Dayananda Sagar Academy of Technology & Management

(Autonomous Institute under VTU)

Accredited by NAAC with A+ grade, Accredited by 6 UG programs by NBA, New Delhi

Opp. Art of Living, Udayapura, Kanakapura Road,Bangalore – 560082

Department of Information Science & Engineering

2024-2025

Computer Networks Laboratory Manual

BCS502

Compiled by

Dr. Rajesh L (Associate Professor)

Dr. Rashmi Amardeep(Associate Professor)

Dr. Pallavi H B (Associate Professor)

COMPUTER NETWORKS LAB / BCS502

Laboratory Component:

1. Implement three nodes point – to – point network with duplex links between them for different topologies. 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 in the network.
3. Implement an Ethernet LAN using n nodes and set multiple traffic nodes and plot congestion window for different source / destination.
4. Write 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 the 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 a 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. Write a program for congestion control using leaky bucket algorithm.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

CIE for the theory component of the IPCC (maximum marks 50)

● IPCC means practical portion integrated with the theory of the course.

● CIE marks for the theory component are **25 marks** and that for the practical component is **25 marks**.

● **25 marks** for the theory component are split into **15 marks** for two Internal Assessment Tests (Two Tests, each of **15 Marks** with 01-hour duration, are to be conducted) and **10 marks** for other assessment methods mentioned in 22OB4.2. The first test at the end of 40-50% coverage of the syllabus and the second test after covering 85-90% of the syllabus.

● Scaled-down marks of the sum of two tests and other assessment methods will be CIE marks for the theory component of IPCC (that is for 25 marks).

● The student has to secure **40% of 25 marks** to qualify in the CIE of the theory component of IPCC. CIE for the practical component of the IPCC

● **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.

● 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.

● 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**.

● 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**.

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

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

SEE for IPCC

Theory SEE will be conducted by University as per the scheduled timetable, with common question papers for the course **(duration 03 hours)**

1. The question paper will have ten questions. Each question is set for 20 marks.
2. There will be 2 questions from each module. Each of the two questions under a module (with a maximum of 3 sub-questions), should have a mix of topics under that module.
3. The students have to answer 5 full questions, selecting one full question from each module.
4. Marks scored by the student shall be proportionally scaled down to 50 Marks

The theory portion of the IPCC shall be for both CIE and SEE, whereas the practical portion will have a CIE component only. Questions mentioned in the SEE paper may include questions from the practical component.

VISION OF THE INSTITUTE

To strive at creating the institution a center of highest caliber of learning, so as to create an overall intellectual atmosphere with each deriving strength from the other to be the best of engineers, scientists with management & design skills.

MISSION OF THE INSTITUTE

- To serve its region, state, the nation and globally by preparing students to make meaningful contributions in an increasing complex global society challenge.
- To encourage, reflection on and evaluation of emerging needs and priorities with state of art infrastructure at institution.
- To support research and services establishing enhancements in technical, economic, human and cultural development.
- To establish inter disciplinary center of excellence, supporting/ promoting student's implementation.
- To increase the number of Doctorate holders to promote research culture on campus.
- To establish IIPC, IPR, EDC, innovation cells with functional MOU's supporting student's quality growth

QUALITY POLICY

Dayananda Sagar Academy of Technology and Management aims at achieving academic excellence through continuous improvement in all spheres of Technical and Management education. In pursuit of excellence cutting-edge and contemporary skills are imparted to the utmost satisfaction of the students and the concerned stakeholders

OBJECTIVES & GOALS

- Creating an academic environment to nurture and develop competent entrepreneurs, leaders and professionals who are socially sensitive and environmentally conscious.
- Integration of Outcome Based Education and cognitive teaching and learning strategies to

enhance learning effectiveness.

- Developing necessary infrastructure to cater to the changing needs of Business and Society.
- Optimum utilization of the infrastructure and resources to achieve excellence in all areas of relevance
- Adopting learning beyond curriculum through outbound activities and creative assignments.

• Imparting contemporary and emerging techno-managerial skills to keep pace with the changing global trends.

• Facilitating greater Industry-Institute Interaction for skill development and employability enhancement.

• Establishing systems and processes to facilitate research, innovation and entrepreneurship for holistic development of students.

• Implementation of Quality Assurance System in all Institutional processes.

VISION OF THE DEPARTMENT

Impart magnificent learning atmosphere establishing innovative practices among the students aiming to strengthen their software application knowledge and technical skills.

MISSION OF THE DEPARTMENT

M1: To deliver quality technical training on software application domain.

M2: To nurture team work in order to transform individual as responsible leader and entrepreneur for future trends.

M3: To inculcate research practices in teaching thus ensuring research blend among students.

M4: To ensure more doctorates in the department, aiming at professional strength.

M5: To inculcate the core information science engineering practices with hardware blend by providing advanced laboratories.

M6: To establish innovative labs, start-ups and patent culture.

Program Educational Objectives (PEOs)

PEO1: Graduates shall have successful careers as information science engineers and will be able to lead and manage teams across the globe.

PEO2: Graduates shall be professional in engineering practice and shall demonstrate good problem solving, communication skills and contribute to address societal issues.

PEO3: Graduates shall be pursuing distinctive education, entrepreneurship and research in an excellent environment which helps in the process of life-long learning.

Program Specific Outcomes (PSOs):

PSO1: Apply appropriate programming knowledge in software development, operations and maintenance of real-time applications.

PSO2: Meet the industry requirements in adapting to cutting edge technologies.

PSO3: Develop business and entrepreneurial ideas to support society requirements.

Program Outcomes (POs)

PO1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2. 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.

PO3. 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 considerations.

PO4. 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.

PO5. 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.

PO6. 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.

PO7. 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.

PO8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO10. 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.

PO11. 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.

PO12. 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.

Dayananda Sagar Academy of Technology & Management

(Autonomous Institute under VTU)

Accredited by NAAC with A+ grade, Accredited by 6 UG programs by NBA, New Delhi

Opp. Art of Living, Udayapura, Kanakapura Road, Bangalore – 560082

Department of Information Science & Engineering

SUBJECT: Computer Network Laboratory

SUBJECT CODE: BCS502

SEMESTER: V

Course outcomes

CO 1	**Explain** the fundamentals of computer networks.
CO 2	**Apply** the concepts of computer networks to demonstrate the working of various layers and protocols in communication network.
CO 3	**Analyze** the principles of protocol layering in modern communication systems.
CO 4	**Demonstrate** various Routing protocols and their services using tools such as Cisco packet tracer.
CO 5	**Develop** and Analyze the performance of wired and wireless Networks.

Experiment No:1-Simulate a three node point to point network with duplex links between them. Set queue size and vary the bandwidth and find number of packets dropped.

Program:

```
set ns [new Simulator] /* Letter S is capital */
set nf [open lab1.nam w] /* open a nam trace file in write mode */
$ns namtrace-all $nf /* nf – nam file */
set tf [open lab1.tr w] /* tf- trace file */
$ns trace-all $tf
proc finish { } { /* provide space b/w proc and finish and all are in small case */
global ns nf tf
$ns flush-trace /* clears trace file contents */
close $nf
close $tf
exec nam lab1.nam &
exit 0
}
set n0 [$ns node] /* creates 4 nodes */
 set n1 [$ns node]
set n2 [$ns node]
 set n3 [$ns node]
$ns duplex-link $n0 $n2 200Mb 10ms DropTail /*Letter M is capital Mb*/
$ns duplex-link $n1 $n2 100Mb 5ms DropTail /*D and T are capital*/
$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] /* Letters A,U,D and P are capital */
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR] /* A,T,C,B and R are capital*/
$cbr0 set packetSize_ 500 /*S is capital, space after underscore*/
$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] /* A and N are capital */
$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
```

<u>AWK file:</u> (Open a new editor using "vi command" and write awk file and save with ".awk" extension) #immediately after BEGIN should open braces '{'

