NS2 Wired Program Examples

1. TCL script for creating nodes, duplex link, orientation, Label and Queue.

2. TCL script for bandwidth and delay configuration between Nodes.

3. TCL script to create a network which consists of eight nodes then set the bandwidth, delay and queue size of the link between the nodes.

4. TCL script to set identification color to links.

5. TCL script to create Tcp agent, Tcp sink and attach the Tcp agent with Tcpsink.

6. TCL script for TCP communication between two Clients and a Endserver.

7. TCL script for TCP communication between four Clients and a Endserver.

8. TCL script for TCP communication between more numbers of nodes.

9. TCL script for UDP communication.

10. TCL script to drop down the packets in particular link at specific time.

11. TCL script to drop the packets in router and Endserver link at 1.6sec.

12. TCL script to drop down the packets in same link at particular time intervals.

13. TCL script to generate rands file.

14. TCL script to generate rands file and generating graph using rands file.

15. TCL script to generate rand file which contains packets sent, received and dropped information.

16. TCL script to handle trace annotation.

17. TCL script for display the nodes activity in simulation window using trace annotation.

1. TCL script for creating nodes, duplex link, orientation, Label and Queue.

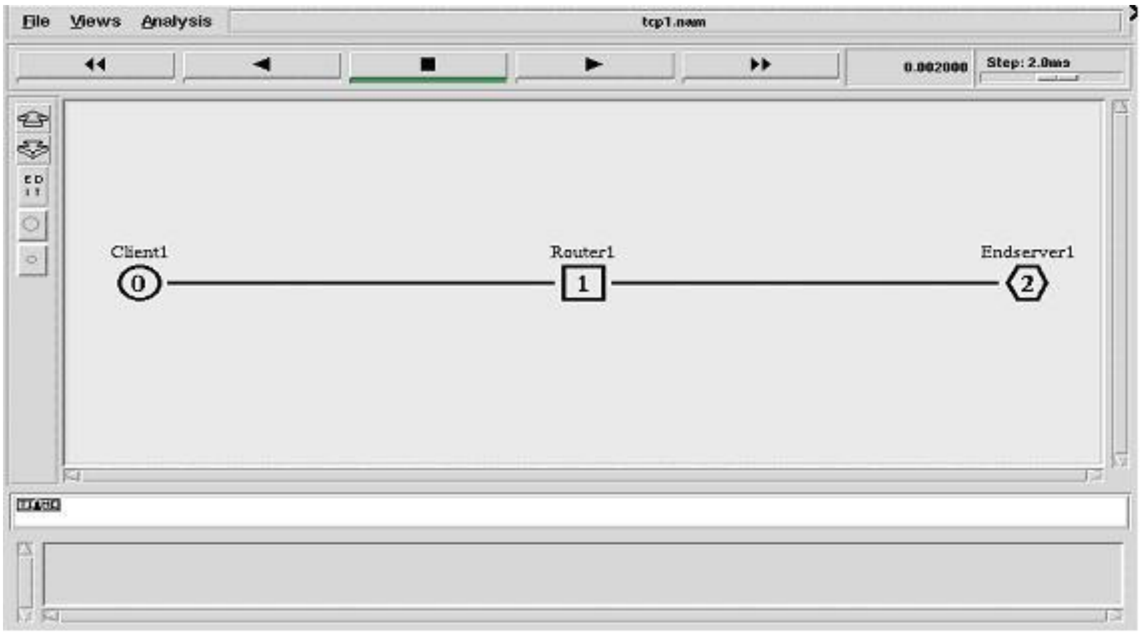
Description:

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.

File name: "tcp1.tcl"

```
#-------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

#-----How to run-----#

$ns tcp1.tcl



#----------Output--------#
```



2. TCL script for bandwidth and delay configuration between Nodes.

Description:

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.

File name: "tcp2.tcl"

```
#-------Event scheduler object creation--------#

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----------------#
```

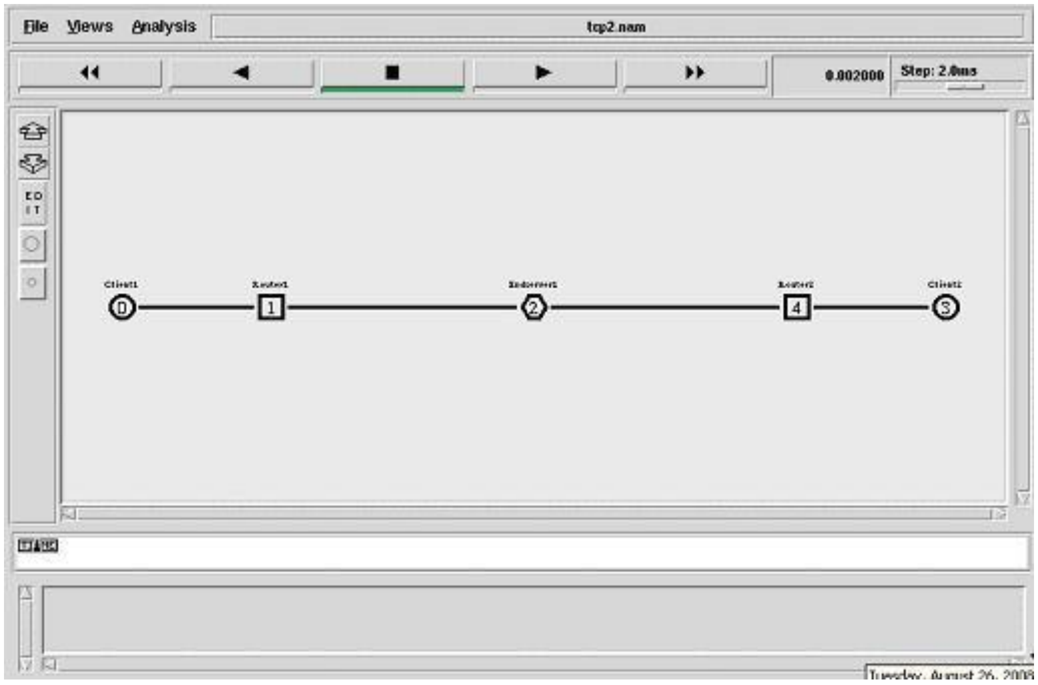
```
set Client1 [$ns node]
set Router1 [$ns node]
set Endserver1 [$ns node]
set Client2 [$ns node]
set Router2 [$ns node]

#---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
```

#-----How to run-----#

```
$ns tcp2.tcl
```

#----------Snapshot----------#



3. TCL script to create a network which consists of eight nodes then set the bandwidth, delay and queue size of the link between the nodes.

Description:

This network consists of 8 nodes (Client1, Client2, Client3, Router1, Router2 Router3, Router4, Endserver1,). The duplex links between Client1,Client2,Client3 and Router1 have 2 Mbps of bandwidth and 100 ms of delay. The duplex link between Router1 and Router2 has 200Mbps of bandwidth and 100 ms of delay. The duplex link between Router2 and Router3 has 200Mbps of bandwidth and 100 ms of delay. The duplex link between Router3 and Router4 has 200Mbps of bandwidth and 100 ms of delay. The duplex link between Router4 and Endserver1 has 200Mbps of bandwidth and 100 ms of delay. Each node uses a DropTail queue, of which the maximum size is 100.

File name: "tcp3.tcl"

#-------Event scheduler object creation--------#

```
set ns [new Simulator ]

#----------creating nam objects----------------#

set nf [open tcp3.nam w]
$ns namtrace-all $nf

#open the trace file
set nt [open tcp3.tr w]
$ns trace-all $nt

set proto rlm

$ns color 51 blue
$ns color 52 yellow
$ns color 53 red

#----- creating client- router- end server node---------#
set Client1 [$ns node]
set Client2 [$ns node]
set Client3 [$ns node]
set Router1 [$ns node]
set Router2 [$ns node]
set Router3 [$ns node]
set Router4 [$ns node]
set Endserver1 [$ns node]
#set Endserver2 [$ns node]

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

$ns duplex-link $Client1 $Router1 2Mb 100ms DropTail
$ns duplex-link $Client2 $Router1 2Mb 100ms DropTail
$ns duplex-link $Client3 $Router1 2Mb 100ms DropTail
$ns duplex-link $Router1 $Router2 100Kb 100ms DropTail
$ns duplex-link $Router2 $Router3 100Kb 100ms DropTail
$ns duplex-link $Router3 $Router4 200Kb 100ms DropTail
$ns duplex-link $Router4 $Endserver1 200Kb 100ms DropTail
#$ns duplex-link $Router4 $Endserver2 200Kb 100ms DropTail

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

$ns duplex-link-op $Client1 $Router1 orient down-right
$ns duplex-link-op $Client2 $Router1 orient right
$ns duplex-link-op $Client3 $Router1 orient up-right
$ns duplex-link-op $Router1 $Router2 orient up-right
$ns duplex-link-op $Router2 $Router3 orient down-right
$ns duplex-link-op $Router3 $Router4 orient right
$ns duplex-link-op $Router4 $Endserver1 orient up

#------------Labeling----------------#

$ns at 0.0 "$Client1 label Client1"
$ns at 0.0 "$Client2 label Client2"
$ns at 0.0 "$Client3 label Client3"
$ns at 0.0 "$Router1 label Router1"
```

```
$ns at 0.0 "$Router2 label Router2"
$ns at 0.0 "$Router3 label Router3"
$ns at 0.0 "$Router4 label Router4"
$ns at 0.0 "$Endserver1 label Endserver1"
```

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

```
#$Endserver1 shape hexagon
#$Router1 shape square
#$Router2 shape square
#$Router3 Shape square
#$Router4 Shape square
```

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

```
$ns duplex-link-op $Client1 $Router1 queuePos 0.1
$ns duplex-link-op $Client2 $Router1 queuePos 0.1
$ns duplex-link-op $Client3 $Router1 queuePos 0.5
$ns duplex-link-op $Router1 $Router2 queuePos 0.1
$ns duplex-link-op $Router2 $Router3 queuePos 0.5
$ns duplex-link-op $Router3 $Router4 queuePos 0.1
$ns duplex-link-op $Router4 $Endserver1 queuePos 0.5
```

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

```
proc finish {} {

            global ns nf nt
            puts "running nam..."
            exec nam tcp3.nam &
            exit 0
            }
```

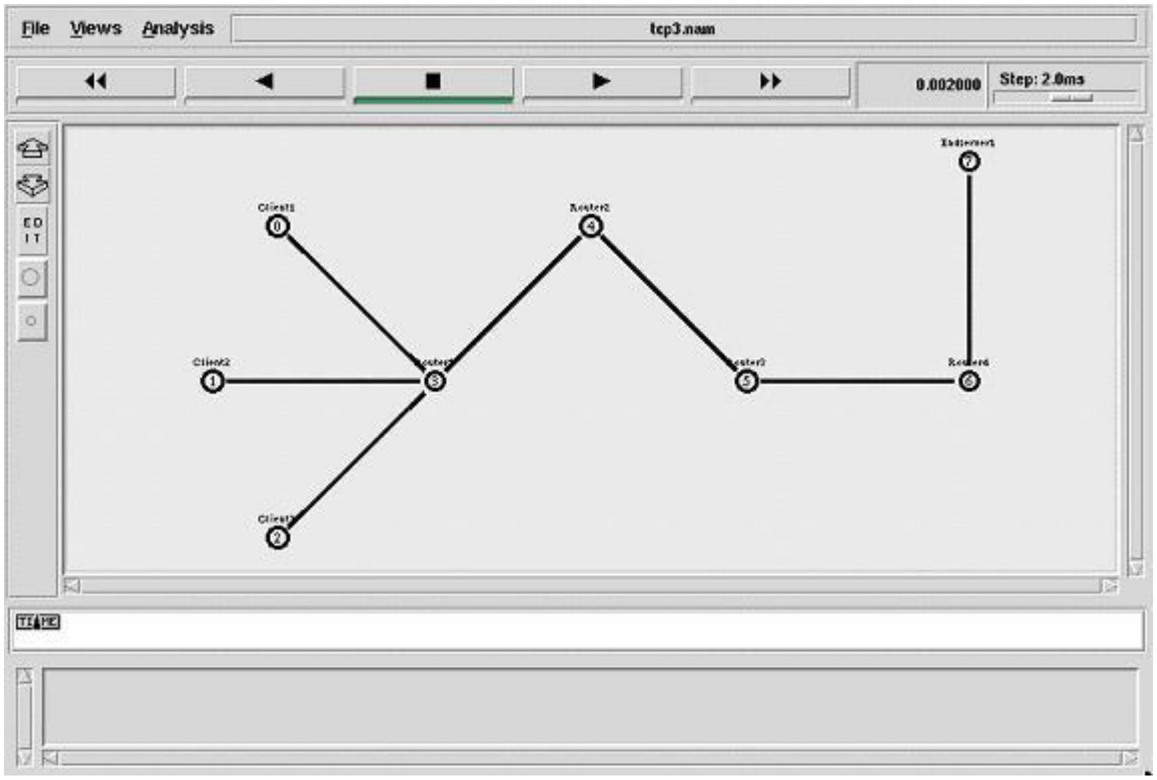
#Calling finish procedure

```
$ns at 35.0 "finish"
$ns run
```

#-----How to run-------------#

```
$ns tcp3.tcl
```

#---------Snapshot-----------#



4. TCL script to set identification color to links.

Description:

This network consists of 9 nodes (C1, C2, C3, C4, R1, R2, R3, R4, ROU1, ROU2 and ROU3). The duplex links between C1, C2, C3, R1, R2, R3 and ROU1 have 2 Mbps of bandwidth and 100 ms of delay. The duplex link between C1 and ROU1 has 1Mbps of bandwidth and 10ms of delay. The duplex link between C2 and ROU1 has 500Kbps of bandwidth and 10ms of delay. The duplex link between C3 and ROU1 has 750Kbps of bandwidth and 10ms of delay. The duplex link between C4 and ROU1 has 1Mbps of bandwidth and 10ms of delay. The duplex link between R1 and ROU1, R2 and ROU1, R3 and ROU1, R4 and ROU3 has 1Mbps of bandwidth and 10ms of delay. Each link uses a DropTail queue.

File name: "tcp4.tcl"

```
#-------Event scheduler object creation--------#


set ns [ new Simulator ]

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

set nf [open tcp4.nam w]
$ns namtrace-all $nf

#Open the trace file
set nt [open tcp4.tr w]
$ns trace-all $nt
```

```
set proto rlm

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

$ns color 1 dodgerblue
$ns color 2 red
$ns color 3 cyan
$ns color 4 green
$ns color 5 yellow
$ns color 6 black
$ns color 7 magenta
$ns color 8 gold
$ns color 9 red

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

set C1 [$ns node]
set C2 [$ns node]
set C3 [$ns node]
set C4 [$ns node]
set R1 [$ns node]
set R2 [$ns node]
set R3 [$ns node]
set R4 [$ns node]
set ROU1 [$ns node]
set ROU2 [$ns node]
set ROU3 [$ns node]

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

$ns duplex-link $C1 $ROU1 1Mb 10ms DropTail
$ns duplex-link $C2 $ROU1 500Kb 10ms DropTail
$ns duplex-link $C3 $ROU1 750Kb 10ms DropTail
$ns duplex-link $C4 $ROU2 1Mb 10ms DropTail
$ns duplex-link $R1 $ROU1 1Mb 10ms DropTail
$ns duplex-link $R2 $ROU1 1Mb 10ms DropTail
$ns duplex-link $R3 $ROU1 1Mb 10ms DropTail
$ns duplex-link $R4 $ROU3 1Mb 10ms DropTail
$ns duplex-link $ROU2 $ROU1 1Mb 10ms DropTail
$ns duplex-link $ROU2 $ROU3 1Mb 10ms DropTail
$ns duplex-link $ROU1 $ROU3 1Mb 10ms DropTail

#-------------QUEUE SIZE DESCRIPTION---------------#

$ns queue-limit $ROU1 $ROU2 18
$ns queue-limit $ROU1 $ROU3 18
$ns queue-limit $ROU2 $ROU1 20
$ns queue-limit $ROU3 $ROU1 20

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


$ns duplex-link-op $C1 $ROU1 orient down
```

```
$ns duplex-link-op $C2 $ROU1 orient down-right
$ns duplex-link-op $C3 $ROU1 orient down-left
$ns duplex-link-op $C4 $ROU2 orient up
$ns duplex-link-op $R1 $ROU1 orient up
$ns duplex-link-op $R2 $ROU1 orient up-right
$ns duplex-link-op $R3 $ROU1 orient up-left
$ns duplex-link-op $R4 $ROU3 orient down
$ns duplex-link-op $ROU1 $ROU2 orient down-right
$ns duplex-link-op $ROU3 $ROU2 orient down-right



# --------------LABELLING -----------------------------#

$ns at 0.0 "$C1 label CL1"
$ns at 0.0 "$C2 label CL2"
$ns at 0.0 "$C3 label CL3"
$ns at 0.0 "$C4 label CL4"
$ns at 0.0 "$R1 label RC1"
$ns at 0.0 "$R2 label RC2"
$ns at 0.0 "$R3 label RC3"
$ns at 0.0 "$R4 label RC4"
$ns at 0.0 "$ROU1 label ROU1"
$ns at 0.0 "$ROU2 label ROU2"
$ns at 0.0 "$ROU3 label ROU3"

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

$ROU1 shape square
$ROU2 shape square
$ROU3 shape square

# ----------------QUEUES POSITIONING AND ESTABLISHMENT -------------#

$ns duplex-link-op $ROU2 $ROU1 queuePos 0.1
#$ns duplex-link-op $ROU2 $C5 queuePos 0.1
$ns duplex-link-op $ROU3 $ROU1 queuePos 0.1



#--------SETTING IDENTIFICATION COLORS TO ROUTER-LINKS----------#

$ns duplex-link-op $ROU1 $ROU2 color cyan
$ns duplex-link-op $ROU1 $ROU3 color cyan
$ns duplex-link-op $ROU2 $ROU3 color cyan



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


proc finish {} {

               global ns nf nt nf1
```

```
                $ns flush-trace
                close $nf
                puts "running nam..."
                exec nam Tcp4.nam &
                exit 0
            }
```

