PART-A

1. Implement three nodes point – to – point network with duplex links between them. Set the queue size, vary the bandwidth and find the number of packets dropped.

```
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 50
$ns queue-limit $n1 $n2 50
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
```

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

/*immediately after BEGIN should open braces '{'

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

Steps for execution:

```
amc@amc-p2-1274il:~/Desktop/NS2/day1$ gedit 1.tcl
amc@amc-p2-1274il:~/Desktop/NS2/day1$ ns 1.tcl
```

Note:

1. Set the queue size fixed from n0 to n2 as 10, n1-n2 to 10 and from n2-n3 as 5.
   Syntax: To set the queue size
   $ns set queue-limit <from> <to> <size> Eg:

   $ns set queue-limit $n0 $n2 10

2. Go on varying the bandwidth from 10, 20 30 . . and find the number of packets dropped at the node 2

Trace file contains 12 columns:-

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

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.

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

set tf [ open lab2.tr w ]
$ns trace-all $tf

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

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

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

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

set p3 [new Agent/Ping]
$ns attach-agent $n2 $p3
$p3 set packetSize_ 30000
$p3 set interval_ 0.00001

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

set p5 [new Agent/Ping]
$ns attach-agent $n5 $p5
$ns queue-limit $n0 $n4 5
$ns queue-limit $n2 $n4 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 "gedit 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:

```
amc@amc-p2-1274il:~/Desktop/NS2/day1$ gedit 2.tcl

amc@amc-p2-1274il:~/Desktop/NS2/day1$ ns 2.tcl
```

```
node 2 received answer from 3 with round trip time 5.3 msec
node 0 received answer from 5 with round trip time 804.9 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 2 received answer from 3 with round trip time 5.3 msec
node 0 received answer from 5 with round trip time 804.9 msec
node 2 received answer from 3 with round trip time 5.3 msec

amc@amc-p2-1274il:~/Desktop/NS2/day1$ awk -f lab2.awk lab2.tr

The number of packets dropped =20

amc@amc-p2-1274il:~/Desktop/NS2/day1$
```

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

```
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]
$n1 color "red"

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

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

set n4 [$ns node]
$n4 shape square

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

$ns make-lan "$n0 $n1 $n2 $n3 $n4" 50Mb 100ms LL Queue/DropTail Mac/802_3

$ns duplex-link $n4 $n5 1Mb 1ms DropTail
$ns duplex-link-op $n4 $n5 orient right
```

```
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 sink0 [new Agent/TCPSink]
$ns attach-agent $n5 $sink0
$ns connect $tcp0 $sink0
set tcp1 [new Agent/TCP]
$ns attach-agent $n2 $tcp1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ftp1 set packetSize_ 600
$ftp1 set interval_ 0.001

set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp1 $sink1

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

set file2 [open file2.tr w]
$tcp1 attach $file2

$tcp0 trace cwnd_
$tcp1 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"
$ns at 0.2 "$ftp1 start"
$ns at 8 "$ftp1 stop"
$ns at 14 "$ftp0 stop"
$ns at 10 "$ftp1 start"
$ns at 15 "$ftp1 stop"
$ns at 16 "finish"

$ns run
```

