1

Problem Definition:

This network consists of 4 nodes (n0, n1, n2, n3) as shown in above figure. The duplex links between n0 and n2, and n1 and n2 have 2 Mbps of bandwidth and 10 ms of delay. The duplex link between n2 and n3 has 1.7 Mbps of bandwidth and 20 ms of delay. Each node uses a DropTail queue, of which the maximum size is 10. A "tcp" agent is attached to n0, and a connection is established to a tcp "sink" agent attached to n3. As default, the maximum size of a packet that a "tcp" agent can generate is 1KByte. A tcp "sink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. A "udp" agent that is attached to n1 is connected to a "null" agent attached to n3. A "null" agent just frees the packets received. A "ftp" and a "cbr" traffic generator are attached to "tcp" and "udp" agents respectively, and the "cbr" is configured to generate 1 KByte packets at the rate of 1 Mbps. The "cbr" is set to start at 0.1 sec and stop at 4.5 sec, and "ftp" is set to start at 1.0 sec and stop at 4.0 sec.

Script:

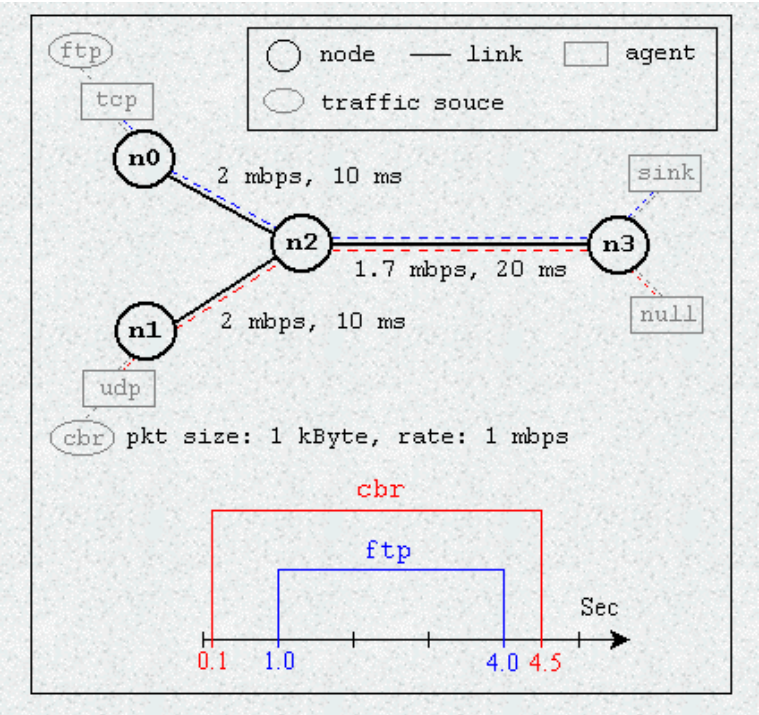
```
#Create a simulator object
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

#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
```

Reference: http://nile.wpi.edu/NS/simple_ns.html

2.

Problem Definition:
Trace Analysis of the following



Script:

```
#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

#Open the Trace file
set tf [open out.tr w]
$ns trace-all $tf

#Define a 'finish' procedure
proc finish {} {
    global ns nf tf
    $ns flush-trace
    #Close the NAM trace file
    close $nf
    #Close the Trace file
    close $tf
    #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
```

Reference: http://nile.wpi.edu/NS/analysis.html

3

Problem Definition:
Relatively simple multicast configuration between two nodes

Script:

```
set ns [new Simulator -multicast on]; # enable multicast
routing;
set group [Node allocaddr]  ; # allocate a multicast address;
set node0 [$ns node]         ;# create multicast capable
nodes;
set node1 [$ns node]
$ns duplex-link $node0 $node1 1.5Mb 10ms DropTail

set tracef [open dm.tr w]
$ns trace-all $tracef

set namtracef [open dm.nam w]
$ns namtrace-all $namtracef

set mproto DM         ; # configure multicast protocol;
set mrthandle [$ns mrtproto $mproto]; # all nodes will contain
multicast protocol agents;

set udp [new Agent/UDP]          ;# create a source agent at
node 0;
$ns attach-agent $node0 $udp
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$udp set dst_addr_ $group
$udp set dst_port_ 0

set receiver [new Agent/LossMonitor];  # create a receiver
agent at node 1;
$ns attach-agent $node1 $receiver
```

