SRI INDU COLLEGE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

LAB MANUAL

ON

R20INF31L2 : DATA COMMUNICATIONS AND NETWORKS LAB

III B. TECH I SEMESTER

DATA COMMUNICATIONS AND NETWORKS LAB

B.Tech. III Year I Sem.

L	T	P	C
0	0	3	1.5

Note:

A. Minimum of 12 Experiments have to be conducted

B. All the Experiments may be Conducted using Network Simulation software like NS-2, NSG-2.1 and Wire SHARK/equivalent software.

Note: For Experiments 2 to 10 Performance may be evaluated through simulation by using the parameters Throughput, Packet Delivery Ratio, Delay etc.

1. Writing a TCL Script to create two nodes and links between nodes
2. Writing a TCL Script to transmit data between nodes
3. Evaluate the performance of various LAN Topologies
4. Evaluate the performance of Drop Tail and RED queue management schemes
5. Evaluate the performance of CBQ and FQ Scheduling Mechanisms
6. Evaluate the performance of TCP and UDP Protocols
7. Evaluate the performance of TCP, New Reno and Vegas
8. Evaluate the performance of AODV and DSR routing protocols
9. Evaluate the performance of AODV and DSDV routing protocols
10. Evaluate the performance of IEEE 802.11 and IEEE 802.15.4
11. Evaluate the performance of IEEE 802.11 and SMAC
12. Capturing and Analysis of TCP and IP Packets
13. Simulation and Analysis of ICMP and IGMP Packets
14. Analyze the Protocols SCTP, ARP, NetBIOS, IPX VINES
15. Analysis of HTTP, DNS and DHCP Protocols

Major Equipment Required:

Required software (Open Source) like NS-2, NSG-2.1 and Wire SHARK

THE FIRST TCL SCRIPT

How to start

Now we are going to write a 'template' that you can use for all of the first Tcl scripts. You can write your Tcl scripts in any text editor like joe or emacs. I suggest that you call this first example 'example1.tcl'.

First of all, you need to create a simulator object. This is done with the command

```
set ns [new Simulator]
```

Now we open a file for writing that is going to be used for the nam trace data.

```
set nf [open out.nam w]

$ns namtrace-all $nf
```

The first line opens the file 'out.nam' for writing and gives it the file handle 'nf'. In the second line we tell the
simulator object that we created above to write all simulation data that is going to be relevant for nam into this file.

The next step is to add a 'finish' procedure that closes the trace file and starts nam.

```
proc finish {} {

        global ns nf

        $ns flush-trace close $nf

        exec nam out.nam & exit 0

}
```

You don't really have to understand all of the above code yet. It will get clearer to you once you see what the code does.

The next line tells the simulator object to execute the 'finish' procedure after 5.0 seconds of simulation time.

```
$ns at 5.0 "finish"
```

You probably understand what this line does just by looking at it. ns provides you with a very simple way to schedule events with the 'at' command.

The last line finally starts the simulation.

```
$ns run
```

You can actually save the file now and try to run it with 'ns example1.tcl'. You are going to get an error message like 'nam: empty trace file out.nam' though, because until now we haven't defined any objects (nodes, links, etc.) or events. We are going to define the objects in the events.

Two nodes, one link

In this section we are going to define a very simple topology with two nodes that are connected by a link. The following two lines define the two nodes. (Note: You have to insert the code in this section **before** the line '$ns run', or even better, before the line '$ns at 5.0 "finish"').

```
set n0 [$ns node] set n1 [$ns node]
```

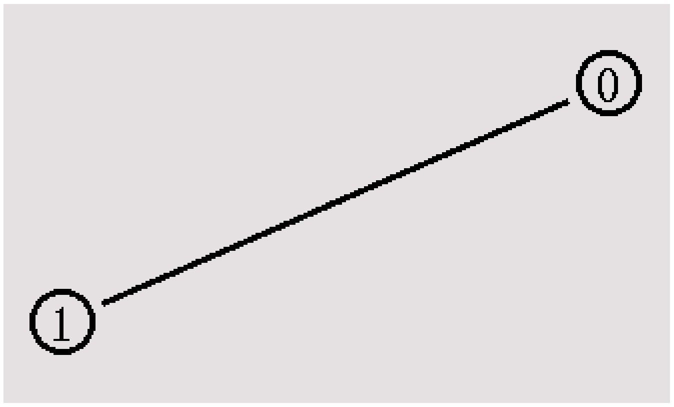
A new node object is created with the command '$ns node'. The above code creates two nodes and assigns them to the handles 'n0' and 'n1'.