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

cwnd:- means congestion window

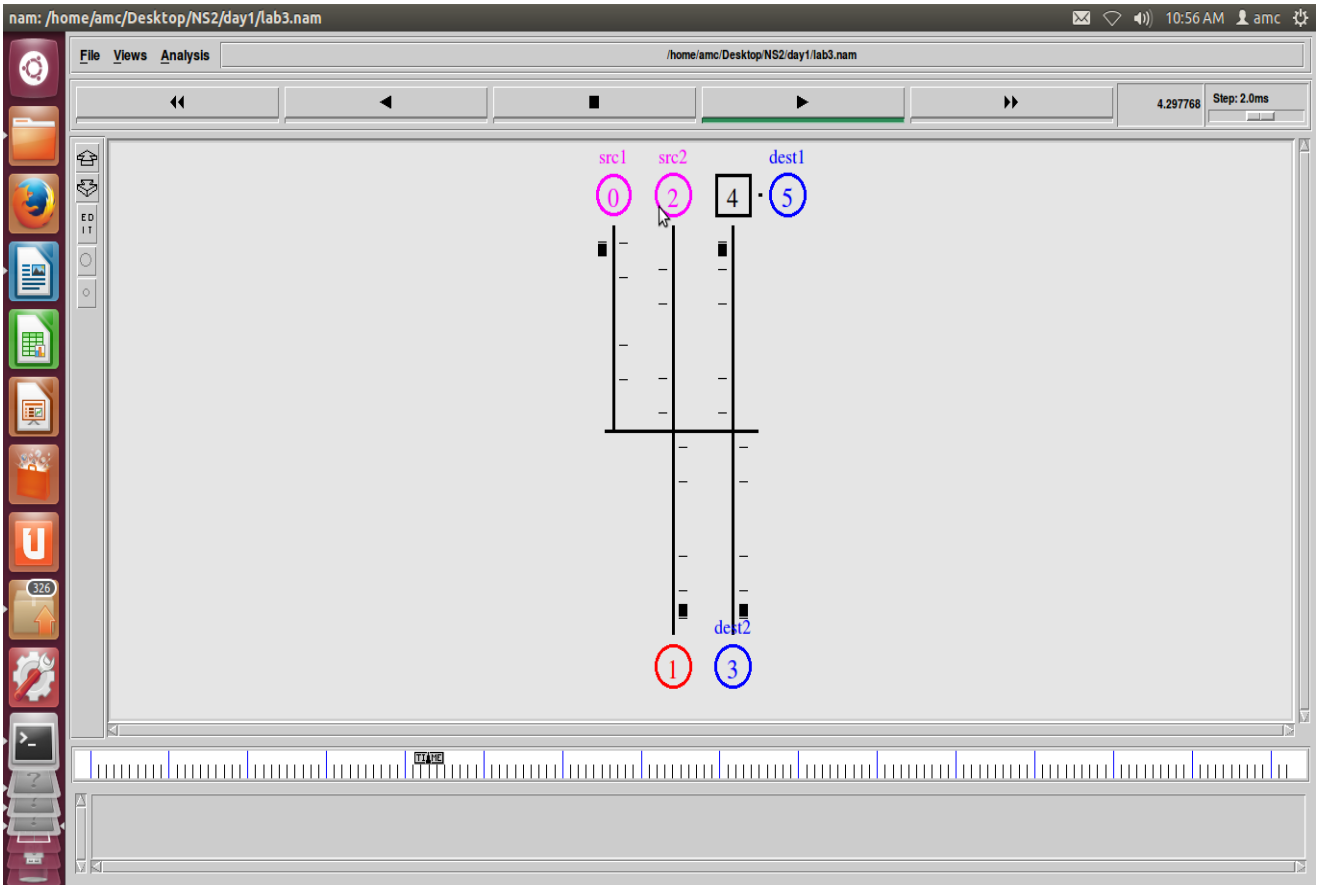
```
BEGIN {
}
{
if($6= ="cwnd_") /* don't leave space after writing cwnd_ */
 printf("%f\t%f\t\n",$1,$7); /* you must put \n in printf */
}
END {
}
```

Steps for execution:

amc@amc-p2-1274il:~/Desktop/NS2/day1$ gedit 3.tcl

amc@amc-p2-1274il:~/Desktop/NS2/day1$ ns 3.tcl

TOPOLOGY:

