

Experiment No: 4**SIMPLE ESS WITH WIRELESS LAN**

Aim: Simulate simple ESS and with transmitting nodes in wireless LAN by simulation and determine the performance with respect to transmission of packets.

Program:

```

set ns [new Simulator]
set tf [open lab4.tr w]
$ns trace-all $tf

set topo [new Topography]
$topo load_flatgrid 1000 1000

set nf [open lab4.nam w]
$ns namtrace-all-wireless $nf 1000 1000
$ns node-config -adhocRouting DSDV \
    -llType LL \
    -macType Mac/802_11 \
    -ifqType Queue/DropTail \
    -ifqLen 50 \
    -phyType Phy/WirelessPhy \
    -channelType Channel/WirelessChannel \
    -propType Propagation/TwoRayGround \
    -antType Antenna/OmniAntenna \
    -topoInstance $topo \
    -agentTrace ON \
    -routerTrace ON
    create-god 3

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]

$n0 label "tcp0"
$n1 label "sink1/tcp1"
$n2 label "sink2"

$n0 set X_ 50
$n0 set Y_ 50
$n0 set Z_ 0

$n1 set X_ 100
$n1 set Y_ 100
$n1 set Z_ 0

$n2 set X_ 600
$n2 set Y_ 600
$n2 set Z_ 0

$ns at 0.1 "$n0 setdest 50 50 15"
$ns at 0.1 "$n1 setdest 100 100 25"
$ns at 0.1 "$n2 setdest 600 600 25"

set tcp0 [new Agent/TCP]

```

```
$ns attach-agent $n0 $tcp0
```

```
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
set sink1 [new Agent/TCPSink]
$ns attach-agent $n1 $sink1
$ns connect $tcp0 $sink1
set tcp1 [new Agent/TCP]
$ns attach-agent $n1 $tcp1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set sink2 [new Agent/TCPSink]
$ns attach-agent $n2 $sink2
$ns connect $tcp1 $sink2
$ns at 5 "$ftp0 start"
$ns at 5 "$ftp1 start"
$ns at 100 "$n1 setdest 550 550 15"
$ns at 190 "$n1 setdest 70 70 15"
proc finish { } {
    global ns nf tf
    $ns flush-trace
    exec nam lab4.nam &
    close $tf
    exit 0
}
$ns at 250 "finish"
$ns run
```

AWK File:

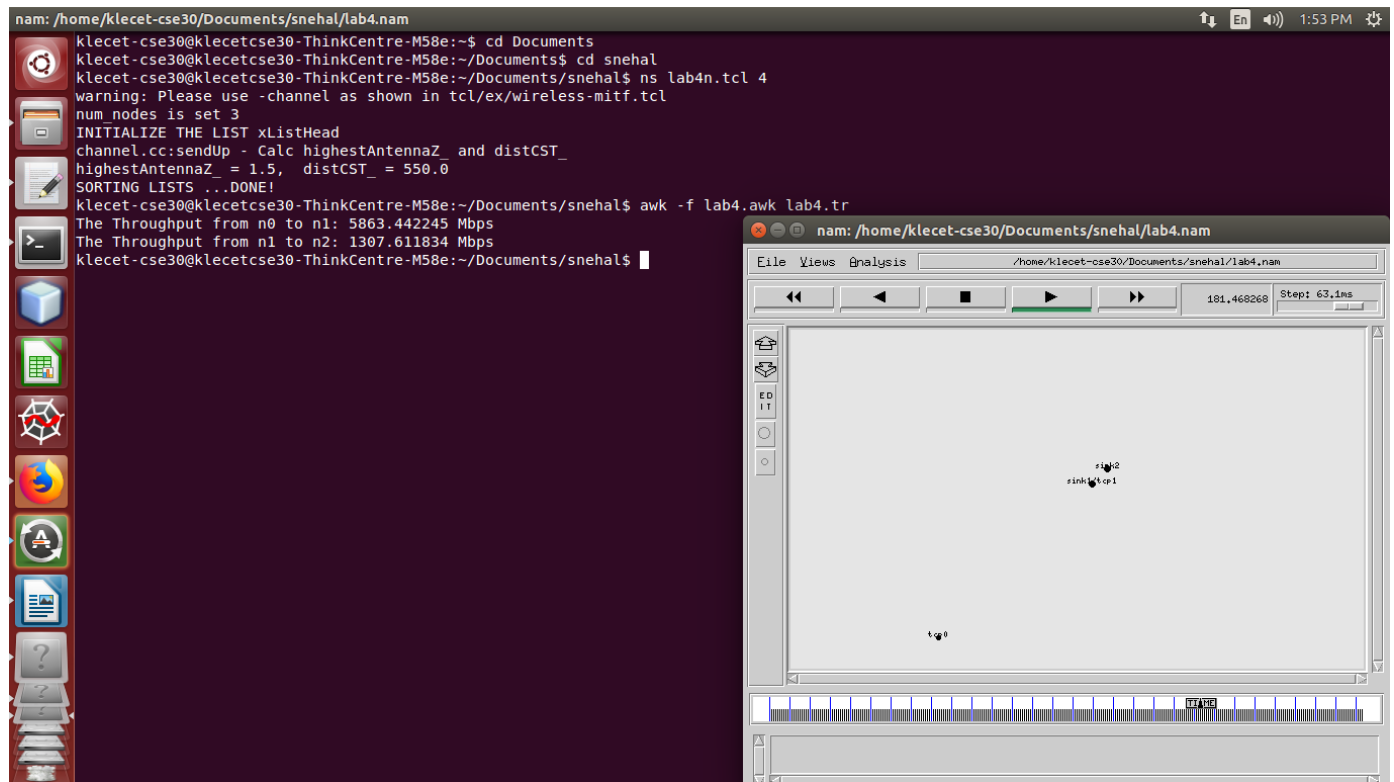
```
BEGIN{
    count1=0
    count2=0
    pack1=0
    pack2=0
    time1=0
    time2=0
}
{
    if($1=="r"&& $3=="_1_" && $4=="AGT")
    {
        count1++ ;
        pack1=pack1+$8 ;
        time1=$2 ;
    }
    if($1=="r" && $3=="_2_" && $4=="AGT")
    {
        count2++ ;
        pack2=pack2+$8 ;
        time2=$2 ;
    }
}
END{
```

```
printf("The Throughput from n0 to n1: %f Mbps \n",
((count1*pack1*8)/(time1*1000000)));
printf("The Throughput from n1 to n2: %f Mbps",
((count2*pack2*8)/(time2*1000000)));
}
```

Procedure:

- 1) Open vi editor and type program. Program name should have the extension “**.tcl**”
[root@localhost ~]# vi lab4.tcl
- 2) Save the program by pressing “**ESC key**” first, followed by “**Shift and :**” keys simultaneously and type “**wq**” and press **Enterkey**.
- 3) Open vi editor and type **awk** program. Program name should have the extension “**.awk**”
[root@localhost ~]# vi lab4.awk
- 4) Save the program by pressing “**ESC key**” first, followed by “**Shift and :**” keys simultaneously and type “**wq**” and press **Enterkey**.
- 5) Run the simulation program
[root@localhost~]# ns lab4.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 begin.
- 6) After simulation is completed run **awk** file to see the output,
[root@localhost~]# awk -f lab4.awk lab4.tr
- 7) To see the trace file contents open the file as,
[root@localhost~]# vi lab4.tr

Output:



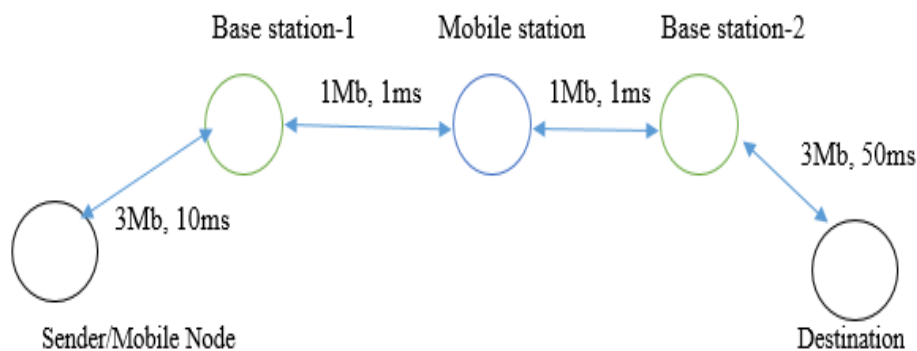
Experiment No: 5**GSM on NS2**

Aim: Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment

Second Generation (2G) technology is based on the technology known as global system for mobile communication (GSM). This technology enabled various networks to provide services like text messages, picture messages and MMS. The technologies used in 2G are either TDMA (Time Division Multiple Access) which divides signal into different time slots or CDMA (Code Division Multiple Access) which allocates a special code to each user so as to communicate over a multiplex physical channel.