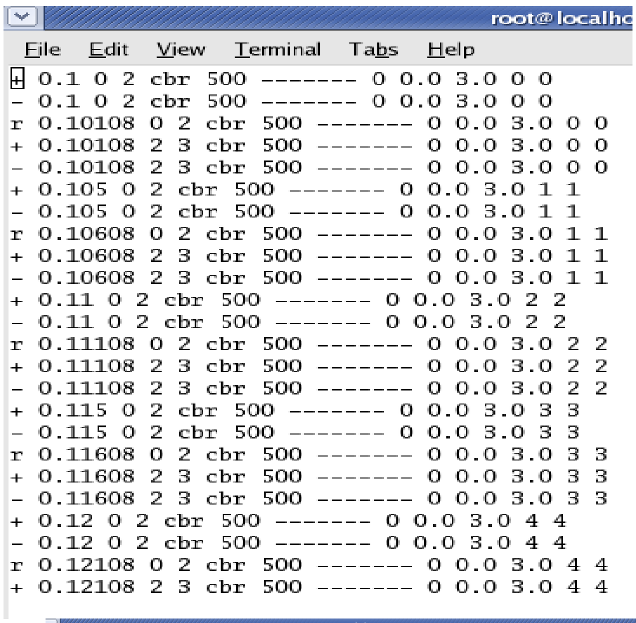
```
BEGIN{ c=0; }
{
  if ($1 = = "d")
 {       c++;
         printf("%s\t%s\n",$5,$11);
  }
 }
END{ printf("The number of packets dropped =%d\n",c); }
```

<u>Steps for execution:</u>

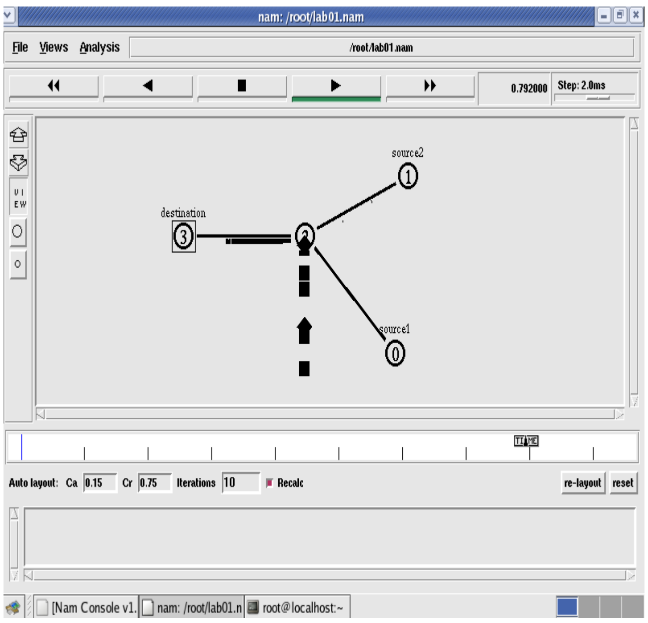
- *Open gedit editor and type program. Program name should have the extension " **.tcl** "*
  [root@localhost ~]# gedit lab1.tcl
- *Save the program*
- *Open editor and type **awk** program. Program name should have the extension "**.awk** "*
  [root@localhost ~]# gedit lab1.awk
- *Save the program*
- *Run the simulation program*
  [root@localhost~]# ns lab1.tcl
- *Here **"ns"** indicates network simulator. We get the topology shown in the snapshot.*
- *Now press the play button in the simulation window and the simulation will begins.*
- *After simulation is completed run **awk file** to see the output ,*
  [root@localhost~]# awk –f lab1.awk lab1.tr
- *To see the trace file contents open the file as ,*
  [root@localhost~]# gedit lab1.tr

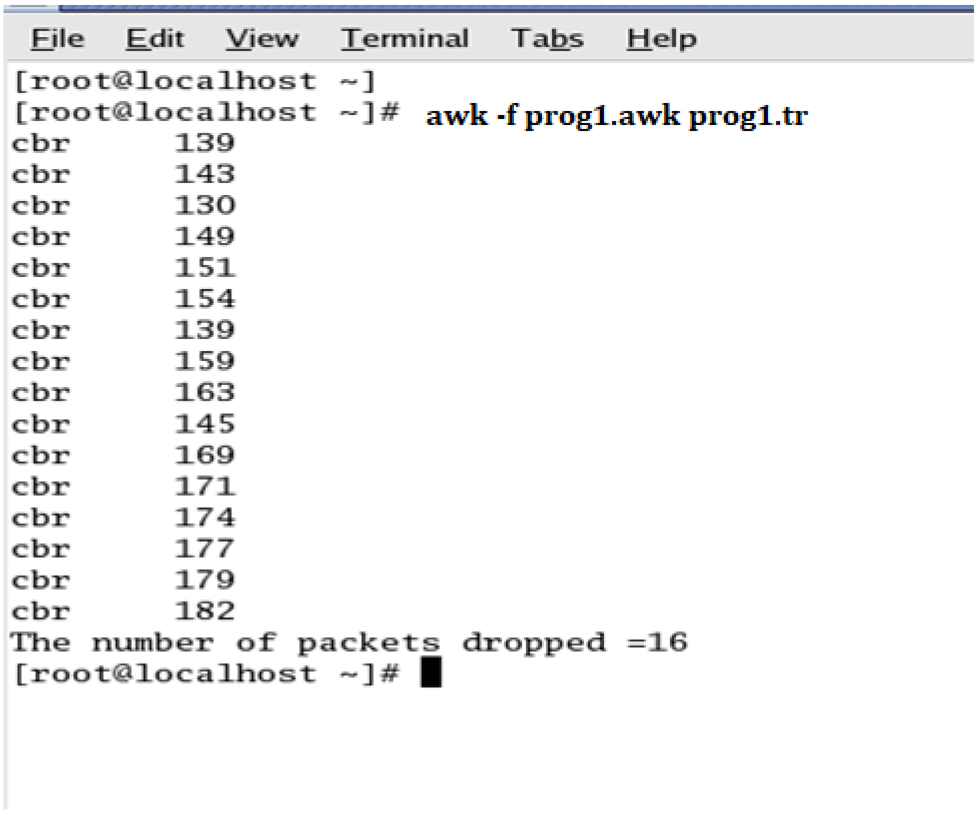
Trace file contains 12 columns:

Event type, Event time, From Node, To Node, Packet Type, Packet Size, Flags (indicated by --------), Flow ID, Source address, Destination address, Sequence ID, Packet ID



```
+ 0.1 0 2 cbr 500 ------- 0 0.0 3.0 0 0
- 0.1 0 2 cbr 500 ------- 0 0.0 3.0 0 0
r 0.10108 0 2 cbr 500 ------- 0 0.0 3.0 0 0
+ 0.10108 2 3 cbr 500 ------- 0 0.0 3.0 0 0
- 0.10108 2 3 cbr 500 ------- 0 0.0 3.0 0 0
+ 0.105 0 2 cbr 500 ------- 0 0.0 3.0 1 1
- 0.105 0 2 cbr 500 ------- 0 0.0 3.0 1 1
r 0.10608 0 2 cbr 500 ------- 0 0.0 3.0 1 1
+ 0.10608 2 3 cbr 500 ------- 0 0.0 3.0 1 1
- 0.10608 2 3 cbr 500 ------- 0 0.0 3.0 1 1
+ 0.11 0 2 cbr 500 ------- 0 0.0 3.0 2 2
- 0.11 0 2 cbr 500 ------- 0 0.0 3.0 2 2
r 0.11108 0 2 cbr 500 ------- 0 0.0 3.0 2 2
+ 0.11108 2 3 cbr 500 ------- 0 0.0 3.0 2 2
- 0.11108 2 3 cbr 500 ------- 0 0.0 3.0 2 2
+ 0.115 0 2 cbr 500 ------- 0 0.0 3.0 3 3
- 0.115 0 2 cbr 500 ------- 0 0.0 3.0 3 3
r 0.11608 0 2 cbr 500 ------- 0 0.0 3.0 3 3
+ 0.11608 2 3 cbr 500 ------- 0 0.0 3.0 3 3
- 0.11608 2 3 cbr 500 ------- 0 0.0 3.0 3 3
+ 0.12 0 2 cbr 500 ------- 0 0.0 3.0 4 4
- 0.12 0 2 cbr 500 ------- 0 0.0 3.0 4 4
r 0.12108 0 2 cbr 500 ------- 0 0.0 3.0 4 4
+ 0.12108 2 3 cbr 500 ------- 0 0.0 3.0 4 4
```





```
[root@localhost ~]
[root@localhost ~]# awk -f prog1.awk prog1.tr
cbr     139
cbr     143
cbr     130
cbr     149
cbr     151
cbr     154
cbr     139
cbr     159
cbr     163
cbr     145
cbr     169
cbr     171
cbr     174
cbr     177
cbr     179
cbr     182
The number of packets dropped =16
[root@localhost ~]# █
```