The next line connects the two nodes.

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

This line tells the simulator object to connect the nodes n0 and n1 with a duplex link with the bandwidth 1Megabit, a delay of 10ms and a DropTail queue.

Now you can save your file and start the script with 'ns example1.tcl'. nam will be started automatically and you should see an output that resembles the picture below.



Sending data

Of course, this example isn't very satisfying yet, since you can only look at the topology, but nothing actually happens, so the next step is to send some data from node n0 to node n1. In ns, data is always being sent from one 'agent' to another. So the next step is to create an agent object that sends data from node n0, and another agent object that receives the data on node n1.

```
#Create  a  UDP  agent  and
attach it to node n0 set udp0
[new Agent/UDP]

$ns attach-agent $n0 $udp0


# Create a CBR traffic source and
attach it to udp0 set cbr0 [new
Application/Traffic/CBR]

$cbr0 set packetSize_ 500

$cbr0 set interval_ 0.005

$cbr0 attach-agent $udp0
```

These lines create a UDP agent and attach it to the node n0, then attach a CBR traffic generatot to the UDP
agent. CBR stands for 'constant bit rate'. Line 7 and 8 should be self-explaining. ThepacketSize is being set to 500 bytes and a packet will be sent every 0.005 seconds (i.e. 200 packets per second). You can find the relevant parameters for each agent type In The next lines create a Null agent which acts as traffic sink and attach it to node n1.

```
set null0 [new Agent/Null]

$ns attach-agent $n1 $null0
```

Now the two agents have to be connected with each other.

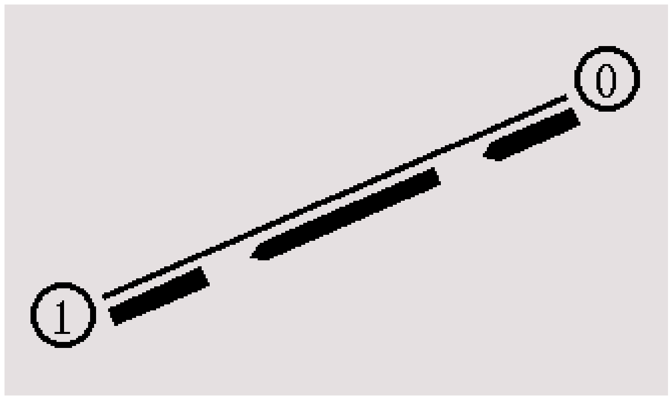
```
$ns connect $udp0 $null0
```

And now we have to tell the CBR agent when to send data and when to stop sending. Note: It's probably best to put the following lines just before the line '$ns at 5.0 "finish"'.

```
$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"
```

This code should be self-explaining again.

Now you can save the file and start the simulation again. When you click on the 'play' button in the nam
window, you will see that after 0.5 simulation seconds, node 0 starts sending data packets to node 1. You might want to slow nam down then with the 'Step' slider.



I suggest that now you start some experiments with nam and the Tcl script. You can click on any packet in the nam window to monitor it, and you can also click directly on the link to get some graphs with statistics. I also suggest that you try to change the 'packetsize_' and 'interval_' parameters in the Tcl script to see what happens. You can download the full example here.

Most of the information that I needed to be able to write this Tcl script was taken directly from the example files in the 'tcl/ex/' directory, while I learned which CBR agent arguments (packetSize_, interval_) I had to set from the ns manual page.