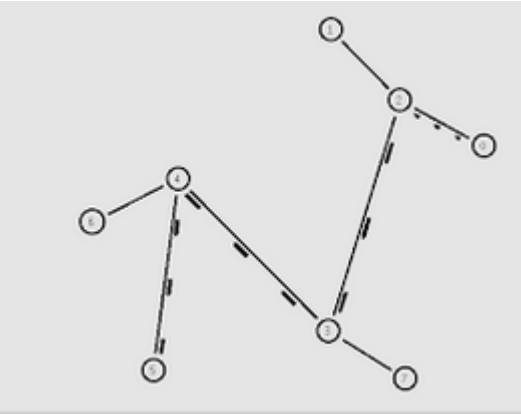
```
$ns at 0.3 "$node1 join-group $receiver $group"
$ns at 0.1 "$cbr start"

$ns at 5.0 "finish"

proc finish {} {
        global ns tracef namtracef
        exec nam dm.nam &
        close $tracef
        close $namtracef
        exit 0
}
$ns run
```

Reference: https://www.nsnam.com/2017/03/multicasting-in-ns2.html

4.

Problem Definition:

Centralized Multicast routing in ns2



Script:

```
#This example is to demonstrate the multicast routing protocol.
set ns [new Simulator -multicast on]
#Turn on Tracing
set tf [open output.tr w]
$ns trace-all $tf

# Turn on nam Tracing
set fd [open mcast.nam w]
$ns namtrace-all $fd

# Create 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]
set n6 [$ns node]
set n7 [$ns node]

# Create links with DropTail Queues
$ns duplex-link $n0 $n2 1.5Mb 10ms DropTail
$ns duplex-link $n1 $n2 1.5Mb 10ms DropTail
```

```
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$ns duplex-link $n3 $n4 1.5Mb 10ms DropTail
$ns duplex-link $n3 $n7 1.5Mb 10ms DropTail
$ns duplex-link $n4 $n5 1.5Mb 10ms DropTail
$ns duplex-link $n4 $n6 1.5Mb 10ms DropTail

# Routing protocol: say distance vector
#Protocols: CtrMcast, DM, ST, BST
#Dense Mode protocol is supported in this example
set mproto DM
set mrthandle [$ns mrtproto $mproto {}]

# Set two groups with group addresses
set group1 [Node allocaddr]
set group2 [Node allocaddr]

# UDP Transport agent for the traffic source for group1
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
$udp0 set dst_addr_ $group1
$udp0 set dst_port_ 0
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp0

# Transport agent for the traffic source for group2
set udp1 [new Agent/UDP]
$ns attach-agent $n1 $udp1
$udp1 set dst_addr_ $group2
$udp1 set dst_port_ 0
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp1

# Create receiver to accept the packets
set rcvr1 [new Agent/Null]
$ns attach-agent $n5 $rcvr1
$ns at 1.0 "$n5 join-group $rcvr1 $group1"
set rcvr2 [new Agent/Null]
$ns attach-agent $n6 $rcvr2
$ns at 1.5 "$n6 join-group $rcvr2 $group1"


set rcvr3 [new Agent/Null]
$ns attach-agent $n7 $rcvr3
$ns at 2.0 "$n7 join-group $rcvr3 $group1"

set rcvr4 [new Agent/Null]
$ns attach-agent $n5 $rcvr1
$ns at 2.5 "$n5 join-group $rcvr4 $group2"

set rcvr5 [new Agent/Null]
$ns attach-agent $n6 $rcvr2
$ns at 3.0 "$n6 join-group $rcvr5 $group2"

set rcvr6 [new Agent/Null]
$ns attach-agent $n7 $rcvr3
```

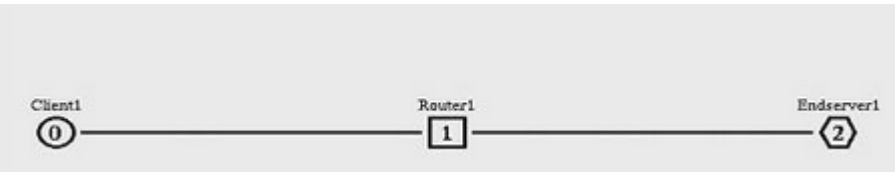
```
#The nodes are leaving the group at specified times
$ns at 3.5 "$n7 join-group $rcvr6 $group2"
$ns at 4.0 "$n5 leave-group $rcvr1 $group1"
$ns at 4.5 "$n6 leave-group $rcvr2 $group1"
$ns at 5.0 "$n7 leave-group $rcvr3 $group1"
$ns at 5.5 "$n5 leave-group $rcvr4 $group2"
$ns at 6.0 "$n6 leave-group $rcvr5 $group2"
$ns at 6.5 "$n7 leave-group $rcvr6 $group2"

# Schedule events

$ns at 0.5 "$cbr1 start"
$ns at 9.5 "$cbr1 stop"
$ns at 0.5 "$cbr2 start"
$ns at 9.5 "$cbr2 stop"

#post-processing

$ns at 10.0 "finish"
proc finish {} {
 global ns tf
  $ns flush-trace
  close $tf
  exec nam mcast.nam &
  exit 0
}

$ns set-animation-rate 3.0ms
$ns run
```