Experiment No: 2-Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.
program:

```
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]

$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] # letters A and P should be capital
$ns attach-agent $n0 $p1
```

```
$p1 set packetSize_ 50000
$p1 set interval_ 0.0001

set p2 [new Agent/Ping] # letters A and P should be capital
$ns attach-agent $n1 $p2

set p3 [new Agent/Ping] # letters A and P should be capital
$ns attach-agent $n2 $p3
$p3 set packetSize_ 30000
$p3 set interval_ 0.00001


set p4 [new Agent/Ping] # letters A and P should be capital
$ns attach-agent $n3 $p4

set p5 [new Agent/Ping] # letters A and P should be capital
$ns attach-agent $n5 $p5

$ns queue-limit $n0 $n4 5
$ns queue-limit $n2 $n4 3
$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"
}
# please provide space between $node_ and id. No space between $ and from. No space
between and $ and rtt */

$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 file*:** *(Open a new editor using "vi command" and write awk file and save with ".awk" extension)*

```
BEGIN{
drop=0;
}
{
 if($1= ="d" )
  {
   drop++;
   }
}
END{

printf("Total number of %s packets dropped due to congestion =%d\n",$5, drop);
}
```

Steps for execution

1) *Open vi editor and type program. Program name should have the extension " **.tcl** "*

   [root@localhost ~]# vi lab2.tcl

2) *Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.*
3) *Open vi editor and type **awk** program. Program name should have the extension "**.awk** "*

   [root@localhost ~]# vi lab2.awk

4) *Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.*
5) *Run the simulation program*

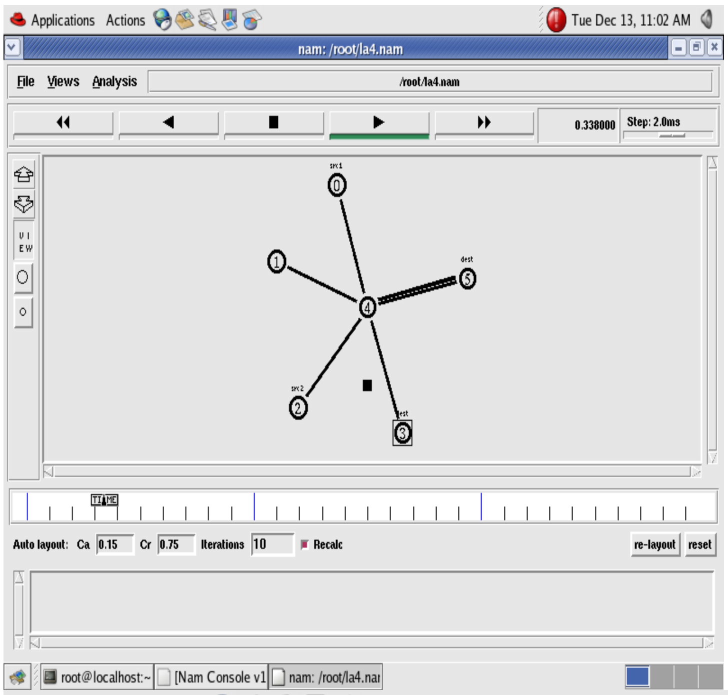
   [root@localhost~]# ns lab2.tcl

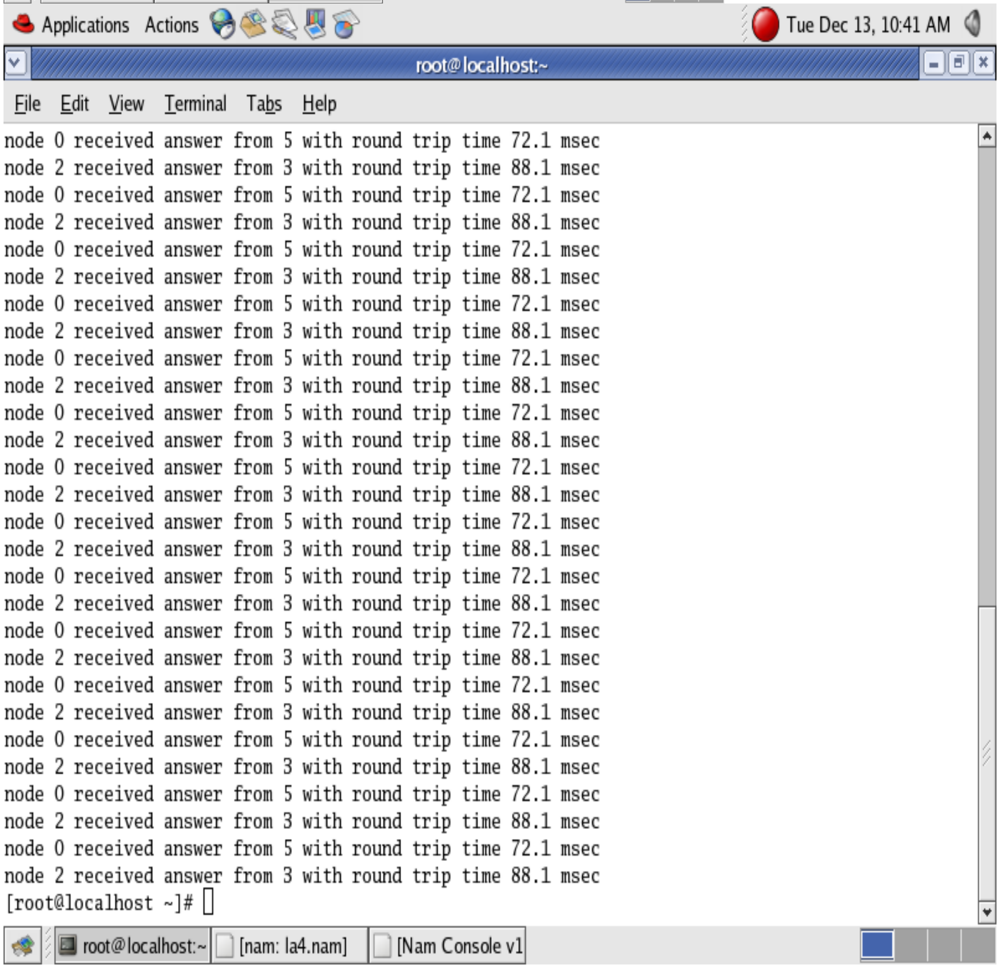
   i) *Here **"ns"** indicates network simulator. We get the topology shown in the snapshot.*
   ii) *Now press the play button in the simulation window and the simulation will begins.*
6) *After simulation is completed run **awk file** to see the output ,*

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

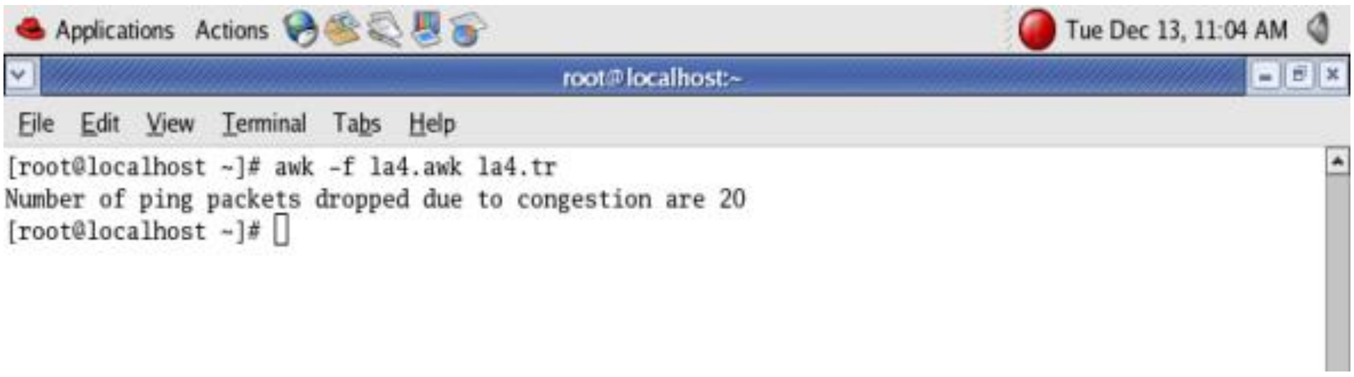
7) *To see the trace file contents open the file as ,*

[root@localhost~]# vi lab2.tr