GSM uses a variation of time division multiple access (TDMA). 2G networks developed as a replacement for first generation (1G) analog cellular networks, and the GSM standard originally described as a digital, circuit-switched network optimized for full duplex voice telephony. This expanded over time to include data communications, first by circuit-switched transport, then by packet data transport via GPRS (General Packet Radio Services).

GSM can be implemented on all the versions of NS2 (Since year 2004: ns-2.27, and later versions of NS2)

Design:**Program :**

```

set stop 100 ;# Stop time.
# Topology
set type gsm ;#type of link:
# AQM parameters
set minth 30 ;
set maxth 0 ;
set adaptive 1 ;# 1 for Adaptive RED, 0 for plain RED
# Traffic generation.
set flows 0 ;# number of long-lived TCP flows
set window 30 ;# window for long-lived traffic
set web 2 ;# number of web sessions
# Plotting statistics.
set opt(wrap) 100 ;# wrap plots?
set opt(srcTrace) is ;# where to plot traffic
set opt(dstTrace) bs2 ;# where to plot traffic

#default downlink bandwidth in bps
set bwDL(gsm) 9600
#default uplink bandwidth in bps
set bwUL(gsm) 9600
  
```

```

#default downlink propagation delay in seconds
set propDL(gsm) .500
#default uplink propagation delay in seconds
set propUL(gsm) .500

set ns [new Simulator]
set tf [open out.tr w]
$ns trace-all $tf

set namf [open out.nam w]
$ns namtrace-all $namf

set nodes(is) [$ns node]
set nodes(ms) [$ns node]
$nodes(ms) label "Mobile Station"
set nodes(bs1) [$ns node]
set nodes(bs2) [$ns node]
set nodes(lp) [$ns node]

proc cell_topo {} {
    global ns nodes
    $ns duplex-link $nodes(lp) $nodes(bs1) 3Mbps 10nodes(ms) DropTail
    $ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED
    $ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED
    $ns duplex-link $nodes(bs2) $nodes(is) 3Mbps 10nodes(ms) DropTail
    puts " GSM Cell Topology"
}

proc set_link_para {t} {
    global ns nodes bwUL bwDL propUL propDL buf
    $ns bandwidth $nodes(bs1) $nodes(ms) $bwDL($t) duplex
    $ns bandwidth $nodes(bs2) $nodes(ms) $bwDL($t) duplex
    $ns delay $nodes(bs1) $nodes(ms) $propDL($t) duplex
    $ns delay $nodes(bs2) $nodes(ms) $propDL($t) duplex
    $ns queue-limit $nodes(bs1) $nodes(ms) 10
    $ns queue-limit $nodes(bs2) $nodes(ms) 10
}

# RED and TCP parameters
Queue/RED set adaptive_ $adaptive
Queue/RED set thresh_ $minth
Queue/RED set maxthresh_ $maxth
Agent/TCP set window_ $window

#Create topology
switch $type {
    gsm -
    gprs -
    umts { cell_topo }
}

set_link_para $type
$ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]
$ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]

# Set up forward TCP connection

```

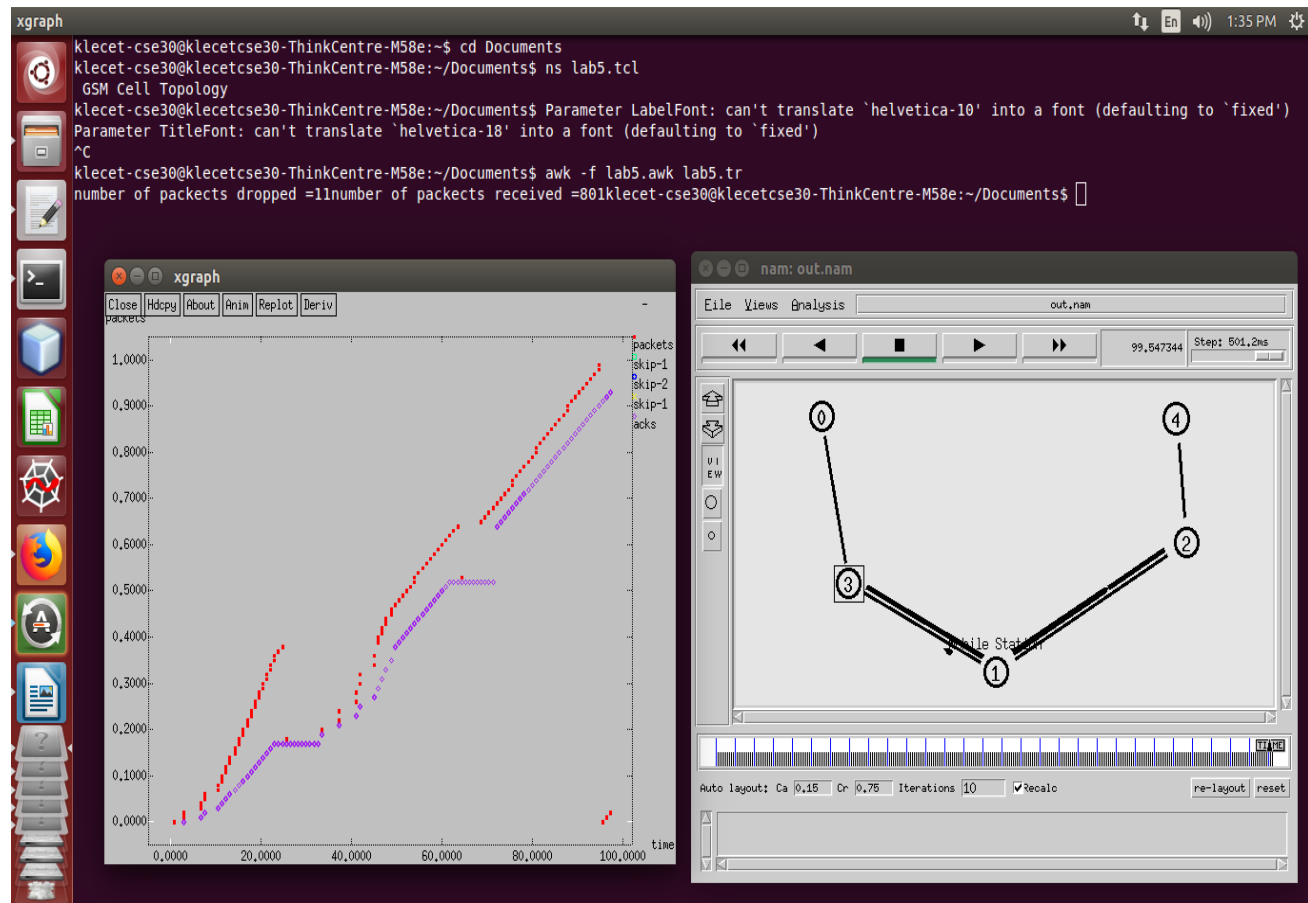
```

if {$flows == 0} {
    set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
    set ftp1 [[set tcp1] attach-app FTP]
    $ns at 0.8 "[set ftp1] start"
}
if {$flows > 0} {
    set tcp1 [$ns create-connection TCP/Sack1 $nodes(is)
    TCPSink/Sack1 $nodes(lp) 0]
    set ftp1 [[set tcp1] attach-app FTP]
    $tcp1 set window_ 100
    $ns at 0.0 "[set ftp1] start"
    $ns at 3.5 "[set ftp1] stop"
    set tcp2 [$ns create-connection TCP/Sack1 $nodes(is)
    TCPSink/Sack1 $nodes(lp) 0]
    set ftp2 [[set tcp2] attach-app FTP]
    $tcp2 set window_ 3
    $ns at 1.0 "[set ftp2] start"
    $ns at 8.0 "[set ftp2] stop"
}

proc stop {} {
    global nodes opt namf
    set wrap $opt(wrap)
    set sid [$nodes($opt(srcTrace)) id]
    set did [$nodes($opt(dstTrace)) id]
    set a "out.tr"

    set GETRC "ns-allinone-2.35/ns-2.35/bin/getrc"
    set RAW2XG "ns-allinone-2.35/ns-2.35/bin/raw2xg"
    exec $GETRC -s $sid -d $did -f 0 out.tr | \
    $RAW2XG -s 0.01 -m $wrap -r > plot.xgr
    exec $GETRC -s $did -d $sid -f 0 out.tr | \
    $RAW2XG -a -s 0.01 -m $wrap >> plot.xgr
    exec nam out.nam &
    exec xgraph -x time -y packets plot.xgr &
    exit 0
}
$ns at $stop "stop"
$ns run

