcp2 attach $file2

$tcp0 trace cwnd_
$tcp2 trace cwnd_

proc finish { } {
global ns nf tf
$ns flush-trace
close $tf
close $nf
exec nam lab3.nam &
exit 0
}

$ns at 0.1 "$ftp0 start"
$ns at 5 "$ftp0 stop"
$ns at 7 "$ftp0 start"
$ns at 0.2 "$ftp2 start"
$ns at 8 "$ftp2 stop"
$ns at 14 "$ftp0 stop"
$ns at 10 "$ftp2 start"
$ns at 15 "$ftp2 stop"
$ns at 16 "finish"
$ns run
```

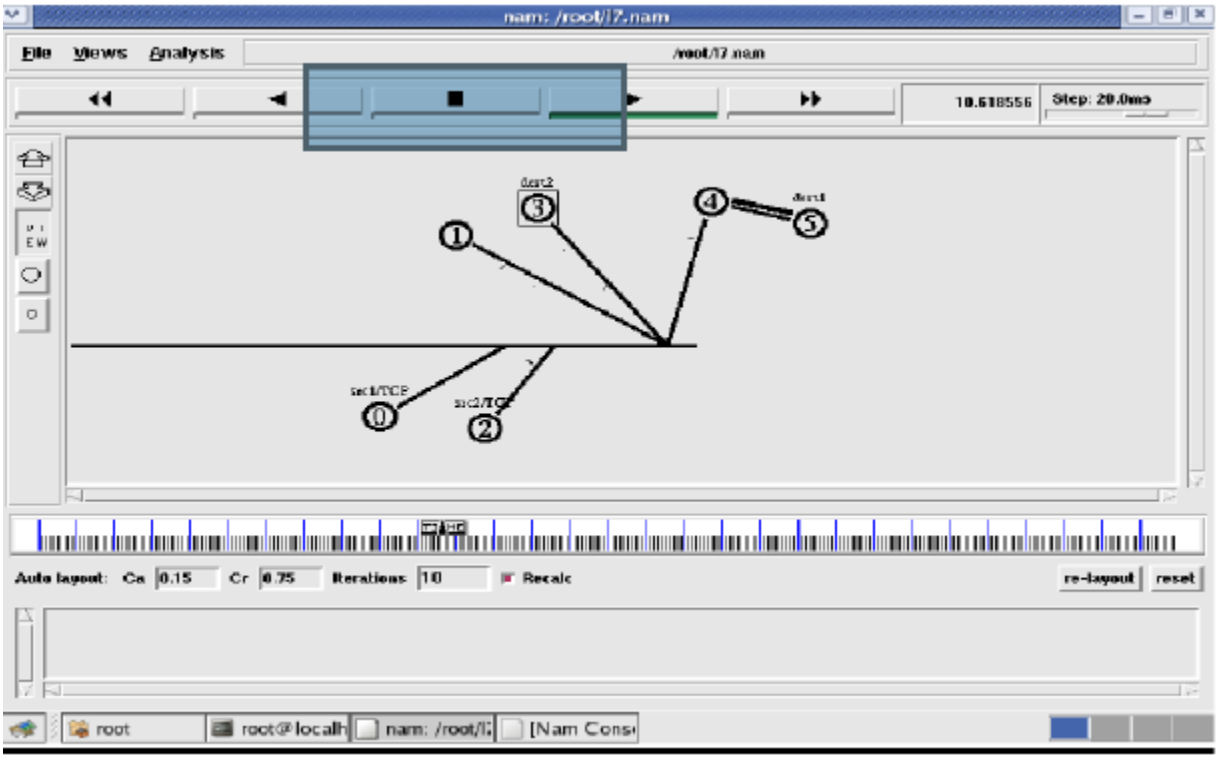
AWK Program:

lab3.awk