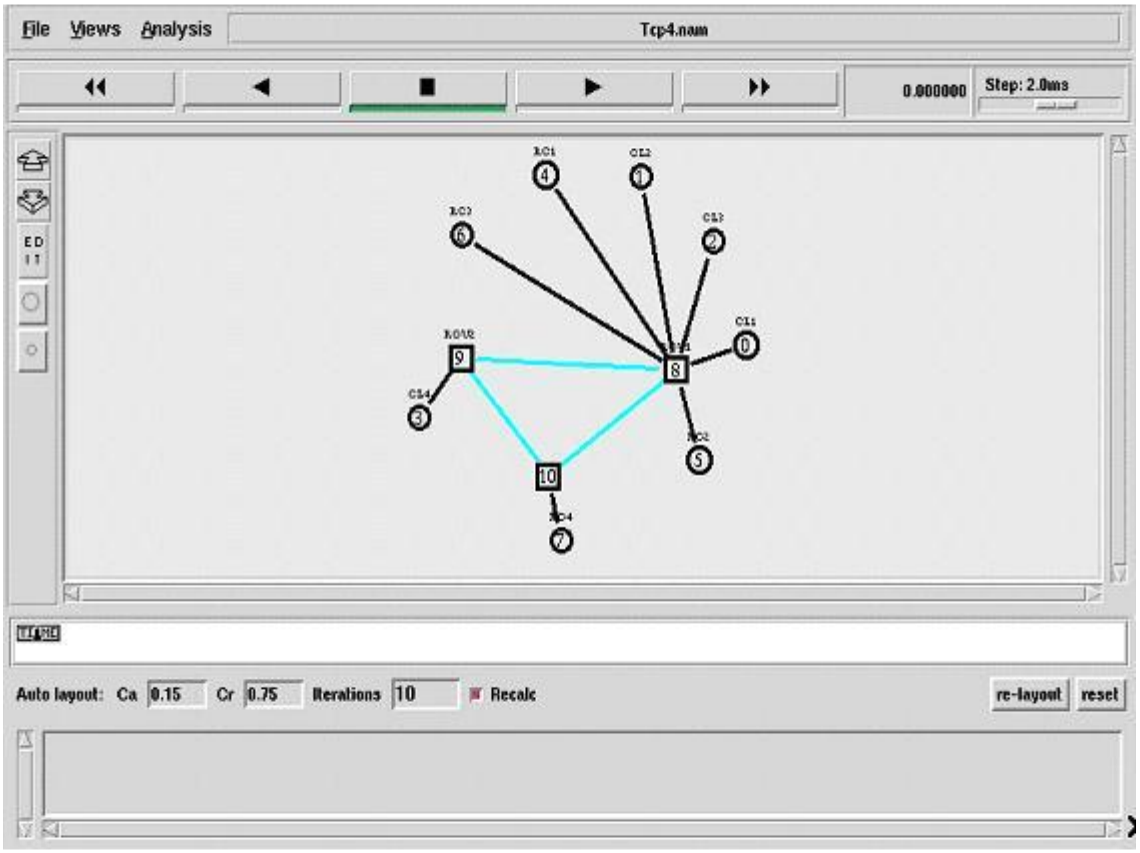
#Calling finish procedure

```
$ns at 20.0 "finish"
$ns run
```

#--------How to run---------#

```
$ns tcp4.tcl
```

#-----------Snapshot------------#



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 Recalc

5. TCL script to create Tcp agent, Tcp sink and attach the Tcp agent with Tcpsink.

Description:

This network consists of 4 nodes (Client1, Router1, Endserver1 and Endserver2). The duplex link between Client1and Router1 has 2 Mbps of bandwidth and 100 ms of delay. The duplex link between Router1 and Endserver1 has 100Kbps of bandwidth and 100 ms of delay. The duplex link between Router1 and Endserver2 has 200Kbps of bandwidth and 100 ms of delay. Each node uses a DropTail queue. A "TCP" agent is attached to Client1. "TCPSink" agent is attached to

Endserver1 and Endserver2. TCP and TCPSink agent is connected. As default, the maximum size of a packet that a "TCP" agent can generate is 1000bytes. A tcp "sink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. The ftp is set to start at 0.50 sec and stop at 8.5 sec.

File name: "tcpred1.tcl"

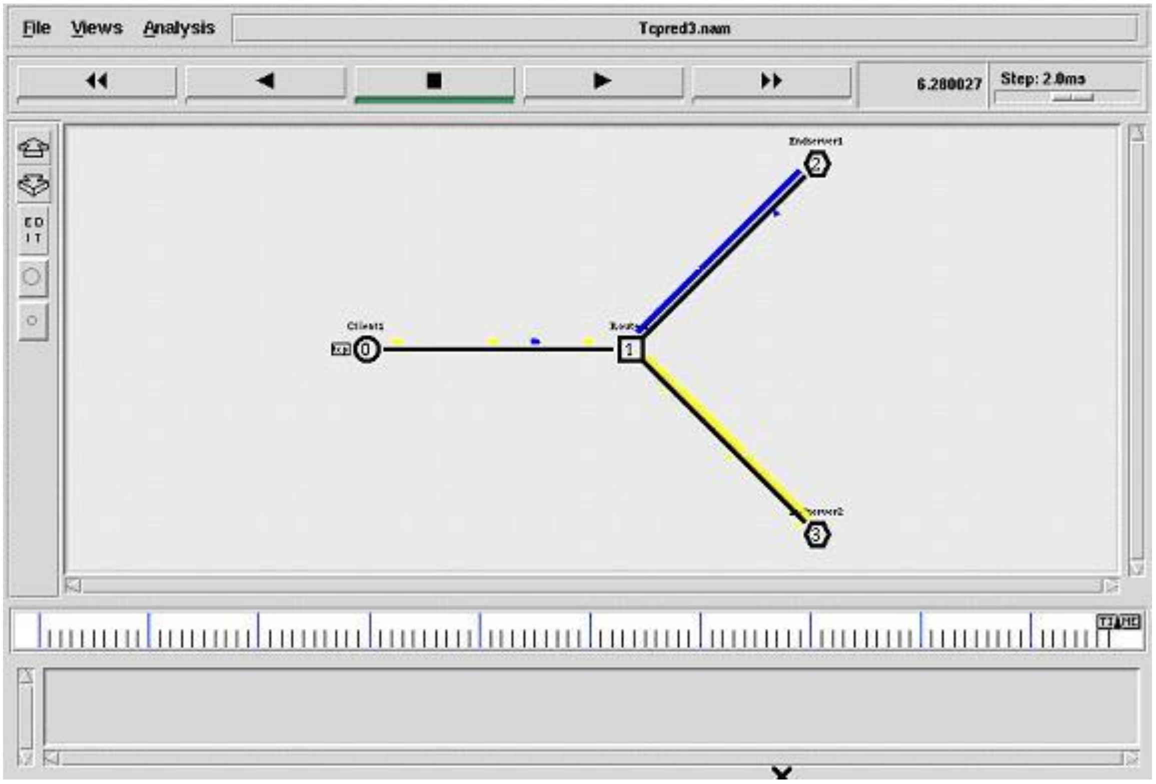
```
#-------Event scheduler object creation--------#
set ns [new Simulator]

#----------creating nam objects----------------#

set nf [open tcpred1.nam w]
$ns namtrace-all $nf

#open the trace file
set nt [open tcpred1.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 [$ns node]
set Endserver2 [$ns node]

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

$ns duplex-link $Client1 $Router1 2Mb 100ms DropTail
$ns duplex-link $Router1 $Endserver1 100Kb 100ms DropTail
$ns duplex-link $Router1 $Endserver2 200Kb 100ms DropTail

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

$ns duplex-link-op $Client1 $Router1 orient right
$ns duplex-link-op $Router1 $Endserver1 orient up-right
$ns duplex-link-op $Router1 $Endserver2 orient down-right




#------------Labeling----------------#

$ns at 0.0 "$Client1 label Client1"
$ns at 0.0 "$Router1 label Router1"
$ns at 0.0 "$Endserver1 label Endserver1"
$ns at 0.0 "$Endserver2 label Endserver2"
```

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

$Endserver1 shape hexagon
$Endserver2 shape hexagon
$Router1 shape square

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

#$ns duplex-link-op $Client1 $Router1 queuePos 0.1
#$ns duplex-link-op $Client2 $Router1 queuePos 0.1
#$ns duplex-link-op $Router1 $Router2 queuePos 0.1
#$ns duplex-link-op $Router2 $Endserver1 queuePos 0.5



#---------Establishing communication-------------#

#-------------Client1 to Endserver1---#

set tcp0 [new Agent/TCP]
$tcp0 set maxcwnd_ 16
$tcp0 set fid_ 1
$ns attach-agent $Client1 $tcp0
set sink0 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink0
$ns connect $tcp0 $sink0
set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ns add-agent-trace $tcp0 tcp
$tcp0 tracevar cwnd_
$ns at 0.50 "$ftp0 start"
$ns at 8.5 "$ftp0 stop"

#-------------Client1 to Endserver2---#

set tcp1 [new Agent/TCP]
$tcp1 set maxcwnd_ 16
$tcp1 set fid_ 2
$ns attach-agent $Client1 $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $Endserver2 $sink1
$ns connect $tcp1 $sink1
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1
$ns add-agent-trace $tcp1 tcp
$tcp0 tracevar cwnd_
$ns at 0.50 "$ftp1 start"
$ns at 8.5 "$ftp1 stop"



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

```
proc finish {} {
         global ns nf nt
         $ns flush-trace
         close $nf
         puts "running nam..."
         exec nam Tcpred1.nam &
         exit 0
      }

#Calling finish procedure
$ns at 10.0 "finish"
$ns run
```

#--------- how to execute the program-----------#
$ ns tcpred1.tcl

#--------------output--------------#



6.TCL script for TCP communication between two Clients and a Endserver.

Description:

This network consists of 4 nodes (Client1, Client2, Router1, Rounter2 and Endserver1). The duplex links between Client1 Client2 and Router1have 2 Mbps of bandwidth and 100 ms of delay. The duplex link between Router1 and Router2 has 2Mbps of bandwidth and 100 ms of delay. The duplex link between Router2 and Endserver1 has 200Kbps of bandwidth and 100 ms of delay. Each link uses a Drop Tail queue. A

"TCP" agent is attached to Client1, and Client2. "TCPSink" agent is attached to Endserver1. Both the agents are connected. 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 ftp is set to start at 0.5 sec and stop at 5.5 sec.

<u>File name: "tcpred1.tcl"</u>

```
#-------Event scheduler object creation--------#

set ns [new Simulator]

#----------creating nam objects----------------#

set nf [open tcpred1.nam w]
$ns namtrace-all $nf

#open the trace file
set nt [open tcpred1.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 [$ns node]
set Client2 [$ns node]
set Router1 [$ns node]
set Router2 [$ns node]
set Endserver1 [$ns node]



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

$ns duplex-link $Client1 $Router1 2Mb 100ms DropTail
$ns duplex-link $Client2 $Router1 2Mb 100ms DropTail
$ns duplex-link $Router1 $Router2 2Mb 100ms DropTail
$ns duplex-link $Router2 $Endserver1 200Kb 100ms DropTail



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

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

#------------Labelling----------------#
```

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

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

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

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

$ns duplex-link-op $Client1 $Router1 queuePos 0.1
$ns duplex-link-op $Client2 $Router1 queuePos 0.1
$ns duplex-link-op $Router1 $Router2 queuePos 0.1
$ns duplex-link-op $Router2 $Endserver1 queuePos 0.5



#---------Establishing communication-------------#
#-------------Client1 to Endserver1---#

set tcp0 [new Agent/TCP]
$tcp0 set maxcwnd_ 16
$tcp0 set fid_ 1
$ns attach-agent $Client1 $tcp0

set sink0 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink0

$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns add-agent-trace $tcp0 tcp
$tcp0 tracevar cwnd_

$ns at 0.5 "$ftp0 start"
$ns at 5.5 "$ftp0 stop"



set tcp1 [new Agent/TCP]
$tcp1 set maxcwnd_ 16
$tcp1 set fid_ 2
$ns attach-agent $Client2 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink1
```

```
$ns connect $tcp1 $sink1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

$ns add-agent-trace $tcp1 tcp
$tcp1 tracevar cwnd_

$ns at 0.5 "$ftp1 start"
$ns at 5.5 "$ftp1 stop"
```

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

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

         puts "running nam..."
         exec nam tcpred1.nam &
         exit 0
      }
```

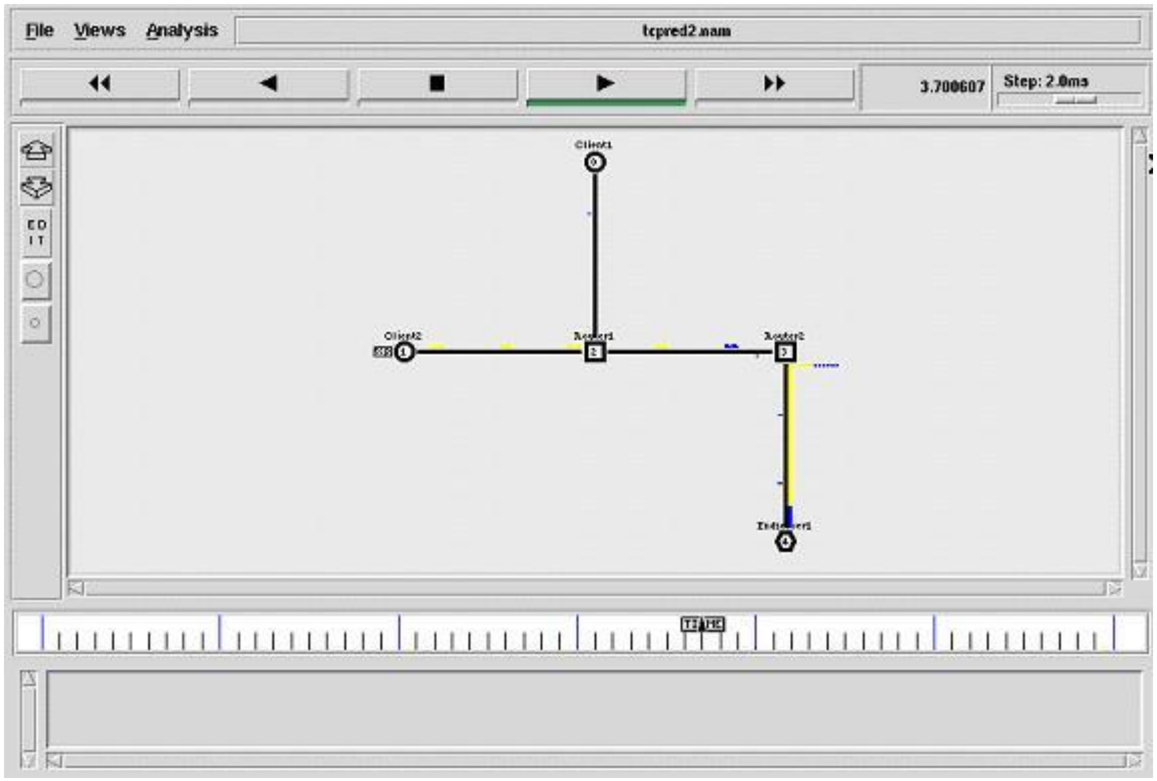
#Calling finish procedure

```
$ns at 6.0 "finish"
$ns run
```

#-------- How to run-----------#

```
$ns tcpred2.tcl
```

#---------- snapshot-----------#



7.TCL script for TCP communication between four Clients and a Endserver.

Description:

This network consists of 8 nodes (Client1, Client2, Client3, Client4, Router1, Router2 Router3, Router4, Router5, Router6 and Endserver1,). The duplex links between Client1, Client2, Client3, Client4 and Router1 have 5 Mbps of bandwidth and 50 ms of delay. The duplex link between Router1 and Router2 has 5Mbps of bandwidth and 50 ms of delay. The duplex link between Router2 and Router3 has 150Kbps of bandwidth and 50 ms of delay and etc. Finally the duplex link between Router6 and Endserver1 has 300Kbps of bandwidth and 50 ms of delay. Each link uses a DropTail queue. A "TCP" agent is attached to Client1, Client2, Client3, Client4 and a connection is established to a "TCPSink" 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 ftp is set to start at 0.50 sec and stop at 28.5 sec.