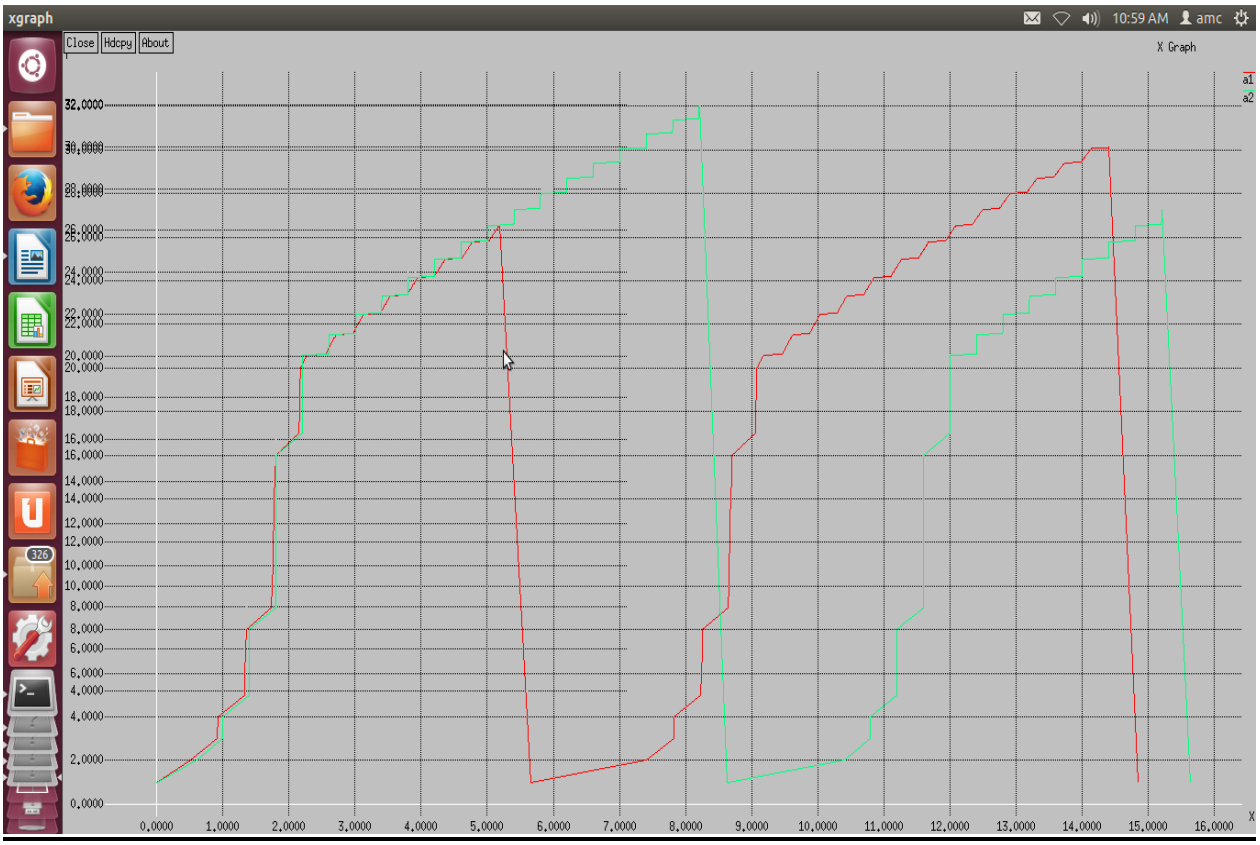


amc@amc-p2-1274il:~/Desktop/NS2/day1$ awk -f cwd.awk file1.tr >a1

amc@amc-p2-1274il:~/Desktop/NS2/day1$ awk -f cwd.awk file2.tr >a2

amc@amc-p2-1274il:~/Desktop/NS2/day1$ xgraph a1 a2

OUTPUT:



amc@amc-p2-1274il:~/Desktop/NS2/day1$

4. **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/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 (Open a new editor using "gedit command" and write awk file and save with ".awk" extension)

```
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)));
}
```