```
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
node 0 received answer from 5 with round trip time 72.1 msec
node 2 received answer from 3 with round trip time 88.1 msec
[root@localhost ~]# 
```



```
[root@localhost ~]# awk -f la4.awk la4.tr
Number of ping packets dropped due to congestion are 20
[root@localhost ~]# 
```

Experiment No: 3 Simulate an Ethernet LAN using ‘n’ nodes and set multiple traffic nodes and plot congestion window for different source / destination.

<u>Program:</u>

```
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"
```

```
$ns make-lan "$n0 $n1 $n2 $n3 $n4" 100Mb 100ms LL Queue/DropTail Mac/802_3
# should come in single line
$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_  # must put underscore ( _ ) after cwnd and no space between them
$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
```

<u>AWK file:</u> *(Open a new editor using "vi command" and write awk file and save with ".awk" extension)*

cwnd:- means congestion window

```
BEGIN {
}
{
if($6= ="cwnd_")  # don't leave space after writing cwnd_
printf("%f\t%f\t\n",$1,$7); # you must put \n in printf
}
END{
}
```

<u>Steps for execution:</u>

1) *Open vi editor and type program. Program name should have the extension " **.tcl** "*

   [root@localhost ~]# vi lab3.tcl

2) *Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.*

3) *Open vi editor and type **awk** program. Program name should have the extension "**.awk** "*

   [root@localhost ~]# vi lab3.awk

4) *Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.*

5) *Run the simulation program*

   [root@localhost~]# ns lab3.tcl

6) *After simulation is completed run **awk file** to see the output ,*

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

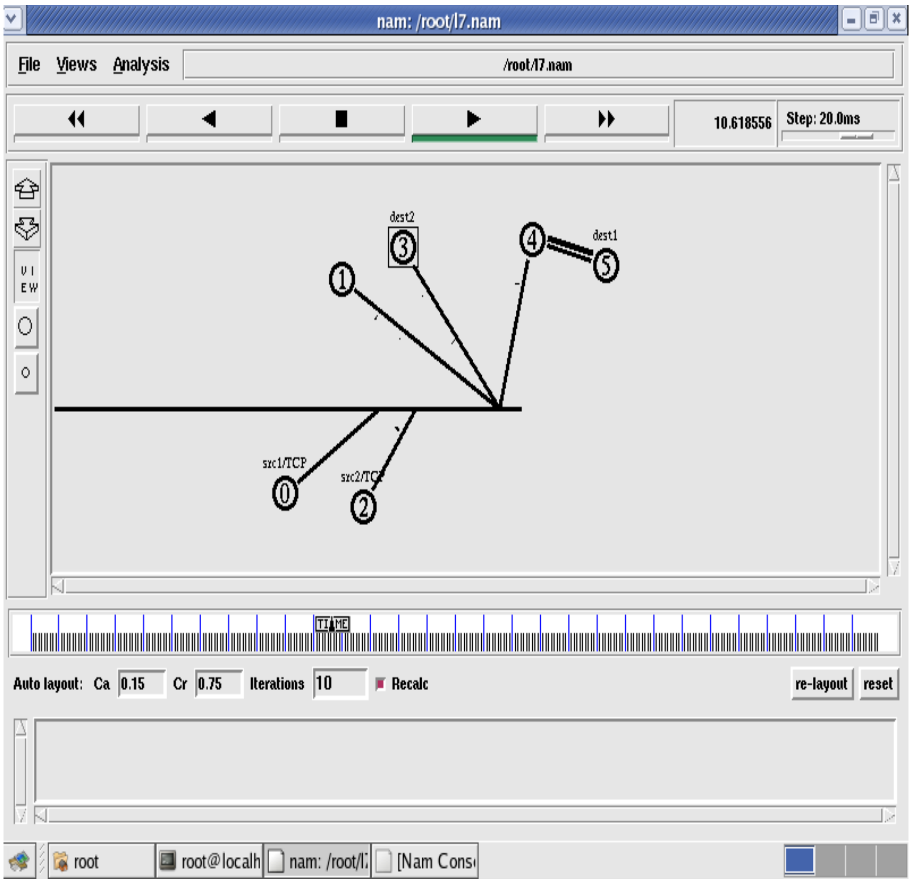
   ii. ***[root@localhost~]# awk –f lab3.awk file2.tr > a2***

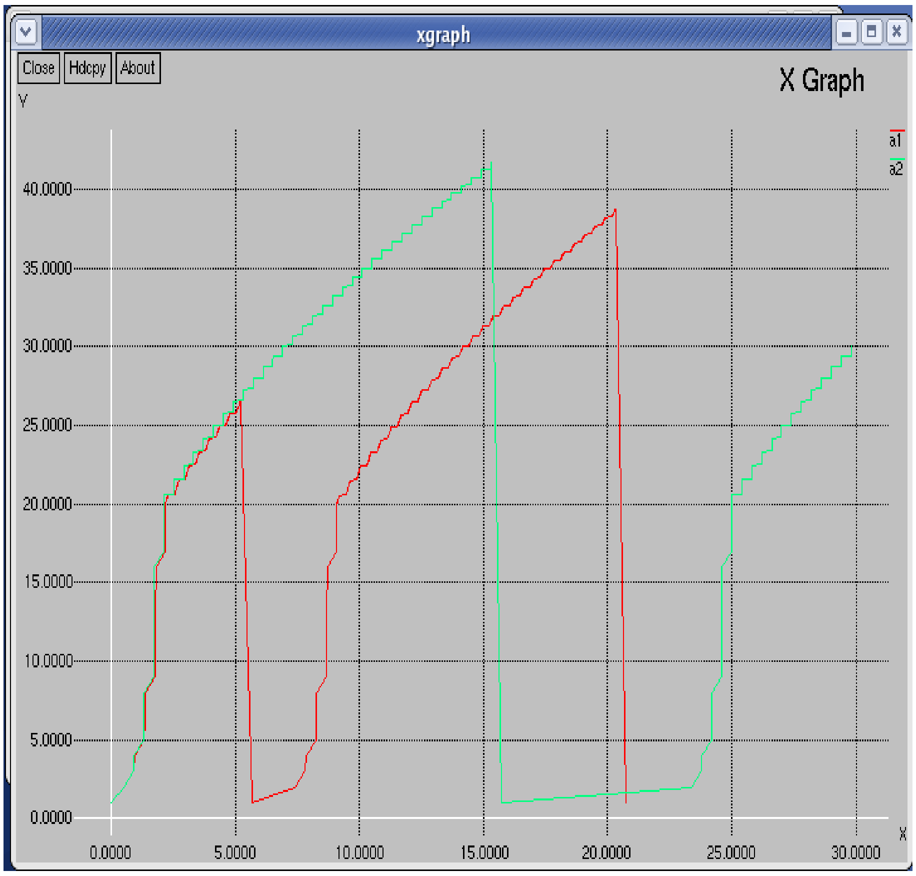
   iii. ***[root@localhost~]# xgraph a1 a2***

7) *Here we are using the congestion window trace files i.e.* ***file1.tr*** *and* ***file2.tr*** *and we are redirecting the contents of those files to new files say* ***a1*** *and* ***a2*** *using* ***output redirection operator (>)****.*

8) *To see the trace file contents open the file as ,*

[root@localhost~]# vi lab3.tr





Java Programs

Java is a general-purpose computer programming language that is simple, concurrent, class-based, object-oriented language. The compiled Java code can run on all platforms that support Java without the need for recompilation hence Java is called as "write once, run anywhere" (WORA).The Java compiled intermediate output called “byte-code” that can run on any Java virtual machine (JVM) regardless of computer architecture. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

In Linux operating system Java libraries are preinstalled. It’s very easy and convenient to compile and run Java programs in Linux environment. To compile and run Java Program is a two-step process:

1. Compile Java Program from Command Prompt

   [root@host ~]# javac Filename.java

The Java compiler (Javac) compiles java program and generates a byte-code with the same file name and .class extension.

2. Run Java program from Command Prompt

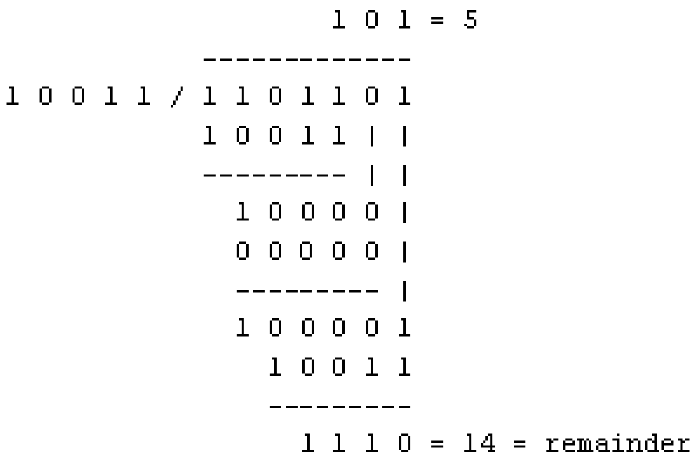
   [root@host ~]# java Filename

The java interpreter (Java) runs the byte-code and gives the respective output. It is important to note that in above command we have omitted the .class suffix of the byte- code (Filename.class).

Experiment No:4 Write a program for error detecting code using CRC-CCITT (16-bits).

Whenever digital data is stored or interfaced, data corruption might occur. Since the beginning of computer science, developers have been thinking of ways to deal with this type of problem. For serial data they came up with the solution to attach a parity bit to each sent byte. This simple detection mechanism works if an odd number of bits in a byte changes, but an even number of false bits in one byte will not be detected by the parity check. To overcome this problem developers have searched for mathematical sound mechanisms to detect multiple false bits. The **CRC** calculation or *cyclic redundancy check* was the result of this. Nowadays CRC calculations are used in all types of communications. All packets sent over a network connections are checked with CRC. Also each data block on your hard disk has a CRC value attached to it. Modern computer world cannot do without these CRC calculations. So let's see why they are so widely used. The answer is simple; they are powerful, detect many types of errors and are extremely fast to calculate especially when dedicated hardware chips are used.

The idea behind CRC calculation is to look at the data as one large binary number. This number is divided by a certain value and the remainder of the calculation is called the CRC. Dividing in the CRC calculation at first looks to cost a lot of computing power, but it can be performed very quickly if we use a method similar to the one learned at school. We will as an example calculate the remainder for the character 'm'—which is 1101101 in binary notation— by dividing it by 19 or 10011. Please note that 19 is an odd number. This is necessary as we will see further on. Please refer to your schoolbooks as the binary calculation method here is not very different from the decimal method you learned when you were young. It might only look a little bit strange. Also notations differ between countries, but the method is similar.



