10a

```
#Create Simulator

set ns [new Simulator]


#Open Trace file and NAM file

set ntrace [open prog1.tr w]

$ns trace-all $ntrace

set namfile [open prog1.nam w]

$ns namtrace-all $namfile


#Finish Procedure

proc Finish {} {

global ns ntrace namfile


#Dump all the trace data and close the files

$ns flush-trace

close $ntrace

close $namfile


#Execute the nam animation file

exec nam prog1.nam &


#Show the number of packets dropped

exec echo "The number of packet drops is " &

exec grep -c "^d" prog1.tr &

exit 0

}


#Create 3 nodes

set n0 [$ns node]
```

```
set n1 [$ns node]

set n2 [$ns node]


#Label the nodes

$n0 label "TCP Source"

$n2 label "Sink"


#Set the color

$ns color 1 blue


#Create Links between nodes

#You need to modify the bandwidth to observe the variation in packet drop

$ns duplex-link $n0 $n1 1Mb 50ms DropTail

$ns duplex-link $n1 $n2 1Mb 50ms DropTail


#Make the Link Orientation

$ns duplex-link-op $n0 $n1 orient right

$ns duplex-link-op $n1 $n2 orient right


#Set Queue Size

#You can modify the queue length as well to observe the variation in packet drop

$ns queue-limit $n0 $n1 5

$ns queue-limit $n1 $n2 2


#Set up a Transport layer connection.

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set sink0 [new Agent/TCPSink]

$ns attach-agent $n2 $sink0

$ns connect $tcp0 $sink0
```

```
#Set up an Application layer Traffic

set cbr0 [new Application/Traffic/CBR]

$cbr0 set type_ CBR

$cbr0 set packetSize_ 100

$cbr0 set rate_ 2Mb

$cbr0 set random_ false

$cbr0 attach-agent $tcp0

$tcp0 set class_ 1


#Schedule Events

$ns at 0.0 "$cbr0 start"

$ns at 5.0 "Finish"


#Run the Simulation

$ns run
```

10b

```
#Create Simulator

set ns [new Simulator]


#Use colors to differentiate the traffic

$ns color 1 Blue

$ns color 2 Red


#Open trace and NAM trace file

set ntrace [open prog2.tr w]

$ns trace-all $ntrace

set namfile [open prog2.nam w]

$ns namtrace-all $namfile


#Finish Procedure

proc Finish {} {

global ns ntrace namfile


#Dump all trace data and close the file

$ns flush-trace

close $ntrace

close $namfile


#Execute the nam animation file

exec nam prog2.nam &


#Find the number of ping packets dropped

puts "The number of ping packets dropped are "
```

```
exec grep "^d" prog2.tr | cut -d " " -f 5 | grep -c "ping" &

exit 0

}


#Create six nodes

for {set i 0} {$i < 6} {incr i} {

set n($i) [$ns node]

}


#Connect the nodes

for {set j 0} {$j < 5} {incr j} {

$ns duplex-link $n($j) $n([expr ($j+1)]) 0.1Mb 10ms DropTail

}


#Define the recv function for the class 'Agent/Ping'

Agent/Ping instproc recv {from rtt} {

$self instvar node_

puts "node [$node_ id] received ping answer from $from with round trip time $rtt


ms"

}


#Create two ping agents and attach them to n(0) and n(5)

set p0 [new Agent/Ping]

$p0 set class_ 1

$ns attach-agent $n(0) $p0


set p1 [new Agent/Ping]

$p1 set class_ 1

$ns attach-agent $n(5) $p1

$ns connect $p0 $p1
```

```
#Set queue size and monitor the queue

#Queue size is set to 2 to observe the drop in ping packets

$ns queue-limit $n(2) $n(3) 2

$ns duplex-link-op $n(2) $n(3) queuePos 0.5


#Create Congestion

#Generate a Huge CBR traffic between n(2) and n(4)

set tcp0 [new Agent/TCP]

$tcp0 set class_ 2

$ns attach-agent $n(2) $tcp0

set sink0 [new Agent/TCPSink]

$ns attach-agent $n(4) $sink0

$ns connect $tcp0 $sink0


#Apply CBR traffic over TCP

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize_ 500

$cbr0 set rate_ 1Mb

$cbr0 attach-agent $tcp0

#Schedule events

$ns at 0.2 "$p0 send"

$ns at 0.4 "$p1 send"

$ns at 0.4 "$cbr0 start"

$ns at 0.8 "$p0 send"

$ns at 1.0 "$p1 send"

$ns at 1.2 "$cbr0 stop"

$ns at 1.4 "$p0 send"

$ns at 1.6 "$p1 send"

$ns at 1.8 "Finish"
```

#Run the Simulation

$ns run

#Fork me!