```
BEGIN {
}
{
if($6=="cwnd_")
printf("%f\t%f\t\n",$1,$7);
}
END {
}
```

OUTPUT:

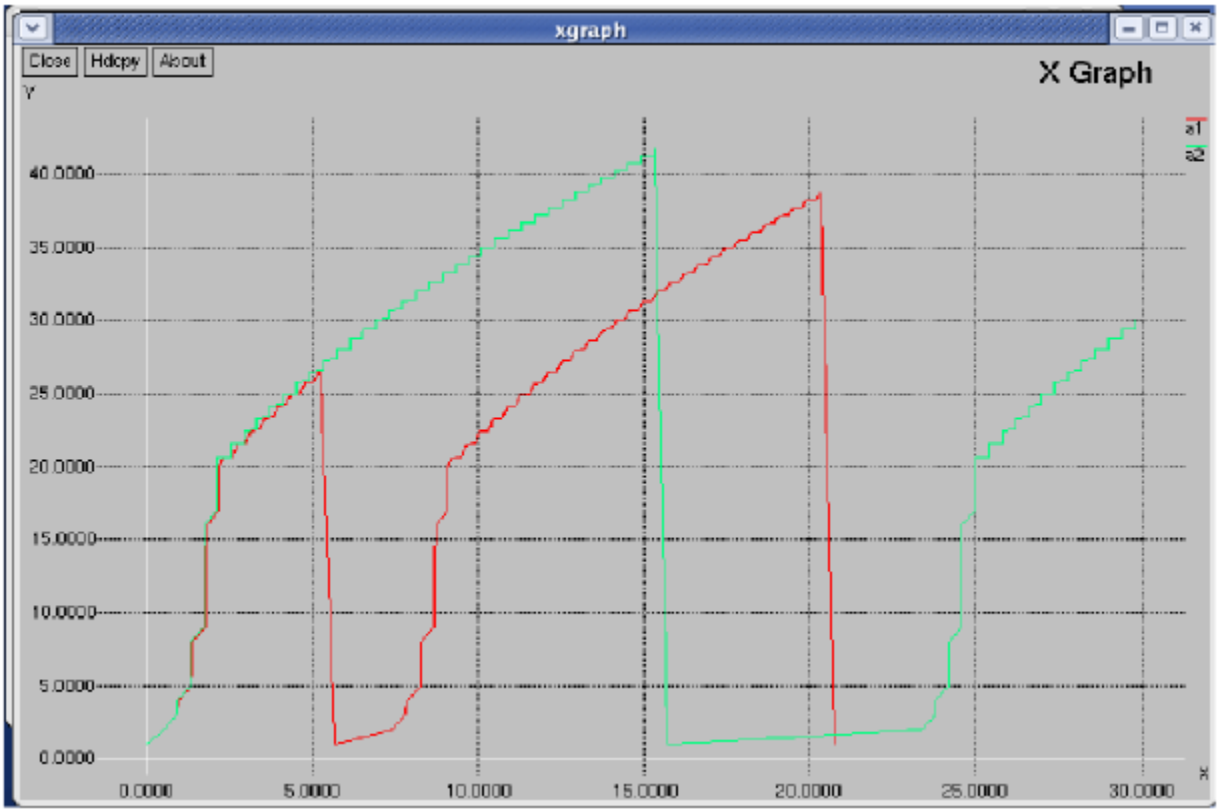
[root@localhost ~]# ns lab3.tcl



[root@localhost~]# awk –f lab3.awk file1.tr > a1

[root@localhost~]# awk –f lab3.awk file2.tr > a2

[root@localhost~]# xgraph a1 a2



PART B

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

Cyclic Redundancy Check Code for Error – Detection

The cyclic Redundancy Check (CRC) id a technique for detecting errors in data transmission, but not for correcting errors when they are detected.

CCITT- Consultative Committee for International Telegraphy and Telephone.

Algorithm:

A) For Computing CRC:

- The CRC Algorithm is based on polynomial arithmetic.
- Let the message that we have to send has k bits (denoted by M(x) in polynomial form having degree (k-1)) the sender and the receiver are agreed upon a generator polynomial having r bits (denoted by G(x) in polynomial form having degree (r-1)). The generator polynomial is also called "Divisor".
- Now, append (r-1) zero bits to the LSB side of the message M(x) so it will now contain (k+r-1) bits and corresponds to the polynomial $x^{(r-1)}$ M (x).
- Divide the polynomial $x^{(r-1)}$ M (x) by Divisor, using modulo-2 subtraction (bit by bit XOR operation). Add the remainder R(x) (called frame check sequence) to $x^{(r-1)}$ M (x), using modulo -2 additions (bit by bit XOR operation). This is the message that will be transmitted by the transmitter denoted by T(x).

B) For Error Detection:

- Suppose that a transmission error occurs, so that the received message at the receiver is T(x) +E(x), instead of T(x). Each 1 bit in E(x) corresponds to a bit that has been inverted.
- The received message at the receiver end is divided by G(x), i.e. [T(x) + E(x) / G(x)]. Since T(x)/G(x) is 0, so the result is simply E(x)/G(x).
- If E(x)/G(x) =0 than there is no error in the received message, otherwise there is an error.
- The following type of errors can be detected using CRC:
  - If G(x) has more than one bit and the coefficient of x^0 is 1, then all single bit errors are detected.
  - If G(x) is not divisible by x (the coefficient of x^0 is 1) , and t is the least positive integer (0<t<n-1) such that G(x) divides x^1 + 1, then all isolated double errors are detected.
  - If G(x) has a factor (x+1), then all odd numbered errors are detected.

Source code:

```
import java.util.*;
public class CRC
{
	void div(int a[],int k)
	{
		int gp[]={1,0,0,0,1,0,0,0,0,0,0,1,0,0,0,0,1};
		int count=0;
		for(int i=0;i<k;i++)
		{
			if(a[i]==gp[0])
			{
				for(int j=i;j<17+i;j++)
				{
					a[j]=a[j]^gp[count++];
				}
				count=0;
			}
		}
	}

	public static void main(String[] args)
	{
		int a[]=new int[100];
		int b[]=new int[100];
		int len,k;
		CRC ob=new CRC();
		System.out.println("Enter the length of Data Frame:");
		Scanner sc=new Scanner(System.in);
```

```
len=sc.nextInt();
int flag=0;
System.out.println("Enter the Message:");
for(int i=0;i<len;i++)
{
        a[i]=sc.nextInt();
}
for(int i=0;i<16;i++)
{
        a[len++]=0;
}
k=len-16;
for(int i=0;i<len;i++)
{
        b[i]=a[i];
}
ob.div(a,k);
for(int i=0;i<len;i++)
a[i]=a[i]^b[i];
System.out.println("Data to be transmitted: ");
for(int i=0;i<len;i++)
{
        System.out.print(a[i]+" ");
}
System.out.println();
System.out.println("Enter the Received Data: ");
for(int i=0;i<len;i++)
{
        a[i]=sc.nextInt();
```

```
                }
                ob.div(a, k);
                for(int i=0;i<len;i++)
                {
                        if(a[i]!=0)
                        {
                                flag=1;
                                break;
                        }
                }
                if(flag==1)
                        System.out.println("error in data");
                else
                        System.out.println("no error");
        }
}
```

OUTPUT

RUN 1

Enter the length of Data Frame:

4

Enter the Message:

1 1 0 1

Data to be transmitted:

1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1

Enter the Received Data:

1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1

no error

RUN 2

Enter the length of Data Frame:

4

Enter the Message:

1 1 0 1

Data to be transmitted:

1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1

Enter the Received Data:

1 0 0 1 1 1 0 1 0 0 0 1 1 0 1 0 1 1 0 1

error in data

6. Write a program to find the shortest path between vertices using bellman-ford algorithm.

Bell man ford algorithm is a procedure used to find all shortest path in a graph from one source to all other nodes. The algorithm was introduced by American mathematicians Richard Bellman and Lester ford. Computes shortest from a single source vertex to all of the other vertices in a weighted diagraph.

Source Code:

```
import java.util.Scanner;
public class BellmanFord
{
   private int d[];
   private int nov;
   public static final int MAX_VALUE = 999;

   public BellmanFord(int nov)
   {
      this.nov = nov;
      d = new int[nov + 1];
   }

   public void BellmanFordEvaluation(int s, int am[][])
   {
      for (int n = 1; n <= nov; n++)
      {
         d[n] = MAX_VALUE;
      }

      d[s] = 0;
      for (int n = 1; n <= nov - 1; n++)
      {
         for (int sn = 1; sn <= nov; sn++)
         {
            for (int dn = 1; dn <= nov; dn++)
                        {
                                    if (am[sn][dn] != MAX_VALUE)
                                    {
                                                if (d[dn] > d[sn] + am[sn][dn])
                                    d[dn] = d[sn] + am[sn][dn];
                        }
            }
         }
      }
```

```
    for (int sn = 1; sn <= nov; sn++)
    {
       for (int dn = 1; dn <= nov; dn++)
            {
                    if (am[sn][dn] != MAX_VALUE)
                    {
                            if(d[dn] > d[sn] + am[sn][dn])