```
                 1 0 1 = 5
             -------------
1 0 0 1 1 / 1 1 0 1 1 0 1
            1 0 0 1 1 | |
            --------- | |
              1 0 0 0 0 |
              0 0 0 0 0 |
              --------- |
              1 0 0 0 0 1
                1 0 0 1 1
                ---------
                  1 1 1 0 = 14 = remainder
```

With decimal calculations you can quickly check that 109 divided by 19 gives a quotient of 5 with 14 as the remainder. But what we also see in the scheme is that every bit extra to check only costs one binary comparison and in 50% of the cases one binary subtraction. You can easily increase the number of bits of the test data string—for example to 56 bits if we use our example value "*Lammert*"—and the result can be calculated with 56 binary comparisons and an average of 28 binary subtractions. This can be implemented in hardware directly with only very few transistors involved. Also software algorithms can be very efficient.

All of the CRC formulas you will encounter are simply checksum algorithms based on modulo-2 binary division where we ignore carry bits and in effect the subtraction will be equal to an *exclusive or* operation. Though some differences exist in the specifics across different CRC formulas, he basic mathematical process is always the same:

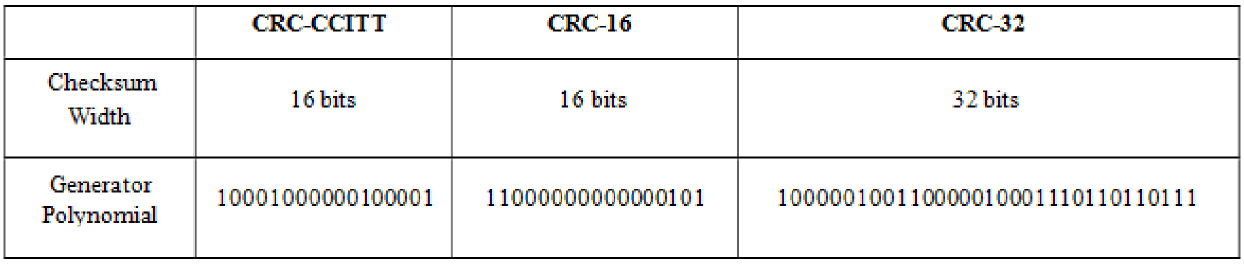
• The message bits are appended with *c* zero bits; this *augmented message* is the dividend

• A predetermined *c+1*-bit binary sequence, called the *generator polynomial*, is the divisor

• The checksum is the *c*-bit remainder that results from the division operation

Table 1 lists some of the most commonly used generator polynomials for 16- and 32-bit CRCs.

Remember that the width of the divisor is always one bit wider than the remainder. So, for example, you'd use a 17-bit generator polynomial whenever a 16-bit checksum is required.



	CRC-CCITT	CRC-16	CRC-32
Checksum Width	16 bits	16 bits	32 bits
Generator Polynomial	10001000000100001	11000000000000101	100000100110000010001110110110111

International Standard CRC Polynomials

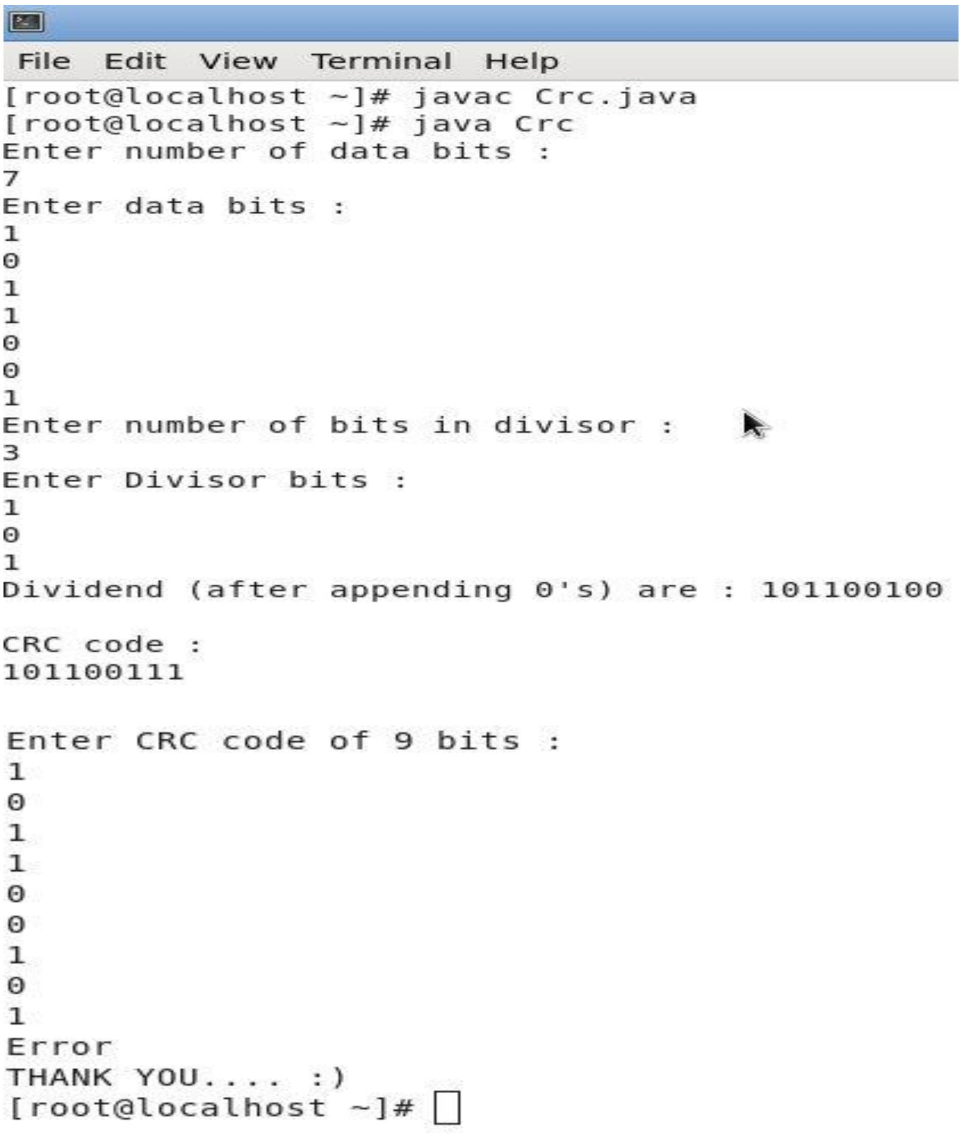
```
import java.io.*;