How to start tcl program

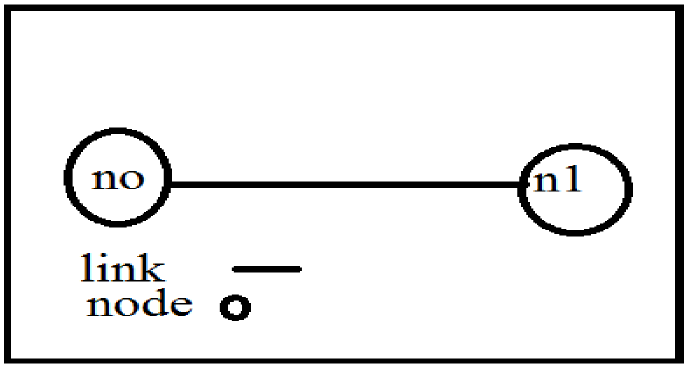
```
set ns [new Simulator]

set nf [open out.nam w]
$ns namtrace-all $nf

proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
}

$ns at 5.0 "finish"

$ns run
```

1. **Writing a TCL Script to create two nodes and links between nodes**

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all
$nf
#Define a 'finish' procedure
proc finish {}
{
global ns nf
$ns flush-trace
#Close the trace file
 close
$nf
#Execute nam on the trace file
        exec nam out.nam &
        exit 0
}
#Create two nodes
set n0 [$ns node]
set n1 [$ns node]
#Create a duplex link between the nodes
$ns duplex-link
$n0
$n1 1Mb 10ms DropTail
#Call the finish procedure after 5 seconds of simulation time
```

```
$ns at 5.0 "finish"

#Run the simulation

$ns run
```

OUT PUT

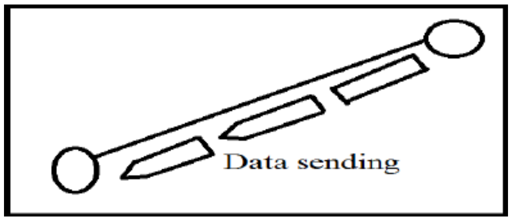


2 Writing a TCL Script to transmit data between nodes

```
    set ns [new Simulator]

set nf [open out.nam w]
$ns namtrace-all $nf

proc finish {} {
        global ns nf
        $ns flush-trace
        close $nf
        exec nam out.nam &
        exit 0
}

set n0 [$ns node]
set n1 [$ns node]
```

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

#Create a UDP agent and attach it to node n0
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n1 $null0

$ns connect $udp0 $null0

$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"

$ns at 5.0 "finish"

$ns run
```