File name: "tcpred3.tcl"

```
#-------Event scheduler object creation--------#


set ns [ new Simulator]

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

```
set nf [open Tcpred3.nam w]
$ns namtrace-all $nf

#Open the trace file
set nt [open Tcpred3.tr w]
$ns trace-all $nt

set proto rlm

$ns color 1 red
$ns color 2 blue
$ns color 3 yellow
$ns color 4 cyan
$ns color 5 maroon

# --------- CREATING CLIENT - ROUTER -ENDSERVER NODES-----------#

set Client1 [$ns node]
set Client2 [$ns node]
set Client3 [$ns node]
set Client4 [$ns node]
set Router1 [$ns node]
set Router2 [$ns node]
set Router3 [$ns node]
set Router4 [$ns node]
set Router5 [$ns node]
set Router6 [$ns node]
set Endserver1 [$ns node]

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

$ns duplex-link $Client1 $Router1  5Mb 50ms DropTail
$ns duplex-link $Client2 $Router1  5Mb 50ms DropTail
$ns duplex-link $Client3 $Router1 5Mb 50ms DropTail
$ns duplex-link $Client4 $Router1 5Mb 50ms DropTail
$ns duplex-link $Router1 $Router2 5Mb 50ms DropTail
$ns duplex-link $Router2 $Router3 150Kb 50ms DropTail
$ns duplex-link $Router3 $Router4 300Kb 50ms DropTail
$ns duplex-link $Router4 $Router5 100Kb 50ms DropTail
$ns duplex-link $Router5 $Router6 300Kb 50ms DropTail
$ns duplex-link $Router6 $Endserver1 300Kb 50ms DropTail
#$ns duplex-link $Router6 $Endserver2 300Kb 50ms DropTail



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

$ns duplex-link-op $Client1 $Router1 orient down-right
$ns duplex-link-op $Client2 $Router1 orient right
$ns duplex-link-op $Client3 $Router1 orient up-right
$ns duplex-link-op $Client4 $Router1 orient up
$ns duplex-link-op $Router1 $Router2 orient right
$ns duplex-link-op $Router2 $Router3 orient down
$ns duplex-link-op $Router3 $Router4 orient right
$ns duplex-link-op $Router4 $Router5 orient up
$ns duplex-link-op $Router5 $Router6 orient right
```

```
$ns duplex-link-op $Router6 $Endserver1 orient up-right
#$ns duplex-link-op $Router6 $Endserver2 orient right



# --------------CREATING LABELLING -----------------------------#

$ns at 0.0 "$Client1 label Client1"
$ns at 0.0 "$Client2 label Client2"
$ns at 0.0 "$Client3 label Client3"
$ns at 0.0 "$Client4 label Client4"
$ns at 0.0 "$Router1 label Router1"
$ns at 0.0 "$Router2 label Router2"
$ns at 0.0 "$Router3 label Router3"
$ns at 0.0 "$Router4 label Router4"
$ns at 0.0 "$Router5 label Router5"
$ns at 0.0 "$Router6 label Router6"
$ns at 0.0 "$Endserver1 label Endserver"
#$ns at 0.0 "$Endserver2 label Endserver2"

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

$Endserver1 shape hexagon
$Router1 shape box
$Router2 shape square
$Router3 shape square
$Router4 shape square
$Router5 shape square
$Router6 shape square

# ----------------ESTABLISHING QUEUES -------------#

$ns duplex-link-op $Client1 $Router1 queuePos 0.1
$ns duplex-link-op $Client2 $Router1 queuePos 0.1
$ns duplex-link-op $Client3 $Router1 queuePos 0.5
$ns duplex-link-op $Client4 $Router1 queuePos 0.5
$ns duplex-link-op $Router1 $Router2 queuePos 0.1
$ns duplex-link-op $Router2 $Router3 queuePos 0.1
$ns duplex-link-op $Router3 $Router4 queuePos 0.1
$ns duplex-link-op $Router4 $Router5 queuePos 0.1
$ns duplex-link-op $Router5 $Router6 queuePos 0.5
$ns duplex-link-op $Router6 $Endserver1 queuePos 0.5

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

#-------CLIENT1 TO ENDSERVER1 -------------#

set tcp0 [new Agent/TCP]
$tcp0 set maxcwnd_ 16
$tcp0 set fid_ 4
$ns attach-agent $Client1 $tcp0

set sink0 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink0
```

```
$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns add-agent-trace $tcp0 tcp
$tcp0 tracevar cwnd_

$ns at 0.5 "$ftp0 start"
$ns at 28.5 "$ftp0 stop"
```

# ---------------- CLIENT2 TO ENDSERVER1 -------------#

```
set tcp1 [new Agent/TCP]
$tcp1 set fid_ 2
$tcp1 set maxcwnd_ 16
$ns attach-agent $Client2 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink1

$ns connect $tcp1 $sink1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

$ns add-agent-trace $tcp1 tcp1
$tcp1 tracevar cwnd_

$ns at 0.58 "$ftp1 start"
$ns at 28.5 "$ftp1 stop"
```

# ----------------CLIENT3 TO ENDSERVER1------------#

```
set tcp2 [new Agent/TCP]
$tcp2 set fid_ 0
$tcp2 set maxcwnd_ 16
$tcp2 set packetsize_ 100
$ns attach-agent $Client3 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink2
$ns connect $tcp2 $sink2

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ns add-agent-trace $tcp2 tcp2
$tcp2 tracevar cwnd_

$ns at 0.65 "$ftp2 start"
$ns at 28.5 "$ftp2 stop"
```

#-------------------CLIENT4 TO ENDSERVER1---------------#

```
set tcp3 [new Agent/TCP]
$tcp3 set fid_ 3
$tcp3 set maxcwnd_ 16
$tcp2 set packetsize_ 100
$ns attach-agent $Client4 $tcp3

set sink3 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink3

$ns connect $tcp3 $sink3

set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3

$ns add-agent-trace $tcp3 tcp3
$tcp3 tracevar cwnd_

$ns at 0.60 "$ftp3 start"
$ns at 28.5 "$ftp3 stop"
```

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

```
proc finish {} {



               global ns nf nt

               $ns flush-trace
               close $nf
               puts "running nam..."
               exec nam Tcpred3.nam &
               exit 0
            }
```

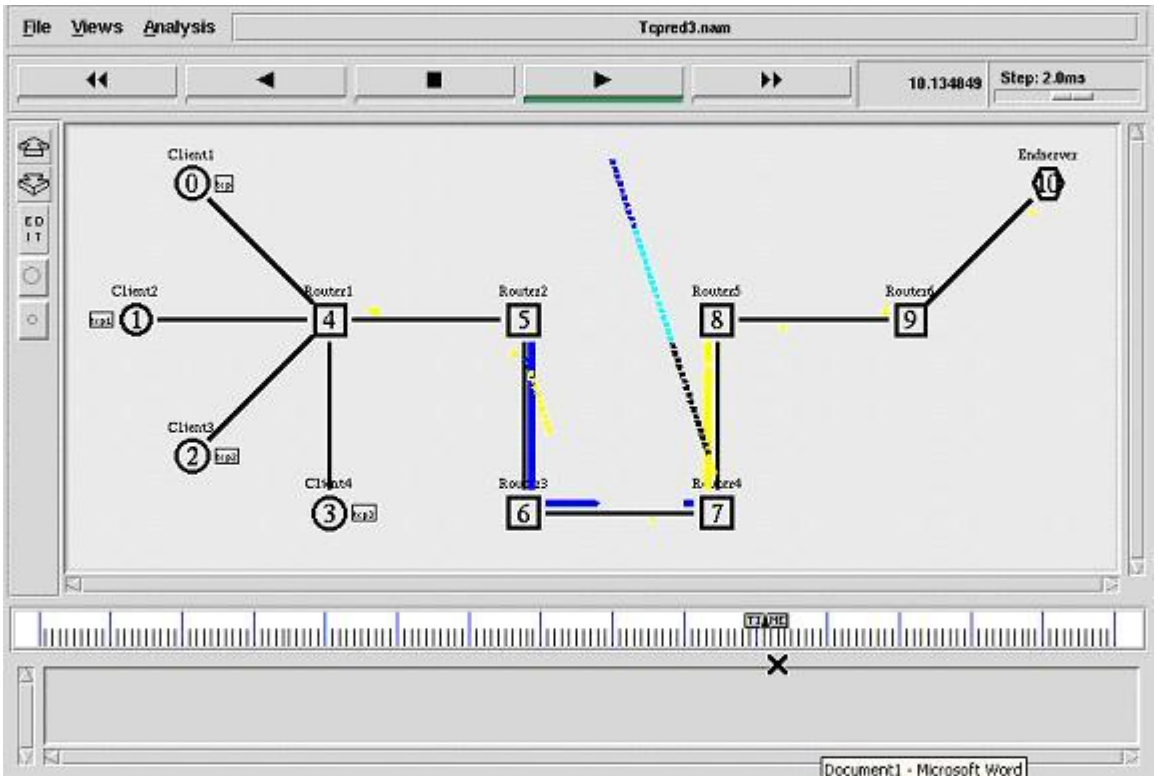
#Calling finish procedure

```
$ns at 15.0 "finish"
$ns run
```

#----------How to run-------------#

```
$ns tcpred4.tcl
```

#------------- Snapshot-------------#



8.TCL script for TCP communication between more numbers of nodes.

Description:

This network consists of 9 nodes (C1, C2, C3, C4, R1, R2, R3, R4, ROU1, ROU2 and ROU3). The duplex link between the links is configured with specific bandwidth and delay. Each link uses a DropTail queue. A "TCP" agent is attached to C1 and a connection is established to a tcp "TCPSink" agent attached to R4. A "TCP" agent is attached to C2 and a connection is established to a tcp "TCPSink" agent attached to R3. A agent is attached to C3 and a connection is established to a "TCPSink" agent attached to R2As default, the maximum size of a packet that a "TCP" agent can generate is 1000bytes. A "TCP" agent is attached to C4 and a connection is established to a "TCPSink" agent attached to R1 A "TCPSink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. Application ftp is attached to all "TCP" agents for packet generation. The ftp is set to start at 0.2 sec and stop at 4.0 sec.

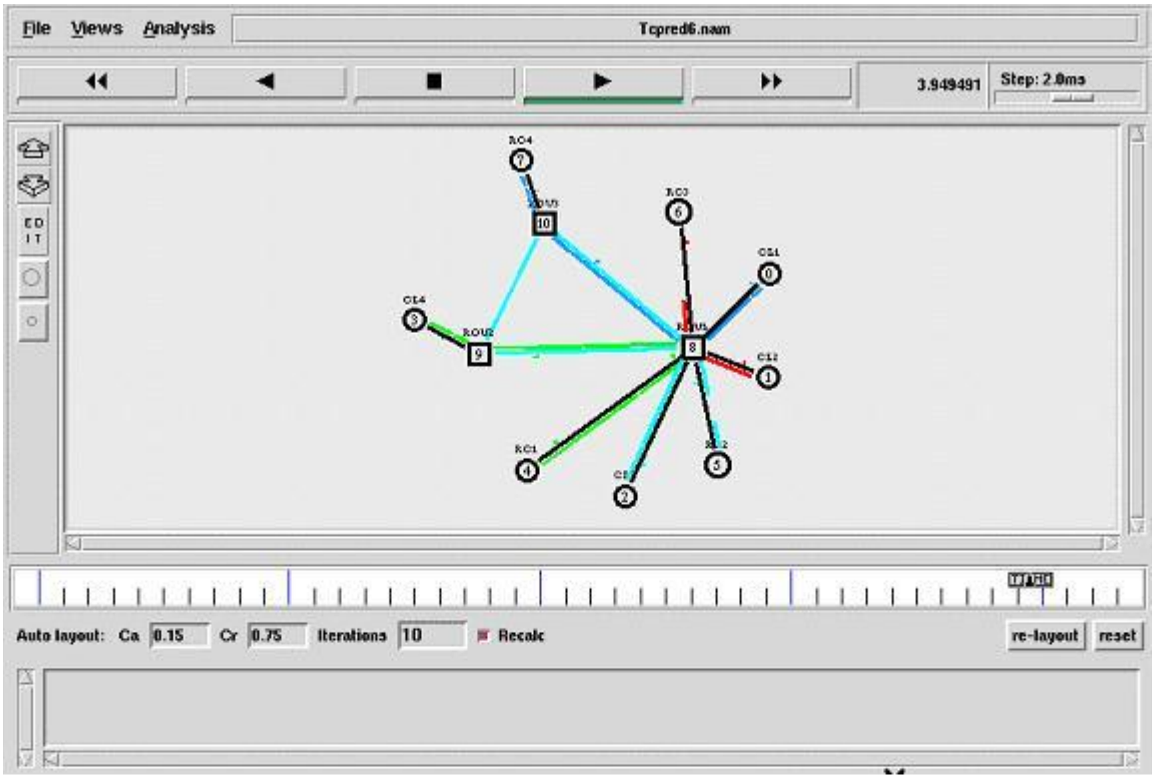
File name: "tcpred5.tcl"

```
#-------Event scheduler object creation--------#

set ns [ new Simulator

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


set nf [open tcpred5.nam w]
$ns namtrace-all $nf
```

```
#Open the trace file
set nt [open tcpred5.tr w]
$ns trace-all $nt



set proto rlm

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

$ns color 1 dodgerblue
$ns color 2 red
$ns color 3 cyan
$ns color 4 green
$ns color 5 yellow
$ns color 6 black
$ns color 7 magenta
$ns color 8 gold
$ns color 9 red



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

set C1 [$ns node]
set C2 [$ns node]
set C3 [$ns node]
set C4 [$ns node]
set R1 [$ns node]
set R2 [$ns node]
set R3 [$ns node]
set R4 [$ns node]
set ROU1 [$ns node]
set ROU2 [$ns node]
set ROU3 [$ns node]



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

$ns duplex-link $C1 $ROU1 1Mb 10ms DropTail
$ns duplex-link $C2 $ROU1 500Kb 10ms DropTail
$ns duplex-link $C3 $ROU1 750Kb 10ms DropTail
$ns duplex-link $C4 $ROU2 1Mb 10ms DropTail
$ns duplex-link $R1 $ROU1 1Mb 10ms DropTail
$ns duplex-link $R2 $ROU1 1Mb 10ms DropTail
$ns duplex-link $R3 $ROU1 1Mb 10ms DropTail
$ns duplex-link $R4 $ROU3 1Mb 10ms DropTail

$ns duplex-link $ROU2 $ROU1 1Mb 10ms DropTail
$ns duplex-link $ROU2 $ROU3 1Mb 10ms DropTail
$ns duplex-link $ROU1 $ROU3 1Mb 10ms DropTail

#-------------QUEUE SIZE DESCRIPTION---------------#
```

```
$ns queue-limit $ROU1 $ROU2 18
$ns queue-limit $ROU1 $ROU3 18
$ns queue-limit $ROU2 $ROU1 20
$ns queue-limit $ROU3 $ROU1 20

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


$ns duplex-link-op $C1 $ROU1 orient down
$ns duplex-link-op $C2 $ROU1 orient down-right
$ns duplex-link-op $C3 $ROU1 orient down-left
$ns duplex-link-op $C4 $ROU2 orient up
$ns duplex-link-op $R1 $ROU1 orient up
$ns duplex-link-op $R2 $ROU1 orient up-right
$ns duplex-link-op $R3 $ROU1 orient up-left
$ns duplex-link-op $R4 $ROU3 orient down

$ns duplex-link-op $ROU1 $ROU2 orient down-right
$ns duplex-link-op $ROU3 $ROU2 orient down-right

#$ns queue-limit $ $n1 15



# --------------LABELLING -----------------------------#

$ns at 0.0 "$C1 label CL1"
$ns at 0.0 "$C2 label CL2"
$ns at 0.0 "$C3 label CL3"
$ns at 0.0 "$C4 label CL4"
$ns at 0.0 "$R1 label RC1"
$ns at 0.0 "$R2 label RC2"
$ns at 0.0 "$R3 label RC3"
$ns at 0.0 "$R4 label RC4"
$ns at 0.0 "$ROU1 label ROU1"
$ns at 0.0 "$ROU2 label ROU2"
$ns at 0.0 "$ROU3 label ROU3"

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

$ROU1 shape square
$ROU2 shape square
$ROU3 shape square

# ----------------QUEUES POSITIONING AND ESTABLISHMENT -------------#

$ns duplex-link-op $ROU2 $ROU1 queuePos 0.1
#$ns duplex-link-op $ROU2 $C5 queuePos 0.1
$ns duplex-link-op $ROU3 $ROU1 queuePos 0.1



#--------SETTING IDENTIFICATION COLORS TO ROUTER-LINKS----------#

$ns duplex-link-op $ROU1 $ROU2 color cyan
```

```
$ns duplex-link-op $ROU1 $ROU3 color cyan
$ns duplex-link-op $ROU2 $ROU3 color cyan



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

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

      #$tcp0 set fid_ 3
      #$Base1 set fid_ 3
      #$tcp0 set window_ 15
      #$ftp0 set packetSize_ 1000
      #$ftp0 set interval_ .05

      set tcp1 [$ns create-connection TCP $C1 TCPSink $R4 1]
      $tcp1 set class_ 1
      $tcp1 set maxcwnd_ 16
      $tcp1 set packetsize_ 4000
      $tcp1 set fid_ 1
      set ftp1 [$tcp1 attach-app FTP]
      $ftp1 set interval_ .005
      $ns at 0.2 "$ftp1 start"
      $ns at 4.0 "$ftp1 stop"

      set tcp2 [$ns create-connection TCP $C2 TCPSink $R3 1]
      $tcp2 set class_ 1
      $tcp2 set maxcwnd_ 16
      $tcp2 set packetsize_ 4000
      $tcp2 set fid_ 2
      set ftp2 [$tcp2 attach-app FTP]
      $ftp2 set interval_ .005
      $ns at 0.7 "$ftp2 start"
      $ns at 4.0 "$ftp2 stop"

      set tcp3 [$ns create-connection TCP $C3 TCPSink $R2 1]
      $tcp3 set class_ 1
      $tcp3 set maxcwnd_ 16
      $tcp3 set packetsize_ 4000
      $tcp3 set fid_ 3
      set ftp3 [$tcp3 attach-app FTP]
      $ftp3 set interval_ .005
      $ns at 1.2 "$ftp3 start"
      $ns at 4.0 "$ftp3 stop"

      set tcp4 [$ns create-connection TCP $C4 TCPSink $R1 1]
      $tcp4 set class_ 1
      $tcp4 set maxcwnd_ 16
      $tcp4 set packetsize_ 4000
      $tcp4 set fid_ 4
      set ftp4 [$tcp4 attach-app FTP]
      $ftp1 set interval_ .005
      $ns at 2.5 "$ftp4 start"
      $ns at 4.0 "$ftp4 stop"
      # ---------------- FINISH PROCEDURE -------------#

       proc finish {} {

               global ns nf nt nf1
               $ns flush-trace
               close $nf
               puts "running nam..."
               exec nam Tcpred5.nam &
               exit 0
            }