                {
                                    System.out.println("The Graph contains negative egde cycle");
                                    break;
                            }
          }
       }
    }

    for (int v = 1; v <= nov; v++)
    {
       System.out.println("Distance of source  " + s + " to " + v + " is " + d[v]);
    }
}

  public static void main(String args[])
  {
     int nov = 0;
     int s;
     Scanner scanner = new Scanner(System.in);


     System.out.println("Enter the number of vertices");
     nov = scanner.nextInt();


     int am[][] = new int[nov + 1][nov + 1];
     System.out.println("Enter the adjacency matrix");
     for (int sn = 1; sn <= nov; sn++)
     {
        for (int dn = 1; dn <= nov; dn++)
        {
           am[sn][dn] = scanner.nextInt();
```

```
                if (sn == dn)
            {
                am[sn][dn] = 0;
                continue;
            }
            if (am[sn][dn] == 0)
            {
                am[sn][dn] = MAX_VALUE;
            }
        }
    }

    System.out.println("Enter the source vertex");
    s = scanner.nextInt();

    BellmanFord bellmanford = new BellmanFord(nov);
    bellmanford.BellmanFordEvaluation(s, am);
    scanner.close();
  }
}
```

OUTPUT:

```
Enter the number of vertices
4
Enter the adjacency matrix
0 5 0 0
5 0 3 4
0 3 0 2
0 4 2 0
Enter the source vertex
2
```

Distance of source 2 to 1 is 5

Distance of source 2 to 2 is 0

Distance of source 2 to 3 is 3

Distance of source 2 to 4 is 4

7. 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. Implement the above program using as message queues or FIFOs as IPC channels.

A socket is an endpoint of a two-way communication link between two programs running on the network. Socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent. Java provides a set of classes, defined in a package called java.net, to enable the rapid development of network applications.

The steps for creating a server program in TCP/IP are:

1. Open the Server Socket:
   ServerSocket server = new ServerSocket(PORT);

2. Wait for the Client Request:
   Socket client = server.accept(); // bind with the port no

3. Create I/O streams for communicating to the client
   InputStream istream = sock.getInputStream( );
   BufferedReader fileRead =new BufferedReader(new InputStreamReader(istream));

4. Perform communication with client
   Receive from client: String fname = fileRead.readLine( );
   Send to client: pwrite.println(str);

5. Close socket:
   client.close();

The steps for creating a client program in TCP/IP are:

1. Create a Socket Object:
   Socket client = new Socket(server, port_id);

2. Create I/O streams for communicating with the server.
   is = new DataInputStream(client.getInputStream());
   os = new DataOutputStream(client.getOutputStream());

3. Perform I/O or communication with the server:
   Receive data from the server: String line = is.readLine();
   Send data to the server: os.writeBytes("Hello\n");

4. Close the socket when done:
   client.close();

Source Code:

TCP Server.java

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
class TCPServer
{
public static void main(String args[])throws Exception
{
while(true)
{
ServerSocket ss=new ServerSocket(5000);
System.out.println ("Waiting for request");
Socket s=ss.accept();
System.out.println ("Connected With "+s.getInetAddress().toString());
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
try
{
String filename="";
filename=din.readUTF();
System.out.println("Receiving file name");
System.out.println("SendGet....Ok");
File f=new File(filename);
FileInputStream fin=new FileInputStream(f);
```

```
long sz=(int) f.length();
byte b[]=new byte [1024];
int read;
dout.writeUTF(Long.toString(sz));
dout.flush();
System.out.println ("File Size: "+sz+" Bytes");
System.out.println ("Receive Buffer size: "+ss.getReceiveBufferSize());
System.out.println("Sending file contents");
while((read = fin.read(b)) != -1)
{
dout.write(b, 0, read);
dout.flush();
}
fin.close();
dout.flush();
System.out.println("Send Complete");
}
catch(Exception e)
{
e.printStackTrace();
System.out.println("An error occured");
}
din.close();
s.close();
ss.close();
}
}
}
```