```

Output:

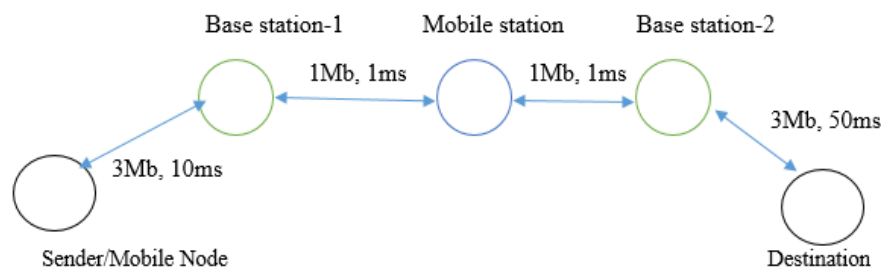
Experiment No: 6

Aim: Implement and study the performance of CDMA on NS2/NS3 (Using stack called Call net) or equivalent environment.

3G networks developed as a replacement for second generation (2G) GSM standard network with full duplex voice telephony. CDMA is used as the access method in many mobile phone standards. IS-95, also called cdmaOne, and its 3G evolution CDMA2000, are often simply referred to as CDMA, but UMTS (The Universal Mobile Telecommunications System is a third generation mobile cellular system for networks based on the GSM standard.), the 3G standard used by GSM carriers, also uses wideband CDMA. Long-Term Evolution (LTE) is a standard for high-speed wireless communication which uses CDMA network technology.

3G technology generally refers to the standard of accessibility and speed of mobile devices. The standards of the technology were set by the International Telecommunication Union (ITU). This technology enables use of various services like GPS (Global Positioning System), mobile television and video conferencing. It not only enables them to be used worldwide, but also provides with better bandwidth and increased speed. The main aim of this technology is to allow much better coverage and growth with minimum investment.

CDMA can be implemented on all the versions of NS2 (Since year 2004: ns-2.27, and later versions of NS2)

Design:**Program:**

```

set stop 100 ;# Stop time.
# Topology
set type cdma ;#type of link:
# AQM parameters
set minth 30 ;
set maxth 0 ;
set adaptive 1 ;# 1 for Adaptive RED, 0 for plain RED
# Traffic generation.
set flows 0 ;# number of long-lived TCP flows
set window 30 ;# window for long-lived traffic
set web 2 ;# number of web sessions
# Plotting statics.
set opt(wrap) 100 ;# wrap plots?
set opt(srcTrace) is ;# where to plot traffic
set opt(dstTrace) bs2 ;# where to plot traffic

#default downlink bandwidth in bps
set bwDL(cdma) 384000
#default uplink bandwidth in bps
set bwUL(cdma) 64000
#default downlink propagation delay in seconds
set propDL(cdma) .150
  
```



```

#default uplink propagation delay in seconds
set propUL(cdma) .150

set ns [new Simulator]
set tf [open out.tr w]
$ns trace-all $tf
set nodes(is) [$ns node]
set nodes(ms) [$ns node]
set nodes(bs1) [$ns node]
set nodes(bs2) [$ns node]
set nodes(lp) [$ns node]

proc cell_topo {} {
    global ns nodes
    $ns duplex-link $nodes(lp) $nodes(bs1) 3Mbps 10nodes(ms) DropTail
    $ns duplex-link $nodes(bs1) $nodes(ms) 1 1 RED
    $ns duplex-link $nodes(ms) $nodes(bs2) 1 1 RED
    $ns duplex-link $nodes(bs2) $nodes(is) 3Mbps 50nodes(ms) DropTail
    puts " cdma Cell Topology"
}

proc set_link_para {t} {
    global ns nodes bwUL bwDL propUL propDL buf
    $ns bandwidth $nodes(bs1) $nodes(ms) $bwDL($t) duplex
    $ns bandwidth $nodes(bs2) $nodes(ms) $bwDL($t) duplex
    $ns delay $nodes(bs1) $nodes(ms) $propDL($t) duplex
    $ns delay $nodes(bs2) $nodes(ms) $propDL($t) duplex
    $ns queue-limit $nodes(bs1) $nodes(ms) 20
    $ns queue-limit $nodes(bs2) $nodes(ms) 20
}

# RED and TCP parameters
Queue/RED set adaptive_ $adaptive
Queue/RED set thresh_ $minth
Queue/RED set maxthresh_ $maxth
Agent/TCP set window_ $window

source ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts/web.tcl
#Create topology
switch $type {
    cdma {cell_topo}
}

set_link_para $type
$ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]
$ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]
# Set up forward TCP connection
if {$flows == 0} {
    set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
    set ftp1 [[set tcp1] attach-app FTP]
    $ns at 0.8 "[set ftp1] start"
}
if {$flows > 0} {
    set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
    set ftp1 [[set tcp1] attach-app FTP]
    $tcp1 set window_ 100
    $ns at 0.0 "[set ftp1] start"
    $ns at 3.5 "[set ftp1] stop"
}

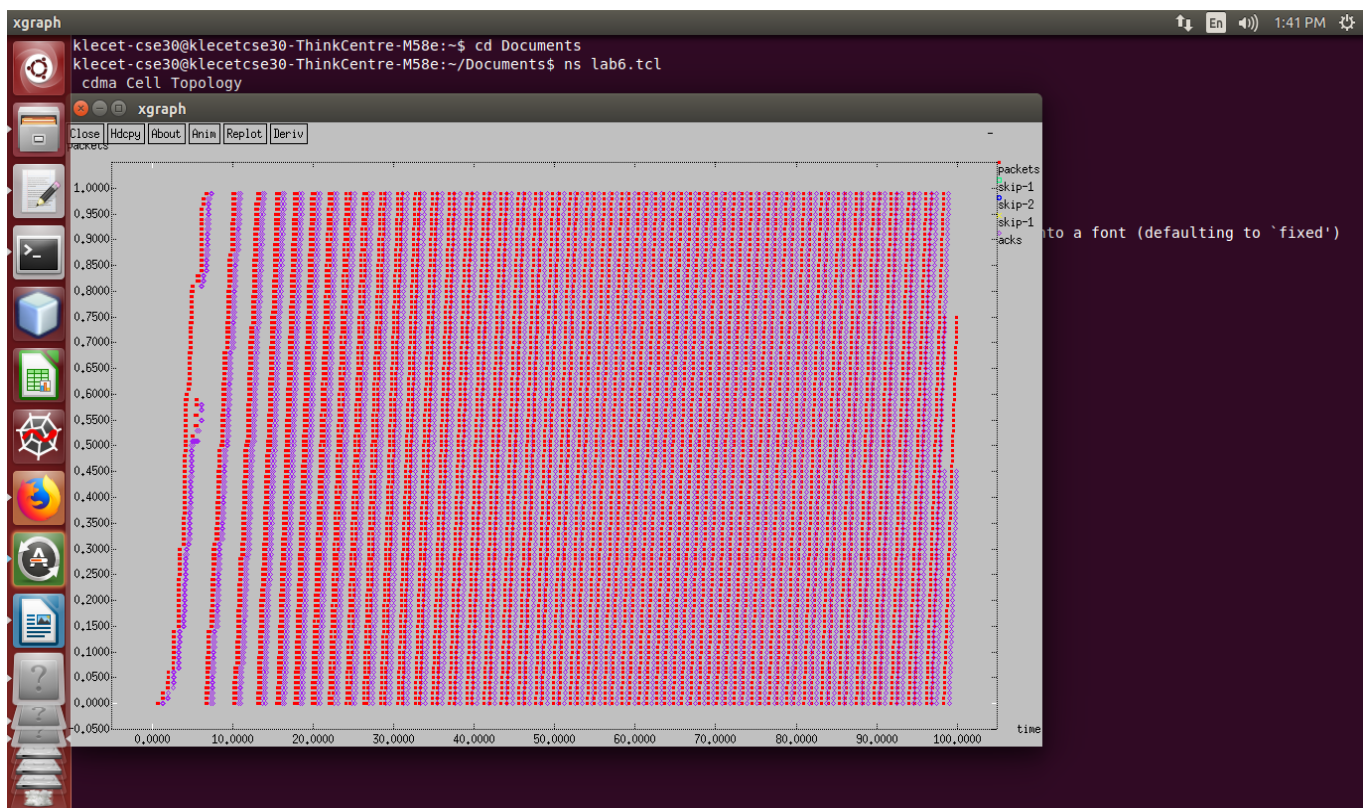
```

```

set tcp2 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
set ftp2 [[set tcp2] attach-app FTP]
$tcp2 set window_ 3
$ns at 1.0 "[set ftp2] start"
$ns at 8.0 "[set ftp2] stop"
}
proc stop {} {
    global nodes opt nf
    set wrap $opt(wrap)
    set sid [$nodes($opt(srcTrace)) id]
    set did [$nodes($opt(dstTrace)) id]
    set a "out.tr"
    set GETRC "ns-allinone-2.35/ns-2.35/bin/getrc"
    set RAW2XG "ns-allinone-2.35/ns-2.35/bin/raw2xg"
    exec $GETRC -s $sid -d $did -f 0 out.tr | \
        $RAW2XG -s 0.01 -m $wrap -r > plot.xg
    exec $GETRC -s $did -d $sid -f 0 out.tr | \
        $RAW2XG -a -s 0.01 -m $wrap >> plot.xg
    exec xgraph -x time -y packets plot.xg &
    exit 0
}
$ns at $stop "stop"
$ns run