11a

```
#Create Simulator

set ns [new Simulator]

#Use colors to differentiate the traffics

$ns color 1 Blue

$ns color 2 Red

#Open trace and NAM trace file

set ntrace [open prog5.tr w]

$ns trace-all $ntrace

set namfile [open prog5.nam w]

$ns namtrace-all $namfile

#Use some flat file to create congestion graph windows

set winFile0 [open WinFile0 w]

set winFile1 [open WinFile1 w]

#Finish Procedure

proc Finish {} {

#Dump all trace data and Close the files

global ns ntrace namfile

$ns flush-trace

close $ntrace

close $namfile

#Execute the NAM animation file

exec nam prog5.nam &

#Plot the Congestion Window graph using xgraph

exec xgraph WinFile0 WinFile1 &

exit 0

}

#Plot Window Procedure

proc PlotWindow {tcpSource file} {

global ns
```

```
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
# To plot graph over x and y axis
puts $file "$now $cwnd"
$ns at [expr $now+$time] "PlotWindow $tcpSource $file"
}
#Create 6 nodes
for {set i 0} {$i<6} {incr i} {
set n($i) [$ns node]
}
#Create duplex links between the nodes
$ns duplex-link $n(0) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 2Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 1.0Mb 100ms DropTail
#Nodes n(3) , n(4) and n(5) are considered in a LAN
set lan [$ns newLan "$n(3) $n(4) $n(5)" 0.5Mb 40ms LL Queue/DropTail MAC/802_3 Channel]
#Orientation to the nodes
$ns duplex-link-op $n(0) $n(2) orient right-down
$ns duplex-link-op $n(1) $n(2) orient right-up
$ns duplex-link-op $n(2) $n(3) orient right
#Setup queue between n(2) and n(3) and monitor the queue
$ns queue-limit $n(2) $n(3) 20
$ns duplex-link-op $n(2) $n(3) queuePos 0.5
#Set error model on link n(2) to n(3) (optional- to analyse the amt of drop removed pkts in tr file)
set loss_module [new ErrorModel]
$loss_module ranvar [new RandomVariable/Uniform]
$loss_module drop-target [new Agent/Null]
$ns lossmodel $loss_module $n(2) $n(3)
#Set up the TCP connection between n(0) and n(4)
```

```
set tcp0 [new Agent/TCP/Newreno]

$tcp0 set fid_ 1

$tcp0 set window_ 8000

$tcp0 set packetSize_ 552

$ns attach-agent $n(0) $tcp0

set sink0 [new Agent/TCPSink/DelAck]

$ns attach-agent $n(4) $sink0

$ns connect $tcp0 $sink0

#Apply FTP Application over TCP

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ftp0 set type_ FTP

#Set up another TCP connection between n(5) and n(1)

set tcp1 [new Agent/TCP/Newreno]

$tcp1 set fid_ 2

$tcp1 set window_ 8000

$tcp1 set packetSize_ 552

$ns attach-agent $n(5) $tcp1

set sink1 [new Agent/TCPSink/DelAck]

$ns attach-agent $n(1) $sink1

$ns connect $tcp1 $sink1

#Apply FTP application over TCP

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ftp1 set type_ FTP

#Schedule Events

$ns at 0.1 "$ftp0 start"

$ns at 0.1 "PlotWindow $tcp0 $winFile0"

$ns at 0.5 "$ftp1 start"

$ns at 0.5 "PlotWindow $tcp1 $winFile1"

$ns at 25.0 "$ftp0 stop"
```

```
$ns at 25.1 "$ftp1 stop"

$ns at 25.2 "Finish"

#Run the simulation

$ns run
```

11b

```
#filename.tcl


#Create a ns simulator

set ns [new Simulator]


#Setup topography object

set topo [new Topography]

$topo load_flatgrid 1500 1500


#Open the NS trace file

set tracefile [open p6.tr w]

$ns trace-all $tracefile


#Open the NAM trace file

set namfile [open p6.nam w]

$ns namtrace-all $namfile

$ns namtrace-all-wireless $namfile 1500 1500


#===================================

# Mobile node parameter setup

#===================================

$ns node-config -adhocRouting DSDV \

   -llType LL \

   -macType Mac/802_11 \

   -ifqType Queue/DropTail \

   -ifqLen 20 \

   -phyType Phy/WirelessPhy \

   -channelType Channel/WirelessChannel \

   -propType Propagation/TwoRayGround \
```

```
    -antType Antenna/OmniAntenna \
    -topoInstance $topo \
    -agentTrace ON \
    -routerTrace ON


#===================================
# Nodes Definition
#===================================
create-god 6
#Create 6 nodes
set n0 [$ns node]
$n0 set X_ 630
$n0 set Y_ 501
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 454
$n1 set Y_ 340
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 785
$n2 set Y_ 326
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 270
$n3 set Y_ 190
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
```

```
set n4 [$ns node]

$n4 set X_ 539

$n4 set Y_ 131

$n4 set Z_ 0.0

$ns initial_node_pos $n4 20

set n5 [$ns node]

$n5 set X_ 964

$n5 set Y_ 177

$n5 set Z_ 0.0

$ns initial_node_pos $n5 20


#=================================

# Agents Definition

#=================================

#Setup a UDP connection

set udp0 [new Agent/UDP]

$ns attach-agent $n0 $udp0

set null1 [new Agent/Null]

$ns attach-agent $n4 $null1

$ns connect $udp0 $null1

$udp0 set packetSize_ 1500


#Setup a TCP connection

set tcp0 [new Agent/TCP]

$ns attach-agent $n3 $tcp0

set sink1 [new Agent/TCPSink]

$ns attach-agent $n5 $sink1

$ns connect $tcp0 $sink1


#=================================

# Applications Definition
```

```
#==================================

#Setup a CBR Application over UDP connection

set cbr0 [new Application/Traffic/CBR]

$cbr0 attach-agent $udp0

$cbr0 set packetSize_ 1000

$cbr0 set rate_ 1.0Mb

$cbr0 set random_ null


#Setup a FTP Application over TCP connection

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0


#==================================

# Termination

#==================================

#Define a 'finish' procedure

proc finish {} {

global ns tracefile namfile

$ns flush-trace

close $tracefile

close $namfile
```