Reference: https://www.nsnam.com/2017/03/multicasting-in-ns2.html

5.

Problem Definition

This network consists of 3 nodes (Client1, Router1 and Endserver1). The duplex link between Client1 and Router1 has 2 Mbps of bandwidth and 100 ms of delay. The duplex link between Router1 and Endserver1 has 200Kbps of bandwidth and 100 ms of delay. Each link between nodes uses a Drop Tail queue.



Script:

```
#-------Event scheduler object creation--------#
set ns [new Simulator]
#----------creating nam objects----------------#

set nf [open tcp1.nam w]
$ns namtrace-all $nf
```

```
#open the trace file
set nt [open tcp1.tr w]
$ns trace-all $nt

set proto rlm

$ns color 1 blue
$ns color 2 yellow
$ns color 3 red

#---------- creating client- router- end server node----------------#

set Client1 [$ns node]
set Router1 [$ns node]
set Endserver1 [$ns node]



#---creating duplex link---------#

$ns duplex-link $Client1 $Router1 2Mb 100ms DropTail
$ns duplex-link $Router1 $Endserver1 200Kb 100ms DropTail



#----------------creating orientation------------------#

$ns duplex-link-op $Client1 $Router1 orient right
$ns duplex-link-op $Router1 $Endserver1 orient right

#------------Labelling----------------#

$ns at 0.0 "$Client1 label Client1"
$ns at 0.0 "$Router1 label Router1"
$ns at 0.0 "$Endserver1 label Endserver1"

#-----------Configuring nodes------------#

$Endserver1 shape hexagon
$Router1 shape square

#----------------Establishing queues---------#

#$ns duplex-link-op $Client1 $Router1 queuePos 0.1
#$ns duplex-link-op $Router1 $Endserver1 queuePos 0.5

#---------finish procedure--------#

proc finish {} {
        global ns nf nt
        $ns flush-trace
        close $nf
        close $nt
```

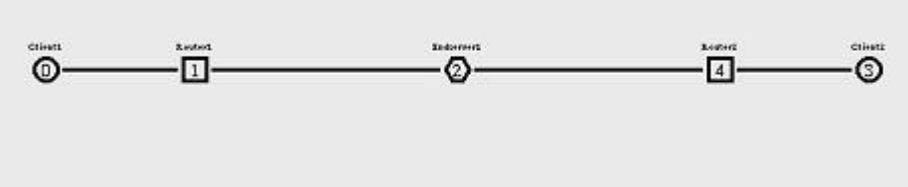
```
    puts "running nam..."
    exec nam tcp1.nam &
    exit 0
  }

#Calling finish procedure
$ns at 6.0 "finish"
$ns run
```

Reference:
http://enggedu.com/TCL_script_for_create_Nodes_duplex_link_orientation_Label_and_Queue/index.php

6.

Problem Definition:

This network consists of 4 nodes (Client1, Router1, Client2, Router2 and Endserver1). The duplex link between Client1 and Router1 has 2 Mbps of bandwidth and 50ms of delay. The duplex link between Router1 and Endserver1 has 100Kbps of bandwidth and 100 ms of delay. The duplex link between Client2 and Router2 has 100Kbps bandwidth and 50ms delay. The duplex link between Router2 and Endserver1 has 100Kbps bandwidth and 100ms of delay. Each link between nodes uses a DropTail queue.