```

Output:



PART-B

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).

1. 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 connection are checked with a 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.

$$\begin{array}{r}
 101 = 5 \\
 10011 / 1101101 \\
 \underline{10011} \\
 100000 \\
 \underline{00000} \\
 100001 \\
 \underline{10011} \\
 1110 = 14 = \text{remainder}
 \end{array}$$

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, the 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

Source Code:

```
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 remainder
        {
            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.... :)");
}

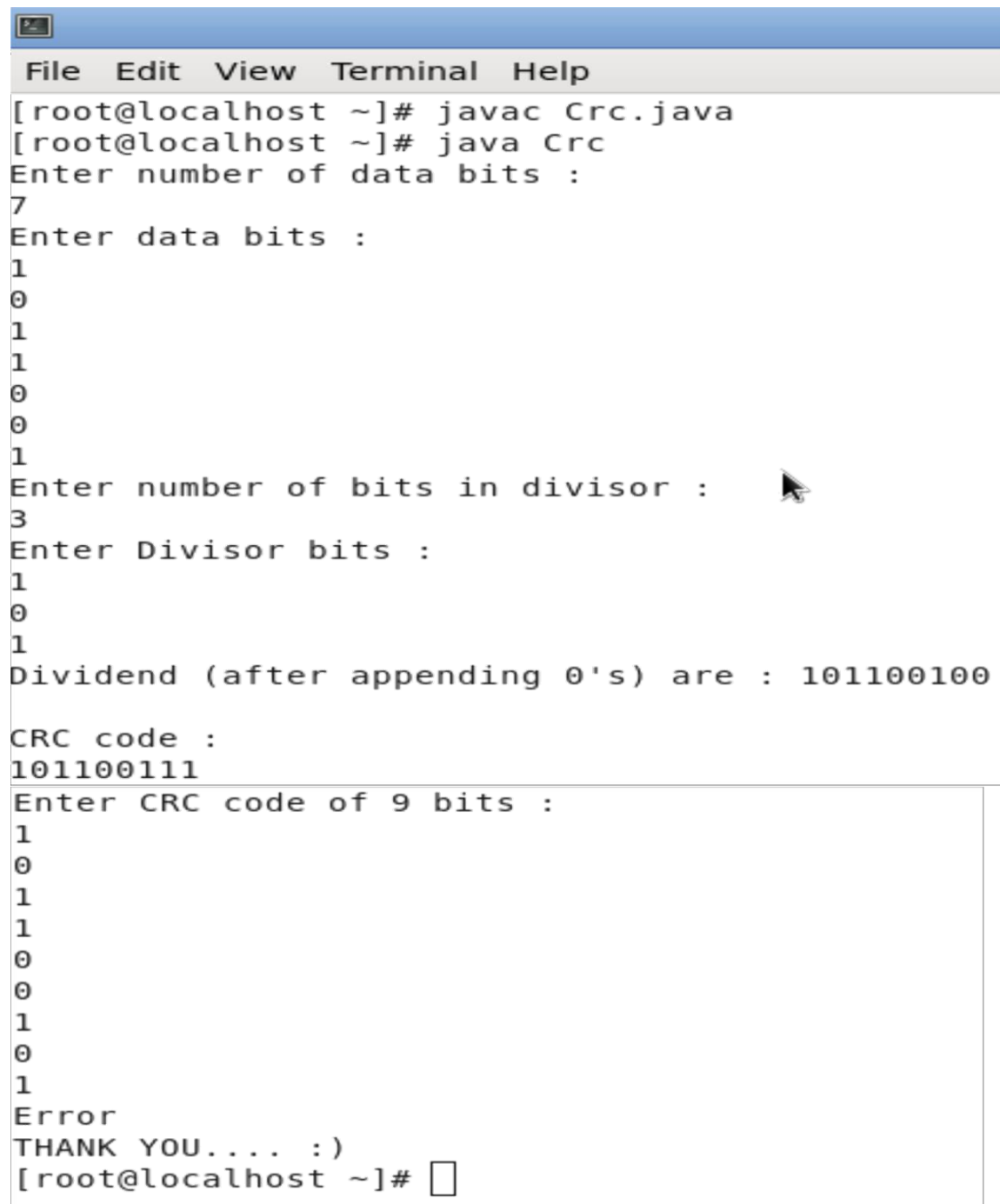
static int[] divide(int div[],int divisor[], int rem[])
{
    int cur=0;
    while(true)
    {
        for(int i=0;i<divisor.length;i++)
            rem[cur+i]=(rem[cur+i]^divisor[i]);

        while(rem[cur]==0 && cur!=rem.length-1)
            cur++;

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

Output:

[root@localhost ~]# vi Crc.java



