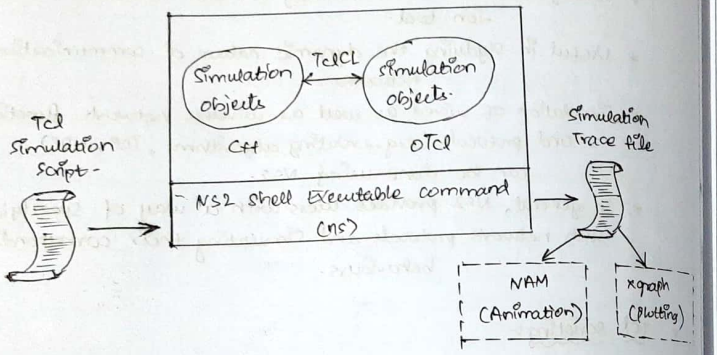
* Basic architecture of NS2



Event	Time	From Node	To Node	PKT Type	PKT size	Flags	Fid	Src Addr	Dest Addr	Seq Num	Pkt Id

Experiment No.

Page No.: 02

Set the time of traffic generation (e.g., CBR, FTP)
Terminate the simulation.

* NS Simulator preliminaries:-

(1) Initialization and termination aspects of the ns simulator
(2) Definition of network nodes, links, queues and topology.
(3) Definition of agents and of applications.
(4) The nam visualization tool.
(5) Tracing and random variables.

* Structure of Traces files:

When tracing into an output ASCII file, the trace is organized in 12 fields as follows in fig shown, The meanings of the fields are:

(1) The first field is the event type. It is given by one of four possible symbols r, +, -, d which correspond respectively to receive (at the output of the link), enqueued, dequeued and dropped.
(2) The second field gives the time at which the event occurs.
(3) Gives the input node of the link at which the event occurs.
(4) Gives the output node of the link at which the event occurs.
(5) Gives the packet type (eg CBR or TCP)
(6) Gives the packet size
(7) Some flags.

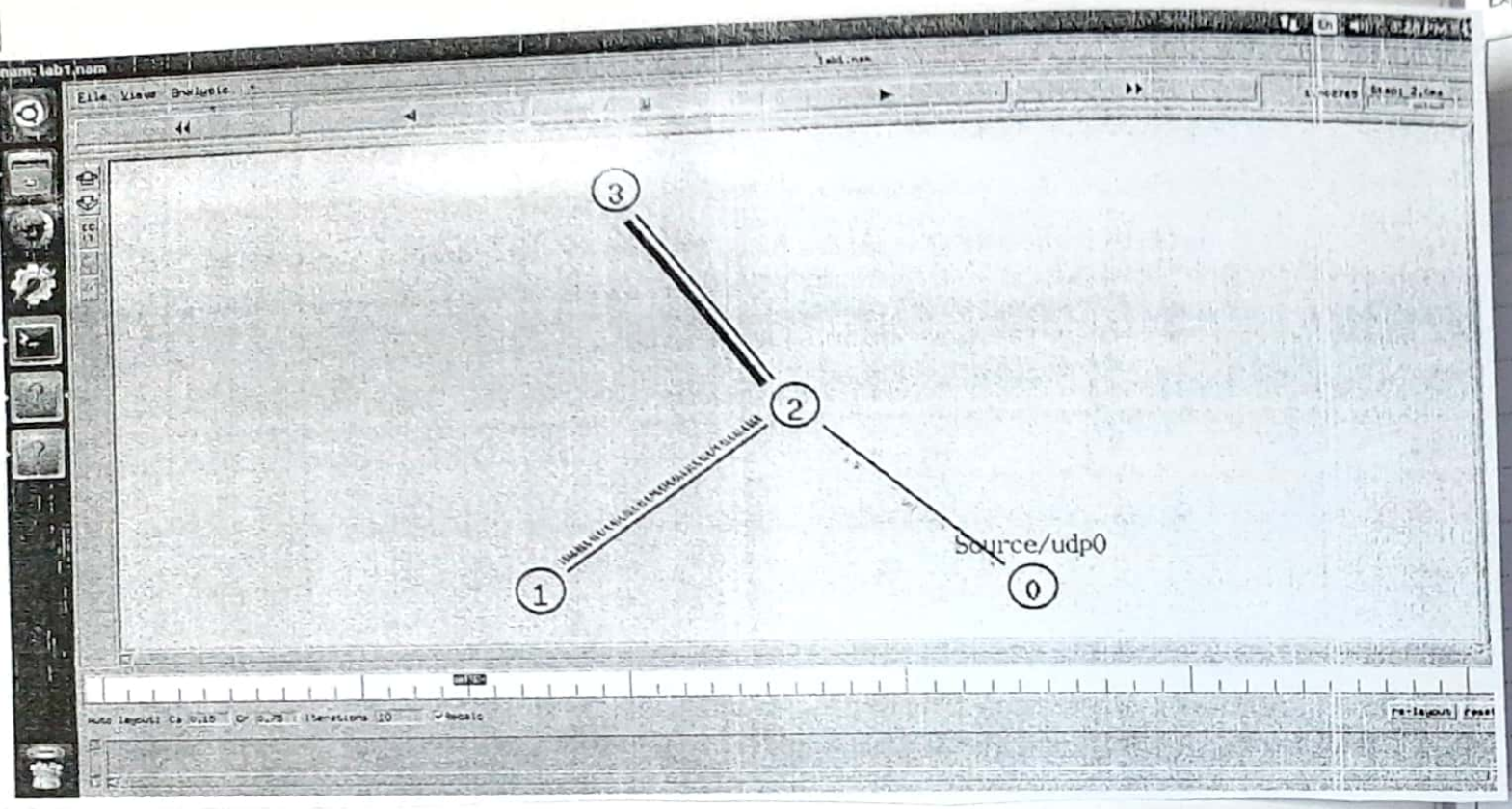
(8) This is flow id (fid) of IPV6 that a user can set for each flow at the input OTcl script one can further use this field for analysis purposes; it is also used when specifying stream color for the NAM display.