      #Calling finish procedure

$ns at 20.0 "finish"
$ns run

#-----How to run-----#

$ns tcpred5.tcl

#----------Snapshot----------#
```



9. TCL script for UDP communication.

Description:

This network consists of 6 nodes. The duplex links between nodes are configured with the specific bandwidth and delay. Each link uses a

DropTail queue. A "TCP" agent is attached to n0 and a connection is established to a tcp "sink" agent attached to n4. As default, the maximum size of a packet that a "TCP" agent can generate is 1KByte. A "TCPSink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. An "UDP" agent is attached to n1 and a connection is established to udp "Null" agent attached to n5. "TCP" agent is attached with "FTP" application and "UDP" agent is attached with "CBR" traffic for packet generation. The ftp is set to start at 1.0 sec and stop at 10.5 sec. The cbr is set to start at 0.1 sec and stop at 10.0 sec.

<u>File name: "tcpred4.tcl"</u>

```
#-------Event scheduler object creation--------#



set ns [new Simulator]



#---Open the Trace files---#

set file1 [open Tcpred4.tr w]
$ns trace-all $file1

#--Open the NAM trace file----#

set file2 [open Tcpred4.nam w]
$ns namtrace-all $file2

#Define different colors for data flows (for NAM)

$ns color 1 Blue
$ns color 2 Red

#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]

#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
$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail

#Give node position (for NAM)
```

```
$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
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down
```

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

```
$ns queue-limit $n2 $n3 20
```

#Setup a 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 window_ 8000
$tcp set packetSize_ 552

#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 $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

$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 10.0 "$ftp stop"
$ns at 10.5 "$cbr stop"
```

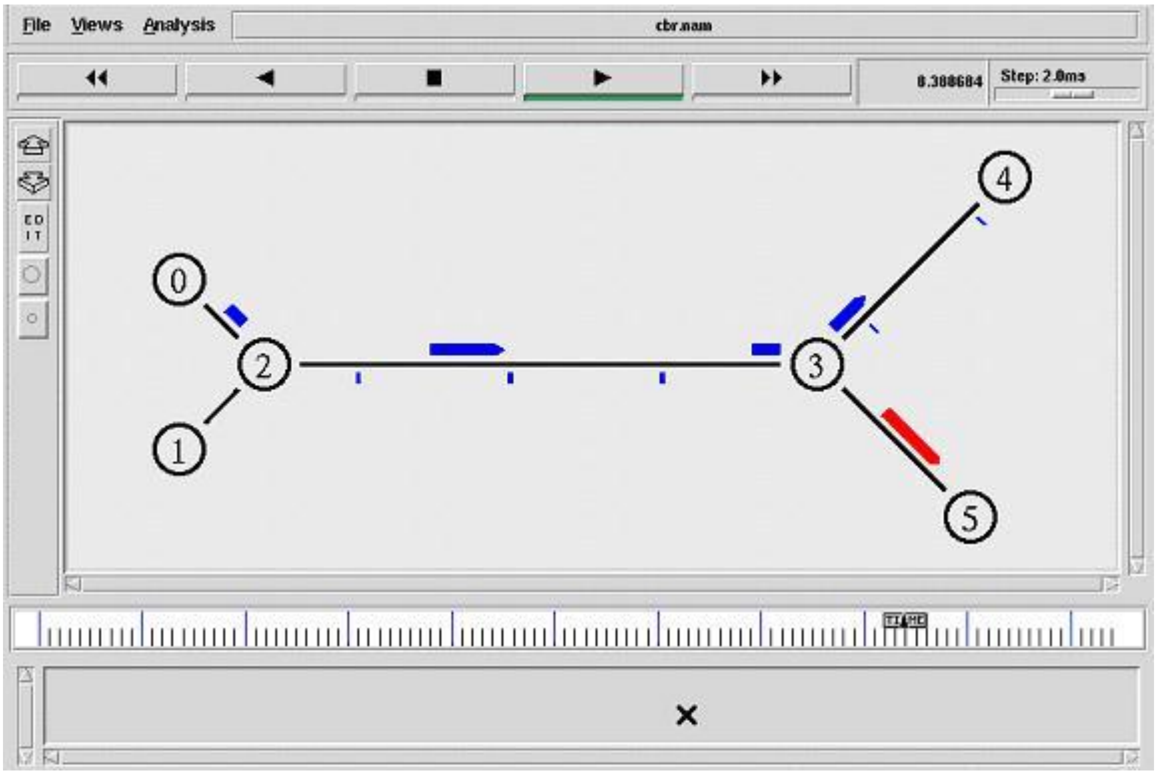
#Define a 'finish' procedure

```
proc finish {} {
        global ns file1 file2
        $ns flush-trace
        close $file1
        close $file2
        exec nam Tcpred4.nam &
        exit 0
}



$ns at 12.0 "finish"
$ns run
```

#----------How to run----------#

```
$ns tcpred4.tcl
```

#--------Snapshot--------#



10. TCL script to drop down the packets in particular link at specific time.

Description:

This network consists of 5 nodes (Client1, Client2, Router1, Router2 and Endserver1). The duplex links between

Client1 and Client2 and Router1 have 5Mbps of bandwidth and 50 ms of delay. The duplex link between Router1 and Router2 has 150Kbps of bandwidth and 50 ms of delay. The duplex link between Router2 and Endserver1 has 300Kbps of bandwidth and 50 ms of delay. Each link uses a DropTail queue. A "TCP" agent is attached to Client1, and Client2 connection is established to a "TCPSink" 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. Link failure model is created at 2.88 between Router1 and Router2. The packets are dropped down between Router1 to Router2 at 2.88 sec. FTP application is attached to "TCP" agent. The ftp is set to start at 0.5 sec and stop at 6.5 sec.

File name: "drop1.tcl"

```
#-------Event scheduler object creation--------#

set ns [new Simulator]

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

set nf [open drop1.nam w]
$ns namtrace-all $nf

set nt [open drop1.tr w]
$ns trace-all $nt

set proto rlm
$ns color 1 red
$ns color 2 cgyan

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

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

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

$ns duplex-link $Client1 $Router1 5Mb 50ms DropTail
$ns duplex-link $Client2 $Router1 5Mb 50ms DropTail
$ns duplex-link $Router1 $Router2 150Kb 50ms DropTail
$ns duplex-link $Router2 $Endserver1 300Kb 50ms DropTail

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

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

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

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

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

# ----------------QUEUES POSITIONING AND ESTABLISHMENT -------------#

```
$ns duplex-link-op $Client1 $Router1 queuePos 0.1
$ns duplex-link-op $Client2 $Router1 queuePos 0.1
$ns duplex-link-op $Router1 $Router2 queuePos 0.5
$ns duplex-link-op $Router2 $Endserver1 queuePos 0.5
```

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

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

```
set tcp0 [new Agent/TCP]
$tcp0 set maxcwnd_ 16
$tcp0 set fid_ 1
$ns attach-agent $Client1 $tcp0

set sink0 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink0

$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns add-agent-trace $tcp0 tcp
$tcp0 tracevar cwnd_

$ns at 0.5 "$ftp0 start"
$ns at 5.5 "$ftp0 stop"
#-----------------------------------------#

set tcp1 [new Agent/TCP]
$tcp1 set maxcwnd_ 16
$tcp1 set fid_ 2
$ns attach-agent $Client2 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink1

$ns connect $tcp1 $sink1
```

```
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

$ns add-agent-trace $tcp1 tcp
$tcp1 tracevar cwnd_

$ns at 0.5 "$ftp1 start"
$ns at 6.5 "$ftp1 stop"

#----------------------drop nodes---------------------------#

$ns rtmodel-at 2.88 down $Router1 $Router2
$ns rtmodel-at 3.52 up $Router2 $Endserver1
```

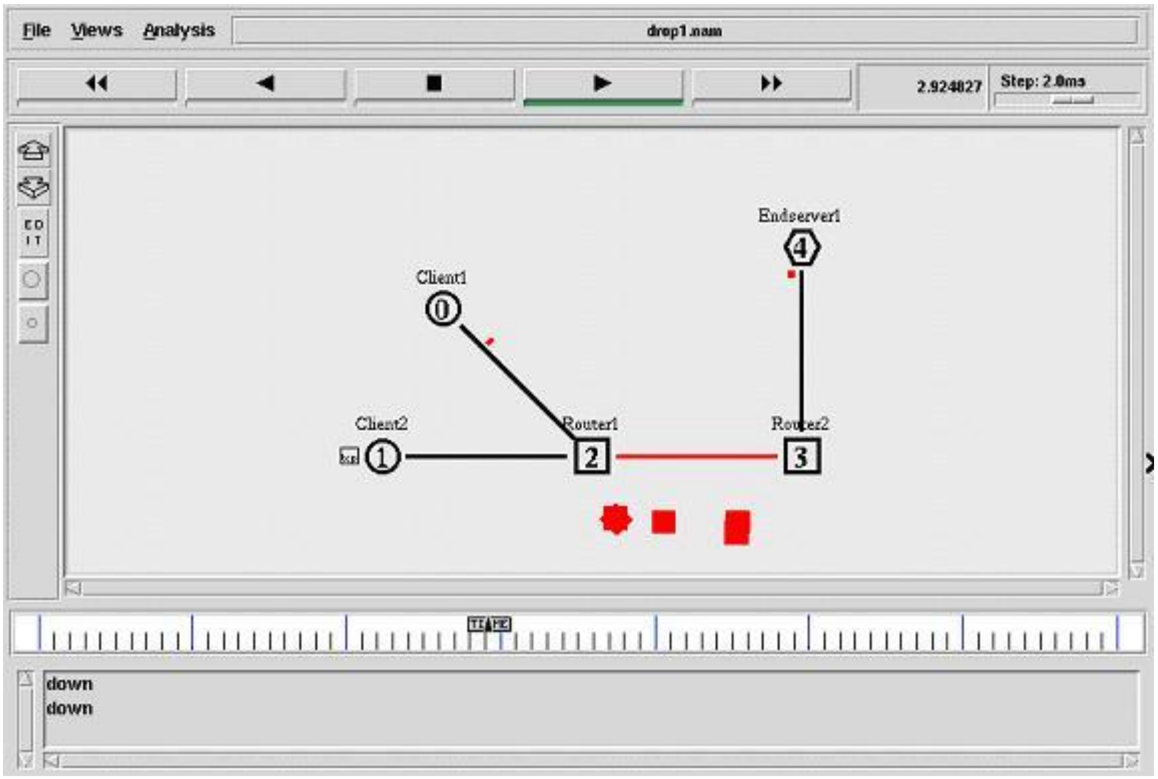
# ---------------- FINISH PROCEDURE -------------#

```
proc finish {} {

 global ns nf nt
 $ns flush-trace
 close $nf
 puts "running nam..."
 exec nam drop1.nam &
 exit 0
 }

#Calling finish procedure
$ns at 7.0 "finish"
$ns run
```

#-----How to run-----#

```
$ns drop1.tcl
```

#----------Snapshot-----------#



```
down
down
```

11. TCL script to drop the packets in router and Endserver link at 1.6sec.

Description:

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.

File name: "drop2.tcl"

```
#-------Event scheduler object creation--------#

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"
```

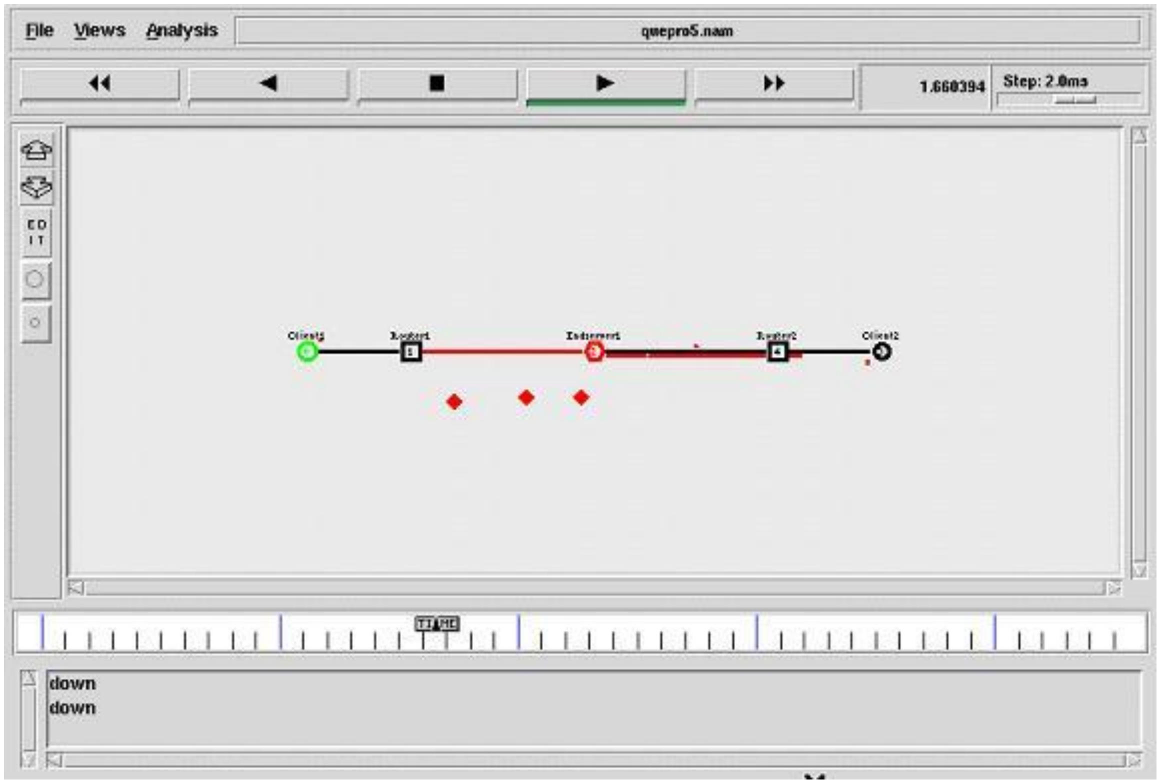
# ---------------- FINISH PROCEDURE -------------#

```
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
```

#-----How to run-----#

```
$ns drop2.tcl
```

#----------Snapshot-----------#



```
down
down
```

12.TCL script to drop down the packets in same link at particular time intervals.

Description:

This network consists of 8 nodes (Client1, Client2, Client3, Client4, Router1, Router2 Router3, Router4, Router5, Router6 and Endserver1). The duplex links between Client1, Client2, Client3, Client4 and Router1 have 5 Mbps of bandwidth and 50 ms of delay. The duplex link between routers and Endserver are configured with specific bandwidth and delay. Each link uses a DropTail queue. A "TCP" agent is attached to Client1, Client2, Client3, Client4 and a connection is established to a "TCPSink" 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 packet dropped down between the Router3 and Router4 at 2.88 and again it dropped down between the Router4 and Router5 at 7.28 sec. The ftp is set to start at 0.60 sec and stop at 28.5 sec.

File name: “drop3.tcl”

```
#-------Event scheduler object creation--------#



set ns [ new Simulator]