TCPClient.java

```
import java.io.*;
import java.net.*;
import java.util.Scanner;
class TCPClient
{
public static void main(String args[])throws Exception
{
String address = "";
Scanner sc=new Scanner(System.in);
System.out.println("Enter Server Address: ");
address=sc.nextLine();
Socket s=new Socket(address,5000);
DataInputStream din=new DataInputStream(s.getInputStream());
DataOutputStream dout=new DataOutputStream(s.getOutputStream());
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
String filename,rcvfile;
System.out.println("Enter File Name: ");
filename=sc.nextLine();
sc.close();
try
{
dout.writeUTF(filename);
dout.flush();
rcvfile="client"+filename;
System.out.println("Saving file as: "+rcvfile);
FileOutputStream fos=new FileOutputStream(new File(rcvfile),true);
long bytesRead;
System.out.println("Receiving file..");
long sz=Long.parseLong(din.readUTF());
System.out.println ("File Size: "+sz+" Bytes");
```

```
byte b[]=new byte [1024];

do
{
bytesRead = din.read(b, 0, b.length);
fos.write(b,0,b.length);
}while(!(bytesRead<1024));
System.out.println("Completed");
fos.close();
dout.close();
s.close();
}
catch(EOFException e)
{
//do nothing
}
}
}
```

OUTPUT:

TCP Server

```
Waiting for request
Connected With /127.0.0.1
Receiving file name
SendGet....Ok
File Size: 17 Bytes
Receive Buffer size: 8192
Sending file contents
```

TCP Client

```
Enter Server Address:
127.0.0.1
Enter File Name:
sample.txt
Saving file as: clientsample.txt
Receiving file..
File Size: 17 Bytes
Completed
```

8. Write a program on datagram socket for client/server to display the messages on client side, typed at the server side.

UDPServer.java

```
import java.net.*;

public class UDPServer
{
	public static void main(String[] args)
	{
		DatagramSocket skt=null;
		try
		{
			skt=new DatagramSocket(6788);
			byte[] buffer = new byte[1000];
			System.out.println("Listening on port 6788");
			while(true)
			{
				DatagramPacket request = new DatagramPacket(buffer,buffer.length);
				skt.receive(request);
				String message = new String(request.getData());
				System.out.println("server received request ");
				String toUpper = message.toUpperCase();
				byte[] sendMsg= toUpper.getBytes();
				System.out.println("server sending response ");
				DatagramPacket reply = new DatagramPacket(sendMsg,sendMsg.length,
				request.getAddress(),request.getPort());
				skt.send(reply);
			}
		}
		catch(Exception ex)
```

```
                {
                }
        }
}
```

UDPClient.java

```
import java.net.*;
import java.util.Scanner;
public class UDPClient
{
        public static void main(String[] args)
        {
                DatagramSocket skt;
                Scanner scan = new Scanner(System.in);
                try
                {
                        System.out.println("Enter Message:");
                        String msg= scan.next();
                        skt=new DatagramSocket();

                        byte[] b = msg.getBytes();
                        InetAddress host=InetAddress.getByName("127.0.0.1");
                        int serverPort=6788;
                        DatagramPacket request =new DatagramPacket (b,b.length,host,serverPort);
                        skt.send(request);
                        byte[] buffer =new byte[1000];
                        DatagramPacket reply= new DatagramPacket(buffer,buffer.length);
                        skt.receive(reply);
```

```
                String s1 = new String(reply.getData());
                System.out.println("Client received: " + s1.trim());
                skt.close();
            }
            catch(Exception ex)
            {
            }
        }
}
```

OUTPUT:

UDPServer

```
Listening on port 6788
server received request
server sending response
```

UDPClient

```
Enter Message: networks
Client received: NETWORKS
```

9. Write a program for simple RSA algorithm to encrypt and decrypt the data.

RSA is an example of public key cryptography. It was developed by Rivest, Shamir and Adelman. The RSA algorithm can be used for both public key encryption and digital signatures. Its security is based on the difficulty of factoring large integers.

The RSA algorithm's efficiency requires a fast method for performing the modular exponentiation operation. A less efficient, conventional method includes raising a number (the input) to a power (the secret or public key of the algorithm, denoted e and d, respectively) and taking the remainder of the division with N. A straight- forward implementation performs these two steps of the operation sequentially: first, raise it to the power and second, apply modulo. The RSA algorithm comprises of three steps, which are depicted below:

Key generation algorithm

1. Generate two large random primes, p and q, of approximately equal size such that their product n=p*q
2. Compute n= p*q and Euler's totient function (@) phi(n) = (p-1)(q-1)
3. Choose an integer e, 1< e < phi, such that gcd(e, phi)=1
4. Compute the secret exponent d, 1< d <phi , such that e*d ≡1(mod phi)
5. The public key is (e,n) and the private key is (d,n). The values of p, q and phi should also be kept secret.