(9) This is the source address given in the form of "node port".

(10) This is the destination address, given in the same form.

(11) This is the network layer protocol's packet sequence number. Even though UDP implementations in a real network do not use sequence number, ns keeps track of UDP packet sequence number for analysis purpose.

(12) The last field shows the Unique id of the packet.



iment No.

Page No.: 05

```
$ns color 1 "red"
$ns color 2 "blue"
$udp0 set class_ 1
$udp1 set class_ 2
set null [new Agent/NULL]
$ns attach-agent $n3 $null
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1

$ns connect $udp0 $null
$ns connect $udp1 $null

$cbr1 set interval_ 0.0005

Proc finish {} {
  global ns tf nf
  $ns flush-trace
  exec nam lab1.nam &
  close $tf
  close $nf
  exit 0
}

$ns at 0.1 "$cbr0 start"
$ns at 0.5 "$cbr1 start"
$ns at 5.0 "finish"
$ns run.
```

Experiment No.

Page No.: 06

⇒ AWK file:- (open a new editor using "vi command" and write awk file and save with ".awk" extension)

```
BEGIN{
  #include<stdio.h>
  count=0
}
{
if($1=="d")
   count++
}
END{
  Printf("packets dropped is %d\n", count)
}
```

⇒ Steps for execution:-

* Open gedit editor and type program. Program name should have the extension ".tcl"

  [root@localhost ~]# gedit lab1.tcl.

* Save the program and close the file.
* Open the editor and type awk program. Program name should have the extension ".awk"

  [root@localhost ~]# gedit lab1.awk.

* Save the program and close the file.
* Run the simulation program

  [root@localhost ~]# ns lab1.tcl.

* Here "ns" indicates network simulator. We get the topology shown in the snapshot

Experiment No.

* 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.

TRANMISSION OF PING MESSAGE

Aim:- Implement transmission of ping message / trace route over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.

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

set no [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$no label "ping0"
$n1 label "ping1"
$n2 label "R1"
$n3 label "R2"
$n4 label "ping2"
$n5 label "ping3"

$ns color 1 Red
$ns color 2 blue
$ns color 3 green
$ns color 4 orange
```

```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail.
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n3 $n5 1Mb 10ms DropTail.

proc finish {} {
    global ns tf nf
    exec nam lab2.nam &
    close $tf
    close $nf
    exit 0
}
set ping0 [new Agent/Ping]
$ns attach-agent $n0 $ping0
set ping1 [new Agent/Ping]
$ns attach-agent $n1 $ping1
set ping4 [new Agent/Ping]
$ns attach-agent $n4 $ping4
set ping5 [new Agent/Ping]
$ns attach-agent $n5 $ping5