OUT PUT



3. Evaluate the performance of various LAN Topologies

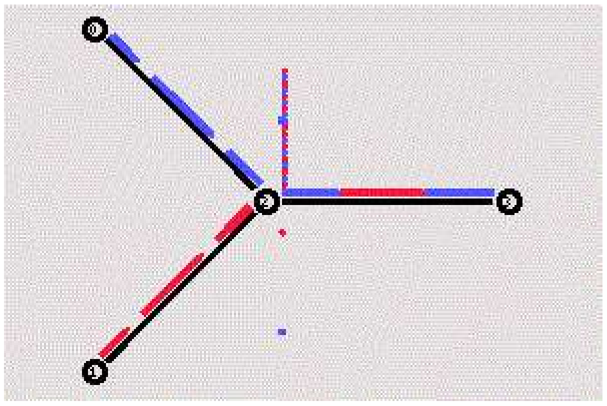
```
#Create a simulator object

set ns [new Simulator]

#Define different colors for data flows

$ns color 1 Blue

$ns color 2 Red

#Open the nam trace file

set nf [open out.nam w]
```

```
$ns  namtrace-all  $nf

#Define a 'finish' procedure

proc finish {} {

        global ns nf

        $ns flush-trace

     #Close the trace file

        close $nf

     #Execute nam on the trace file

        exec nam out.nam &

        exit 0

}

#Create four nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

#Create links between the nodes

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

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

$ns duplex-link $n3 $n2 1Mb 10ms SFQ


$ns duplex-link-op $n0 $n2 orient right-down

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

$ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for the link between node 2 and node 3

$ns duplex-link-o
```

```
p $n2 $n3 queuePos 0.5

#Create a UDP agent and attach it to node n0

set udp0 [new Agent/UDP]

$udp0 set class_ 1

$ns attach-agent $n0 $udp0

# Create a CBR traffic source and attach it to udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize_ 500

$cbr0 set interval_ 0.005

$cbr0 attach-agent $udp0

#Create a UDP agent and attach it to node n1

set udp1 [new Agent/UDP]

$udp1 set class_ 2

$ns attach-agent $n1 $udp1

# Create a CBR traffic source and attach it to udp1

set cbr1 [new Application/Traffic/CBR]

$cbr1 set packetSize_ 500

$cbr1 set interval_ 0.005

$cbr1 attach-agent $udp1

#Create a Null agent (a traffic sink) and attach it to node n3

set null0 [new Agent/Null]

$ns attach-agent $n3 $null0


#Connect the traffic sources with the traffic sink

$ns connect $udp0 $null0

$ns connect $udp1 $null0
```

```
#Schedule events for the CBR agents

$ns at 0.5 "$cbr0 start"

$ns at 1.0 "$cbr1 start"

$ns at 4.0 "$cbr1 stop"

$ns at 4.5 "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run
```

OUT PUT



4. Evaluate the performance of Drop Tail and RED queue management schemes

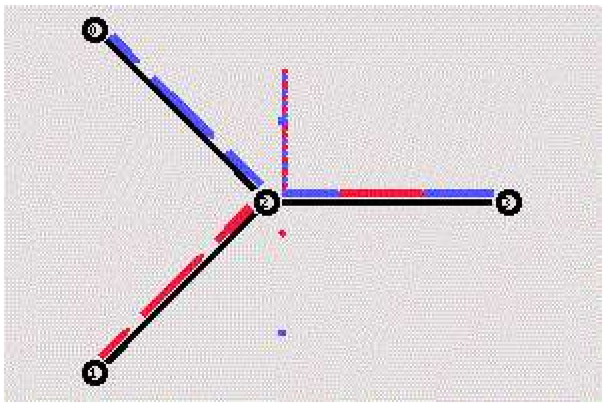
```
#Create a simulator object

set ns [new Simulator]

#Define different colors for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red
```

```
#Open the NAM trace file

set nf [open out.nam w]

$ns namtrace-all $nf

#Define a 'finish' procedure

proc finish {} {

        global ns nf

        $ns flush-trace

        #Close the NAM trace file

        close $nf


  #Execute NAM on the trace file

        exec nam out.nam &

        exit 0

}

#Create four nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

#Create links between the nodes

$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail

$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10

$ns queue-limit $n2 $n3 10
```

```
#Give node position (for NAM)

$ns duplex-link-op $n0 $n2 orient right-down

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

$ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)

$ns duplex-link-op $n2 $n3 queuePos 0.5

#Setup a TCP connection

set tcp [new Agent/TCP]

$tcp set class_ 2

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]

$ns attach-agent $n3 $sink

$ns connect $tcp $sink

$tcp set fid_ 1

#Setup a FTP over TCP connection

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set type_  FTP

#Setup a UDP connection

set udp [new Agent/UDP]

$ns attach-agent $n1 $udp

set null [new Agent/Null]

$ns attach-agent $n3 $null

$ns connect $udp $null

$udp set fid_ 2
```

```
#Setup a CBR over UDP connection

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set type_ CBR

$cbr set packet_size_ 1000

$cbr set rate_ 1mb

$cbr set random_ false

#Schedule events for the CBR and FTP agents

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 4.0 "$ftp stop"

$ns at 4.5 "$cbr stop"

#Detach tcp and sink agents (not really necessary)

$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3 $sink"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Print CBR packet size and interval

puts "CBR packet size = [$cbr set packet_size_]"

puts "CBR interval = [$cbr set interval_]"

#Run the simulation

$ns run
```