```
File Edit View Terminal Help
[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 ~]#
```


2. 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 num_ver;
    public static final int MAX_VALUE = 999;

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

    public void BellmanFordEvaluation(int source, int A[][])
    {
        for (int node = 1; node <= num_ver; node++)
        {
            D[node] = MAX_VALUE;
        }
        D[source] = 0;
        for (int node = 1; node <= num_ver - 1; node++)
```

```
        {
            for (int sn = 1; sn <= num_ver; sn++)
            {
                for (int dn = 1; dn <= num_ver; dn++)
                {
                    if (A[sn][dn] != MAX_VALUE)
                    {
                        if (D[dn] > D[sn] + A[sn][dn])
                            D[dn] = D[sn] + A[sn][dn];
                    }
                }
            }
        }
        for (int sn = 1; sn <= num_ver; sn++)
        {
            for (int dn = 1; dn <= num_ver; dn++)
            {
                if (A[sn][dn] != MAX_VALUE)
                {
                    if (D[dn] > D[sn] + A[sn][dn])
                        System.out.println("The Graph contains negative egde cycle");
                }
            }
        }
        for (int vertex = 1; vertex <= num_ver; vertex++)
        {
            System.out.println("distance of source " + source + " to " + vertex + " is " + D[vertex]);
        }
    }
}
```

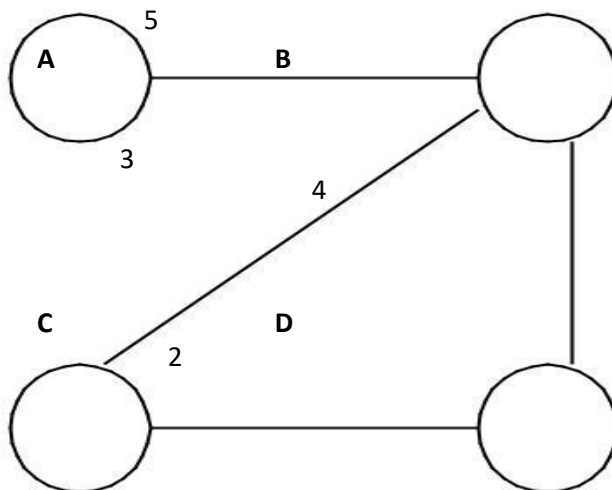
```
public static void main(String[] args)
{
    int num_ver = 0;
    int source;
    Scanner scanner = new
    Scanner(System.in);
    System.out.println("Enter the number of
    vertices"); num_ver = scanner.nextInt();
    int A[][] = new int[num_ver + 1][num_ver + 1];
    System.out.println("Enter the adjacency matrix");
    for (int sn = 1; sn <= num_ver; sn++)
    {
        for (int dn = 1; dn <= num_ver; dn++)
        {
            A[sn][dn] = scanner.nextInt();
            if (sn == dn)
            {
                A[sn][dn] = 0;
                continue;
            }
        }
    }
}
```

```

        if (A[sn][dn] == 0)
        {
            A[sn][dn] = MAX_VALUE;
        }
    }
}
System.out.println("Enter the source vertex");
source = scanner.nextInt();
BellmanFord b = new BellmanFord
(num_ver); b.BellmanFordEvaluation(source,
A); scanner.close();
}
}

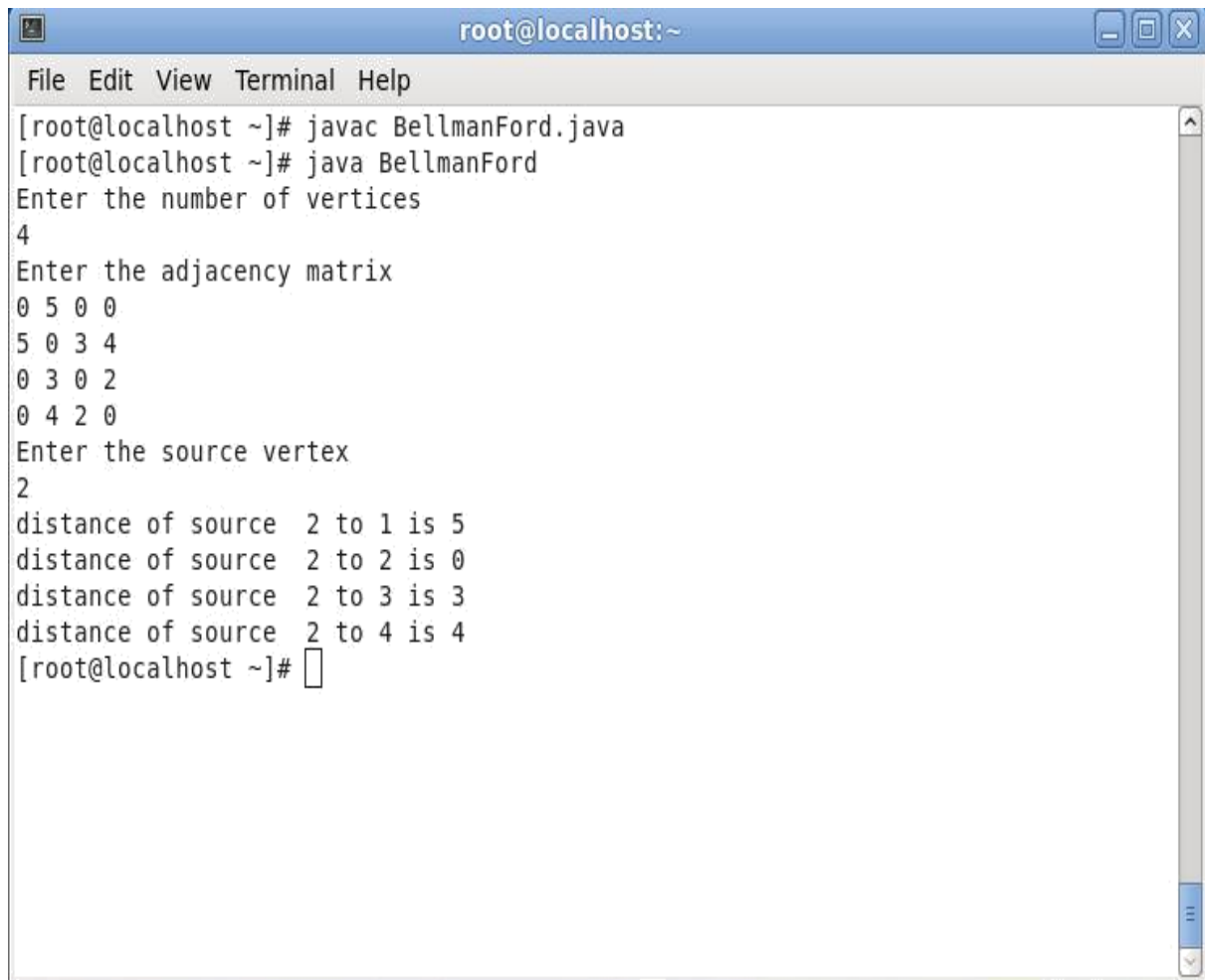
```

Input graph:



Output:

[root@localhost ~]# vi BellmanFord.java



```
root@localhost:~
File Edit View Terminal Help
[root@localhost ~]# javac BellmanFord.java
[root@localhost ~]# java BellmanFord
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
[root@localhost ~]#
```

3. 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.