$ns connect $ping0 $ping4
$ns connect $ping1 $ping5

proc sendingPacket {} {
    global ns ping0 ping1
    set interevalTime 0.01
    set now [$ns now]
```

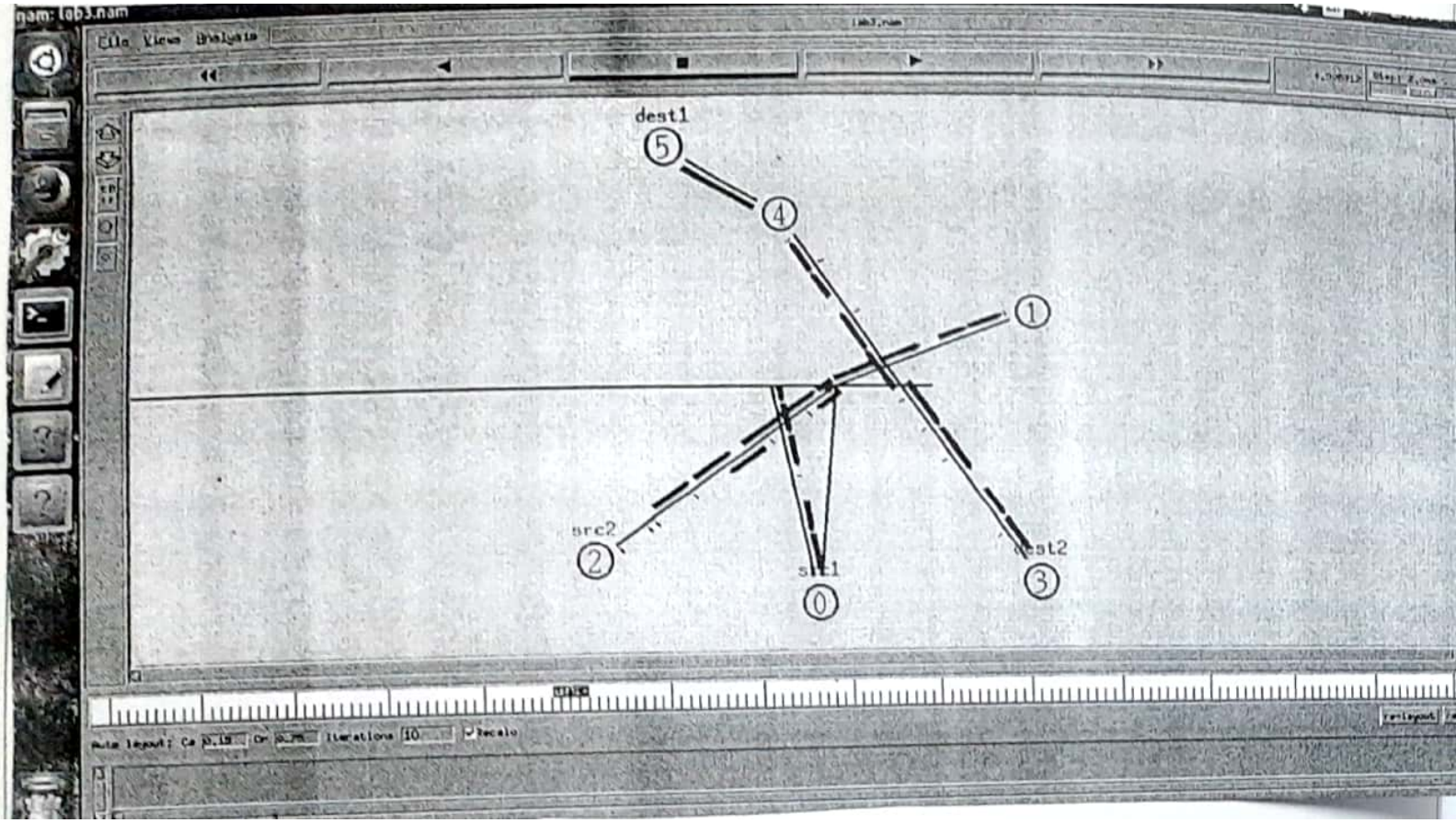
```
$ns duplex-link $n0 $n2 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 10ms DropTail
$ns duplex-link $n2 $n3 1Mb 10ms DropTail.
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n3 $n5 1Mb 10ms DropTail.