Encryption

Sender A does the following:

1. Using the public key(e,n)
2. Represents the plaintext message as a positive integer M
3. Computes the cipher text C= M^e mod n.
4. Sends the cipher text C to B (Receiver)

Decryption

Recipient B does the following:

1. Uses his private key (d,n) to compute M=C^d mod n
2. Extracts the plaintext from the integer representative m.

Source Code:

```
import java.util.*;
import java.io.*;
public class RSA
{
   static int gcd(int m,int n)
   {
      while(n!=0)
      {
          int r=m%n;
         m=n;
         n=r;
      }
      return m;
   }
   public static void main(String args[])
   {
      int p=0,q=0,n=0,e=0,d=0,phi=0;
      int nummes[]=new int[100];
      int encrypted[]=new int[100];
      int decrypted[]=new int[100];
      int i=0,j=0,nofelem=0;
      Scanner sc=new Scanner(System.in);
      String message ;
      System.out.println("Enter the Message to be encrypted:");
      message= sc.nextLine();
      System.out.println("\nEnter value of p and q");
      p=sc.nextInt();
```

```
q=sc.nextInt();
n=p*q;
phi=(p-1)*(q-1);
for(i=2;i<phi;i++)
   if(gcd(i,phi)==1)
       break;
e=i;
for(i=2;i<phi;i++)
   if((e*i-1)%phi==0)
      break;
d=i;
nofelem=message.length();

for(i=0;i< nofelem;i++)
{
   char c = message.charAt(i);
   nummes[i]=c-96;
}
for(i=0;i<nofelem;i++)
{
   encrypted[i]=1;
   for(j=0;j<e;j++)
      encrypted[i] =(encrypted[i]*nummes[i])%n;
}
System.out.println("\nEncrypted message\n");
for(i=0;i<nofelem;i++)
{
   System.out.print(encrypted[i]);
```

```
        }

        for(i=0;i<nofelem;i++)
        {
            decrypted[i]=1;
            for(j=0;j<d;j++)
               decrypted[i]=(decrypted[i]*encrypted[i])%n;
        }
        System.out.println("\n\nDecrypted message\n ");
        for(i=0;i<nofelem;i++)
           System.out.print((char)(decrypted[i]+96));
    }
}
```

OUTPUT:

```
Enter the Message to be encrypted:
Computer Networks Lab

Enter value of p and q
7
17

Encrypted message
-7136136721903186-64-86319010936864466-64-90132

Decrypted message
 Computer Networks Lab
```

10. **Write a program for congestion control using leaky bucket algorithm**.

What is **congestion**?
A state occurring in network layer when the message traffic is so heavy that it slows down network response time.
Effects of Congestion

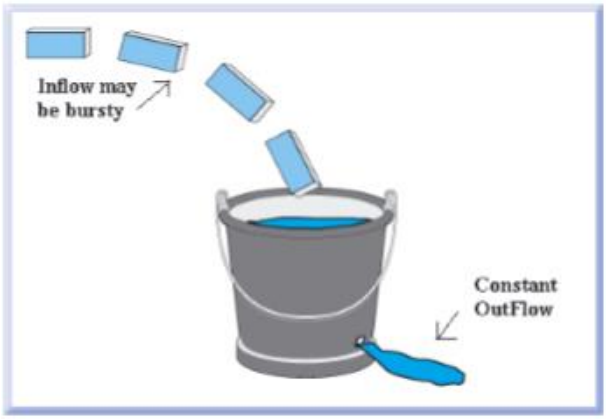
- As delay increases, performance decreases.
- If delay increases, retransmission occurs, making situation worse.

Traffic shaping or policing: To control the amount and rate of traffic is called Traffic shaping or policing. Traffic shaping term is used when the traffic leaves a network. Policing term is used when the data enters the network. Two techniques can shape or police the traffic leaky bucket and token bucket.

Congestion control algorithms

- **Leaky Bucket Algorithm**

Let us consider an example to understand
Imagine a bucket with a small hole in the bottom. No matter at what rate water enters the bucket, the outflow is at constant rate. When the bucket is full with water additional water entering spills over the sides and is lost.



Similarly, each network interface contains a leaky bucket and the following **steps** are involved in leaky bucket algorithm:

1. When host wants to send packet, packet is thrown into the bucket.
2. The bucket leaks at a constant rate, meaning the network interface transmits packets at a constant rate.
3. Bursty traffic is converted to a uniform traffic by the leaky bucket.
4. In practice the bucket is a finite queue that outputs at a finite rate.