OUT PUT



5. Evaluate the performance of CBQ and FQ Scheduling Mechanisms

```
#Create a simulator object

set ns [new Simulator]

#Tell the simulator to use dynamic routing

$ns rtproto DV

#Open the nam trace file

set nf [open out.nam w]

$ns namtrace-all $nf”

#Define a 'finish' procedure

proc finish {} {

    global ns nf

    $ns flush-trace

        #Close the trace file

    close $nf

        #Execute nam on the trace file

    exec nam out.nam &

    exit 0

}
```

```
#Create seven nodes

for {set i 0} {$i < 7} {incr i} {

        set n($i) [$ns node]

}

#Create links between the nodes

for {set i 0} {$i < 7} {incr i} {

        $ns duplex-link $n($i) $n([expr ($i+1)%7]) 1Mb 10ms DropTail

}

#Create a UDP agent and attach it to node n(0)

set udp0 [new Agent/UDP]

$ns attach-agent $n(0) $udp0

# Create a CBR traffic source and attach it to udp0

set cbr0 [new Application/Traffic/CBR]

$cbr0 set packetSize_ 500

$cbr0 set interval_ 0.005

$cbr0 attach-agent $udp0

#Create a Null agent (a traffic sink) and attach it to node n(3)

set null0 [new Agent/Null]

$ns attach-agent $n(3) $null0

#Connect the traffic source with the traffic sink

$ns connect $udp0 $null0

#Schedule events for the CBR agent and the network dynamics

$ns at 0.5 "$cbr0 start"

$ns rtmodel-at 1.0 down $n(1) $n(2)

$ns rtmodel-at 2.0 up $n(1) $n(2)
```

$ns at 4.5 "$cbr0 stop"

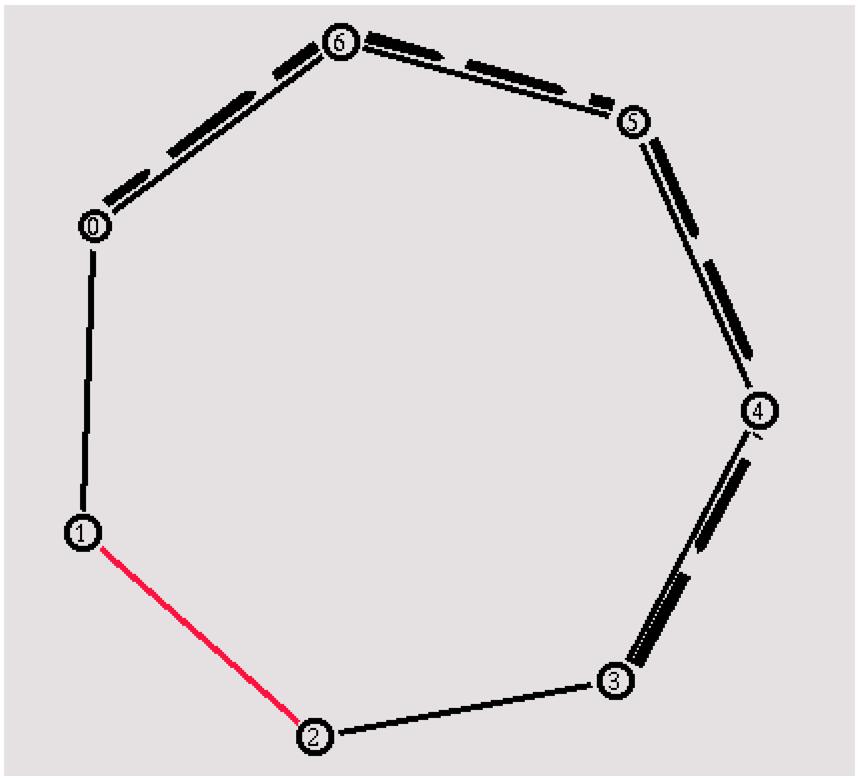
#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run