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

```
set nf [open drop3.nam w]
$ns namtrace-all $nf

#Open the trace file
set nt [open drop3.tr w]
$ns trace-all $nt

set proto rlm

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

$ns color 1 red
$ns color 2 blue
$ns color 3 yellow
$ns color 4 cyan
$ns color 5 maroon

# --------- CREATING CLIENT - ROUTER -ENDSERVER NODES-----------#

set Client1 [$ns node]
set Client2 [$ns node]
set Client3 [$ns node]
set Client4 [$ns node]
set Router1 [$ns node]
set Router2 [$ns node]
set Router3 [$ns node]
set Router4 [$ns node]
set Router5 [$ns node]
set Router6 [$ns node]
set Endserver1 [$ns node]
#set Endserver2 [$ns node]

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

$ns duplex-link $Client1 $Router1 5Mb 50ms DropTail
$ns duplex-link $Client2 $Router1 5Mb 50ms DropTail
$ns duplex-link $Client3 $Router1 5Mb 50ms DropTail
$ns duplex-link $Client4 $Router1 5Mb 50ms DropTail
$ns duplex-link $Router1 $Router2 5Mb 50ms DropTail
$ns duplex-link $Router2 $Router3 150Kb 50ms DropTail
$ns duplex-link $Router3 $Router4 300Kb 50ms DropTail
$ns duplex-link $Router4 $Router5 100Kb 50ms DropTail
$ns duplex-link $Router5 $Router6 300Kb 50ms DropTail
$ns duplex-link $Router6 $Endserver1 300Kb 50ms DropTail
#$ns duplex-link $Router6 $Endserver2 300Kb 50ms DropTail



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

$ns duplex-link-op $Client1 $Router1 orient down-right
$ns duplex-link-op $Client2 $Router1 orient right
$ns duplex-link-op $Client3 $Router1 orient up-right
$ns duplex-link-op $Client4 $Router1 orient up
$ns duplex-link-op $Router1 $Router2 orient right
```

```
$ns duplex-link-op $Router2 $Router3 orient down
$ns duplex-link-op $Router3 $Router4 orient right
$ns duplex-link-op $Router4 $Router5 orient up
$ns duplex-link-op $Router5 $Router6 orient right
$ns duplex-link-op $Router6 $Endserver1 orient up-right
#$ns duplex-link-op $Router6 $Endserver2 orient right



# --------------LABELLING ---------------------#

$ns at 0.0 "$Client1 label Client1"
$ns at 0.0 "$Client2 label Client2"
$ns at 0.0 "$Client3 label Client3"
$ns at 0.0 "$Client4 label Client4"
$ns at 0.0 "$Router1 label Router1"
$ns at 0.0 "$Router2 label Router2"
$ns at 0.0 "$Router3 label Router3"
$ns at 0.0 "$Router4 label Router4"
$ns at 0.0 "$Router5 label Router5"
$ns at 0.0 "$Router6 label Router6"
$ns at 0.0 "$Endserver1 label Endserver"
#$ns at 0.0 "$Endserver2 label Endserver2"

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

$Endserver1 shape hexagon
$Router1 shape box
$Router2 shape square
$Router3 shape square
$Router4 shape square
$Router5 shape square
$Router6 shape square

# ----------------ESTABLISHING QUEUES -------------#

$ns duplex-link-op $Client1 $Router1 queuePos 0.1
$ns duplex-link-op $Client2 $Router1 queuePos 0.1
$ns duplex-link-op $Client3 $Router1 queuePos 0.5
$ns duplex-link-op $Client4 $Router1 queuePos 0.5
$ns duplex-link-op $Router1 $Router2 queuePos 0.1
$ns duplex-link-op $Router2 $Router3 queuePos 0.1
$ns duplex-link-op $Router3 $Router4 queuePos 0.1
$ns duplex-link-op $Router4 $Router5 queuePos 0.1
$ns duplex-link-op $Router5 $Router6 queuePos 0.5
$ns duplex-link-op $Router6 $Endserver1 queuePos 0.5

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

#----------- CLIENT1 TO ENDSERVER -------------#

set tcp0 [new Agent/TCP]
$tcp0 set maxcwnd_ 16
$tcp0 set fid_ 4
$ns attach-agent $Client1 $tcp0
```

```
set sink0 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink0

$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns add-agent-trace $tcp0 tcp
$tcp0 tracevar cwnd_

$ns at 0.5 "$ftp0 start"
$ns at 28.5 "$ftp0 stop"

# ---------------- CLIENT2 TO ENDSERVER -------------#



set tcp1 [new Agent/TCP]
$tcp1 set fid_ 2
$tcp1 set maxcwnd_ 16
$ns attach-agent $Client2 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink1

$ns connect $tcp1 $sink1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

$ns add-agent-trace $tcp1 tcp1
$tcp1 tracevar cwnd_

$ns at 0.58 "$ftp1 start"
$ns at 28.5 "$ftp1 stop"
```

# ---------------- CLIENT3 TO ENDSERVER -------------#

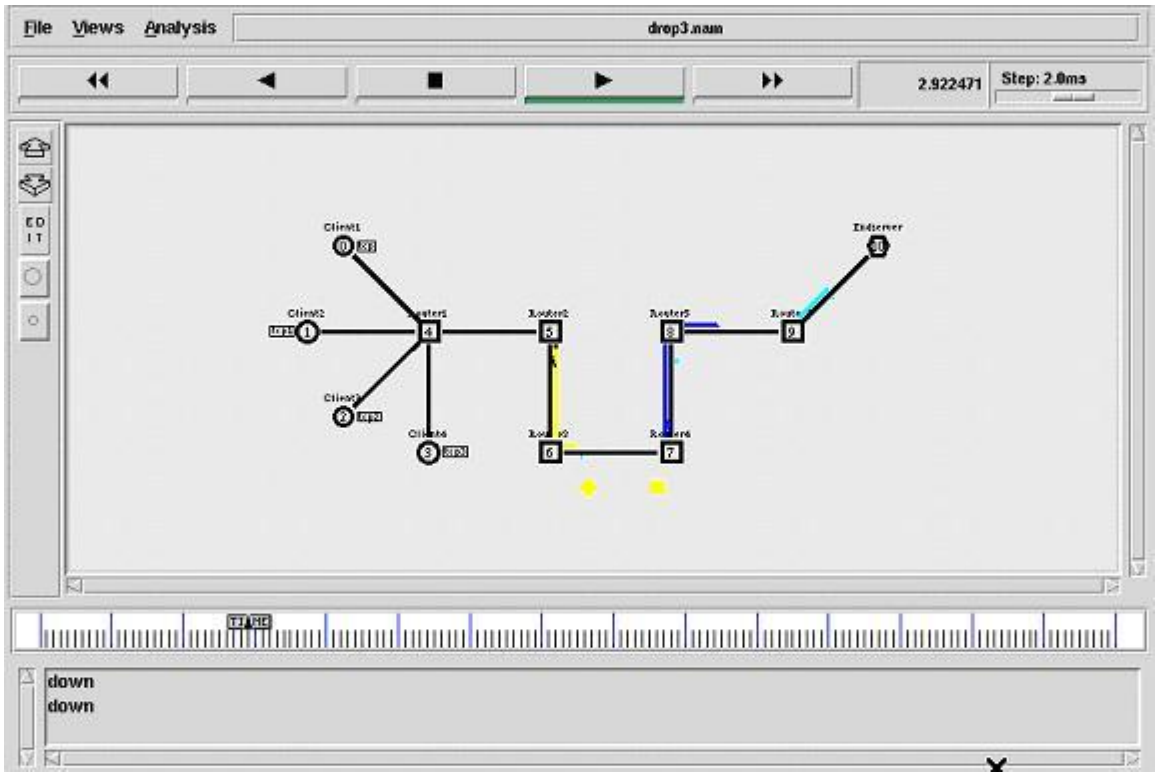
```
set tcp2 [new Agent/TCP]
$tcp2 set fid_ 0
$tcp2 set maxcwnd_ 16
$tcp2 set packetsize_ 100
$ns attach-agent $Client3 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink2
$ns connect $tcp2 $sink2

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ns add-agent-trace $tcp2 tcp2
$tcp2 tracevar cwnd_
```

```
$ns at 0.65 "$ftp2 start"
$ns at 28.5 "$ftp2 stop"

#--------------------CLIENT4 TO ENDSERVER----------------#

set tcp3 [new Agent/TCP]
$tcp3 set fid_ 3
$tcp3 set maxcwnd_ 16
$tcp2 set packetsize_ 100
$ns attach-agent $Client4 $tcp3

set sink3 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink3

$ns connect $tcp3 $sink3

set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3

$ns add-agent-trace $tcp3 tcp3
$tcp3 tracevar cwnd_

$ns at 0.60 "$ftp3 start"
$ns at 28.5 "$ftp3 stop"

#--------------Creating drops-------------#

$ns rtmodel-at 2.880511 down $Router3 $Router4
$ns rtmodel-at 2.880511 up $Router3 $Router4

$ns rtmodel-at 7.299242 down $Router5 $Router6
$ns rtmodel-at 7.299242 up $Router5 $Router6



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


proc finish {} {



               global ns nf nt

               $ns flush-trace
               close $nf
               puts "running nam..."
               exec nam drop3.nam &
               exit 0
            }

#Calling finish procedure
$ns at 15.0 "finish"
$ns run
```

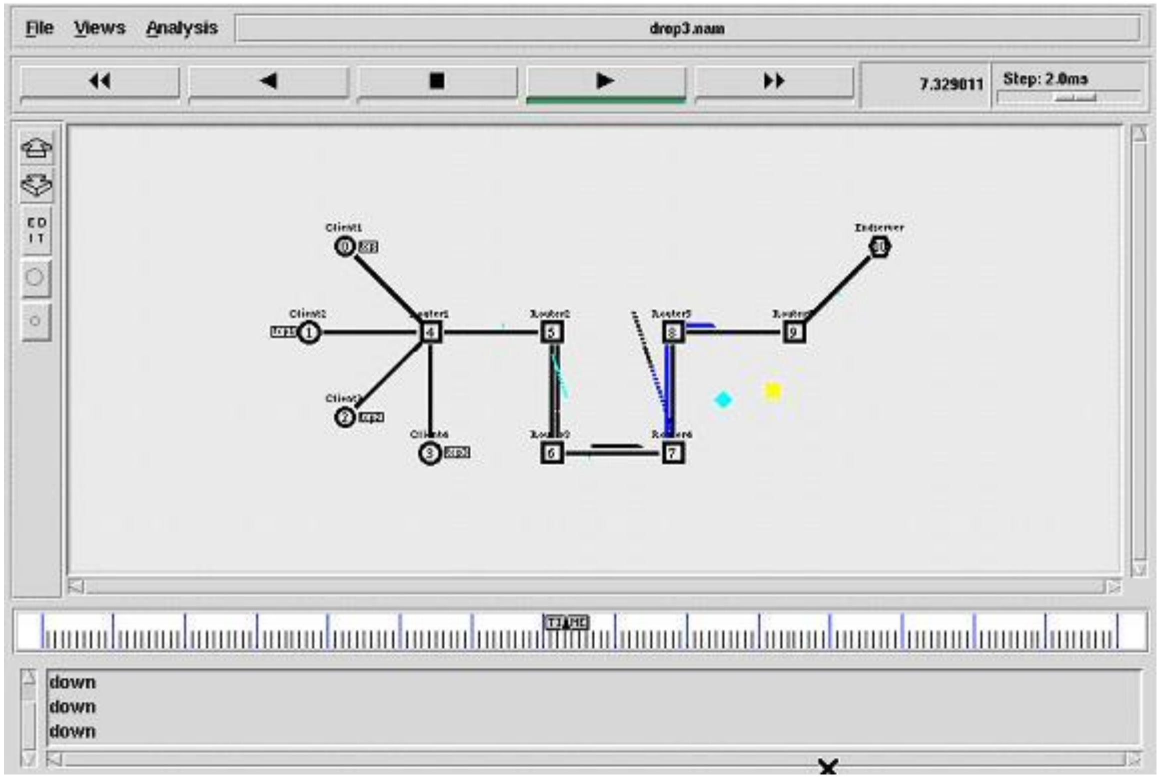
#-----How to run-----#

```
$ns drop3.tcl
```

#----------Snapshot----------#



```
down
down
```



```
down
down
down
```

13. TCL script to generate rands file.

Description:

This network consists of 4 nodes (Client1, Client2, Router1, Router2 and Endserver1).The duplex links between Client1 Client2 and Router1have 5 Mbps of bandwidth and 50 ms of delay. The duplex link between Router1 and Router2 has 150Kbps of bandwidth and 50 ms of delay. The duplex link between Router2 and Endserver1 has 300Kbps of bandwidth and 50 ms of delay. Each link uses a Drop Tail queue. A "TCP" agent is attached to Client1, and Client2 connection is established to a "TCPSink" agent attached to Endserver1 and E. 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 ftp is set to start at 0.5 sec and stop at 5.5 sec. Here we generate the rand file. It is used to create a graph to identify the how many data were sent, how many data were dropped and so on...

File name: "Rands1.tcl"

```
#-------Event scheduler object creation--------#

set ns [new Simulator]

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

set nf [open Rands1.nam w]
$ns namtrace-all $nf

set nt [open Rands1.tr w]
$ns trace-all $nt

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

set proto rlm
$ns color 1 red
$ns color 2 cgyan

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

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

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

$ns duplex-link $Client1 $Router1 5Mb 50ms DropTail
$ns duplex-link $Client2 $Router1 5Mb 50ms DropTail
$ns duplex-link $Router1 $Router2 150Kb 50ms DropTail
$ns duplex-link $Router2 $Endserver1 300Kb 50ms DropTail
```

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

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

# --------------LABELLING -----------------------------#

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

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

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

# ----------------QUEUES POSITIONING AND ESTABLISHMENT -------------#



$ns duplex-link-op $Client1 $Router1 queuePos 0.1
$ns duplex-link-op $Client2 $Router1 queuePos 0.1
$ns duplex-link-op $Router1 $Router2 queuePos 0.5
$ns duplex-link-op $Router2 $Endserver1 queuePos 0.5

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

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

set tcp0 [new Agent/TCP]
$tcp0 set maxcwnd_ 16
$tcp0 set fid_ 1
$ns attach-agent $Client1 $tcp0

set sink0 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink0

$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns add-agent-trace $tcp0 tcp
$tcp0 tracevar cwnd_

$ns at 0.5 "$ftp0 start"
$ns at 5.5 "$ftp0 stop"
#---------------------------------------#
```

```
set tcp1 [new Agent/TCP]
$tcp1 set maxcwnd_ 16
$tcp1 set fid_ 2
$ns attach-agent $Client2 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink1

$ns connect $tcp1 $sink1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

$ns add-agent-trace $tcp1 tcp
$tcp1 tracevar cwnd_

$ns at 0.5 "$ftp1 start"
$ns at 6.5 "$ftp1 stop"



$ns rtmodel-at 2.88 down $Router1 $Router2
$ns rtmodel-at 3.52 up $Router2 $Endserver1

$ns at 1.1025 "$ns trace-annotate \"Time: 1.1025 Pkt Transfer Path
thru node_(1)..\""
$ns at 2.88 "$ns trace-annotate \"Time: 2.88 node_(3) drop and block
pkt transfer..\""
$ns at 5.1936 "$ns trace-annotate \"Time: 5.1936 Pkt Transfer Path
thru node_(1)..\""
```

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

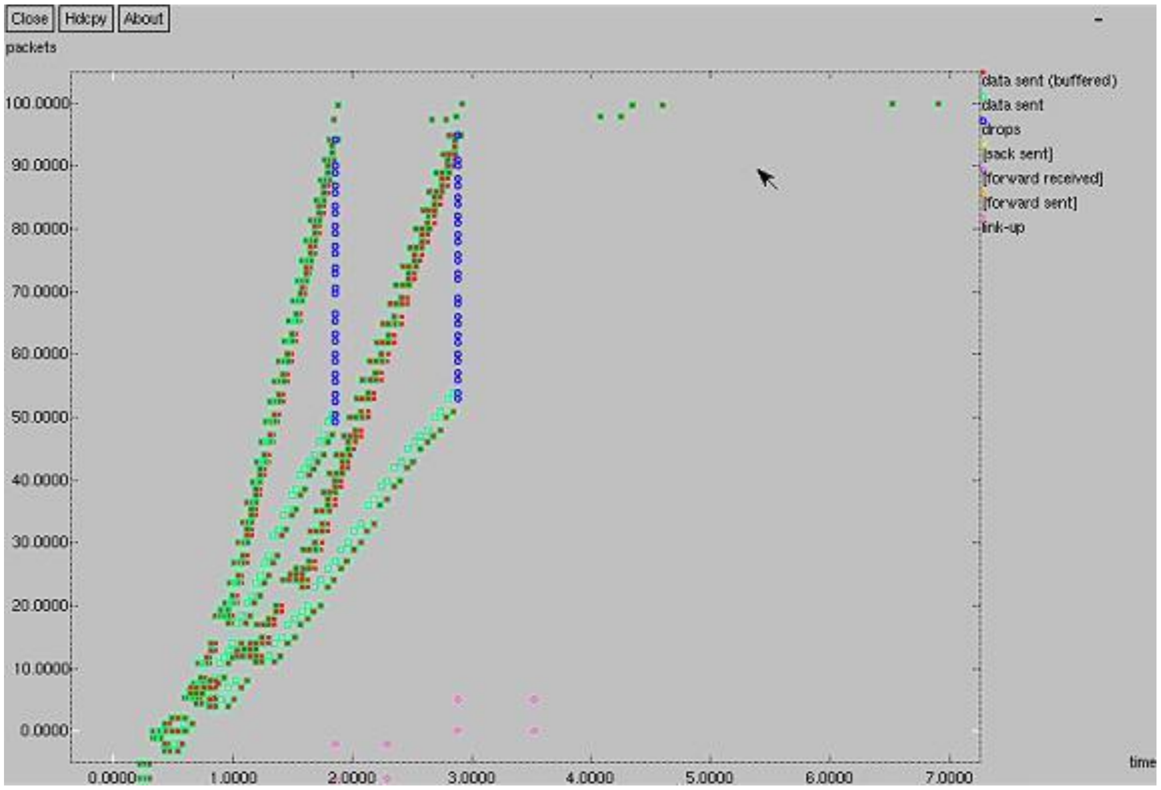
```
proc finish {} {

 global ns nf nt
set PERL "/bin/perl5.8.2"
               set NSHOME "/home/ns-allinone-2.30"  ;#change path
               set XGRAPH "$NSHOME/bin/xgraph"
               set SETFID "$NSHOME/ns-2.30/bin/set_flow_id"
               set RAW2XG_SCTP "$NSHOME/ns-2.30/bin/raw2xg-sctp"
               $ns flush-trace
               close $nf
               exec $PERL $SETFID -s Rands1.tr | \
               $PERL $RAW2XG_SCTP -A -q > Rands1.rands
               exec $XGRAPH -bb -tk -nl -m -x time -y packets
Rands1.rands &
 $ns flush-trace
 close $nf
 puts "running nam..."
 exit 0
 }
```

```
#Calling finish procedure
$ns at 7.0 "finish"
$ns run
```

#-----How to run-----#