Steps for execution:

amc@amc-p2-1274il:~/Desktop/NS2/day1$ gedit 4.tcl

amc@amc-p2-1274il:~/Desktop/NS2/day1$ ns 4.tcl

OUTPUT:

amc@amc-p2-1274il:~/Desktop/NS2/day1$ awk -f lab4.awk lab4.tr

The Throughput from n0 to n1: 5863.442245 Mbps

The Throughput from n1 to n2: 1307.611834 Mbps

amc@amc-p2-1274il:~/Desktop/NS2/day1$

```
# General Parameters
set opt(title) zero       ;
set opt(stop) 100         ;# Stop time.
set opt(ecn) 0            ;
# Topology
set opt(type) gsm         ;#type of link:
set opt(secondDelay) 55         ;# average delay of access links in ms
# AQM parameters
set opt(minth) 30         ;
set opt(maxth) 0          ;
set opt(adaptive) 1       ;# 1 for Adaptive RED, 0 for plain RED
# Traffic generation.
set opt(flows) 0          ;# number of long-lived TCP flows
set opt(window) 30        ;# window for long-lived traffic
set opt(web) 2            ;# number of web sessions
# Plotting statistics.
set opt(quiet)   0        ;# popup anything?
set opt(wrap)    100      ;# wrap plots?
set opt(srcTrace) is      ;# where to plot traffic
set opt(dstTrace) bs2     ;# where to plot traffic
set opt(gsmbuf) 10                ; # buffer size for gsm

#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
#default buffer size in packets
set buf(gsm)    10

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 10ms 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 50ms DropTail
  puts "Cell Topology"
}
proc set_link_params {t} {
```

```
  global ns nodes bwUL bwDL propUL propDL buf
  $ns bandwidth $nodes(bs1) $nodes(ms) $bwDL($t) simplex
  $ns bandwidth $nodes(ms) $nodes(bs1) $bwUL($t) simplex
  $ns bandwidth $nodes(bs2) $nodes(ms) $bwDL($t) simplex
  $ns bandwidth $nodes(ms) $nodes(bs2) $bwUL($t) simplex
  $ns delay $nodes(bs1) $nodes(ms) $propDL($t) simplex
  $ns delay $nodes(ms) $nodes(bs1) $propDL($t) simplex
  $ns delay $nodes(bs2) $nodes(ms) $propDL($t) simplex
  $ns delay $nodes(ms) $nodes(bs2) $propDL($t) simplex
  $ns queue-limit $nodes(bs1) $nodes(ms) $buf($t)
  $ns queue-limit $nodes(ms) $nodes(bs1) $buf($t)
  $ns queue-limit $nodes(bs2) $nodes(ms) $buf($t)
  $ns queue-limit $nodes(ms) $nodes(bs2) $buf($t)
}
# RED and TCP parameters
Queue/RED set summarystats_ true
Queue/DropTail set summarystats_ true
Queue/RED set adaptive_ $opt(adaptive)
Queue/RED set q_weight_ 0.0
Queue/RED set thresh_ $opt(minth)
Queue/RED set maxthresh_ $opt(maxth)
Queue/DropTail set shrink_drops_ true
Agent/TCP set ecn_ $opt(ecn)
Agent/TCP set window_ $opt(window)
DelayLink set avoidReordering_ true

source web.tcl

#Create topology
switch $opt(type) {
 gsm -
 gprs -
 umts {cell_topo}
}
  set_link_params $opt(type)
  $ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]
  $ns insert-delayer $nodes(bs1) $nodes(ms) [new Delayer]
  $ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]
  $ns insert-delayer $nodes(bs2) $nodes(ms) [new Delayer]


# Set up forward TCP connection
if {$opt(flows) == 0} {
        set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
        set ftp1 [[set tcp1] attach-app FTP]
        $ns at 0.8 "[set ftp1] start"
}
if {$opt(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($opt(dstTrace)) id]
        if {$opt(srcTrace) == "is"} {
                set a "-a out.tr"
        } else {
                set a "out.tr"
        }
        set GETRC "../../../bin/getrc"
      set RAW2XG "../../../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 $GETRC -s $sid -d $did -f 1 out.tr | \
        $RAW2XG -s 0.01 -m $wrap -r >> plot.xgr
      exec $GETRC -s $did -d $sid -f 1 out.tr  | \
        $RAW2XG -s 0.01 -m $wrap -a  >> plot.xgr

        exec ./xg2gp.awk plot.xgr
      if {!$opt(quiet)} {
          exec xgraph -bb -tk -nl -m -x time -y packets plot.xgr &
      }
        exit 0
}
$ns at $opt(stop) "stop"
$ns run
```