OUT PUT



#Create a simulator object

set ns [new Simulator]

#Open the output files

set f0 [open out0.tr w]

set f1 [open out1.tr w]

set f2 [open out2.tr w]

```
#Create 5 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
#Connect the nodes
$ns duplex-link $n0 $n3 1Mb 100ms DropTail
$ns duplex-link $n1 $n3 1Mb 100ms DropTail
$ns duplex-link $n2 $n3 1Mb 100ms DropTail
$ns duplex-link $n3 $n4 1Mb 100ms DropTail
#Define a 'finish' procedure
proc finish {} {
        global f0 f1 f2
        #Close the output files
        close $f0
        close $f1
        close $f2
        #Call xgraph to display the results
        exec xgraph out0.tr out1.tr out2.tr -geometry 800x400 &
    exit 0
}
#Define a procedure that attaches a UDP agent to a previously created node
#'node' and attaches an Expoo traffic generator to the agent with the
#characteristic values 'size' for packet size 'burst' for burst time,
#'idle' for idle time and 'rate' for burst peak rate. The procedure connects
#the source with the previously defined traffic sink 'sink' and returns the
#source object.
proc attach-expoo-traffic { node sink size burst idle rate } {
        #Get an instance of the simulator
        set ns [Simulator instance]
        #Create a UDP agent and attach it to the node
        set source [new Agent/UDP]
        $ns attach-agent $node $source
        #Create an Expoo traffic agent and set its configuration parameters
        set traffic [new Application/Traffic/Exponential]
        $traffic set packetSize_ $size
        $traffic set burst_time_ $burst
        $traffic set idle_time_ $idle
        $traffic set rate_ $rate
    # Attach traffic source to the traffic generator
    $traffic attach-agent $source
        #Connect the source and the sink
        $ns connect $source $sink
        return $traffic
}
#Define a procedure which periodically records the bandwidth received by the
#three traffic sinks sink0/1/2 and writes it to the three files f0/1/2.
proc record {} {
    global sink0 sink1 sink2 f0 f1 f2
```

```
        #Get an instance of the simulator
        set ns [Simulator instance]
        #Set the time after which the procedure should be called again
    set time 0.5
        #How many bytes have been received by the traffic sinks?
    set bw0 [$sink0 set bytes_]
    set bw1 [$sink1 set bytes_]
    set bw2 [$sink2 set bytes_]
        #Get the current time
    set now [$ns now]
        #Calculate the bandwidth (in MBit/s) and write it to the files
    puts $f0 "$now [expr $bw0/$time*8/1000000]"
    puts $f1 "$now [expr $bw1/$time*8/1000000]"
    puts $f2 "$now [expr $bw2/$time*8/1000000]"
        #Reset the bytes_ values on the traffic sinks
    $sink0 set bytes_ 0
    $sink1 set bytes_ 0
    $sink2 set bytes_ 0
        #Re-schedule the procedure
    $ns at [expr $now+$time] "record"
}

#Create three traffic sinks and attach them to the node n4
set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]

$ns attach-agent $n4 $sink0

$ns attach-agent $n4 $sink1

$ns attach-agent $n4 $sink2

#Create three traffic sources

set source0 [attach-expoo-traffic $n0 $sink0 200 2s 1s 100k]

set source1 [attach-expoo-traffic $n1 $sink1 200 2s 1s 200k]

set source2 [attach-expoo-traffic $n2 $sink2 200 2s 1s 300k]

#Start logging the received bandwidth

$ns at 0.0 "record"

#Start the traffic sources

$ns at 10.0 "$source0 start"

$ns at 10.0 "$source1 start"

$ns at 10.0 "$source2 start"

#Stop the traffic sources

$ns at 50.0 "$source0 stop"

$ns at 50.0 "$source1 stop"

$ns at 50.0 "$source2 stop"

#Call the finish procedure after 60 seconds simulation time

$ns at 60.0 "finish"

#Run the simulation

$ns run
```