Source Code:

```
import java.util.Scanner;
public class LeakyBucket
{
	public static void main(String args[])
	{
		int n, outgoing, store, bucketSize;
		int incoming[];
		Scanner scan = new Scanner(System.in);
		System.out.println("Enter number of inputs");
		n = scan.nextInt();
		incoming = new int[n];
		for(int i = 0 ;i< n ; i++)
		{
			System.out.println("Enter incoming packet size "+(i+1));
			incoming[i] = scan.nextInt();
		}
		System.out.println("Enter bucket size");
		bucketSize = scan.nextInt();
		System.out.println("Enter outgoing rate");
		outgoing = scan.nextInt();
		store = 0;
		int i = 0;
		System.out.println("Packet Recieved | Packet Dropped | Packet Sent | Packet Left");

		do
		{
```

```
int pktReceived = 0, pktSent = 0,pktDrop = 0;
if(i < n)
{
        pktReceived = incoming[i];

        if(pktReceived  <= (bucketSize - store))
        {
                store += pktReceived ;
        }
        else
        {

                pktDrop = pktReceived -(bucketSize - store);
                store = bucketSize;
        }
}
if(store > outgoing)
{
        store -= outgoing;
        pktSent = outgoing;
}
else
{
        pktSent = store;
        store = 0;
}
System.out.println(pktReceived +"\t\t"+pktDrop+"\t\t"+pktSent+"\t\t"+store);
try
```

```
                    {
                        Thread.sleep(2000);
                    }
                    catch(Exception e)
                    {
                    }
                    i++;

                }while(store != 0 || i < n);

        }
}
```

OUTPUT

```
Enter number of inputs
5
Enter incoming packet size 1
20
Enter incoming packet size 2
60
Enter incoming packet size 3
40
Enter incoming packet size 4
75
Enter incoming packet size 5
30
```

Enter bucket size

50

Enter outgoing rate

25

Packet Received | Packet Dropped | Packet Sent | Packet Left

20	0	20	0
60	10	25	25
40	15	25	25
75	50	25	25
30	5	25	25
0	0	25	0

Viva Questions

1. What are functions of different layers?
2. Differentiate between TCP/IP Layers and OSI Layers
3. Why header is required?
4. What is the use of adding header and trailer to frames?
5. What is encapsulation?
6. Why fragmentation requires?
7. What is MTU?
8. Which layer imposes MTU?
9. Differentiate between flow control and congestion control.
10. Differentiate between Point-to-Point Connection and End-to-End connections.
11. What are protocols running in different layers?
12. What is Protocol Stack?
13. Differentiate between TCP and UDP.
14. Differentiate between Connectionless and connection oriented connection.
15. Why frame sorting is required?
16. What is meant by subnet?
17. What is meant by Gateway?
18. What is an IP address?
19. What is MAC address?
20. Why IP address is required when we have MAC address?
21. What is meant by port?
22. What are ephemerical port number and well known port numbers?
23. What is a socket?
24. What are the parameters of socket()?
25. Describe bind(), listen(), accept(), connect(), send() and recv().
26. What are system calls? Mention few of them.
27. What is meant by file descriptor?
28. What is meant by traffic shaping?
29. How do you classify congestion control algorithms?

30. How do you implement Leaky bucket?
31. How do you generate busty traffic?
32. What is the polynomial used in CRC-CCITT?
33. . What are the other error detection algorithms?
34. What are Routing algorithms?
35. How do you classify routing algorithms? Give examples for each.
36. What are drawbacks in distance vector algorithm?
37. How routers update distances to each of its neighbor?
38. How do you overcome count to infinity problem?
39. What is cryptography?
40. How do you classify cryptographic algorithms?
41. What is public key?
42. What is private key?
43. What are key cipher text and plaintext?
44. What is simulation?
45. What are advantages of simulation?
46. Differentiate between Simulation and Emulation.
47. What is meant by router, bridge, switch, hub?
48. Differentiate between route, bridge, switch and hub.
49. What is ping and telnet?
50. What is FTP?
51. What is collision?
52. How do you generate multiple traffics across different sender-receiver pairs?
53. What is meant by mobile host?
54. Name few other Network simulators
55. Differentiate between logical and physical address.
56. Which address gets affected if a system moves from one place to anotherplace?
57. What is ICMP? What are uses of ICMP? Name few.
58. Which layer implements security for data?
59. What is connectionless and connection oriented protocol?
60. What is Congestion window?