     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());


                /* System.out.print("Data bits are : ");
for(int i=0; i< data_bits; i++)

                       System.out.print(data[i]);
                       System.out.println();

                       System.out.print("divisor bits are : ");
               for(int i=0; i< divisor_bits; i++)
                       System.out.print(divisor[i]);
```

```
                        System.out.println();

                */ tot_length=data_bits+divisor_bits-1;

                div=new int[tot_length];
                rem=new int[tot_length];
                crc=new int[tot_length];
                /*------------------ CRC GENERATION-----------------------*/
                for(int i=0;i<data.length;i++)
                        div[i]=data[i];

                System.out.print("Dividend (after appending 0's) are : ");
                for(int i=0; i< div.length; i++)
                System.out.print(div[i]);
                System.out.println();
          for(int j=0; j<div.length; j++){
                rem[j] = div[j];
     }
     rem=divide(div, divisor, rem);
     for(int i=0;i<div.length;i++) //append dividend and ramainder
     {
                   crc[i]=(div[i]^rem[i]);
     }
                System.out.println();
                System.out.println("CRC code");

              for(int i=0;i<crc.length;i++)
              System.out.print(crc[i]);
     /*-------------------ERROR DETECTION---------------------*/
        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());

     /* System.out.print("crc bits are : ");
     for(int i=0; i< crc.length; i++)
        System.out.print(crc[i]);
        System.out.println();
       */
     for(int j=0; j<crc.length; j++){
rem[j] = crc[j];
}
```

```
rem=divide(crc, divisor, rem);

for(int i=0; i< rem.length; i++)
{
if(rem[i]!=0)
{
System.out.println("Error");
break;
}
if(i==rem.length-1)
System.out.println("No Error");
}

System.out.println("THANK YOU.... :)");
}

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

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

Output:

```
[root@localhost ~]# vi Crc.java
```



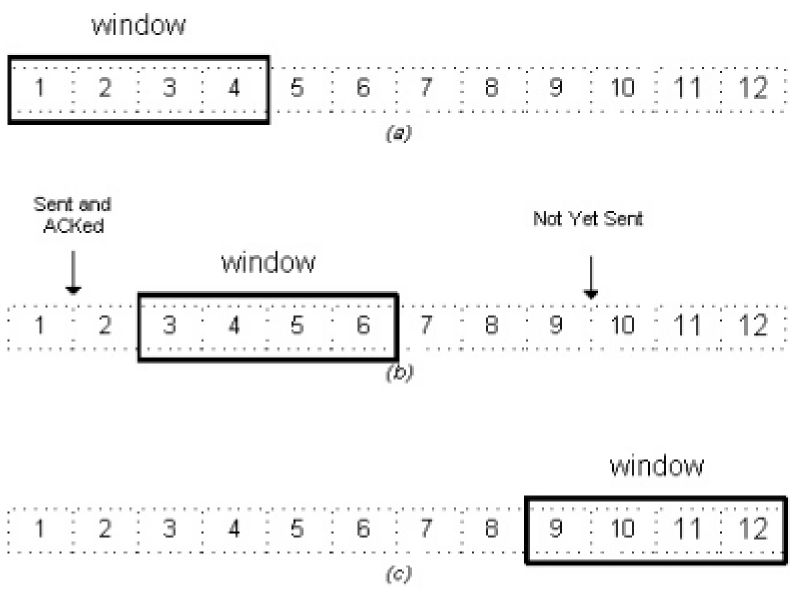
```
[root@localhost ~]# javac Crc.java
[root@localhost ~]# java Crc
Enter number of data bits :
7
Enter data bits :
1
0
1
1
0
0
1
Enter number of bits in divisor :
3
Enter Divisor bits :
1
0
1
Dividend (after appending 0's) are : 101100100

CRC code :
101100111

Enter CRC code of 9 bits :
1
0
1
1
0
0
1
0
1
Error
THANK YOU.... :)
[root@localhost ~]# 
```

Experiment No:5 Develop a program to implement a sliding window protocol in the data link layer.

In computer networks sliding window protocol is a method to transmit data on a network. Sliding window protocol is applied on the Data Link Layer of OSI model. At data link layer data is in the form of frames. In Networking, Window simply means a buffer which has data frames that needs to be transmitted.

Both sender and receiver agrees on some window size. If window size=w then after sending w frames sender waits for the acknowledgement (ack) of the first frame.

As soon as sender receives the acknowledgement of a frame it is replaced by the next frames to be transmitted by the sender. If receiver sends a collective or cumulative acknowledgement to sender then it understands that more than one frames are properly received, for eg:- if ack of frame 3 is received it understands that frame 1 and frame 2 are received properly.



In this simple example, there is a 4-byte sliding window. Moving from left to right, the window "slides" as bytes in the stream are sent and acknowledged.

Most sliding window protocols also employ ARQ (Automatic Repeat reQuest) mechanism. In ARQ, the sender waits for a positive acknowledgement before proceeding to the next frame. If no acknowledgement is received within a certain time interval it retransmits the frame. ARQ is of two types : Go-Back-N and Selective Repeat Protocols.

```
#include<stdio.h>
int main()
{
    int w,i,f,frames[50];
    printf("Enter window size: ");
    scanf("%d",&w);
 printf("\nEnter number of frames to transmit: ");
    scanf("%d",&f);
printf("\nEnter %d frames: ",f);
    for(i=1;i<=f;i++)
        scanf("%d",&frames[i]);
    printf("\nWith sliding window protocol the frames will be sent in the following manner
(assuming no corruption of frames)\n\n");
```

```
    printf("After sending %d frames at each stage sender waits for acknowledgement sent by
the receiver\n\n",w);
    for(i=1;i<=f;i++)
    {
        if(i%w==0)
        {
            printf("%d\n",frames[i]);
            printf("Acknowledgement of above frames sent is received by sender\n\n");
        }
        else
            printf("%d ",frames[i]);
  }
if(f%w!=0)
        printf("\nAcknowledgement of above frames sent is received by sender\n");
    return 0;
}
```

Experiment No:6 Write a program to find the shortest path between vertices using Bellman-ford algorithm.

Distance Vector Algorithm is a decentralized routing algorithm that requires that each router simply inform its neighbors of its routing table. For each network path, the receiving routers pick the neighbor advertising the lowest cost, then add this entry into its routing table for re-advertisement. To find the shortest path, Distance Vector Algorithm is based on one of two basic algorithms: the Bellman-Ford and the Dijkstra algorithms.

Routers that use this algorithm have to maintain the distance tables (which is a one-dimension array -- "a vector"), which tell the distances and shortest path to sending packets to each node in the network. The information in the distance table is always up date by exchanging information with the neighboring nodes. The number of data in the table equals to that of all nodes in networks (excluded itself). The columns of table represent the directly attached neighbors whereas the rows represent all destinations in the network. Each data contains the path for sending packets to each destination in the network and distance/or time to transmit on that path (we call this as "cost"). The measurements in this algorithm are the number of hops, latency, the number of outgoing packets, etc.