proc finish {} {
    global ns tf nf
    exec nam lab2-nam &
    close $tf
    close $nf
    exit 0
}
set ping0 [new Agent/Ping]
$ns attach-agent $n0 $ping0
set ping1 [new Agent/Ping]
$ns attach-agent $n1 $ping1
set ping4 [new Agent/Ping]
$ns attach-agent $n4 $ping4
set ping5 [new Agent/Ping]
$ns attach-agent $n5 $ping5

$ns connect $ping0 $ping4
$ns connect $ping1 $ping5

proc sendingPacket {} {
    global ns ping0 ping1
    set interval Time 0.01
    set now [$ns now]
```

```
    $ns at [expr $now + $interval Time] "$ping0 send"
    $ns at [expr $now + $interval Time] "$ping1 send"
    $ns at [expr $now + $interval Time] "sendingPacket"
}

Agent/Ping instproc recv {from rtt} {
        $self instvar node_
        puts "The node [$node_ id] received an ACK from
                the node $from with RTT $rtt ms"
}

$ping0 set class_ 1
$ping1 set class_ 2
$ping2 set class_ 3
$ping3 set class_ 4

$ns at 0.01 "sendingPacket"
$ns at 4.0 "finish"
$ns run
```

⇒ AWK file: (Open a new editor using "gedit command" & write awk file & save with ".awk" extension)

```
BEGIN {
  #include<stdio.h>
  count=0
}
{
  if ($1 == "d")
```

```
        count++
    }
}
END {
    printf ("packets dropped is :%d\n", count)
}
```



```
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp0 set packetSize_ 5000
$ftp0 set interval_ 0.0001
set sink5 [new Agent/TCPSink]
$ns attach-agent $n5 $sink5

$ns connect $tcp0 $sink5

set tcp2 [new Agent/TCP]
$ns attach-agent $n2 $tcp2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ftp2 set packetSize_ 600
$ftp2 set interval_ 0.001
set sink3 [new Agent/TCPSink]
$ns attach-agent $n3 $sink3

$ns connect $tcp2 $sink3

set file1 [open file1.tr w]
$tcp0 attach $file1
set file2 [open file2.tr w]
$tcp2 attach $file2

$tcp0 trace cwnd_
$tcp2 trace cwnd_
```

```
proc finish {} {
    global ns tf nf
    $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"
$ns at 14 "$ftp0 stop"

$ns at 0.2 "ftp2 start"
$ns at 6 "ftp2 stop"
$ns at 8 "ftp2 start"
$ns at 15 "ftp2 stop".

$ns at 16 "finish"
$ns run
```

AWK file:

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

SIMPLE ESS WITH WIRELESS LAN

Aim: Implement simple ESS and with transmitting nodes in wire-less LAN by simulation and determine the performance with respect to transmission of packets.

```
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/Wireless channel \
                -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 "tcp1/sink1"
$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_ 200
$n2 set Y_ 200
$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 200 200 35"

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
#
Proc finish {} {
    global $ns tf nf
    $ns flush-trace
    close $tf
    close $nf
    exec nam lam64.nam &
    exit 0
}

$ns at 0.5 "$ftp0 start"
$ns at 0.5 "$ftp1 start"
$ns at 10 "$n2 setdest 700 700 60"
$ns at 80 "$n2 setdest 200 200 25"
$ns at 60 "finish"
$ns run.

AWK file:-

BEGIN {
    #include <stdio.h>
    count1=0;
```

```
        count2 = 0;
        time1 = 0;
        time2 = 0;
        packet1 = 0;
        packet2 = 0;
}
{
    if ($1 == "r" && $3 == "_1_" && $4 == "AGT")
    {
        count1++
        packet1 = packet1 + $8
        time1 = $2
    }
    if ($1 == "r" && $3 == "_2_" && $4 == "AGT")
    {
        count2++
        packet2 = packet2 + $8
        time2 = $2
    }
END {
    printf (" The throughput from n0 to n1 is %f Mbps \n",
    ((count1 * packet1 * 8) / (time1 * 1000000)));
    printf (" The throughput from n1 to n2 is %f Mbps \n",
    ((count2 * packet2 * 8) / (time2 * 1000000)));
}
```

Error Detecting code using CRC-CCITT (16-bit)

Aim:- Write a program for ERROR detecting code using CRC-CCITT (16bit).

Source code:-