Steps for execution:

amc@amc-p2-1274il:~$ cd ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts/

amc@amc-p2-1274il:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ gedit mtp-gsm.tcl

amc@amc-p2-1274il:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ ns mtp-gsm.tcl

Cell Topology

amc@amc-p2-1274il:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$

```
# General Parameters
set opt(title) zero       ;
set opt(stop) 100         ;# Stop time.
set opt(ecn) 0            ;
# Topology
set opt(type) umts        ;#type of link:
set opt(secondDelay) 55         ;# average delay of access links in ms
# AQM parameters
set opt(minth) 30         ;
set opt(maxth) 0          ;
set opt(adaptive) 1       ;# 1 for Adaptive RED, 0 for plain RED
# Traffic generation.
set opt(flows) 0          ;# number of long-lived TCP flows
set opt(window) 30        ;# window for long-lived traffic
set opt(web) 2            ;# number of web sessions
# Plotting statistics.
set opt(quiet)   0        ;# popup anything?
set opt(wrap)    100      ;# wrap plots?
set opt(srcTrace) is      ;# where to plot traffic
set opt(dstTrace) bs2     ;# where to plot traffic
set opt(umtsbuf) 10               ; # buffer size for umts

#default downlink bandwidth in bps
set bwDL(umts) 384000
#default uplink bandwidth in bps
set bwUL(umts) 64000
#default downlink propagation delay in seconds
set propDL(umts) .150
#default uplink propagation delay in seconds
set propUL(umts) .150
#default buffer size in packets
set buf(umts)   20

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 10ms 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 50ms DropTail
  puts "Cell Topology"
}
proc set_link_params {t} {
  global ns nodes bwUL bwDL propUL propDL buf
```

```
  $ns bandwidth $nodes(bs1) $nodes(ms) $bwDL($t) simplex
  $ns bandwidth $nodes(ms) $nodes(bs1) $bwUL($t) simplex
  $ns delay $nodes(bs1) $nodes(ms) $propDL($t) simplex
  $ns delay $nodes(ms) $nodes(bs1) $propDL($t) simplex
  $ns queue-limit $nodes(bs1) $nodes(ms) $buf($t)
  $ns queue-limit $nodes(ms) $nodes(bs1) $buf($t)
  $ns bandwidth $nodes(bs2) $nodes(ms) $bwDL($t) simplex
  $ns bandwidth $nodes(ms) $nodes(bs2) $bwUL($t) simplex
  $ns delay $nodes(bs2) $nodes(ms) $propDL($t) simplex
  $ns delay $nodes(ms) $nodes(bs2) $propDL($t) simplex
  $ns queue-limit $nodes(bs2) $nodes(ms) $buf($t)
  $ns queue-limit $nodes(ms) $nodes(bs2) $buf($t)
}

# RED and TCP parameters
Queue/RED set summarystats_ true
Queue/DropTail set summarystats_ true
Queue/RED set adaptive_ $opt(adaptive)
Queue/RED set q_weight_ 0.0
Queue/RED set thresh_ $opt(minth)
Queue/RED set maxthresh_ $opt(maxth)
Queue/DropTail set shrink_drops_ true
Agent/TCP set ecn_ $opt(ecn)
Agent/TCP set window_ $opt(window)
DelayLink set avoidReordering_ true

source web.tcl

#Create topology
switch $opt(type) {
 umts {cell_topo}
}
  set_link_params $opt(type)
  $ns insert-delayer $nodes(ms) $nodes(bs1) [new Delayer]
  $ns insert-delayer $nodes(bs1) $nodes(ms) [new Delayer]
  $ns insert-delayer $nodes(ms) $nodes(bs2) [new Delayer]
  $ns insert-delayer $nodes(bs2) $nodes(ms) [new Delayer]

# Set up forward TCP connection
if {$opt(flows) == 0} {
        set tcp1 [$ns create-connection TCP/Sack1 $nodes(is) TCPSink/Sack1 $nodes(lp) 0]
        set ftp1 [[set tcp1] attach-app FTP]
        $ns at 0.8 "[set ftp1] start"
}
if {$opt(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($opt(dstTrace)) id]
        if {$opt(srcTrace) == "is"} {
                set a "-a out.tr"
        } else {
                set a "out.tr"
        }
        set GETRC "../../../bin/getrc"
      set RAW2XG "../../../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 $GETRC -s $sid -d $did -f 1 out.tr | \
        $RAW2XG -s 0.01 -m $wrap -r >> plot.xgr
      exec $GETRC -s $did -d $sid -f 1 out.tr  | \
        $RAW2XG -s 0.01 -m $wrap -a  >> plot.xgr

        exec ./xg2gp.awk plot.xgr
      if {!$opt(quiet)} {
          exec xgraph -bb -tk -nl -m -x time -y packets plot.xgr &
      }
        exit 0
}
$ns at $opt(stop) "stop"
$ns run
```