11b awk

```
#AWK file (filename.awk)

BEGIN{

count1=0

count2=0

pack1=0

pack2=0

time1=0
```

```
time2=0

}

{

if($1=="r" && $3=="_1_" && $4=="RTR")

{

count1++

pack1=pack1+$8

time1=$2

}

if($1=="r" && $3=="_2_" && $4=="RTR")

{

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 \n", ((count2*pack2*8)/(time2*1000000)));

}
```

12a

```
set opt(title) zero ;

set opt(stop) 100 ;

set opt(ecn) 0 ;

set opt(type) gsm ;

set opt(secondDelay) 55 ;

set opt(minth) 30 ;

set opt(maxth) 0 ;

set opt(adaptive) 1 ;

set opt(flows) 0 ;

set opt(window) 30 ;

set opt(web) 2 ;

set opt(quiet) 0 ;

set opt(wrap) 100 ;

set opt(srcTrace) is ;

set opt(dstTrace) bs2 ;

set opt(gsmbuf) 10 ;

set bwDL(gsm) 9600

set bwUL(gsm) 9600

set propDL(gsm) .500

set propUL(gsm) .500

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)
}
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

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]

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]

$tcp1 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 "/home/ubuntu/ns-allinone-2.35/ns-2.35/bin/getrc"

set RAW2XG "/home/ubuntu/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 $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 /home/ubuntu/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts/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

exec nam p6.nam &

exec echo "Number of packets dropped is : " &

exec grep -c "^D" p6.tr &
```

```
exit 0

}


$ns at 1.0 "$cbr0 start"

$ns at 2.0 "$ftp0 start"

$ns at 180.0 "$ftp0 stop"

$ns at 200.0 "$cbr0 stop"

$ns at 200.0 "finish"

$ns at 70 "$n4 setdest 100 60 20"

$ns at 100 "$n4 setdest 700 300 20"

$ns at 150 "$n4 setdest 900 200 20"

$ns run
```

12b

```
set opt(title) zero ;

set opt(stop) 100 ;

set opt(ecn) 0 ;


set opt(type) umts  ;

set opt(secondDelay) 55  ;



set opt(minth) 30  ;

set opt(maxth)  0  ;

set opt(adaptive) 1  ;


set opt(flows) 0  ;

set opt(window) 30  ;

set opt(web) 2  ;


set opt(quiet) 0  ;

set opt(wrap) 100  ;

set opt(srcTrace) is  ;


set opt(dstTrace) bs2  ;

set opt(umtsbf) 10  ;


set bwDL(umts) 384000


set bwUL(umts) 64000


set propDL(umts) .150
```

```
set propUL(umts) .150

set buf(umts) 20

set ns [new Simulator]
set tf [open 6.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)
}

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
```

web.tcl

```
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]


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 0.8 "[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]

$tcp1 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 6.tr"

} else {

  set a "6.tr"

}



set GETRC "/home/ubuntu/ns-allinone-2.35/ns-2.35/bin/getrc"

set RAW2XG "/home/ubuntu/ns-allinone-2.35/ns-2.35/bin/raw2xg"



exec $GETRC -s $sid -d $did -f 0 6.tr | \

$RAW2XG -s 0.01 -m $wrap -r > plot.xgr

exec $GETRC -s $did -d $sid -f 0 6.tr | \

$RAW2XG -a -s 0.01 -m $wrap -r >> plot.xgr



exec $GETRC -s $sid -d $did -f 1 6.tr | \

$RAW2XG -s 0.01 -m $wrap -r >>plot.xgr

exec $GETRC -s $did -d $sid -f 1 6.tr | \

$RAW2XG -s 0.01 -m $wrap -a >> plot.xgr



exec /home/ubuntu/ns-allinone-2.35/ns-2.35/tcl/ex/wireless-scripts/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
```