PART A

Experiment No:1 **THREE NODE POINT TO POINT NETWORK**

Aim: *Simulate a three node point to point network with duplex links between them. Set queue size and vary the bandwidth and find number of packets dropped.*

Program:

```
set ns [new Simulator]
set f [open lab1.tr w]
set nf [open lab1.nam w]
$ns trace-all $f
$ns namtrace-all $nf

proc finish {} {
    global f nf ns
    $ns flush-trace
    close $f
    close $nf
    exec nam lab1.nam &
    exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$n0 label "TCP Source"
$n1 label "UDP Source"
$n2 label "Sink"
$ns color 1 red
$ns color 2 yellow

$ns duplex-link $n0 $n1 1Mb 10ms DropTail
$ns duplex-link $n1 $n2 1Mb 20ms DropTail
$ns queue-limit $n1 $n2 10
```

```
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n1 $n2 orient right

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
set null0 [new Agent/Null]
$ns attach-agent $n2 $null0
$ns connect $udp0 $null0
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
set sink [new Agent/TCPSink]
$ns attach-agent $n2 $sink
$ftp0 set maxPkts_ 1000
$ns connect $tcp0 $sink

$udp0 set class_ 1
$tcp0 set class_ 2

$ns at 0.1 "$cbr0 start"
$ns at 1.0 "$ftp0 start"
$ns at 4.0 "$ftp0 stop"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

Steps for execution:

- *Open vi editor and type program. Program name should have the extension " **.tcl** "*
  [root@localhost ~]# vi lab1.tcl
- *Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.*
- *Run the simulation program*
  [root@localhost~]# ns lab1.tcl
- *Here **"ns"** indicates network simulator. We get the topology shown in the network animator. Now press the play button in the simulation window and the simulation will begins.*
- *To calculate the number of packet dropped execute the following command.*
  [root@localhost~]# grep ^d lab1.tr| grep "cbr" -c
  [root@localhost~]# grep ^d lab1.tr | grep "tcp" -c

- *Write the value of number of packet dropped in observation sheet. Repeat the above step by changing the bandwidth to [***0.5Mb, 1Mb,1.5Mb, 2Mb***]to the following line of the program.*

  $ns duplex-link $n0 $n1 0.5Mb 10ms DropTail
  $ns duplex-link $n1 $n2 0.5Mb 20ms DropTail
- *Plot a graph with x- axis with bandwidth and y-axis with number of packet dropped of TCP and UDP.*

Experiment No: 2 **TRANSMISSION OF PING MESSAGE**

Aim:*Simulate the transmission of ping messages over a network topology consisting of 6 nodes and find the number of packets dropped due to congestion.*

Program:

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

proc finish {} {
        global ns f nf
        $ns flush-trace
        close $f
        close $nf
        exec nam lab2.nam &
        exit 0
}
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$n0 label "ping0"
$n1 label "ping1"
$n2 label "R1"
$n3 label "R2"
$n4 label "ping4"
$n5 label "ping5"

$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 0.5Mb 30ms DropTail
$ns duplex-link  $n3 $n4 1Mb 10ms DropTail
$ns duplex-link  $n3 $n5 1Mb 10ms DropTail




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 sendPingPacket {} {
        global ns ping0 ping1 ping4 ping5
        set intervalTime 0.001
        set now [$ns now]
        $ns at [expr $now + $intervalTime] "$ping0 send"
        $ns at [expr $now + $intervalTime] "$ping1 send"
```

```
        $ns at [expr $now + $intervalTime] "$ping4 send"
        $ns at [expr $now + $intervalTime] "$ping5 send"

        $ns at [expr $now + $intervalTime] "sendPingPacket"
}

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

$ping0  set class_  1
$ping1  set class_  2
$ping4  set class_  4
$ping5  set class_  5
$ns at 0.01 "sendPingPacket"
$ns at 10.0 "finish"
$ns run
```

Steps for execution:

- *Open vi editor and type program. Program name should have the extension " **.tcl** "*
  [root@localhost ~]# vi lab2.tcl
- *Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.*
- *Run the simulation program*
  [root@localhost~]# ns lab2.tcl
- *Here **"ns"** indicates network simulator. We get the topology shown in the network animator. Now press the play button in the simulation window and the simulation will begin.*
- *To calculate the number of packet dropped. Execute the following command.*
  [root@localhost~]#grep ^d lab2.tr -c

- *Write the value of number of packet sent of TCP and UDP in observation sheet. Repeat the above step by changing the bandwidth to [***0.5Mb, 1Mb,1.5Mb, 2Mb***]to the following line of the program.*

  $ns duplex-link $n0 $n2 1Mb 10ms DropTail
  $ns duplex-link $n1 $n2 1Mb 10ms DropTail

$ns duplex-link $n2 $n3 0.5Mb 30ms DropTail
$ns duplex-link $n3 $n4 1Mb 10ms DropTail
$ns duplex-link $n3 $n5 1Mb 10ms DropTail

- *Plot a graph with x- axis with bandwidth and y-axis with number of packet dropped due to ping.*

Experiment No: 3 **ETHERNET LAN USING N-NODES WITH MULTIPLE TRAFFIC**

Aim:*Simulate an Ethernet LAN using 'n' nodes and set multiple traffic nodes and plot congestion window for different source / destination.*

Program:

```
set ns [new Simulator]
set f [open lab3.tr w]
set nf [open lab3.nam w]
$ns trace-all $f
$ns namtrace-all $nf


proc finish {} {
        global ns f nf outFile1 outFile2
        $ns flush-trace
         close $f
        close $nf
        exec nam lab3.nam &
        exec xgraph Congestion1.xg -geometry 400x400 &
        exec xgraph Congestion2.xg -geometry 400x400 &
        exit 0
}
$ns color 1 red
$ns color 2 green


set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
set n6 [$ns node]
set n7 [$ns node]


$n0 label "TCP FTP Source"
```

```
$n3 label "Sink Destination"
$n5 label "TCP Telnet Source"
$n7 label "Sink Destination"

$ns make-lan "$n0 $n1 $n2 $n3 $n4 $n5 $n6 $n7" 10Mb 30ms LL
Queue/DropTail Mac/802_3

set tcp1 [new Agent/TCP]
$ns attach-agent $n0 $tcp1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n3 $sink1
$ns connect $tcp1 $sink1
$tcp1 set class_ 1

set tcp2 [new Agent/TCP]
$ns attach-agent $n5 $tcp2
set telnet1 [new Application/FTP]
$telnet1 attach-agent $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $n7 $sink2
$ns connect $tcp2 $sink2
$telnet1 set type_ $sink2
$tcp2 set class_ 2

set outFile1 [open Congestion1.xg w]
set outFile2 [open Congestion2.xg w]

puts $outFile1  "TitleText: Congestion Window Plot for TCP1"
puts $outFile1  "XUnitText: SimulationTime(Secs)"
puts $outFile1  "YUnitText: CongestionWindowSize"
```

```
puts $outFile2  "TitleText: Congestion Window Plot for TCP2"
puts $outFile2  "XUnitText: SimulationTime(Secs)"

 puts $outFile2 "YUnitText: CongestionWindowSize"

proc findWindowSize {tcpSource outFile} {
        global ns
        set now [$ns now]
        set cWindSize [$tcpSource set cwnd_]
        puts $outFile "$now $cWindSize"
        $ns at [expr $now + 0.1] "findWindowSize $tcpSource $outFile"
}

$ns at 0.0 "findWindowSize $tcp1 $outFile1"
$ns at 0.1 "findWindowSize $tcp2 $outFile2"
$ns at 0.3 "$ftp1 start"
$ns at 0.5 "$telnet1 start"
$ns at 50.0 "$ftp1 stop"
$ns at 50.0 "$telnet1 stop"
$ns at 50.0 "finish"
$ns run
```

Steps for execution:

- *Open vi editor and type program. Program name should have the extension " **.tcl** "*
  [root@localhost ~]# vi lab3.tcl
- *Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.*
- *Run the simulation program*
  [root@localhost~]# ns lab3.tcl
- *Here **"ns"** indicates network simulator. We get the topology shown in the network animator. Now press the play button in the simulation window and the simulation will begin.*
- *The xgraph automatically calculates and plot the two graph of Congestion window with TCP1 and TCP2.*

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:

```
if {$argc != 1} {
      error "Command: ns <ScriptName.tcl><Number_of_Nodes>"
      exit 0
}


#Define the simulation options
set val(chan)       Channel/WirelessChannel
set val(prop)       Propagation/TwoRayGround
set val(ant)        Antenna/OmniAntenna
set val(ll)         LL
set val(ifq)        Queue/DropTail/PriQueue
set val(ifqlen)     50
set val(netif)      Phy/WirelessPhy
set val(mac)        Mac/802_11
set val(rp)         AODV
set val(nn)         [lindex $argv 0]
set opt(x)          750
set opt(y)          750
set val(stop)       100

set ns [new Simulator]
set trfd [open lab4.tr w]
set namfd [open lab4.nam w]
$ns trace-all $trfd
$ns namtrace-all-wireless $namfd $opt(x) $opt(y)
set topo [new Topography]
$topo load_flatgrid $opt(x) $opt(y)
set god_ [create-god $val(nn)]
```

```
proc stop {} {
        global ns trfd namfd
        close $trfd
        close $namfd
        exec nam lab4.nam &
        exit 0
}



$ns node-config -adhocRouting $val(rp) \
                -llType $val(ll) \
                -macType $val(mac) \
                -ifqType $val(ifq) \
                -channelType $val(chan) \
                -propType $val(prop) \
                -antType $val(ant) \
                -ifqLen $val(ifqlen) \
                -phyType $val(netif) \
                -topoInstance $topo \
                -agentTrace ON \
                -routerTrace ON \
                -macTrace OFF \
                 -movementTrace OFF
for {set i 0} {$i < $val(nn)} {incr i} {
        set n($i) [$ns node]
}

for {set i 0} {$i < $val(nn)} {incr i} {
        set XX [expr rand()*750]
        set YY [expr rand()*750]
        $n($i) set X_ $XX
        $n($i) set Y_ $YY
```

```
}

for {set i 0} {$i < $val(nn)} {incr i} {
        $ns initial_node_pos $n($i) 30
}
set tcp1 [new Agent/TCP]
$ns attach-agent $n(1) $tcp1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $n(3) $sink1
$ns connect $tcp1 $sink1
$ns at 0.0 "destination"
proc destination {} {
        global ns val n
        set now [$ns now]
        set time 5.0
        for {set i 0} {$i < $val(nn)} {incr i} {
                set XX [expr rand()*750]
                set YY [expr rand()*750]
                $ns at [expr $now + $time] "$n($i) setdest $XX $YY 20.0"
        }
        $ns at [expr $now + $time] "destination"
}
for {set i 0} {$i < $val(nn)} {incr i} {
        $ns at $val(stop) "$n($i) reset"
}

$ns at 5.0 "$ftp1 start"
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns run
```

Steps for execution:

- *Open vi editor and type program. Program name should have the extension " **.tcl** "*
  [root@localhost ~]# vi lab4.tcl
- *Save the program by pressing **"ESC key"** first, followed by **"Shift and :"** keys simultaneously and type **"wq"** and press **Enter key**.*
- *Run the simulation program*
  [root@localhost~]# ns lab4.tcl [no. of nodes]
- *Here **"ns"** indicates network simulator. We get the topology shown in the network animator. Now press the play button in the simulation window and the simulation will begin.*
- *To calculate the throughput. Execute the following command.*
  For calculating number of received packets
  [root@localhost~]#grep ^r lab6.tr | grep "AGT" | grep "tcp" | awk '{s+=$8}END{print s}'
  For calculating total time
  [root@localhost~]#grep ^r lab6.tr | grep "AGT" | grep "tcp" | awk '{s+=$2}END{print s}'

  Throughput = (Packet received/ Total Time) (bps)

- *Repeat the above step by changing the number of nodes for each output like [**4,8,12,16**] while running of the program.*
  [root@localhost~]# ns lab6.tcl [no. of nodes]

- *Plot a graph with x- axis with error rate and y-axis with throughput.*

Experiment 5 & 6:

Implement and study the performance of GSM & CDMA on NS3 (Using MAC layer) or equivalent Environment.