```
$ns rands1.tcl
       Or
$xgraph rands1.rands
```

#----------Snapshot----------#



14. TCL script to generate rands file and generating graph using rands file.

Description:

This network consists of 9 nodes (C1, C2, C3, C4, R1, R2 R3, R4, ROU1, ROU2 and ROU3). The duplex links between the nodes have 1 Mbps of bandwidth and 10 ms of delay. Each link uses a DropTail queue. A "TCP" agent is attached to C1 and a connection is established to a "TcpSink" agent attached to R4. A "tcp" agent is attached to C2 and a connection is established to a tcp "sink" agent attached to R3. A "tcp" agent is attached to C3 and a connection is established to a tcp "sink" agent attached to R2As default, the maximum size of a packet that a "tcp" agent can generate is 1KByte. A "tcp" agent is attached

to C4 and a connection is established to a tcp "sink" agent attached to R1 A tcp "sink" agent generates and sends ACK packets to the sender (tcp agent) and frees the received packets. The ftp is set to start at 0.2 sec and stop at 4.5 sec. Here we generate the rand file. It is used to create a graph to identify the how many data were sent, how many data were dropped and so on... it will separated by the different colors.

<u>File name: "Rands2.tcl"</u>

#-------Event scheduler object creation--------#

```
set ns [ new Simulator -multicast on]

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

set nf [open rands2.nam w]
$ns namtrace-all $nf

#Open the trace file
set nt [open rands2.tr w]
$ns trace-all $nt



set proto rlm

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

$ns color 1 dodgerblue
$ns color 2 red
$ns color 3 cyan
$ns color 4 green
$ns color 5 yellow
$ns color 6 black
$ns color 7 magenta
$ns color 8 gold
$ns color 9 red

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

set C1 [$ns node]
set C2 [$ns node]
set C3 [$ns node]
set C4 [$ns node]
set R1 [$ns node]
set R2 [$ns node]
set R3 [$ns node]
set R4 [$ns node]
set ROU1 [$ns node]
set ROU2 [$ns node]
set ROU3 [$ns node]
```

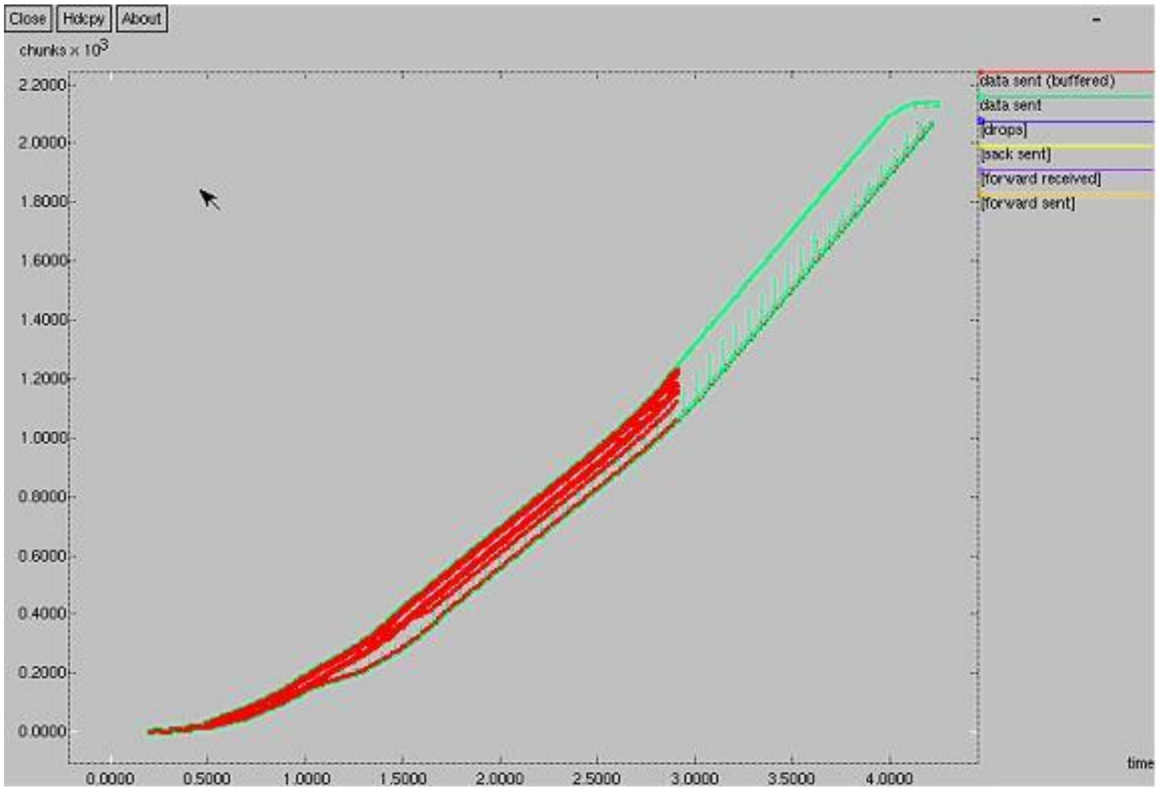
```
# --------------CREATING DUPLEX LINK -----------------------#

$ns duplex-link $C1 $ROU1 1Mb 10ms DropTail
$ns duplex-link $C2 $ROU1 500Kb 10ms DropTail
$ns duplex-link $C3 $ROU1 750Kb 10ms DropTail
$ns duplex-link $C4 $ROU2 1Mb 10ms DropTail
$ns duplex-link $R1 $ROU1 1Mb 10ms DropTail
$ns duplex-link $R2 $ROU1 1Mb 10ms DropTail
$ns duplex-link $R3 $ROU1 1Mb 10ms DropTail
$ns duplex-link $R4 $ROU3 1Mb 10ms DropTail

$ns duplex-link $ROU2 $ROU1 1Mb 10ms DropTail
$ns duplex-link $ROU2 $ROU3 1Mb 10ms DropTail
$ns duplex-link $ROU1 $ROU3 1Mb 10ms DropTail

#-------------QUEUE SIZE DESCRIPTION---------------#

$ns queue-limit $ROU1 $ROU2 18
$ns queue-limit $ROU1 $ROU3 18
$ns queue-limit $ROU2 $ROU1 20
$ns queue-limit $ROU3 $ROU1 20



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

$ns duplex-link-op $C1 $ROU1 orient down
$ns duplex-link-op $C2 $ROU1 orient down-right
$ns duplex-link-op $C3 $ROU1 orient down-left
$ns duplex-link-op $C4 $ROU2 orient up
$ns duplex-link-op $R1 $ROU1 orient up
$ns duplex-link-op $R2 $ROU1 orient up-right
$ns duplex-link-op $R3 $ROU1 orient up-left
$ns duplex-link-op $R4 $ROU3 orient down

$ns duplex-link-op $ROU1 $ROU2 orient down-right
$ns duplex-link-op $ROU3 $ROU2 orient down-right

#$ns queue-limit $ $n1 15



# --------------CREATING LABELLING -----------------------------#

$ns at 0.0 "$C1 label CL1"
$ns at 0.0 "$C2 label CL2"
$ns at 0.0 "$C3 label CL3"
$ns at 0.0 "$C4 label CL4"
$ns at 0.0 "$R1 label RC1"
$ns at 0.0 "$R2 label RC2"
$ns at 0.0 "$R3 label RC3"
$ns at 0.0 "$R4 label RC4"
$ns at 0.0 "$ROU1 label ROU1"
$ns at 0.0 "$ROU2 label ROU2"
$ns at 0.0 "$ROU3 label ROU3"
```

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

$ROU1 shape square
$ROU2 shape square
$ROU3 shape square

# ----------------QUEUES POSITIONING AND ESTABLISHMENT -------------#

$ns duplex-link-op $ROU2 $ROU1 queuePos 0.1
#$ns duplex-link-op $ROU2 $C5 queuePos 0.1
$ns duplex-link-op $ROU3 $ROU1 queuePos 0.1



#--------SETTING IDENTIFICATION COLORS TO ROUTER-LINKS--------#

$ns duplex-link-op $ROU1 $ROU2 color cyan
$ns duplex-link-op $ROU1 $ROU3 color cyan
$ns duplex-link-op $ROU2 $ROU3 color cyan



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

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

      #$tcp0 set fid_ 3
      #$Base1 set fid_ 3
      #$tcp0 set window_ 15
      #$ftp0 set packetSize_ 1000
      #$ftp0 set interval_ .05


      set tcp1 [$ns create-connection TCP $C1 TCPSink $R4 1]
      $tcp1 set class_ 1
      $tcp1 set maxcwnd_ 16
      $tcp1 set packetsize_ 4000
      $tcp1 set fid_ 1
      set ftp1 [$tcp1 attach-app FTP]
      $ftp1 set interval_ .005
      $ns at 0.2 "$ftp1 start"
      $ns at 4.0 "$ftp1 stop"



      set tcp2 [$ns create-connection TCP $C2 TCPSink $R3 1]
      $tcp2 set class_ 1
      $tcp2 set maxcwnd_ 16
      $tcp2 set packetsize_ 4000
      $tcp2 set fid_ 2
      set ftp2 [$tcp2 attach-app FTP]
      $ftp2 set interval_ .005
      $ns at 0.7 "$ftp2 start"
      $ns at 4.0 "$ftp2 stop"
      set tcp3 [$ns create-connection TCP $C3 TCPSink $R2 1]
      $tcp3 set class_ 1
      $tcp3 set maxcwnd_ 16
      $tcp3 set packetsize_ 4000
      $tcp3 set fid_ 3
      set ftp3 [$tcp3 attach-app FTP]
      $ftp3 set interval_ .005
      $ns at 1.2 "$ftp3 start"
      $ns at 4.0 "$ftp3 stop"


      set tcp4 [$ns create-connection TCP $C4 TCPSink $R1 1]
      $tcp4 set class_ 1
      $tcp4 set maxcwnd_ 16
      $tcp4 set packetsize_ 4000
      $tcp4 set fid_ 4
      set ftp4 [$tcp4 attach-app FTP]
      $ftp1 set interval_ .005
      $ns at 2.5 "$ftp4 start"
      $ns at 4.0 "$ftp4 stop"


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


proc finish {} {

               global ns nf nt nf1
               set PERL "/bin/perl5.8.2"
               set NSHOME "/home/ns-allinone-2.30"
                      ;#change path
               set XGRAPH "$NSHOME/bin/xgraph"
               set SETFID "$NSHOME/ns-2.30/bin/set_flow_id"
               set RAW2XG_SCTP "$NSHOME/ns-2.30/bin/raw2xg-sctp"
               $ns flush-trace
               close $nf
               exec $PERL $SETFID -s Rands2.tr | \
               $PERL $RAW2XG_SCTP -A -q > Rands2.rands
               exec $XGRAPH -bb -tk -nl -m -x time -y packets
Rands2.rands &

               $ns flush-trace
               close $nf

               exit 0
             }

#Calling finish procedure
$ns at 20.0 "finish"
$ns run
```

#-----How to run-----#

```
$ns Rands2.tcl

          OR
$xgraph -geometry 600X500 Rands2.rands
```

```
#----------Snapshot----------#
```



15.TCL script to generate rand file which contains packets sent, received and dropped information.

Description:

This network consists of 5 nodes. The duplex links between n1 and n2 have 0.3 Mbps of bandwidth and 10 ms of delay. The duplex link between n2 and n3 has 0.3Mbps of bandwidth and 10 ms of delay. The duplex link between n3 and n4, n5 has 0.3Mbps of bandwidth and 10 ms of delay. Each link uses a DropTail queue. An "UDP" agent is attached to n1 and a connection is established to udp "Null" agent attached to n4. As default, the maximum size of a packet that “UDP" agent can generate is 1000bytes. An udp "Null" agent generates and sends ACK

packets to the sender (tcp agent) and frees the received packets. The cbr is set to start at 0.4 sec and stop at 2.0 sec. The cbr is set to start at 0.4 sec and stop at 2.0 sec. Here we generate the rand file. It is used to create a graph to identify the amount of data sent, received and dropped. Lines in the graph will be separated by the different colors.

<u>File name: "Rands3.tcl"</u>

#-------Event scheduler object creation--------#

```
set ns [new Simulator]



set nf [open Rands3.tr w]
$ns trace-all $nf
$ns namtrace-all [open Rands3.nam w]

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

$ns color 1 red
$ns color 30 purple
$ns color 31 green

# allocate a multicast address;
set group [Node allocaddr]

# nod is the number of nodes
set nod 5

# create multicast capable nodes;
for {set i 1} {$i <= $nod} {incr i} {
   set n($i) [$ns node]
}

#Create links between the nodes
$ns duplex-link $n(1) $n(2) 0.3Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 0.3Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 0.5Mb 10ms DropTail
#$ns duplex-link $n(2) $n(5) 0.3Mb 10ms DropTail
$ns duplex-link $n(3) $n(4) 0.3Mb 10ms DropTail
$ns duplex-link $n(4) $n(5) 0.5Mb 10ms DropTail
#$ns duplex-link $n(4) $n(6) 0.5Mb 10ms DropTail
#$ns duplex-link $n(5) $n(6) 0.5Mb 10ms DropTail

# configures multicast protocol;
BST set RP_($group) $n(2)
$ns mrtproto BST

set udp1 [new Agent/UDP]
set udp2 [new Agent/UDP]

$ns attach-agent $n(1) $udp1
$ns attach-agent $n(2) $udp2
```

```
set src1 [new Application/Traffic/CBR]
$src1 attach-agent $udp1
$udp1 set dst_addr_ $group
$udp1 set dst_port_ 0
$src1 set random_ false



set src2 [new Application/Traffic/CBR]
$src2 attach-agent $udp2
$udp2 set dst_addr_ $group
$udp2 set dst_port_ 1
$src2 set random_ false

# create receiver agents
set rcvr [new Agent/LossMonitor]

# joining and leaving the group;
$ns at 0.6 "$n(3) join-group $rcvr $group"
$ns at 1.3 "$n(4) join-group $rcvr $group"
$ns at 1.6 "$n(5) join-group $rcvr $group"
#$ns at 1.9 "$n(4) leave-group $rcvr $group"
#$ns at 2.3 "$n(6) join-group $rcvr $group"
$ns at 3.5 "$n(3) leave-group $rcvr $group"

$ns at 0.4 "$src1 start"
$ns at 2.0 "$src2 start"

$ns at 4.0 "finish"

proc finish {} {
        global ns nf
set PERL "/bin/perl5.8.2"
               set NSHOME "/home/ns-allinone-2.30"
                   ;#change path
               set XGRAPH "$NSHOME/bin/xgraph"
               set SETFID "$NSHOME/ns-2.30/bin/set_flow_id"
               set RAW2XG_SCTP "$NSHOME/ns-2.30/bin/raw2xg-sctp"
               $ns flush-trace
               close $nf
               exec $PERL $SETFID -s Rands3.tr | \
               $PERL $RAW2XG_SCTP -A -q > Rands3.rands
               exec $XGRAPH -bb -tk -nl -m -x time -y packets
Rands3.rands &

                #exec xgraph /home/project/STAIR/STAIR2/aso1.tr &
                #exec xgraph /home/project/STAIR/STAIR2/aso2.tr &




                # puts "filtering..."
                 #exec tclsh /home/ns-allinone-2.30/nam-
1.12/bin/namfilter.tcl Rands3.nam
```

```
                  puts "running nam..."


          $ns flush-trace
          exec nam Rands3.rands &
          exit 0
}
```

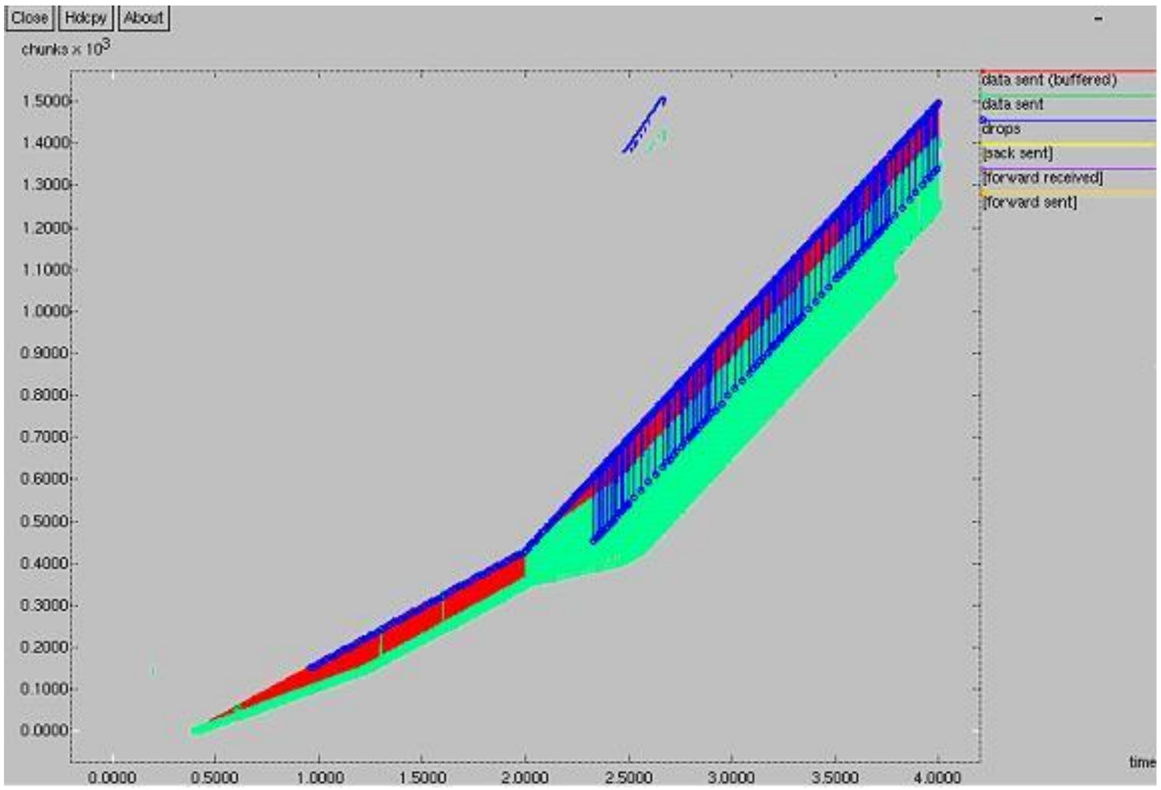
#Calling finish procedure

```
$ns run
```

#-----How to run-----#

```
$ns filename.tcl
    or
$xgraph -geometry 600X700 filename.rands
```

#----------Snapshot-----------#



<u>16.TCL script to handle trace annotation.</u>

Description:

This network consists of 6 nodes. The duplex links between n0, n1, and n2 have 2 Mbps of bandwidth and 10 ms of delay. The simplex

link between n2 and n3 has 0.3Mbps of bandwidth and 100 ms of delay. The duplex link between n3 and n4 has 0.5Mbps of bandwidth and 40ms of delay. Each link uses DropTail queue. A "TCP" agent is attached to n0 and a connection is established to a "TCPSink" agent attached to n4. 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. "UDP" agent is attached to n1 and connection is established to udp "Null" agent attached to n5. Here we will generate the trace-annotate file. It will provide the information about the events at each time. The cbr is set to start at 0.1 sec and stop at 7.5 sec. The ftp is set to start at 1.0 sec and stop at 7.0 sec.