```
import java.util.*;
class crc
{    void div(int a[], int k)
     {  int gp[] = {1,0,0,0,1,0,0,0,0,0,0,0,1,0,0,0,0,1};
        int count=0;
        for(int i=0; i<k; i++)
        {
           if(a[i]==gp[0])
           {
              for(int j=i; j<17+i; j++)
              {  a[j]=a[j]^gp[count++];
              }
              count=0;
           }
        }
     }
  public static void main(String args[])
  {
     int a[]=new int[100];
     int b[]=new int[100];
     int len,k;
     crc ob=new crc();
     System.out.println("Enter the length of Data frame");
```

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

Output:

```
Enter the length of Data frame : 4
Enter the message : 1011
Data to be transmitted : 1011110111000101101011
Enter the received data : 1011110111000001101011
Error in data
```



```
                flag=1;
                break;
            }
        }
        if (flag==1)
            System.out.println("Error in data");
        else
            System.out.println("No error in data");
    }
}
```

Bellman-ford algorithm

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

Source-code:

```
import java.util.*;
public class bellman_ford {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n,s;
        System.out.println("Enter the no of vertices");
        n=sc.nextInt();
        int c[][]=new int[50][50];
        int d[]=new int[20];
        System.out.println("Enter the weighted matrix:");
        for(int i=1;i<=n;i++) {
            for(int j=1;j<=n;j++) {
                c[i][j]=sc.nextInt();
            }
        }
        System.out.println("Enter the source vertex:");
        s=sc.nextInt();
        for(int i=1;i<=n;i++)
            d[i]=99;
        d[s]=0;
        System.out.println("\n Distance vector before:{");
        for(int i=1;i<=n;i++)
            System.out.println(d[i]+" ");
```

```
System.out.println("\n");
for(int i=1; i<=n-1; i++)
{
  for(int u=1; u<=n; u++)
  {
    for(int v=1; v<=n; v++)
    {
      if (c[u][v]!=99)
      {
        if (d[v]>d[u]+c[u][v])
          d[v] = d[u]+c[u][v];
      }
    }
  }
}
for(int u=1; u<=n; u++) {
  for(int v=1; v<=n; v++) {
    if (d[v]>d[u]+c[u][v])
    {
      System.out.println("Negative weighted
                          cycle");
    }
  }
}
System.out.println("Distance vector is: \n");
for(int i=1; i<=n; i++)
    System.out.println(d[i]+" ");
System.out.println("\n");
}
}
```

Congestion control for Leaky Bucket Algorithm

Aim:- C program for congestion control using Leaky Bucket algorithm.

Source code:-

```
package pkg5;
import java.util.Scanner;
public class lekyb {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the bucket size:");
        int n = sc.nextInt();
        System.out.println("Enter the flow rate:");
        int r = sc.nextInt();
        System.out.println("Enter the number of
                time intervals:");
        int ti = sc.nextInt();
        int[] pktrcv = new int[ti];
        int time = 0;
        int[] pktleft = new int[ti];
        int[] pktdrop = new int[ti];
        for(int i = 0; i<ti; i++)
        {
            System.out.println("Enter the pkt size
            received for time interval" + (time=time+1));
            pktrcv[i] = sc.nextInt();
        }
```

```
int[] pktsent = new int[ti];
for(int i=0; i<ti; i++)
{
    if(pktrcv[i] > r)
    {
        pktsent[i] = r;
    }
    else if((pktrcv[i] + pktleft[i]) < r)
    {
        pktsent[i] = pktrcv[i] + pktleft[i];
    }
    else if((pktrcv[i] + pktleft[i]) > r)
    {
        pktsent[i] = r;
    }
    else
    {
        pktsent[i] = r;
    }
}
pktleft[0] = pktrcv[0] - pktsent[0];
for(int i=1; i<ti; i++)
{
    pktleft[i] = (pktleft[i-1] + pktrcv[i]) -
                 pktsent[i];
    if(pktleft[i] > n)
    {
        pktdrop[i] = pktleft[n] - n;
        pktleft[i] = n;
    }
```

```
        {
        System.out.println();
        System.out.println("Time \t pktreceived \t
          pktsent \t pktleft \t pktdropped :");
        System.out.println();
        for(int i=0; i<ti; i++)
         {
            System.out.print((i+1)+" \t \t "+
              pktrcv[i]+" \t \t"+pktsent[i]+" \t \t"
              +pktleft[i]+" \t \t"+pktdrop[i]+"
                  \t \t");
            System.out.println();
         }
      }
   }
```