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.io.BufferedReader;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.EOFException;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStreamReader;
import java.net.Socket;
import java.util.Scanner;

class Client
{
    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();
        //create the socket on port 5000
        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));

        System.out.println("Send Get to start...");
        String str="",filename="";
        try
        {
            while(!str.equals("start"))
            {
                str=br.readLine();
                dout.writeUTF(str);
                dout.flush();
                filename=din.readUTF();
                System.out.println("Receiving file: "+filename);
                filename="client"+filename;
                System.out.println("Saving as file: "+filename);
            }
        }
    }
}
```

```
        long sz=Long.parseLong(din.readUTF()); System.out.println
        ("File Size: "+(sz/(1024*1024))+" MB");
        byte b[]=new byte [1024];
        System.out.println("Receiving file..");
        FileOutputStream fos=new FileOutputStream(new File(filename),true);
        long bytesRead;
        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
    }
}
}
```

TCP Server

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;
class Server
{
    public static void main(String args[])throws Exception
    {
        String filename;
        System.out.println("Enter File Name: ");
        Scanner sc=new Scanner(System.in);
        filename=sc.nextLine();
        sc.close();
        while(true)
        {
            //create server socket on port 5000
            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 str="";
    str=din.readUTF();
    System.out.println("SendGet....Ok");
    if(!str.equals("stop")){
        System.out.println("Sending File: "+filename);
        dout.writeUTF(filename);
        dout.flush();
        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 ("Size: "+sz);
        System.out.println ("Buf size:
        "+ss.getReceiveBufferSize()); while((read = fin.read(b)) !=
        -1) {
            dout.write(b, 0, read);
            dout.flush();
        }
        fin.close();
        System.out.println("..ok");
        dout.flush();
    }
    dout.writeUTF("stop");
    System.out.println("Send Complete");
    dout.flush();
}
catch(Exception e)
{
    e.printStackTrace();
    System.out.println("An error occured");
}
din.close();
s.close();
ss.close();
}
}
}

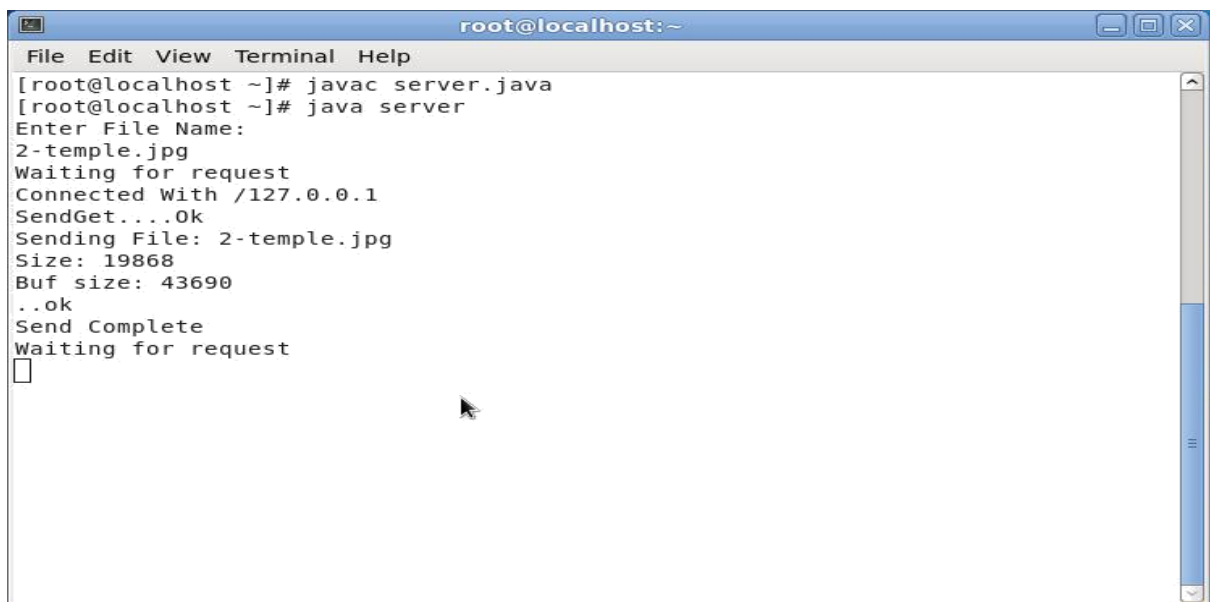
```

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.

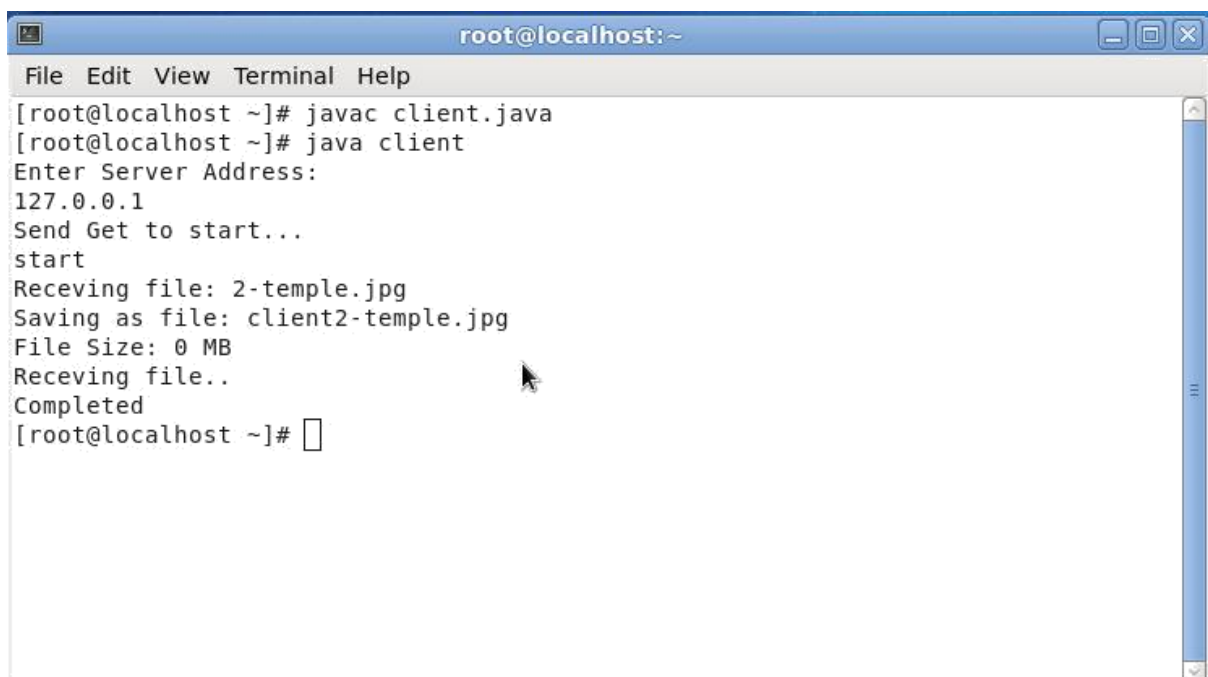
Output:

At server side:



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


4. Write a program on 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;
        try
        {
            skt=new DatagramSocket();
            String msg= "text message ";
            byte[] b = msg.getBytes();
            InetAddress host=InetAddress.getByName("127.0.0.1");
            int serverSocket=6788;
            DatagramPacket request =new DatagramPacket
                (b,b.length,host,serverSocket); skt.send(request);
            byte[] buffer =new byte[1000];
            DatagramPacket reply= new
                DatagramPacket(buffer,buffer.length); skt.receive(reply);
            System.out.println("client received:" +new
                String(reply.getData())); skt.close();
        }
        catch(Exception ex)
        {
        }
    }
}
```

UDP Server

```

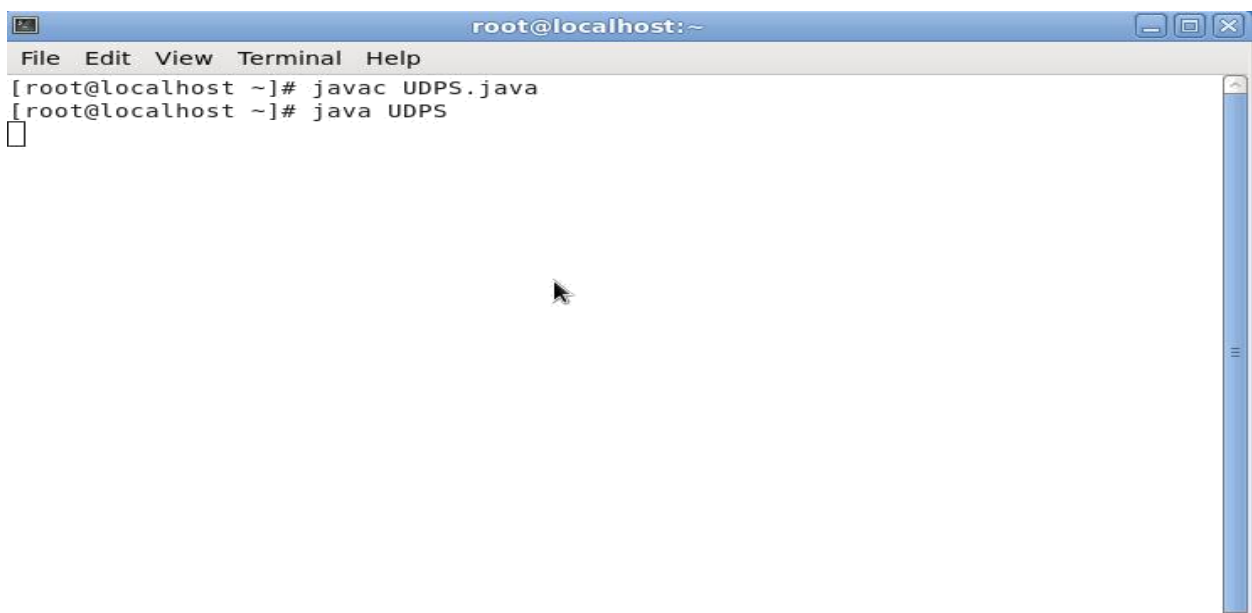
import java.io.*;
import java.net.*;