amc@amc-p2-1274il:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ gedit mtp-umts.tcl

amc@amc-p2-1274il:~/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts$ ns mtp-umts.tcl

Cell Topology

SOURCE CODE:

```
import java.util.*;
import java.io.*;
public class CRC
{
   char t[]=new char[200];
   char cs[]=new char[200];
   char g[]=new char[200];
   int a,e,c;

   void xor()
   {
      for(int i=1;i<17;i++)
         cs[i]=((cs[i]==g[i])?'0':'1');
   }
   void crc()
   {
      for(e=0;e<17;e++)
         cs[e]=t[e];
      do
      {
         if(cs[0]=='1')
            xor();
         for(c=0;c<16;c++)
            cs[c]=cs[c+1];
            cs[c]=t[e++];
      }while(e<=a+16);
   }
   void operation()
   {

      Scanner read=new Scanner(System.in);
      String msg;
      String gs="10001000000100001";
      for(int i=0;i<gs.length();i++)
      g[i]=gs.charAt(i);

      System.out.println("enter the polynomial");
      msg=read.next();
      for(int i=0;i<msg.length();i++)
         t[i]=msg.charAt(i);

      System.out.print("\n generating polynomial is=");
      for(int i=0;i<gs.length();i++)
      System.out.print(g[i]);

      a=msg.length();
      for(e=a;e<a+16;e++)
         t[e]='0';
      System.out.print(" \n Modified message is=");
      for(int i=0;i<msg.length()+16;i++)
      System.out.print(t[i]);
```

```
        crc();

        System.out.println("\n Checksum is:");
        for(int i=0;i<16;i++)
            System.out.print(cs[i]);
        for(e=a;e<a+16;e++)
            t[e]=cs[e-a];
       System.out.println("\n final codeword is:");

       for(int i=0;i<a+16;i++)
        System.out.print(t[i]);
       System.out.println("\nTest error detection 0(yes)/1(no):");
       e=read.nextInt();
       if(e==0)
       {
          System.out.println("\nenter the position where error is to be inserted:");
           e=read.nextInt();
           t[e]=(t[e]=='0')?'1':'0';

            System.out.println("errornous data:")   ;
            for(int i=0;i<a+16;i++)
                System.out.print(t[i]);
        }
         crc();
        for(e=0;(e<16)&&(cs[e]!='1');e++);
        if(e<16)
        System.out.println("error detected");
        else
        System.out.println("no error detected");
    }
    public static void main(String[] args)
    {
      CRC ob=new CRC();
      ob.operation();
    }

}
```

Output:

Run1:

```
enter the polynomial
1011101

 generating polynomial is=10001000000100001
 Modified message is=10111010000000000000000
 Checksum is:
1000101101011000
 final codeword is:
10111011000101101011000
Test error detection 0(yes)/1(no):
0

enter the position where error is to be inserted:
```

Source code:

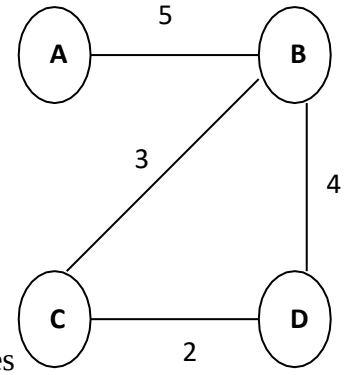
```
import java.util.Scanner;
public class BellmanFord
{
private final int D[];
private final int num_ver;
public static final int MAX_VALUE = 999;
        public BellmanFord(int num_ver)
        {
                this.num_ver = num_ver;
           D = new int[num_ver + 1];
        }

        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();
        }
}
```

Output:

Run1:




```
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

Run2:

```
Enter the number of vertices
4
Enter the adjacency matrix
0 5 0 0
5 0 -2 4
0 -3 0 -5
0 4 2 0
Enter the source vertex
2
The Graph contains negative egde cycle
The Graph contains negative egde cycle
The Graph contains negative egde cycle
The Graph contains negative egde cycle
distance of source 2 to 1 is -5
distance of source 2 to 2 is -15
distance of source 2 to 3 is -15
distance of source 2 to 4 is -17
```

9. 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: Server

```
import java.net.*;
import java.io.*;
public class tcps
{
	public static void main(String args[]) throws IOException
	{
		ServerSocket sersock=new ServerSocket(5000);
		System.out.println("server ready for connection");
		Socket sock=sersock.accept();
		System.out.println("connection is successful");
		InputStream istream=sock.getInputStream();
		BufferedReader fileRead=new BufferedReader(new InputStreamReader(istream));
		String fname=fileRead.readLine();
		BufferedReader contentRead=new BufferedReader(new FileReader(fname));
		OutputStream ostream=sock.getOutputStream();
		PrintWriter pwrite=new PrintWriter(ostream,true);
		String str;
		while((str=contentRead.readLine())!=null)
		{
```

```
                pwrite.println(str);
            }

            sock.close();
            sersock.close();
            pwrite.close();
            fileRead.close();
            contentRead.close();
        }
}
```

Source Code: Client

```
import java.net.*;
import java.io.*;
public class tcpc
{
        public static void main(String args[]) throws IOException
        {
              Socket sock=new Socket("127.0.0.1",5000);
            System.out.println("Enter the File Name");
            BufferedReader keyRead=new BufferedReader(new InputStreamReader(System.in));
            String fname=keyRead.readLine();
            OutputStream ostream=sock.getOutputStream();
            PrintWriter pwrite=new PrintWriter(ostream,true);
            System.out.println();
            pwrite.println(fname);
            InputStream istream=sock.getInputStream();
            BufferedReader socketRead=new BufferedReader(new InputStreamReader(istream));
            String str;
            while((str=socketRead.readLine())!=null)
            {
                System.out.println(str);
            }
            sock.close();
            pwrite.close();
            keyRead.close();
            socketRead.close();
        }
}
```

output:

serverside

```
amc@amc-p2-1274il:~$ gedit abc.txt
amc@amc-p2-1274il:~$ gedit tcps.java
amc@amc-p2-1274il:~$ javac tcps.java
amc@amc-p2-1274il:~$ java tcps
server ready for connection
connection is successful
amc@amc-p2-1274il:~$
```

clientside

```
amc@amc-p2-1274il:~$ javac tcpc.java
amc@amc-p2-1274il:~$ java tcpc
Enter the File Name
abc.txt
computer network lab
Information Science and Engg
amc@amc-p2-1274il:~$
```

10. 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: Server