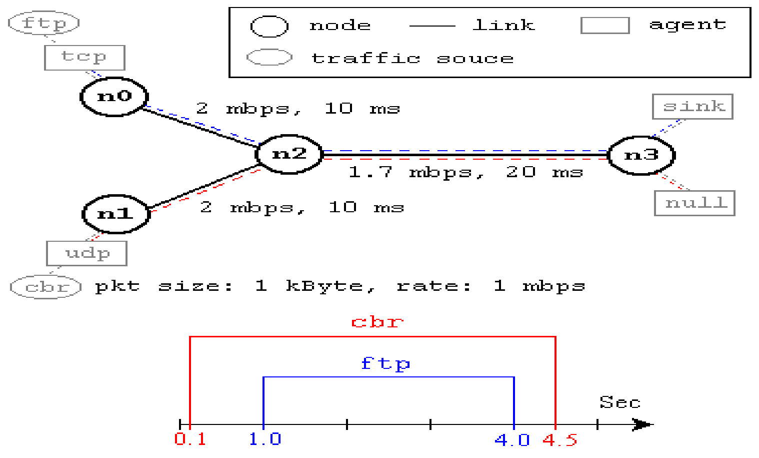
6. Evaluate the performance of TCP and UDP Protocols

```
#Create a simulator object

set ns [new Simulator]
```

```
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#Open the NAM trace file
set nf [open out.nam w]
$ns namtrace-all $nf

#Define a 'finish' procedure
proc finish {} {
        global ns nf
        $ns flush-trace
        #Close the NAM trace file
        close $nf
        #Execute NAM on the trace file
        exec nam out.nam &
        exit 0
}

#Create four nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]

#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.7Mb 20ms DropTail

#Set Queue Size of link (n2-n3) to 10
$ns queue-limit $n2 $n3 10

#Give node position (for NAM)
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n1 $n2 orient right-up
$ns duplex-link-op $n2 $n3 orient right

#Monitor the queue for link (n2-n3). (for NAM)
$ns duplex-link-op $n2 $n3 queuePos 0.5


#Setup a TCP connection
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
```

```
$tcp set fid_ 1

#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP


#Setup a UDP connection
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#Setup a CBR over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
$cbr set random_ false


#Schedule events for the CBR and FTP agents
$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 4.0 "$ftp stop"
$ns at 4.5 "$cbr stop"

#Detach tcp and sink agents (not really necessary)
$ns at 4.5 "$ns detach-agent $n0 $tcp ; $ns detach-agent $n3
$sink"

#Call the finish procedure after 5 seconds of simulation time
$ns at 5.0 "finish"

#Print CBR packet size and interval
puts "CBR packet size = [$cbr set packet_size_]"
puts "CBR interval = [$cbr set interval_]"

#Run the simulation
$ns run
```