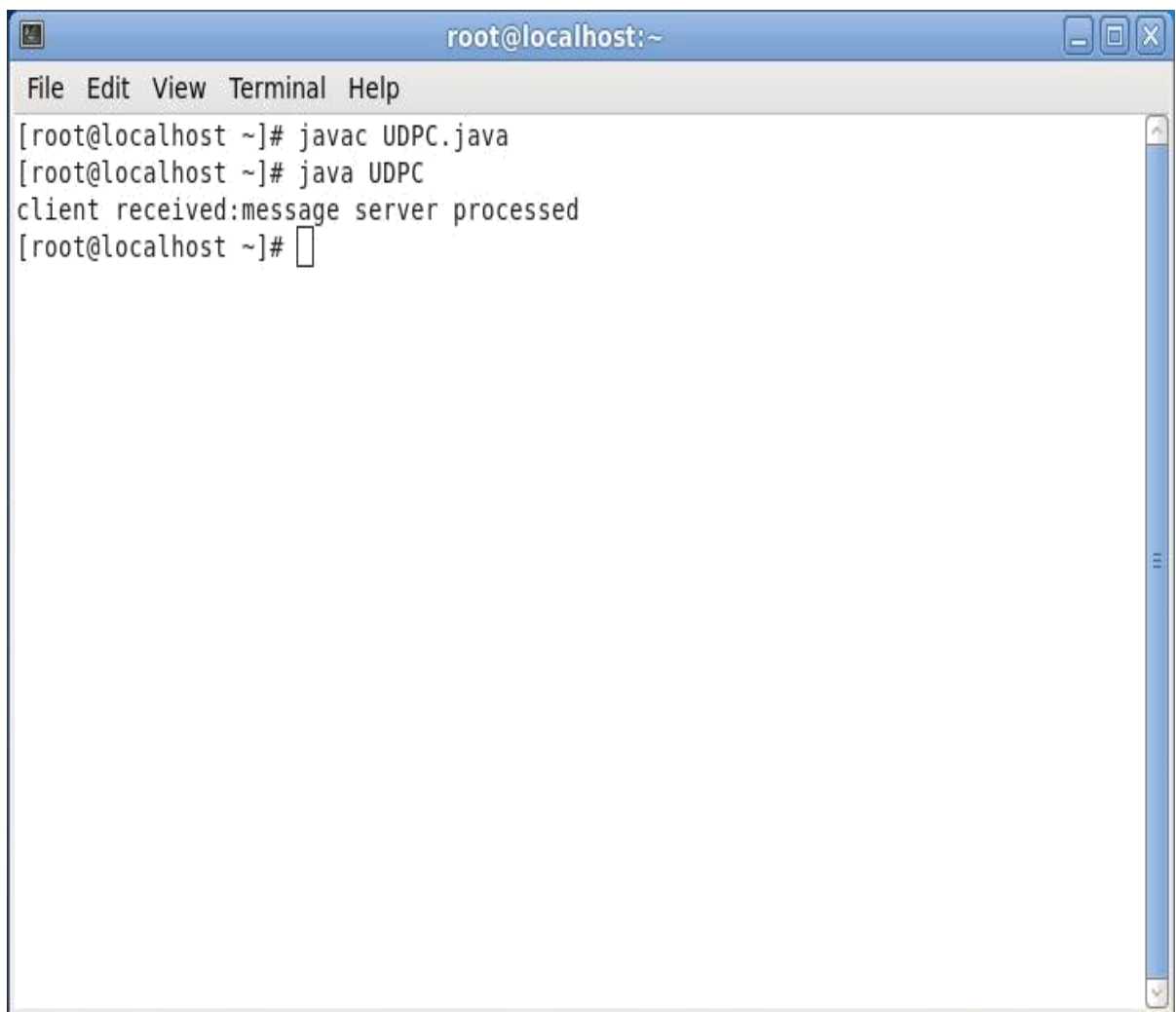
public class UDPS
{
    public static void main(String[] args)
    {
        DatagramSocket skt=null;
        try
        {
            skt=new DatagramSocket(6788);
            byte[] buffer = new byte[1000];
            while(true)
            {
                DatagramPacket request = new
                    DatagramPacket(buffer,buffer.length); skt.receive(request);
                String[] message = (new String(request.getData())).split(" ");
                byte[] sendMsg= (message[1]+ " server processed").getBytes();
                DatagramPacket      reply      =      new
                    DatagramPacket(sendMsg,sendMsg.length,request.getAddress
                    (),request.getPort());
                skt.send(reply);
            }
        }
        catch(Exception 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.

At Server side:

At Client side:A screenshot of a terminal window titled 'root@localhost:~'. The window has a menu bar with 'File', 'Edit', 'View', 'Terminal', and 'Help'. The terminal content shows the following commands and output:

```
[root@localhost ~]# javac UDPC.java
[root@localhost ~]# java UDPC
client received:message server processed
[root@localhost ~]#
```

The terminal has a scrollbar on the right side.

5. 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 $n = p \cdot q$
2. Compute $n = p \cdot q$ 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 $e \cdot d \equiv 1 \pmod{\phi}$.
5. The public key is (e, n) and the private key is (d, n) . The values of p , q , and ϕ 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 $C = M^e \pmod{n}$.
4. Sends the cipher text C to B (Receiver).

