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 10

$ns queue-limit $n1 $n2 10

set udp0 [new Agent/UDP] /\* Letters **A**,**U**,**D** and **P** are capital \*/

$ns attach-agent $n0 $udp0

set cbr0 [new Application/Traffic/CBR] /\* **A**,**T**,**C**,**B** and **R** are capital\*/

$cbr0 set packetSize\_ 500 /\***S** is capital, space after underscore\*/

$cbr0 set interval\_ 0.005

$cbr0 attach-agent $udp0

set udp1 [new Agent/UDP]

$ns attach-agent $n1 $udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 attach-agent $udp1 set udp2 [new Agent/UDP]

$ns attach-agent $n2 $udp2

set cbr2 [new Application/Traffic/CBR]

$cbr2 attach-agent $udp2

set null0 [new Agent/Null] /\* **A** and **N** are capital \*/

$ns attach-agent $n3 $null0

$ns connect $udp0 $null0

$ns connect $udp1 $null0

$ns at 0.1 "$cbr0 start"

$ns at 0.2 "$cbr1 start"

$ns at 1.0 "finish"

$ns run

### AWK file (Open a new editor using “vi command” and write awk file and save with “.awk” extension)

**/\*immediately after BEGIN should open braces ‘{‘**

BEGIN { c=0;

}

{

If ($1= ="d")

{

c++;

printf("%s\t%s\n",$5,$11);

}

}

### /\*immediately after END should open braces ‘{‘

END{

printf("The number of packets dropped =%d\n",c);

}

### 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 “vi command” and write awk file and save with “.awk” extension)

BEGIN{

drop=0;

}

{

if($1= ="d" )

{

drop++;

}

}

END{

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

}

### 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] set n2 [$ns node]

$n2 color "magenta"

$n2 label "src2" set n3 [$ns node]

$n3 color "blue"

$n3 label "dest2" set n4 [$ns node] set n5 [$ns node]

$n5 color "blue"

$n5 l

abel "dest1"

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

### /\* should come in single line \*/

$ns duplex-link $n4 $n5 1Mb 1ms DropTail

set tcp0 [new Agent/TCP]

$ns attach-agent $n0 $tcp0

set ftp0 [new Application/FTP]

$ftp0 attach-agent $tcp0

$ftp0 set packetSize\_ 500

$ftp0 set interval\_ 0.0001

set sink5 [new Agent/TCPSink]

$ns attach-agent $n5 $sink5

$ns connect $tcp0 $sink5 set tcp2 [new Agent/TCP]

$ns attach-agent $n2 $tcp2

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

$ftp2 set packetSize\_ 600

$ftp2 set interval\_ 0.001

set sink3 [new Agent/TCPSink]

$ns attach-agent $n3 $sink3

$ns connect $tcp2 $sink3 set file1 [open file1.tr w]

$tcp0 attach $file1

set file2 [open file2.tr w]

$tcp2 attach $file2

$tcp0 trace cwnd\_ /\* must put **underscore** ( \_ ) after **cwnd and no space between them**\*/

$tcp2 trace cwnd\_

proc finish { } { global ns nf tf

$ns flush-trace close $tf

close $nf

exec nam lab3.nam & exit 0

}

$ns at 0.1 "$ftp0 start"

$ns at 5 "$ftp0 stop"

$ns at 7 "$ftp0 start"

$ns at 0.2 "$ftp2 start"

$ns at 8 "$ftp2 stop"

$ns at 14 "$ftp0 stop"

$ns at 10 "$ftp2 start"

$ns at 15 "$ftp2 stop"

$ns at 16 "finish"

$ns run

### AWK file (Open a new editor using “vi command” and write awk file and save with “.awk” extension)

**cwnd:- means congestion window**

BEGIN {

}

{

if($6= ="cwnd\_") /\* don’t leave space after writing **cwnd\_** \*/ printf("%f\t%f\t\n",$1,$7); /\* you must put **\n** in printf \*/

} END {

}

1. **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 “vi 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)));

}

1. **Implement and study the performance of GSM on NS2/NS3 (Using MAC layer) or equivalent environment.**

**Source Code:**

# General Parameters

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 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(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 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 " 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

source web.tcl

#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 nf

set wrap $opt(wrap)

set sid [$nodes($opt(srcTrace)) id] set did [$nodes($opt(dstTrace)) id] 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 xgraph -x time -y packets plot.xgr &

exit 0

}

$ns at $stop "stop"

$ns run

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

**Source Code:**

# General Parameters

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 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 "../../../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 xgraph -x time -y packets plot.xgr &

exit 0

}

$ns at $stop "stop"

$ns run