Script:

```
set ns [new Simulator]

#----------creating nam object---------#

set nf [open tcp2.nam w]
$ns namtrace-all $nf

set nt [open tcp2.tr w]
$ns trace-all $nt

set proto rlm

$ns color 1 red
$ns color 2 blue

#---------- creating client- router- end server node----------------#

set Client1 [$ns node]
set Router1 [$ns node]
set Endserver1 [$ns node]
set Client2 [$ns node]
set Router2 [$ns node]

#---creating duplex link---------#
```

```
$ns duplex-link $Client1 $Router1 2Mb 50ms DropTail
$ns duplex-link $Router1 $Endserver1 100Kb 100ms DropTail
$ns duplex-link $Client2 $Router2 100Kb 50ms DropTail
$ns duplex-link $Router2 $Endserver1 100Kb 100ms DropTail

#----------creating orientation----------#

$ns duplex-link-op $Client1 $Router1 orient right
$ns duplex-link-op $Router1 $Endserver1 orient right
$ns duplex-link-op $Endserver1 $Router2 orient right
$ns duplex-link-op $Router2 $Client2 orient right

#----------Creating Labeling----------#

$ns at 0.0 "$Client1 label Client1"
$ns at 0.0 "$Router1 label Router1"
$ns at 0.0 "$Endserver1 label Endserver1"
$ns at 0.0 "$Router2 label Router2"
$ns at 0.0 "$Client2 label Client2"

#----------Configuring nodes----------#

$Endserver1 shape hexagon
$Router1 shape square
$Router2 shape square
#----------Establishing queues---------#

$ns duplex-link-op $Client1 $Router1 queuePos 0.5
$ns duplex-link-op $Router1 $Endserver1 queuePos 0.5
$ns duplex-link-op $Client2 $Router2 queuePos 0.5
$ns duplex-link-op $Router2 $Endserver1 queuePos 0.5

#---------finish procedure--------#

proc finish {} {

global ns nf nt
$ns flush-trace
close $nf
puts "running nam..."
exec nam tcp2.nam &
exit 0
}

#Calling finish procedure

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

Reference:

http://enggedu.com/TCL_script_for_Bandwidth_and_delay_configaration_between_Nodes/index.php

7.

Problem Definition:

This network consists of 5 nodes (Client1, Client2, Router1, Router2 and Endserver1). The duplex links between Client1 and Router1 have 2 Mbps of bandwidth and 50 ms of delay. The duplex link between Router1 and Endserver1 has 100Kbps of bandwidth and 100 ms of delay. The duplex link between Client2 and Router2 has 100Kbps of bandwidth and 50 ms of delay. The duplex link between Endserver1 and Router2 has 100Mbps of bandwidth and 100 ms of delay. Each link uses a DropTail queue. A "TCP" agent is attached to Client1 and client2 connection is established to a tcp "sink" agent attached to Endserver1. As default, the maximum size of a packet that a "TCP" agent can generate is 1000bytes. A "TCPsink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. The packets are dropped down between Router1 to Endserver1 at 1.6 sec. The ftp is set to start at 0.5 sec and stop at 3.5 sec.

Script:

```
set ns [new Simulator]

#------------ CREATING NAM OBJECTS -----------------#

set nf [open drop2.nam w]
$ns namtrace-all $nf

set nt [open drop2.tr w]
$ns trace-all $nt

set proto rlm

#------------COLOR DESCRIPTION---------------#

$ns color 1 red
$ns color 2 blue

# --------- CREATING SENDER - RECEIVER - ROUTER NODES-----------#

set Client1 [$ns node]
set Router1 [$ns node]
set Endserver1 [$ns node]
set Client2 [$ns node]
set Router2 [$ns node]

# --------------CREATING DUPLEX LINK ----------------------#

$ns duplex-link $Client1 $Router1 2Mb 50ms DropTail
$ns duplex-link $Router1 $Endserver1 100Kb 100ms DropTail
$ns duplex-link $Client2 $Router2 100Kb 50ms DropTail
$ns duplex-link $Router2 $Endserver1 100Kb 100ms DropTail
```

#----------CREATING ORIENTATION ----------#

```
$ns duplex-link-op $Client1 $Router1 orient right
$ns duplex-link-op $Router1 $Endserver1 orient right
$ns duplex-link-op $Endserver1 $Router2 orient right
$ns duplex-link-op $Router2 $Client2 orient right

# --------------LABELLING ----------#
$ns at 0.0 "$Client1 label Client1"
$ns at 0.0 "$Router1 label Router1"
$ns at 0.0 "$Endserver1 label Endserver1"
$ns at 0.0 "$Router2 label Router2"
$ns at 0.0 "$Client2 label Client2"
```

 # ---------- CONFIGURING NODES ----------#