Decryption

Recipient B does the following:-

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

Source Code:

```
import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;
public class RSA
{
    private BigInteger p;
    private BigInteger q;
    private BigInteger N;
    private BigInteger phi;
    private BigInteger e;
    private BigInteger d;
```

```
private int bitlength = 1024;
private Random r;
public RSA()
{
    r = new Random();
    p = BigInteger.probablePrime(bitlength, r);
    q = BigInteger.probablePrime(bitlength, r);
    N = p.multiply(q);
    phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
    e = BigInteger.probablePrime(bitlength / 2, r);
    while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0)
    {
        e.add(BigInteger.ONE);
    }
    d = e.modInverse(phi);
}
public RSA(BigInteger e, BigInteger d, BigInteger N)
{
    this.e = e;
    this.d = d;
    this.N = N;
}
public static void main(String[] args) throws IOException
{
    RSA rsa = new RSA();
    DataInputStream in = new DataInputStream(System.in);
    String teststring;
    System.out.println("Enter the plain text:");
    teststring = in.readLine();
    System.out.println("Encrypting String: " + teststring);
    System.out.println("String in Bytes: "
        + bytesToString(teststring.getBytes()));
    // encrypt
    byte[] encrypted = rsa.encrypt(teststring.getBytes());
    // decrypt
    byte[] decrypted = rsa.decrypt(encrypted);

    System.out.println("Decrypting Bytes: " + bytesToString(decrypted));
    System.out.println("Decrypted String: " + new String(decrypted));
}
private static String bytesToString(byte[] encrypted)
{
    String test = "";
    for (byte b : encrypted)
    {
        test += Byte.toString(b);
    }
    return test;
}
// Encrypt message
public byte[] encrypt(byte[] message)
```

```
{
return (new BigInteger(message)).modPow(e, N).toByteArray();
}
// Decrypt message
public byte[] decrypt(byte[] message)
{
return (new BigInteger(message)).modPow(d, N).toByteArray();
}
}import java.io.DataInputStream;
import java.io.IOException;
import java.math.BigInteger;
import java.util.Random;
public class RSA
{
private BigInteger p;
private BigInteger q;
private BigInteger N;
private BigInteger phi;
private BigInteger e;
private BigInteger d;
private int bitlength = 1024;
private Random r;
public RSA()
{
r = new Random();
p = BigInteger.probablePrime(bitlength, r);
q = BigInteger.probablePrime(bitlength, r);
N = p.multiply(q);
phi = p.subtract(BigInteger.ONE).multiply(q.subtract(BigInteger.ONE));
e = BigInteger.probablePrime(bitlength / 2, r);
while (phi.gcd(e).compareTo(BigInteger.ONE) > 0 && e.compareTo(phi) < 0)
{
e.add(BigInteger.ONE);
}
d = e.modInverse(phi);
}
public RSA(BigInteger e, BigInteger d, BigInteger N)
{
this.e = e;
this.d = d;
this.N = N;
}
public static void main(String[] args) throws IOException
{
RSA rsa = new RSA();
DataInputStream in = new DataInputStream(System.in);
String teststring;
System.out.println("Enter the plain text:");
teststring = in.readLine();
System.out.println("Encrypting String: " + teststring);
System.out.println("String in Bytes: "
```

```
+ bytesToString(teststring.getBytes());
// encrypt
byte[] encrypted = rsa.encrypt(teststring.getBytes());
// decrypt
byte[] decrypted = rsa.decrypt(encrypted);

System.out.println("Decrypting Bytes: " + bytesToString(decrypted));
System.out.println("Decrypted String: " + new String(decrypted));
}
private static String bytesToString(byte[] encrypted)
{
    String test = "";
    for (byte b : encrypted)
    {
        test += Byte.toString(b);
    }
    return test;
}
// Encrypt message
public byte[] encrypt(byte[] message)
{
    return (new BigInteger(message)).modPow(e, N).toByteArray();
}
// Decrypt message
public byte[] decrypt(byte[] message)
{
    return (new BigInteger(message)).modPow(d, N).toByteArray();
}
}
```

Output:

```
$ javac RSA.java
```

```
$ java RSA
```

```
Enter the plain text: Sanfoundry
```

```
Encrypting String: Sanfoundry
```

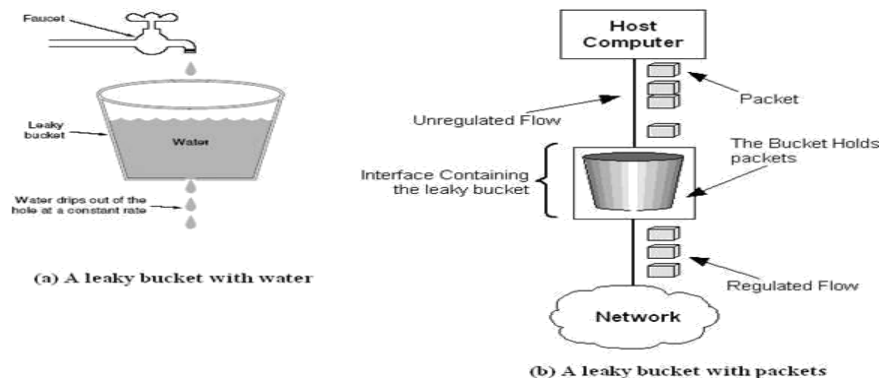
```
String in Bytes: 8397110102111117110100114121
```

```
Decrypting Bytes: 8397110102111117110100114121
```

```
Decrypted String: Sanfoundry
```

6. 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)).



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.io.*;
import java.util.*;
class Queue
{
    int q[],f=0,r=0,size;
    void insert(int n)
    {
        Scanner in = new Scanner(System.in);
        q=new int[10];
        for(int i=0;i<n;i++)
        {
            System.out.print("\nEnter " + i + " element: ");
            int ele=in.nextInt();
            if(r+1>10)
            {
```



```
        System.out.println("\nQueue is full \nLost Packet: "+ele);
        break;
    }
    else
    {
        r++;
        q[i]=ele;
    }
}
}

void delete()
{
    Scanner in = new Scanner(System.in);
    Thread t=new Thread();
    if(r==0)
        System.out.print("\nQueue empty ");
    else
    {
        for(int i=f;i<r;i++)
        {
            try
            {
                t.sleep(1000);
            }
            catch(Exception e){ }
            System.out.print("\nLeaked Packet: "+q[i]);
            f++;
        }
        System.out.println();
    }
}

}

class Leaky extends Thread
{
    public static void main(String ar[]) throws Exception
    {
        Queue q=new Queue();
        Scanner src=new Scanner(System.in);
        System.out.println("\nEnter the packets to be sent:");
        int size=src.nextInt();
        q.insert(size);
        q.delete();
    }
}
```

Output:

```
File Edit View Terminal Help
```

```
Enter the packets to be sent:  
12
```

```
Enter 0 element: 2
```

```
Enter 1 element: 3
```

```
Enter 2 element: 5
```

```
Enter 3 element: 6
```

```
Enter 4 element: 8
```

```
Enter 5 element: 9
```

```
Enter 6 element: 4
```

```
Enter 7 element: 5
```

```
Enter 8 element: 6
```

```
Enter 9 element: 2
```

```
Enter 10 element: 3
```

```
Queue is full  
Lost Packet: 3
```

```
Leaked Packet: 2
```

```
Leaked Packet: 3
```

```
Leaked Packet: 5
```

```
Leaked Packet: 6
```

```
Leaked Packet: 8
```

```
Leaked Packet: 9
```

```
Leaked Packet: 4
```

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 ephemeral 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 IPC? Name three techniques.
28. Explain mkfifo(), open(),close()with parameters.
29. What is meant by file descriptor?
30. What is meant by traffic shaping?
31. How do you classify congestion control algorithms?
32. Differentiate between Leaky bucket and Token bucket.
33. How do you implement Leaky bucket?
34. How do you generate busty traffic?
35. What is the polynomial used in CRC-CCITT?
36. What are the other error detection algorithms?
37. What is difference between CRC and Hamming code?
38. Why Hamming code is called 7,4code?
39. What is odd parity and even parity?
40. What is meant by syndrome?
41. What is generator matrix?
42. What are Routing algorithms?

43. How do you classify routing algorithms? Give examples for each.
44. What are drawbacks in distance vector algorithm?
45. How routers update distance stoeacho fits neighbor?
46. How do you overcome count to infinity problem?
47. What is cryptography?
48. How do you classify cryptographic algorithms?
49. What is public key?
50. What is private key?
51. What are key cipher text and plaintext?
52. What is simulation?
53. What are advantages of simulation?
54. Differentiate between Simulation and Emulation.
55. What is meant by router?
56. What is meant by bridge?
57. What is meant by switch?
58. What is meant by hub?
59. Differentiate between route, bridge, switch and hub.
60. What is ping and telnet?
61. What is FTP?
62. What is BER?
63. What is meant by congestion window?
64. What is BSS?
65. What is incoming through put and outgoing throughput?
66. What is collision?
67. How do you generate multiple traffics across different sender-receiver pairs?
68. How do you setup Ethernet LAN?
69. What is meant by mobile host?
70. What is meant by NCTUns?
71. What are dispatcher, coordinator and nctunscient?
72. Name few other Network simulators
73. Differentiate between logical and physical address.
74. Which address gets affected if a system moves from one place to another place?
75. What is ICMP? What are uses of ICMP? Name few.
76. Which layer implements security for data?
77. What is connectionless and connection oriented protocol?
78. What is Contention?
79. What is Congestion window?
80. What is flow control?