OUT PUT



7, (10.) Evaluate the performance of IEEE 802.11 and IEEE 802.15.4

```
# Simulator Instance Creation

set ns [new Simulator]

#Fixing the co-ordinate of simutaion area

set val(x) 500

set val(y) 500

# Define options

set val(chan) Channel/WirelessChannel ;# channel type

set val(prop) Propagation/TwoRayGround ;# radio-propagation model

set val(netif) Phy/WirelessPhy ;# network interface type

set val(mac) Mac/802_11 ;# MAC type

set val(ifq) Queue/DropTail/PriQueue ;# interface queue type

set val(ll) LL ;# link layer type

set val(ant) Antenna/OmniAntenna ;# antenna model

set val(ifqlen) 50 ;# max packet in ifq

set val(nn) 2 ;# number of mobilenodes
set val(rp) AODV ;# routing protocol

set val(x) 500 ;# X dimension of topography

set val(y) 400 ;# Y dimension of topography

set val(stop) 10.0 ;# time of simulation end

# set up topography object

set topo [new Topography]

$topo load_flatgrid $val(x) $val(y)

#Nam File Creation nam – network animator

set namfile [open sample1.nam w]

#Tracing all the events and cofiguration

$ns namtrace-all-wireless $namfile $val(x) $val(y)

#Trace File creation

set tracefile [open sample1.tr w]

#Tracing all the events and cofiguration

$ns trace-all $tracefile

# general operational descriptor- storing the hop details in the network

create-god $val(nn)

# configure the nodes

$ns node-config -adhocRouting $val(rp) \

-llType $val(ll) \

-macType $val(mac) \

-ifqType $val(ifq) \

-ifqLen $val(ifqlen) \

-antType $val(ant) \

-propType $val(prop) \
-phyType $val(netif) \
-channelType $val(chan) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace ON
# Node Creation
set node1 [$ns node]
# Initial color of the node
$node1 color black
#Location fixing for a single node
$node1 set X_ 200
$node1 set Y_ 100
$node1 set Z_ 0
set node2 [$ns node]
$node2 color black
$node2 set X_ 200
$node2 set Y_ 300
$node2 set Z_ 0
# Label and coloring
$ns at 0.1 "$node1 color blue"
$ns at 0.1 "$node1 label Node1"
$ns at 0.1 "$node2 label Node2"
#Size of the node
```

```
$ns initial_node_pos $node1 30

$ns initial_node_pos $node2 30

# ending nam and the simulation

$ns at $val(stop) "$ns nam-end-wireless $val(stop)"

$ns at $val(stop) "stop"

#Stopping the scheduler

$ns at 10.01 "puts \"end simulation\" ; $ns halt"

#$ns at 10.01 "$ns halt"

proc stop {} {

global namfile tracefile ns

$ns flush-trace

close $namfile

close $tracefile

#executing nam file

exec nam sample1.nam &

}

#Starting scheduler

$ns run
```

8(15.) Analysis of HTTP, DNS and DHCP Protocols. 10, 11 EXPERMENT PROGRAM

#Lan simulation

```
set ns [new Simulator]

#define color for data flows

$ns color 1 Blue

$ns color 2 Red
```

```
#open tracefiles

set tracefile1 [open out.tr w]

set winfile [open winfile w]

$ns trace-all $tracefile1

#open nam file

set namfile [open out.nam w]

$ns namtrace-all $namfile

#define the finish procedure

proc finish {} {

global ns tracefile1 namfile

$ns flush-trace

close $tracefile1

close $namfile

exec nam out.nam &

exit 0

} #create six nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$n1 color Red

$n1 shape box

#create links between the nodes
```

```
$ns duplex-link $n0 $n2 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 2Mb 10ms DropTail

$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail

$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail

set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail

MAC/Csma/Cd Channel]

#Give node position

$ns duplex-link-op $n0 $n2 orient right-down

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

$ns simplex-link-op $n2 $n3 orient right

$ns simplex-link-op $n3 $n2 orient left

#set queue size of link(n2-n3) to 20

$ns queue-limit $n2 $n3 20

#setup TCP connection

set tcp [new Agent/TCP/Newreno]

$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink/DelAck]

$ns attach-agent $n4 $sink

$ns connect $tcp $sink

$tcp set fid_ 1

$tcp set packet_size_ 552

#set ftp over tcp connection

set ftp [new Application/FTP]

$ftp attach-agent $tcp

#setup a UDP connection
```

```
set udp [new Agent/UDP]

$ns attach-agent $n1 $udp

set null [new Agent/Null]

$ns attach-agent $n5 $null

$ns connect $udp $null

$udp set fid_ 2

#setup a CBR over UDP connection

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set type_ CBR

$cbr set packet_size_ 1000

$cbr set rate_ 0.01Mb

$cbr set random_ false

#scheduling the events

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 124.0 "$ftp stop"

$ns at 125.5 "$cbr stop"

proc plotWindow {tcpSource file} {

global ns

set time 0.1

set now [$ns now]

set cwnd [$tcpSource set cwnd_]

puts $file "$now $cwnd"

$ns at [expr $now+$time] "plotWindow $tcpSource $file"
}

$ns at 0.1 "plotWindow $tcp $winfile"

$ns at 125.0 "finish"

$ns run
```