```
$Endserver1 shape hexagon
$Router1 shape square
$Router2 shape square

#-------------QUEUE SIZE DESCRIPTION---------------#
$ns duplex-link-op $Client1 $Router1 queuePos 0.5
$ns duplex-link-op $Router1 $Endserver1 queuePos 0.5
$ns duplex-link-op $Client2 $Router2 queuePos 0.5
$ns duplex-link-op $Router2 $Endserver1 queuePos 0.5
```

 # ----------ESTABLISHING COMMUNICATION ----------#

#----------TCP CONNECTION BETWEEN NODES----------#

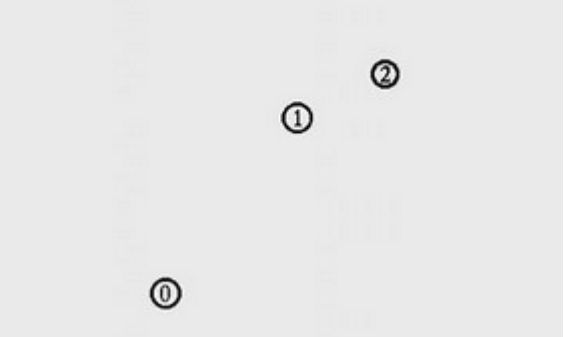
```
set tcp1 [$ns create-connection TCP $Client1 TCPSink $Endserver1 0]
     $tcp1 set fid_ 1
     set ftp1 [$tcp1 attach-app FTP]
     $ftp1 set packetSize_ 1000
     $ftp1 set interval_ 0.5
     $ns at 0.5 "$Client1 color green"
     $ns at 1.5 "$Endserver1 color red"
     $ns at 0.5 "$ftp1 start"
     $ns at 3.0 "$ftp1 stop"

     $ns rtmodel-at 1.6 down $Router1 $Endserver1
     #$ns rtmodel-at 2.0 up $Router1 $Endserver1

#---------- client2 to endserver1------#

set tcp2 [$ns create-connection TCP $Client2 TCPSink $Endserver1 0]
     $tcp2 set fid_ 1
     set ftp2 [$tcp2 attach-app FTP]
     $ftp2 set packetSize_ 1000
     $ftp2 set interval_ 0.5
     $ns at 0.5 "$ftp2 start"
     $ns at 3.0 "$ftp2 stop"
```

```
# ---------- FINISH PROCEDURE ----------#

proc finish {} {

global ns nf nt
$ns flush-trace
close $nf
puts "running nam..."
exec nam drop2.nam &
exit 0
}

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

Reference:
http://enggedu.com/TCL_script_to_drop_the_packets_in_router_and_endserver_link_at_1_6_sec/index.php

8.

Problem Definition:
Three wireless nodes are created and they are configured with specific parameters of a mobile wireless node. After creating the nam file and trace file, we set up topography object. set node_ ($i) [$ns node] is used to create the nodes. Location of the nodes is fixed by specifying X, Y coordinates. Z coordinate is always zero. Here we set the initial size for the every node by using initial_node_pos. AODV routing protocol is used here. $val(stop) specifies the end time of the simulation.



Script:

```
set ns [new Simulator]

# Creating trace file and nam file

set tracefd [open wireless1.tr w]
set namtrace [open wireless1.nam w]
#Fixing the co-ordinate of simulation 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) 3 ;# number of mobilenodes
set val(rp) AODV ;# routing protocol
set val(x) 500 ;# X dimension of topography
set val(y) 500 ;# Y dimension of topography
set val(stop) 10.0 ;# time of simulation end

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

set god_ [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

## Creating node objects...
for {set i 0} {$i < $val(nn) } { incr i } {
          set node_($i) [$ns node]
     }
     for {set i 0} {$i < $val(nn)  } {incr i } {
          $node_($i) color black
          $ns at 0.0 "$node_($i) color black"
     }

# Provide initial location of mobile nodes
$node_(0) set X_ 50.0
$node_(0) set Y_ 50.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 200.0
$node_(1) set Y_ 250.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 300.0
$node_(2) set Y_ 300.0
$node_(2) set Z_ 0.0

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
```

```
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
   $ns at $val(stop) "$node_($i) reset";
}

# Ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 10.01 "puts \"end simulation\"; $ns halt"
#stop procedure:
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
exec nam wireless1.nam &
}
$ns run
```

Reference: http://enggedu.com/Tcl_script_to_create_fixed_wireless_nodes/index.php

9.