File name: "trace.tcl"

#-------Event scheduler object creation--------#

```
set ns [new Simulator]

#Open the Trace files

set file1 [open trace.tr w]
$ns trace-all $file1

#Open the NAM trace file

set file2 [open trace.nam w]
$ns namtrace-all $file2

$ns color 1 Blue
$ns color 2 Red

#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]

#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
$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail

#Give node position (for NAM)
```

```
$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
$ns duplex-link-op $n3 $n4 orient right-up
$ns duplex-link-op $n3 $n5 orient right-down



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

$ns queue-limit $n2 $n3 20

#Setup a 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 window_ 8000
$tcp set packetSize_ 552

#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 $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

$ns at 0.1 "$cbr start"
$ns at 1.0 "$ftp start"
$ns at 7.0 "$ftp stop"
$ns at 7.5 "$cbr stop"

#-------------Trace file creation---------#
```

```
$ns at 1.1025 "$ns trace-annotate \"Time: 1.1025 Pkt Transfer Path
thru node_(1)..\""
$ns at 2.1025 "$ns trace-annotate \"Time: 2.1025 Pkt Transfer Path
thru node_(1)..\""
$ns at 3.1025 "$ns trace-annotate \"Time: 3.1025 Pkt Transfer Path
thru node_(1)..\""

#Define a 'finish' procedure

proc finish {} {
        global ns file1 file2
        $ns flush-trace
        close $file1
        close $file2
        exec nam trace.nam &
        exit 0
}
```

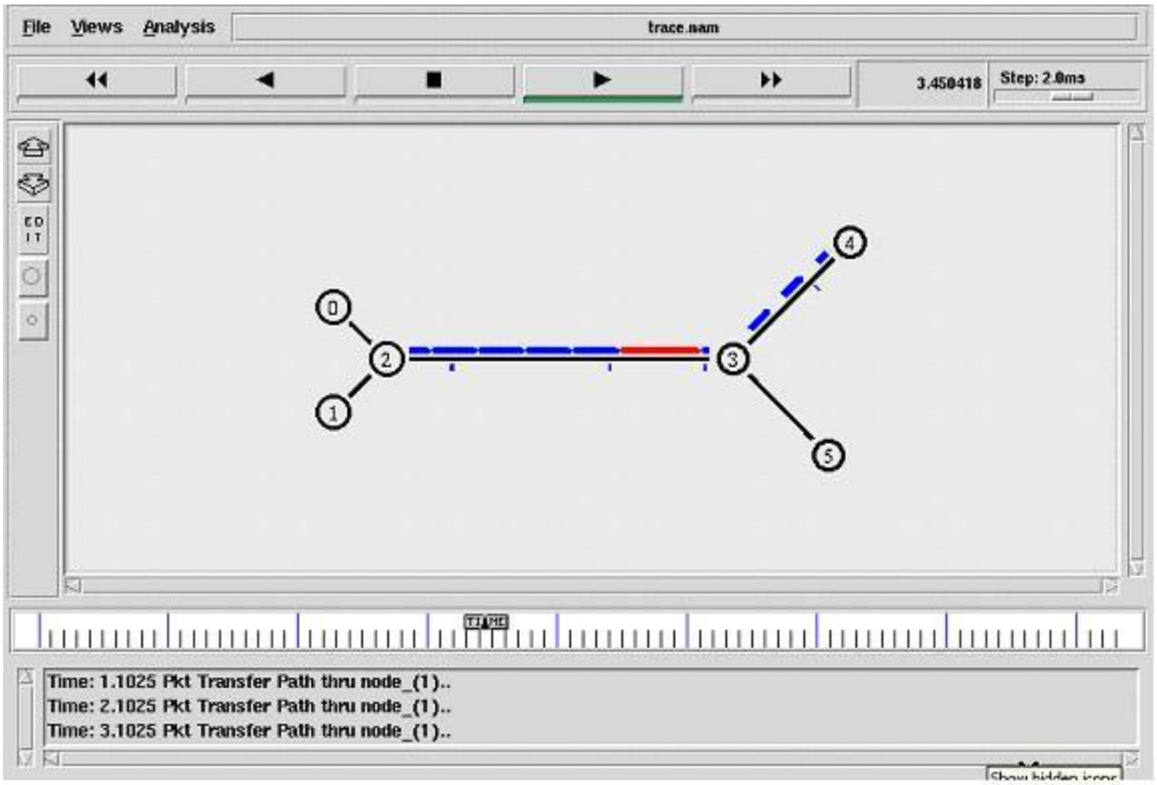
#Calling finish procedure

```
$ns at 10.0 "finish"
$ns run
```

#-----How to run-----#

```
$ns filename.tcl
```

#----------Snapshot----------#



```
Time: 1.1025 Pkt Transfer Path thru node_(1)..
Time: 2.1025 Pkt Transfer Path thru node_(1)..
Time: 3.1025 Pkt Transfer Path thru node_(1)..
```

17.TCL script for display the nodes activity in simulation window using trace annotation.

Description:

This network consists of 8 nodes (Client1, Client2, Client3, Router1, Router2 Router3, Router4, and Endserver1). The duplex links between Client1, Client2, Client3 and Router1 have 2 Mbps of bandwidth and 100 ms of delay. The duplex link between Router1 and Router2 has 200Mbps of bandwidth and 100 ms of delay. The duplex link between Router2 and Router3 has 10Kbps of bandwidth and 100 ms of delay. The duplex link between Router3 and Router4 has 100Kbps of bandwidth and 100 ms of delay. The duplex link between Router4 and Endserver1 has 200Kbps of bandwidth and 100 ms of delay. Each node uses a DropTail queue, of which the maximum size is 100. . A "TCP" agent is attached to Client1, Client2, Client3 and a connection is established to a "TCPSink" 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. Here we will generate the trace-annotate file. It will provide the information about the packet transfer event at the bottom of the simulation window. The ftp starts at .60 sec and ends at 28.5sec

File name: "trace1.tcl"

#-------Event scheduler object creation--------#

```
set ns [new Simulator ]

#----------creating nam objects----------------#

set nf [open trace1.nam w]
$ns namtrace-all $nf

#open the trace file

set nt [open trace1.tr w]
$ns trace-all $nt

set proto rlm

$ns color 51 blue
$ns color 52 yellow
$ns color 53 red

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

set Client1 [$ns node]
set Client2 [$ns node]
set Client3 [$ns node]
set Router1 [$ns node]
set Router2 [$ns node]
set Router3 [$ns node]
set Router4 [$ns node]
set Endserver1 [$ns node]
#set Endserver2 [$ns node]

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

$ns duplex-link $Client1 $Router1 2Mb 100ms DropTail
$ns duplex-link $Client2 $Router1 2Mb 100ms DropTail
$ns duplex-link $Client3 $Router1 2Mb 100ms DropTail
$ns duplex-link $Router1 $Router2 100Kb 100ms DropTail
$ns duplex-link $Router2 $Router3 100Kb 100ms DropTail
$ns duplex-link $Router3 $Router4 200Kb 100ms DropTail
$ns duplex-link $Router4 $Endserver1 200Kb 100ms DropTail
#$ns duplex-link $Router4 $Endserver2 200Kb 100ms DropTail

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

$ns duplex-link-op $Client1 $Router1 orient down-right
$ns duplex-link-op $Client2 $Router1 orient right
$ns duplex-link-op $Client3 $Router1 orient up-right
$ns duplex-link-op $Router1 $Router2 orient up-right
$ns duplex-link-op $Router2 $Router3 orient down-right
$ns duplex-link-op $Router3 $Router4 orient right
$ns duplex-link-op $Router4 $Endserver1 orient up



#------------Labeling----------------#

$ns at 0.0 "$Client1 label Client1"
$ns at 0.0 "$Client2 label Client2"
$ns at 0.0 "$Client3 label Client3"
$ns at 0.0 "$Router1 label Router1"
$ns at 0.0 "$Router2 label Router2"
$ns at 0.0 "$Router3 label Router3"
$ns at 0.0 "$Router4 label Router4"
$ns at 0.0 "$Endserver1 label Endserver1"

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

#$Endserver1 shape hexagon
#$Router1 shape square
#$Router2 shape square
#$Router3 Shape square
#$Router4 Shape square

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

$ns duplex-link-op $Client1 $Router1 queuePos 0.1
$ns duplex-link-op $Client2 $Router1 queuePos 0.1
$ns duplex-link-op $Client3 $Router1 queuePos 0.5
$ns duplex-link-op $Router1 $Router2 queuePos 0.1
$ns duplex-link-op $Router2 $Router3 queuePos 0.5
```

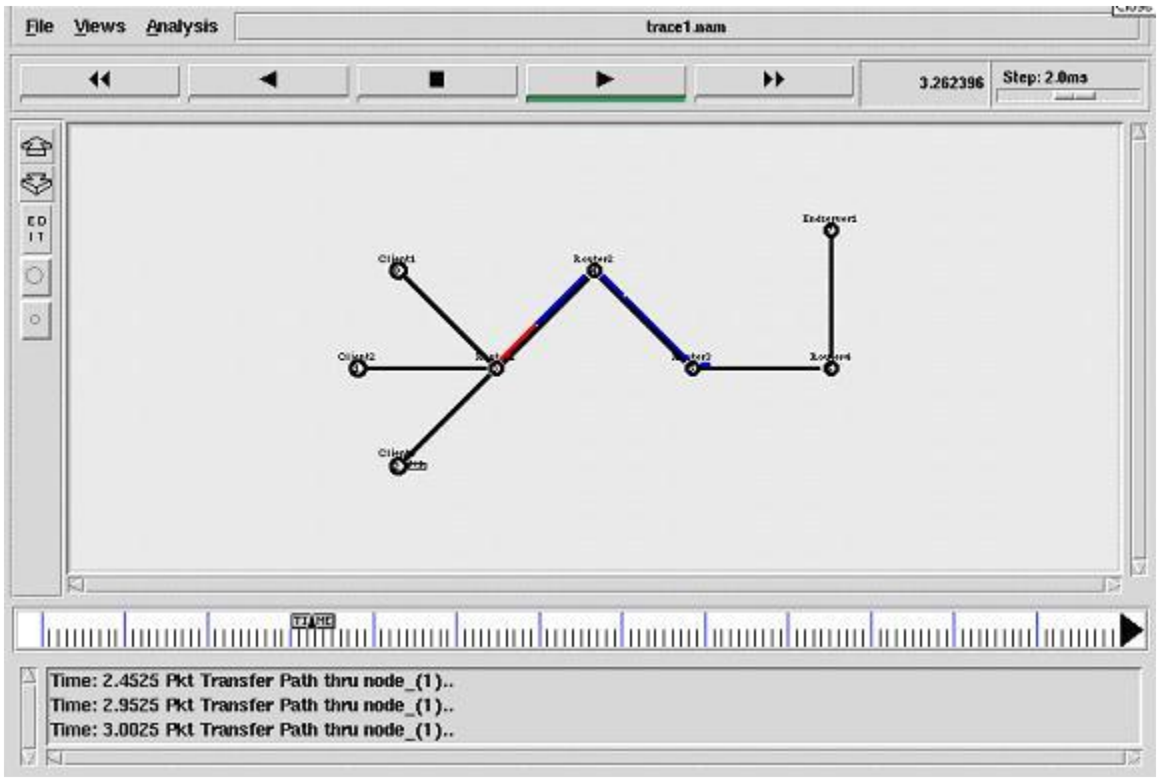
```
$ns duplex-link-op $Router3 $Router4 queuePos 0.1
$ns duplex-link-op $Router4 $Endserver1 queuePos 0.5

#---------Establishing communication-------------#

#-------------Client1 to Endserver1---#

set tcp0 [new Agent/TCP]
$tcp0 set maxcwnd_ 16
$tcp0 set fid_ 51
$ns attach-agent $Client1 $tcp0

set sink0 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink0

$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns add-agent-trace $tcp0 tcp
$tcp0 tracevar cwnd_

$ns at 0.50 "$ftp0 start"
$ns at 28.5 "$ftp0 stop"

#--------------Client2 to Endserver1---------#

set tcp1 [new Agent/TCP]
$tcp1 set maxcwnd_ 16
$tcp1 set fid_ 52
$ns attach-agent $Client2 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink1

$ns connect $tcp1 $sink1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp0

$ns add-agent-trace $tcp1 tcp
$tcp1 tracevar cwnd_

$ns at 0.60 "$ftp1 start"
$ns at 28.5 "$ftp1 stop"

#------------- Client3 to End server----------#

set tcp2 [new Agent/TCP]
$tcp2 set maxcwnd_ 16
$tcp2 set fid_ 53
$ns attach-agent $Client3 $tcp2
```

```
set sink2 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink2

$ns connect $tcp2 $sink2
set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2

$ns add-agent-trace $tcp2 tcp
$tcp2 tracevar cwnd_

$ns at 0.70 "$ftp2 start"
$ns at 28.5 "$ftp2 stop"
#-------------Trace file creation---------#

$ns at 2.4525 "$ns trace-annotate \"Time: 2.4525 Pkt Transfer Path
thru node_(1)..\""
$ns at 2.9525 "$ns trace-annotate \"Time: 2.9525 Pkt Transfer Path
thru node_(1)..\""
$ns at 3.0025 "$ns trace-annotate \"Time: 3.0025 Pkt Transfer Path
thru node_(1)..\""

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

proc finish {} {



               global ns nf nt
               puts "running nam..."
              exec nam trace1.nam &
              exit 0
            }



#Calling finish procedure

$ns at 35.0 "finish"
$ns run
```

#-----How to run-----#

```
$ns filename.tcl
```

#----------Snapshot-----------#



```
Time: 2.4525 Pkt Transfer Path thru node_(1)..
Time: 2.9525 Pkt Transfer Path thru node_(1)..
Time: 3.0025 Pkt Transfer Path thru node_(1)..
```

18.TCL script for display packets transfer information using annotation.