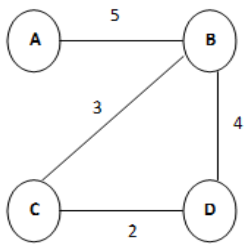
The Bellman–Ford algorithm is an algorithm that computes shortest paths from a single source vertex to all of the other vertices in a weighted digraph. It is slower than Dijkstra's algorithm for the same problem, but more versatile, as it is capable of handling graphs in which some of the edge weights are negative numbers. Negative edge weights are found in various applications of graphs, hence the usefulness of this algorithm. If a graph contains a "negative cycle" (i.e. a cycle whose edges sum to a negative value) that is reachable from the source, then there is no cheapest path: any path that has a point on the negative cycle can be made cheaper by one more walk around the negative cycle. In such a case, the Bellman–Ford algorithm can detect negative cycles and report their existence

Source code:

```
import java.util.Scanner;
public class BellmanFord
{
private int D[];
private int n;
public static final int MAX_VALUE = 999;
public BellmanFord(int n)
{
this.n=n;
D = new int[n+1];
}
public void shortest(int s,int A[][])
{
for (int i=1;i<=n;i++)
D[i]=MAX_VALUE;
 D[s] = 0;
for(int k=1;k<=n-1;k++)
for(int i=1;i<=n;i++)
for(int j=1;j<=n;j++)
 if(A[i][j]!=MAX_VALUE)
{
 if(D[j]>D[i]+A[i][j])
 D[j]=D[i]+A[i][j];
}
for(int i=1;i<=n;i++)
System.out.println("Distance of source " + s + " to "+ i + " is " + D[i]);
}
public static void main(String[ ] args)
{
int n=0,s;
Scanner sc = new Scanner(System.in);
 System.out.println("Enter the number of vertices");
 n = sc.nextInt();
int A[][] = new int[n+1][n+1];
System.out.println("Enter the Weighted matrix");
 for(int i=1;i<=n;i++)
for(int j=1;j<=n;j++)
{
A[i][j]=sc.nextInt();
 if(i==j)
{
 A[i][j]=0;
continue;
}
 if(A[i][j]==0)
A[i][j]=MAX_VALUE;
}
System.out.println("Enter the source vertex");
 s=sc.nextInt();
BellmanFord b = new BellmanFord(n);
 b.shortest(s,A);
sc.close();
}
```

```
}
```

Input graph:



Experiment No:07. 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.

Socket is an interface which enables the client and the server to communicate and pass on information from one another. Sockets provide the communication mechanism between two computers using TCP. A client program creates a socket on its end of the communication and attempts to connect that socket to a server. When the connection is made, the server creates a socket object on its end of the communication. The client and the server can now communicate by writing to and reading from the socket.

Source Code:

TCP Client

```
import java.net.*;
import java.io.*;
public class ReceiveFile {
 public static void main(String[] args) throws Exception{
 //create client Socket
 Socket s=new Socket("localhost",123);
 //accept file name from keyboard
 BufferedReader k=new BufferedReader(new InputStreamReader(System.in));
 System.out.print("Enter file name: ");
String filename=k.readLine();
//send file name to server using DataOutputStream
DataOutputStream d=new DataOutputStream(s.getOutputStream());
d.writeBytes(filename +"\n");
//to read data sent from server
BufferedReader i=new BufferedReader(new InputStreamReader(s.getInputStream()));
String st;
//read first line from server into st
st=i.readLine();
//if file is found on server side, then send "Yes" else "No"
if(st.equals("Yes"))
{
 //read and display the file contents coming from Server
 while((st=i.readLine())!=null)
 System.out.println(st);
//close all connections
i.close();
d.close();k.close();
s.close();
}
 else
System.out.println("File not found");
}
}
```

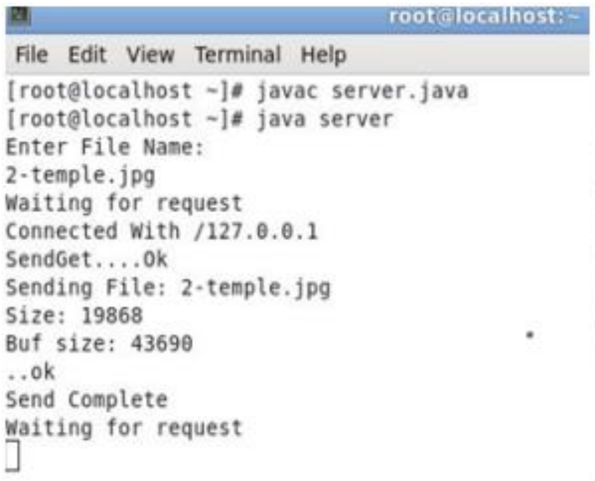
TCP Server

```
//create a server that send file contents to client
import java.net.*;
import java.io.*;
public class SendFile {
 public static void main(String[] args) throws Exception {
 //create a server socket
 ServerSocket se=new ServerSocket(123);
 //let the server wait until the connection is accepted by client
 Socket q=se.accept();
 System.out.println("Connection established successfully");
 //to receive file name from client
 BufferedReader v=new BufferedReader(new InputStreamReader(q.getInputStream()));
 //to transfer file contents to client
 DataOutputStream dr=new DataOutputStream(q.getOutputStream());
 //read file name from client
 String g=v.readLine();
 FileReader f=null;
 BufferedReader ff=null;
 boolean b;
 //create file class object with file name
 File r=new File(g);
 //test if file exists or not
 if(r.exists())
b=true;
 else
 b=false;
 //if file exists, send 'yes' to client else send 'no'
 if(b==true) dr.writeBytes("Yes"+ "\n");
 else dr.writeBytes("No"+"\n");
 if(b==true)
 {
 //attach file to fileReader to read data
 f=new FileReader(g);
 //attach FileReader to BufferedReader
 ff=new BufferedReader(f);
 String qq;
 //read from BufferedReader and write to DataOutputStream
 while((qq=ff.readLine())!=null)
{
 dr.writeBytes(qq+"\n");
 }
 dr.close();
 ff.close();
 v.close();
 se.close();
 q.close();
 f.close();
 }
 }
}
```

Note: Create two different files Client.java and Server.java. Follow the steps given:
1. Open a terminal run the server program and provide the filename to send
2. Open one more terminal run the client program and provide the IP address of the server. We can give localhost address “127.0.0.1” as it is running on same machine or give the IP address of the machine.
3. Send any start bit to start sending file.
4. Refer https://www.tutorialspoint.com/java/java_networking.htm for all the parameters, methods description in socket communication.

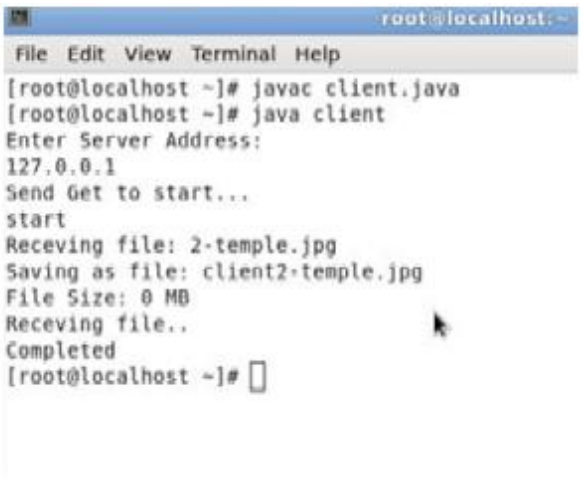
Output:

At server side:



```
[root@localhost ~]# javac server.java
[root@localhost ~]# java server
Enter File Name:
2-temple.jpg
Waiting for request
Connected With /127.0.0.1
SendGet....Ok
Sending File: 2-temple.jpg
Size: 19868
Buf size: 43690
..ok
Send Complete
Waiting for request
```

At client side:



```
[root@localhost ~]# javac client.java
[root@localhost ~]# java client
Enter Server Address:
127.0.0.1
Send Get to start...
start
Receving file: 2-temple.jpg
Saving as file: client2-temple.jpg
File Size: 0 MB
Receving file..
Completed
[root@localhost ~]#
```

Experiment No:08. Develop a program on a datagram socket for client/server to display the messages on client side, typed at the server side.