Problem Definition:

Eight wireless nodes are created and they are configured with specific parameters of a mobile wireless node. After creating the nam file and trace file, we set up topography object. set node_ ($i) [$ns node] is used to create the nodes. Location of the nodes is fixed by specifying X, Y coordinates. Z coordinate is always zero. Here we set the initial size for the every node by using initial_node_pos. AODV routing protocol is used here. $val(stop) specifies the end time of the simulation. In this program nodes are given with cyan color

Script:

```
set ns          [new Simulator]

#Creating trace file and nam file.
set tracefd     [open wireless2.tr w]
set namtrace    [open wireless2.nam w]

# 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) 8 ;# 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

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo [new Topography]

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

set god_ [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

# Creating node objects..
for {set i 0} {$i < $val(nn) } { incr i } {
          set node_($i) [$ns node]
    }
    for {set i 0} {$i < $val(nn)  } {incr i } {
          $node_($i) color cyan
          $ns at 0.0 "$node_($i) color cyan"
    }

# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 30.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 50.0
$node_(1) set Y_ 25.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 200.0
$node_(2) set Y_ 90.0
$node_(2) set Z_ 0.0
```

```
$node_(3) set X_ 350.0
$node_(3) set Y_ 160.0
$node_(3) set Z_ 0.0

$node_(4) set X_ 100.0
$node_(4) set Y_ 250.0
$node_(4) set Z_ 0.0

$node_(5) set X_ 300.0
$node_(5) set Y_ 100.0
$node_(5) set Z_ 0.0

$node_(6) set X_ 400.0
$node_(6) set Y_ 350.0
$node_(6) set Z_ 0.0

$node_(7) set X_ 3s50.0
$node_(7) set Y_ 470.0
$node_(7) set Z_ 0.0

# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
   $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 10.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
exec nam wireless2.nam &
}

$ns run
```

Reference: http://enggedu.com/Tcl_script_to_create_fixed_color_wireless_nodes/index.php

10.

Problem Definition:

Number of nodes (3) is fixed in the program. Nodes are configured with specific parameters of a mobile wireless node. After creating the nam file and trace file, we set up topography object. set node_ ($i) [$ns node] is used to create the nodes. Initial location of the nodes is fixed. Specific X, Y coordinates are assigned to every node. Nodes are given mobility with fixed speed and fixed destination location. Here we set the initial size for the every node by using initial_node_pos. AODV

routing protocol is used here. $val(stop) specifies the end time of the simulation. TCP agent is attached to node_ (0). TCPSink agent is attached to node_(1). Both the agents are connected and FTP application is attached to TCP agent. Now communication set up for node_(0) and node_(1) is established. Similarly communication between node_(1) and node_(2) is established.

Script:

```
set ns            [new Simulator]
#creating trace file and nam file
set tracefd       [open wireless1.tr w]
set windowVsTime2 [open win.tr w]
set namtrace      [open wireless1.nam w]
# 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) 3 ;# number of mobilenodes
set val(rp) AODV ;# routing protocol
set val(x) 500 ;# X dimension of topography
set val(y) 500 ;# Y dimension of topography
set val(stop) 10.0 ;# time of simulation end

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo       [new Topography]

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

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

    for {set i 0} {$i < $val(nn) } { incr i } {
        set node_($i) [$ns node]
    }

# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
$node_(0) set Y_ 5.0
$node_(0) set Z_ 0.0

$node_(1) set X_ 490.0
$node_(1) set Y_ 285.0
$node_(1) set Z_ 0.0

$node_(2) set X_ 150.0
$node_(2) set Y_ 240.0
$node_(2) set Z_ 0.0

# Generation of movements
$ns at 10.0 "$node_(0) setdest 250.0 250.0 3.0"
$ns at 15.0 "$node_(1) setdest 45.0 285.0 5.0"
$ns at 19.0 "$node_(2) setdest 480.0 300.0 5.0"

# Set a TCP connection between node_(0) and node_(1)
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"

set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(1) $tcp
$ns attach-agent $node_(2) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"

# Printing the window size
proc plotWindow {tcpSource file} {
global ns
set time 0.01
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 10.0 "plotWindow $tcp $windowVsTime2"
```

```
# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
   $ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at 150.01 "puts \"end simulation\" ; $ns halt"
proc stop {} {
   global ns tracefd namtrace
   $ns flush-trace
   close $tracefd
   close $namtrace
exec nam simwrls.nam &
}

$ns run
```

Reference:

http://enggedu.com/Tcl_script_to_make_TCP_communication_between_nodes/index.php