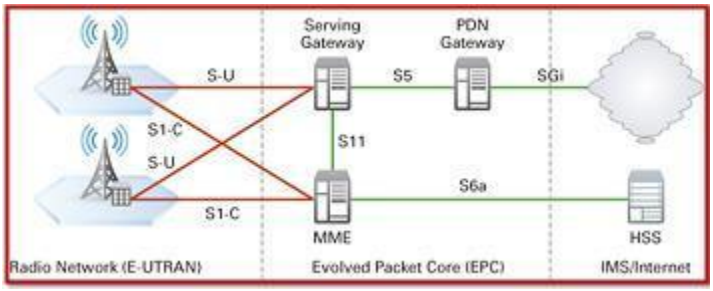
NS3 LTE Simulation

LTE is the latest high-speed cellular transmission network.LTE is a 4G technology with download speeds that run the gamut from 3 to 28 Mbps worldwide.4G LTE is one of several competing 4G standards along with Ultra Mobile Broadband (UMB) andWiMax (IEEE 802.16).Ns3 is the best choice among network simulator for simulating LTE framework.We provide ***customized NS3 LTE Simulation Projects*** based on customer Requirements.

Advantages of LTE:

- LTE will supports seamless connection to existing networks like GSM,CDMA and WCDMA.
- It has simple architecture because of low operating expenditure
- Time required for connecting network and is in range of a few hundred ms and power savings states can now be entered and exited very quickly
- High data rates can be achieved in both downlink as well as uplink.
- Both FDD and TDD can be used on same platform.
- Optimized signaling for connection establishment and other air interface and mobility management procedures have further improved the user experience.

Architecture of LTE:



LTE parameters:

- Transmission bandwidth.
- Mobility.
- Frequency range.

- Duplexing.
- Channel bandwidth.
- Channel coding.
- MIMO.
- Multi-antenna technology.

Sample code for LTE:

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/lte-module.h"
#include "ns3/config-store-module.h"
 using namespace ns3;

int main (int argc, char *argv[])
{

CommandLine cmd; cmd.Parse (argc, argv); ConfigStore
inputConfig;

inputConfig.ConfigureDefaults (); cmd.Parse (argc, argv);

Ptr<LteHelper> lteHelper = CreateObject<LteHelper> ();
lteHelper->SetAttribute ("PathlossModel", StringValue ("ns3::FriisSpectrumPropagationLossModel"));
NodeContainer enbNodes;
NodeContainer ueNodes; enbNodes.Create (1); ueNodes.Create (3);
MobilityHelper mobility;
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (enbNodes);
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (ueNodes);
```

```
NetDeviceContainer enbDevs;
NetDeviceContainer ueDevs;
enbDevs = lteHelper->InstallEnbDevice (enbNodes);
ueDevs = lteHelper->InstallUeDevice (ueNodes);
lteHelper->Attach (ueDevs, enbDevs.Get (0));
enum EpsBearer::Qci q = EpsBearer::GBR_CONV_VOICE;
EpsBearer bearer (q);
lteHelper->ActivateDataRadioBearer (ueDevs, bearer);
Simulator::Stop (Seconds (0.5));
lteHelper->EnablePhyTraces ();
lteHelper->EnableMacTraces ();
lteHelper->EnableRlcTraces ();

double distance_temp [] = { 1000,1000,1000};
std::vector<double> userDistance;
userDistance.assign (distance_temp, distance_temp + 3);

for (int i = 0; i < 3; i++)
{
Ptr<ConstantPositionMobilityModel> mm = ueNodes.Get (i)->GetObject<ConstantPositionMobilityModel> ();
mm->SetPosition (Vector (userDistance[i], 0.0, 0.0));
}
        Simulator::Run
        (); Simulator::Destroy (); return 0;
        }
```