A datagram socket is the one for sending or receiving point for a packet delivery service. Each packet sent or received on a datagram socket is individually addressed and routed. Multiple packets sent from one machine to another may be routed differently, and may arrive in any order.

Source Code:

UDP Client

```
import java.io.*;
import java.net.*;
public class UDPC
{
public static void main(String[] args)
{
DatagramSocket skt;
String str;
try
{
skt=new DatagramSocket(3000);
byte[] b = new byte[1000];
while(true){
 DatagramPacket reply = new DatagramPacket(b,b.length);
skt.receive(reply);
str = new String(reply.getData(),0,reply.getLength());
System.out.println("client received:"+str);
}
}
catch(Exception ex)
{
 System.out.println(ex);
}
}
}
```

UDP Server

```
import java.io.*;
import java.net.*;
public class UDPS
{
public static void main(String[] args)
{
DatagramSocket skt=null;
 int ch =0;
try
{
skt=new DatagramSocket();
InetAddress ip = new InetAddress.getByName("localhost");
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
do
{
System.out.println("Enter the message:");
String msg = br.readLine();
DatagramPacket dp = new DatagramPacket(msg.getBytes(),msg.length(),ip,3000);
skt.send(dp);
System.out.println("Do you wish to continue: 1 for no 0 for yes");
ch = Integer.parseInt(br.readLine());
}while(ch==0);
} catch(Exception ex)
{
 System.out.println(ex):
}
}
}
```

Note: Create two different files UDPC.java and UDPS.java. Follow the following steps:
1. Open a terminal run the server program.
2. Open one more terminal run the client program, the sent message will be received.

Experiment No:09. Develop a program for a 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 (φ) 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:

RSA Key Generation

```
import java.util.*;
import java.math.BigInteger;
import java.lang.*;
public class RSA {
public static void main(String[] args)
{
Scanner scan = new Scanner(System.in);
Random rand1=new Random(System.currentTimeMillis());
Random rand2=new Random(System.currentTimeMillis()*10);
System.out.println("Enter the Public Key:");
int pubkey=scan.nextInt();
BigInteger bigB_p=BigInteger.probablePrime(32, rand1);
BigInteger bigB_q=BigInteger.probablePrime(32, rand2);
BigInteger bigB_n=bigB_p.multiply(bigB_q);
BigInteger bigB_p_1=bigB_p.subtract(new BigInteger("1"));
BigInteger bigB_q_1=bigB_q.subtract(new BigInteger("1"));
BigInteger bigB_p_1_q_1=bigB_p_1.multiply(bigB_q_1);
while(true)
{
BigInteger BigB_GCD=bigB_p_1_q_1.gcd(new BigInteger(""+pubkey));
if(BigB_GCD.equals(BigInteger.ONE))
{
break;
}
pubkey++;
}
BigInteger bigB_pubkey=new BigInteger(""+pubkey);
BigInteger bigB_prvkey=bigB_pubkey.modInverse(bigB_p_1_q_1);
System.out.println("public key : "+bigB_pubkey);
System.out.println("private key : "+bigB_prvkey);
System.out.println("n : "+bigB_n);
System.out.println("Enter The plain text:");
BigInteger bigB_val=scan.nextBigInteger();
BigInteger bigB_cipherVal=bigB_val.modPow(bigB_pubkey,bigB_n);
System.out.println("Cipher text: " + bigB_cipherVal);
BigInteger bigB_plainVal=bigB_cipherVal.modPow(bigB_prvkey,bigB_n);
int plainVal=bigB_plainVal.intValue();
System.out.println("Plain text:" + plainVal);
}
}
```

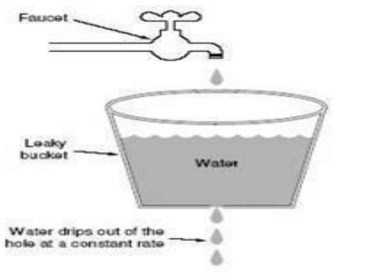
Output:

```
Enter the Public Key:
8
public key : 17
private key : 2271120043244956433
n : 9652260190212882127
Enter the plain text:
34
Cipher text: 7808382075030773626
Plain text: 34
```

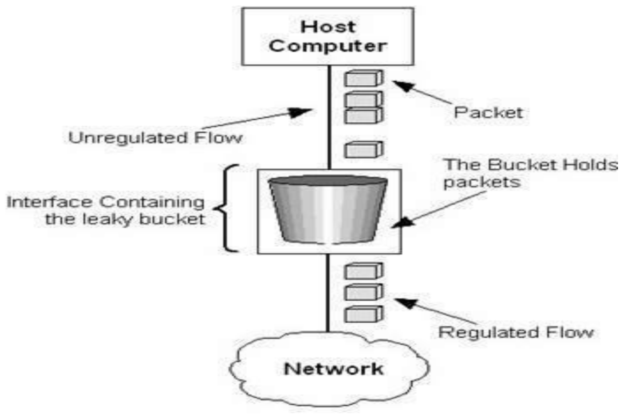
Experiment No:10. Write a program for congestion control using leaky bucket algorithm.

The main concept of the leaky bucket algorithm is that the output data flow remains constant despite the variant input traffic, such as the water flow in a bucket with a small hole at the bottom. In case the bucket contains water (or packets) then the output flow follows a constant rate, while if the bucket is full any additional load will be lost because of spillover. In a similar way if the bucket is empty the output will be zero.
From network perspective, leaky bucket consists of a finite queue (bucket) where all the

incoming packets are stored in case there is space in the queue, otherwise the packets are discarded. In order to regulate the output flow, leaky bucket transmits one packet from the queue in a fixed time (e.g. at every clock tick). In the following figure we can notice the main rationale of leaky bucket algorithm, for both the two approaches (e.g. leaky bucket with water (a) and with packets (b)).



(a) A leaky bucket with water



(b) A leaky bucket with packets

While leaky bucket eliminates completely bursty traffic by regulating the incoming data flow its main drawback is that it drops packets if the bucket is full. Also, it doesn't take into account the idle process of the sender which means that if the host doesn't transmit data for some time the bucket becomes empty without permitting the transmission of any packet.

Source Code:

```
import java.util.Scanner;
public class LeakyBucket {
public static int min(int a, int b)
{
        if(a<b)
                return a;
        else
                return b;
}
public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
    int cap, oprt, nsec, cont, i=0,dr=0, x, res;
```

```
        int [] inp = new int [25];
        int ch;
            System.out.println("\n\nLEAKY BUCKET ALGORITHM\n");
            System.out.println("\nEnter bucket size :   ");
            cap = sc.nextInt();
            System.out.println("\nEnter output rate (no..of pkts/sec)  :  ");
            oprt = sc.nextInt();

     do{
            System.out.println("\nEnter the no..of packets entering at
  second:"+(i+1));
            inp[i++] = sc.nextInt();
            System.out.println("Enter 1 to insert packets or 0 to quit  :   ");
            ch = sc.nextInt();
        }  while(ch==1);
          nsec=i;
     System.out.println("\nSecond : Packets recvd to  bucket : Packets sent : In
  bucket: Dropped\n");

            for(cont=i=0;  (cont>0) || (i<nsec) ; i++)
            {
                    System.out.print("    :"+(i+1));
                    System.out.print("     \t: "+inp[i]);
                    res = min(cont+inp[i],oprt);
                    System.out.print("     \t: "+res);
                    if((x=cont+inp[i]-oprt)>0)
                    {
                            if(x>cap)
                            {
                                    cont=cap;
                                    dr=x-cap;
                            }
                            else
                    {       cont=x;
                            dr=0;
                    }
                }
                else
                cont=0;
                System.out.print("   \t\t:"+cont);
            System.out.print("   \t\t:"+dr+"\n");
        }
    }
}
```

Output:

```
LEAKY BUCKET ALGORITHM

Enter bucket size :
3
Enter output rate (no..of pkts/sec)  :
2
Enter the no..of packets entering at second:1
6
Enter 1 to insert packets or 0 to quit  :
0

Second :   Packets sent :   Packets recvd :   In bucket:    Dropped

   (1)          : 6             : 2               :3            1
   (2)          : 0             : 2               :1            0
   (3)          : 0             : 1               :0            0
```