```
import java.io.*;
 import java.net.*;
public class UDPS
{
        public static void main(String[] args)
        {
                DatagramSocket skt=null;
                try
                {
                skt=new DatagramSocket(6789);
                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)
                {
                }
        }
}
```

Source Code: Client

```
import java.io.*;
```

```
import java.net.*;
public class UDPC
{
	public static void main(String[] args)
	{
		DatagramSocket skt; try
		{
			skt=new DatagramSocket();
			String msg= "network lab ";
			byte[] b = msg.getBytes();
			InetAddress host=InetAddress.getByName("127.0.0.1");
			int serverSocket=6789;
			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)
		{
		}
	}
}
```

Output:

serverside

```
amc@amc-p2-1274il:~/Desktop/NS2/day2/java$ gedit UDPS.java
amc@amc-p2-1274il:~/Desktop/NS2/day2/java$ javac UDPS.java
amc@amc-p2-1274il:~/Desktop/NS2/day2/java$ java UDPS
```

clientside

```
amc@amc-p2-1274il:~/Desktop/NS2/day2/java$ gedit UDPC.java
amc@amc-p2-1274il:~/Desktop/NS2/day2/java$ javac UDPC.java
amc@amc-p2-1274il:~/Desktop/NS2/day2/java$ java UDPC
client received:lab server processed
```

```
import java.util.*;
import java.io.*;

public class RSA
{

   public static int mul(int x, int y, int n)
   {
      int k=1,j;
      for(j=1;j<=y;j++)
         k=(k*x)%n;
      return k;
   }

   public static void main(String[] args)
   {
      String msg;
      int pt[]=new int[100];
      int ct[]=new int[100];
      int n, d, e, p, q,i;
      Scanner read=new Scanner(System.in);
      System.out.println("enter the message to encrypt:");
      msg=read.next();
      for(i=0;i<msg.length();i++)
        pt[i]=msg.charAt(i);
          n=253;d=17;e=13;
      System.out.println("\n cipher text is=");
      for(i=0;i<msg.length();i++)
         ct[i]=mul(pt[i],e,n);
      for(i=0;i<msg.length();i++)
          System.out.print(" "+ ct[i]);
      for(i=0;i<msg.length();i++)
         pt[i]=mul(ct[i],d,n);
      System.out.println("\ndecrypted message is= ");
      for(i=0;i<msg.length();i++)
         System.out.print("    "+(char)pt[i]);

   }

}
```

Output:

```
enter the message to encrypt:
AMCEC

 cipher text is=
 76 110 111 115 111
decrypted message is=
   A    M    C    E    C
```

```
import java.util.*;
 public class Bucket
 {
    static void solution(int pktsize, int output)
    {
       int buketsize=512;
       if(pktsize>buketsize)
       {
          System.out.println("Bucket overflow");
       }
       else
       {
          while(pktsize>output)
          {
            System.out.println(output+"bytes outputed");
            pktsize=pktsize-output;
          }
          if(pktsize>0)
          {
            System.out.println( pktsize+"bytes outputed");
          }
       }
    }

    public static void main(String[] args)
    {
      int output,pktsize,n;
      Scanner read=new Scanner(System.in);
      Random rand=new Random();
      System.out.println( "Enter output rate");
      output=read.nextInt();
       System.out.println( "Enter the number of packets");
       n=read.nextInt();
       for(int i=1;i<=n;i++)
       {
          pktsize=rand.nextInt(1000);
          System.out.println( "packetno:"+i+"packetsize="+pktsize);
          solution(pktsize,output);
       }
    }

}
```

Output:

```
Enter output rate
50
Enter the number of packets
5
packetno:1packetsize=417
50bytes outputed
```

50bytes outputed
50bytes outputed
50bytes outputed
50bytes outputed
50bytes outputed
50bytes outputed
50bytes outputed
17bytes outputed
packetno:2packetsize=917
Bucket overflow
packetno:3packetsize=866
Bucket overflow
packetno:4packetsize=721
Bucket overflow
packetno:5packetsize=20
20bytes outputed