Description:

This network consists of 8 nodes (Client1, Client2, Client3, Router1, Router2, Router3, Router4, and Endserver1). The duplex links between Client1, Client2, Client3 and Router1 have 5 Mbps of bandwidth and 50 ms of delay. The duplex link between Router1 and Router2 has 5Mbps of bandwidth and 50 ms of delay. The duplex link between Router2 and Router3 has 150Kbps of bandwidth and 50 ms of delay. The duplex link between Router3 and Router4, Router4 and Router5, Router5 and Router6, Router6 and Endserver1 has 300Kbps of bandwidth and 50 ms of delay. Each link uses a DropTail queue. A "TCP" agent is attached to Client1, Client2, Client3, Client4 and a connection is established to a "TCPSink" 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. Link failure and recovery model is created at various times. Here we will generate the trace-annotate file. It will provide the information about the packet transfer event at the bottom of the simulation window. The ftp starts at .50 sec and ends at 28.5sec

File name: "trace2.tcl"

```
#-------Event scheduler object creation--------#

set ns [ new Simulator]

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

```
set nf [open trace2.nam w]
$ns namtrace-all $nf

#Open the trace file
set nt [open trace2.tr w]
$ns trace-all $nt

set proto rlm

$ns color 1 red
$ns color 2 blue
$ns color 3 yellow
$ns color 4 cyan
$ns color 5 maroon

                  # --------- CREATING CLIENT - ROUTER -ENDSERVER
NODES-----------#

set Client1 [$ns node]
set Client2 [$ns node]
set Client3 [$ns node]
set Client4 [$ns node]
set Router1 [$ns node]
set Router2 [$ns node]
set Router3 [$ns node]
set Router4 [$ns node]
set Router5 [$ns node]
set Router6 [$ns node]
set Endserver1 [$ns node]
#set Endserver2 [$ns node]

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

$ns duplex-link $Client1 $Router1  5Mb 50ms DropTail
$ns duplex-link $Client2 $Router1  5Mb 50ms DropTail
$ns duplex-link $Client3 $Router1 5Mb 50ms DropTail
$ns duplex-link $Client4 $Router1 5Mb 50ms DropTail
$ns duplex-link $Router1 $Router2 5Mb 50ms DropTail
$ns duplex-link $Router2 $Router3 150Kb 50ms DropTail
$ns duplex-link $Router3 $Router4 300Kb 50ms DropTail
$ns duplex-link $Router4 $Router5 100Kb 50ms DropTail
$ns duplex-link $Router5 $Router6 300Kb 50ms DropTail
$ns duplex-link $Router6 $Endserver1 300Kb 50ms DropTail
#$ns duplex-link $Router6 $Endserver2 300Kb 50ms DropTail



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

$ns duplex-link-op $Client1 $Router1 orient down-right
$ns duplex-link-op $Client2 $Router1 orient right
$ns duplex-link-op $Client3 $Router1 orient up-right
$ns duplex-link-op $Client4 $Router1 orient up
$ns duplex-link-op $Router1 $Router2 orient right
$ns duplex-link-op $Router2 $Router3 orient down
$ns duplex-link-op $Router3 $Router4 orient right
```

```
$ns duplex-link-op $Router4 $Router5 orient up
$ns duplex-link-op $Router5 $Router6 orient right
$ns duplex-link-op $Router6 $Endserver1 orient up-right
#$ns duplex-link-op $Router6 $Endserver2 orient right



# --------------LABELLING -----------------------------#

$ns at 0.0 "$Client1 label Client1"
$ns at 0.0 "$Client2 label Client2"
$ns at 0.0 "$Client3 label Client3"
$ns at 0.0 "$Client4 label Client4"
$ns at 0.0 "$Router1 label Router1"
$ns at 0.0 "$Router2 label Router2"
$ns at 0.0 "$Router3 label Router3"
$ns at 0.0 "$Router4 label Router4"
$ns at 0.0 "$Router5 label Router5"
$ns at 0.0 "$Router6 label Router6"
$ns at 0.0 "$Endserver1 label Endserver"
#$ns at 0.0 "$Endserver2 label Endserver2"

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

$Endserver1 shape hexagon
$Router1 shape box
$Router2 shape square
$Router3 shape square
$Router4 shape square
$Router5 shape square
$Router6 shape square

# ----------------ESTABLISHING QUEUES -------------#

$ns duplex-link-op $Client1 $Router1 queuePos 0.1
$ns duplex-link-op $Client2 $Router1 queuePos 0.1
$ns duplex-link-op $Client3 $Router1 queuePos 0.5
$ns duplex-link-op $Client4 $Router1 queuePos 0.5
$ns duplex-link-op $Router1 $Router2 queuePos 0.1
$ns duplex-link-op $Router2 $Router3 queuePos 0.1
$ns duplex-link-op $Router3 $Router4 queuePos 0.1
$ns duplex-link-op $Router4 $Router5 queuePos 0.1
$ns duplex-link-op $Router5 $Router6 queuePos 0.5
$ns duplex-link-op $Router6 $Endserver1 queuePos 0.5

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

#--------CLIENT1 TO ENDSERVER -------------#

set tcp0 [new Agent/TCP]
$tcp0 set maxcwnd_ 16
$tcp0 set fid_ 4
$ns attach-agent $Client1 $tcp0
```

```
set sink0 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink0

$ns connect $tcp0 $sink0

set ftp0 [new Application/FTP]
$ftp0 attach-agent $tcp0

$ns add-agent-trace $tcp0 tcp
$tcp0 tracevar cwnd_

$ns at 0.5 "$ftp0 start"
$ns at 28.5 "$ftp0 stop"

# ----------------CLIENT2 TO ENDSERVER1 -------------#



set tcp1 [new Agent/TCP]
$tcp1 set fid_ 2
$tcp1 set maxcwnd_ 16
$ns attach-agent $Client2 $tcp1

set sink1 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink1

$ns connect $tcp1 $sink1

set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

$ns add-agent-trace $tcp1 tcp1
$tcp1 tracevar cwnd_

$ns at 0.58 "$ftp1 start"
$ns at 28.5 "$ftp1 stop"

# ---------------- CLIENT3 TO ENDSERVER -------------#



set tcp2 [new Agent/TCP]
$tcp2 set fid_ 0
$tcp2 set maxcwnd_ 16
$tcp2 set packetsize_ 100
$ns attach-agent $Client3 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink2
$ns connect $tcp2 $sink2

set ftp2 [new Application/FTP]
$ftp2 attach-agent $tcp2
$ns add-agent-trace $tcp2 tcp2
$tcp2 tracevar cwnd_
```

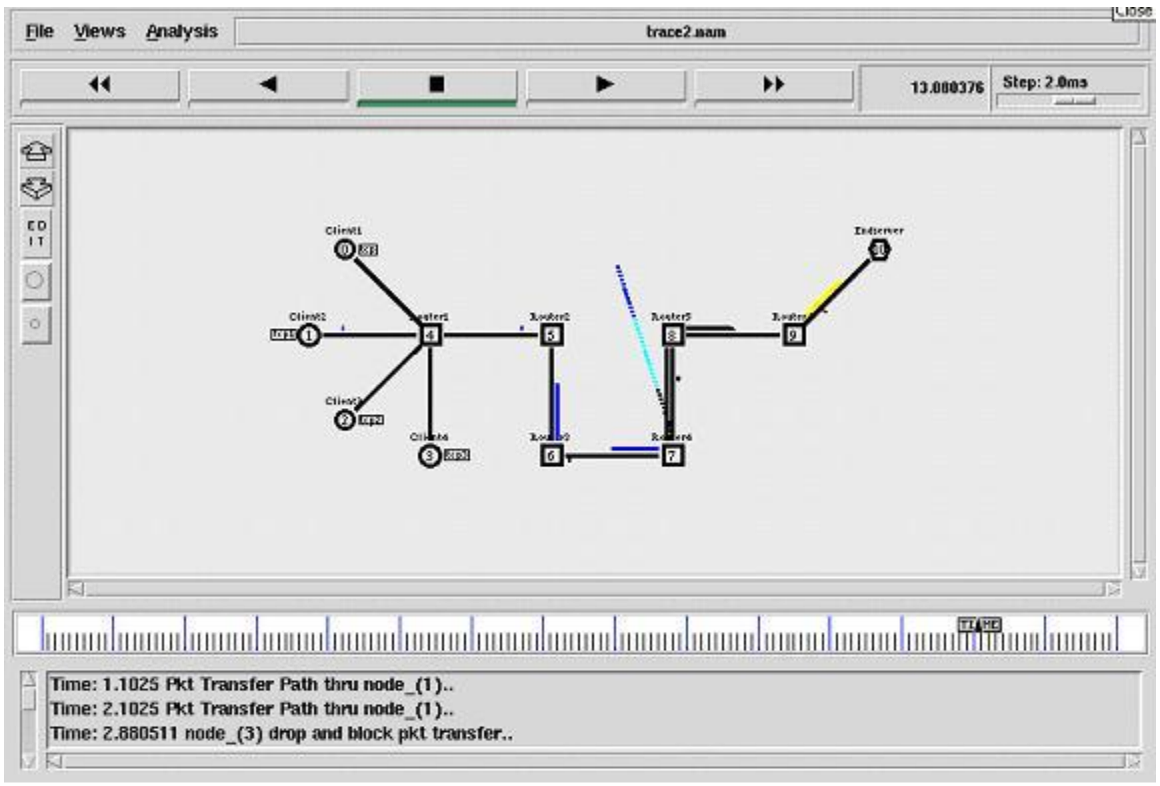
```
$ns at 0.65 "$ftp2 start"
$ns at 28.5 "$ftp2 stop"

#--------------------CLIENT4 TO ENDSERVER----------------#

set tcp3 [new Agent/TCP]
$tcp3 set fid_ 3
$tcp3 set maxcwnd_ 16
$tcp2 set packetsize_ 100
$ns attach-agent $Client4 $tcp3

set sink3 [new Agent/TCPSink]
$ns attach-agent $Endserver1 $sink3

$ns connect $tcp3 $sink3

set ftp3 [new Application/FTP]
$ftp3 attach-agent $tcp3

$ns add-agent-trace $tcp3 tcp3
$tcp3 tracevar cwnd_

$ns at 0.60 "$ftp3 start"
$ns at 28.5 "$ftp3 stop"



$ns rtmodel-at 2.880511 down $Router3 $Router4
$ns rtmodel-at 2.880511 up $Router3 $Router4

$ns rtmodel-at 7.299242 down $Router5 $Router6
$ns rtmodel-at 7.299242 up $Router5 $Router6

#-------------Trace file creation---------#

$ns at 1.1025 "$ns trace-annotate \"Time: 1.1025 Pkt Transfer Path
thru node_(1)..\""
$ns at 2.1025 "$ns trace-annotate \"Time: 2.1025 Pkt Transfer Path
thru node_(1)..\""
$ns at 2.880511 "$ns trace-annotate \"Time: 2.880511 node_(3) drop and
block pkt transfer..\""
$ns at 7.299242 "$ns trace-annotate \"Time: 7.299242 node_(3) drop and
block pkt transfer..\""



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


proc finish {} {



               global ns nf nt
```

```
                $ns flush-trace
                close $nf
                puts "running nam..."
                exec nam trace2.nam &
                exit 0
             }

#Calling finish procedure
$ns at 15.0 "finish"
$ns run
```

#-----How to run-----#

```
$ns filename.tcl
```

#----------Snapshot----------#



```
Time: 1.1025 Pkt Transfer Path thru node_(1)..
Time: 2.1025 Pkt Transfer Path thru node_(1)..
Time: 2.880511 node_(3) drop and block pkt transfer..
```

19.TCL script for Random node communication.

Description:

This network consists of 9 nodes (C1, C2, C3, C4, R1, R2 R3, R4, ROU1, ROU2 and ROU3). The duplex links between have 1Mbps of bandwidth and 10 ms of delay. Each link uses a DropTail queue. TCP connection between the nodes is randomly made and packet transfer is done over the connection.

File name: “RandomTx.tcl”

```
#-------Event scheduler object creation--------#


set ns [ new Simulator ]

#----------creating nam objects----------------#

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

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



set proto rlm

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

$ns color 1 dodgerblue
$ns color 2 red
$ns color 3 cyan
$ns color 4 green
$ns color 5 yellow
$ns color 6 black
$ns color 7 magenta
$ns color 8 gold
$ns color 9 red



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

set C(1) [$ns node]
set C(2) [$ns node]
set C(3) [$ns node]
set C(4) [$ns node]
set R(1) [$ns node]
set R(2) [$ns node]
set R(3) [$ns node]
set R(4) [$ns node]
set ROU(1) [$ns node]
set ROU(2) [$ns node]
set ROU(3) [$ns node]



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

$ns duplex-link $C(1) $ROU(1)  1Mb 10ms DropTail
$ns duplex-link $C(2) $ROU(1)  500Kb 10ms DropTail
$ns duplex-link $C(3) $ROU(1)  750Kb 10ms DropTail
```

```
$ns duplex-link $C(4) $ROU(2)  1Mb 10ms DropTail
$ns duplex-link $R(1) $ROU(1)  1Mb 10ms DropTail
$ns duplex-link $R(2) $ROU(1)  1Mb 10ms DropTail
$ns duplex-link $R(3) $ROU(1)  1Mb 10ms DropTail
$ns duplex-link $R(4) $ROU(3)  1Mb 10ms DropTail

$ns duplex-link $ROU(2) $ROU(1)  1Mb 10ms DropTail
$ns duplex-link $ROU(2) $ROU(3)  1Mb 10ms DropTail
$ns duplex-link $ROU(1) $ROU(3)  1Mb 10ms DropTail

#-------------QUEUE SIZE DESCRIPTION---------------#

$ns queue-limit $ROU(1) $ROU(2) 18
$ns queue-limit $ROU(1) $ROU(3) 18
$ns queue-limit $ROU(2) $ROU(1) 20
$ns queue-limit $ROU(3) $ROU(1) 20



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


$ns duplex-link-op $C(1) $ROU(1) orient down
$ns duplex-link-op $C(2) $ROU(1) orient down-right
$ns duplex-link-op $C(3) $ROU(1) orient down-left
$ns duplex-link-op $C(4) $ROU(2) orient up
$ns duplex-link-op $R(1) $ROU(1) orient up
$ns duplex-link-op $R(2) $ROU(1) orient up-right
$ns duplex-link-op $R(3) $ROU(1) orient up-left
$ns duplex-link-op $R(4) $ROU(3) orient down

$ns duplex-link-op $ROU(1) $ROU(2) orient down-right
$ns duplex-link-op $ROU(3) $ROU(2) orient down-right

# --------------LABELLING -----------------------------#

$ns at 0.0 "$C(1) label CL1"
$ns at 0.0 "$C(2) label CL2"
$ns at 0.0 "$C(3) label CL3"
$ns at 0.0 "$C(4) label CL4"
$ns at 0.0 "$R(1) label RC1"
$ns at 0.0 "$R(2) label RC2"
$ns at 0.0 "$R(3) label RC3"
$ns at 0.0 "$R(4) label RC4"
$ns at 0.0 "$ROU(1) label ROU1"
$ns at 0.0 "$ROU(2) label ROU2"
$ns at 0.0 "$ROU(3) label ROU3"

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

$ROU(1) shape square
$ROU(2) shape square
$ROU(3) shape square

# ----------------QUEUES POSITIONING AND ESTABLISHMENT -------------#
```

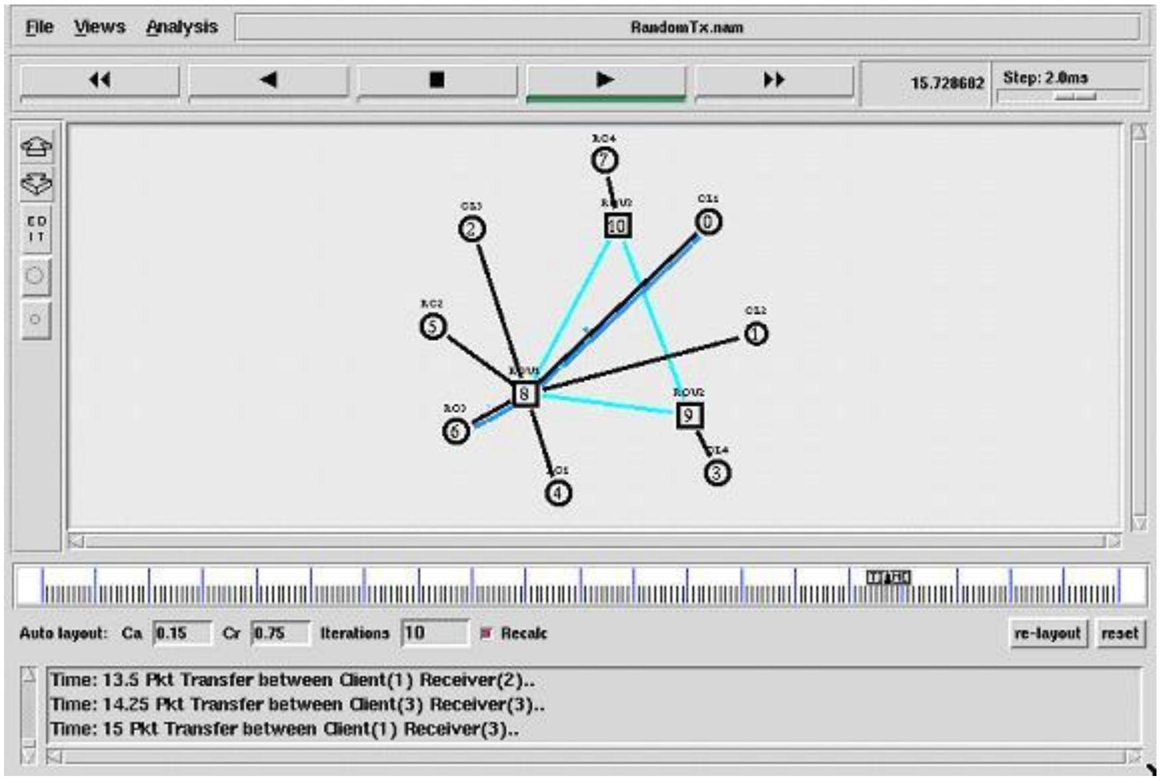
```
$ns duplex-link-op $ROU(2) $ROU(1) queuePos 0.1
#$ns duplex-link-op $ROU(2) $C(5) queuePos 0.1
$ns duplex-link-op $ROU(3) $ROU(1) queuePos 0.1



#-----SETTING IDENTIFICATION COLORS TO ROUTER-LINKS---------------#

$ns duplex-link-op $ROU(1) $ROU(2) color cyan
$ns duplex-link-op $ROU(1) $ROU(3) color cyan
$ns duplex-link-op $ROU(2) $ROU(3) color cyan



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

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


$ns at 0.0 "Tranmission"
proc Tranmission {} {
      global C ROU R ns

      set now [$ns now]
      set time 0.75
      set x [expr round(rand()*4)];if {$x==0} {set x 2}
      set y [expr round(rand()*4)];if {$y==0} {set y 3}

      set tcp1 [$ns create-connection TCP $C($x) TCPSink $R($y) 1]
      $ns at $now "$ns trace-annotate \"Time: $now Pkt Transfer
between Client($x) Receiver($y)..\""
      $tcp1 set class_ 1
      $tcp1 set maxcwnd_ 16
      $tcp1 set packetsize_ 4000
      $tcp1 set fid_ 1
      set ftp1 [$tcp1 attach-app FTP]
      $ftp1 set interval_ .005
      $ns at $now "$ftp1 start"
      $ns at [expr $now+$time] "$ftp1 stop"
      $ns at [expr $now+$time] "Tranmission"
}




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


proc finish {} {

               global ns nf nt nf1

               $ns flush-trace
               close $nf
               puts "running nam..."
```

```
                exec nam RandomTx.nam &
                exit 0
            }



#Calling finish procedure

$ns at 20.0 "finish"
$ns run
```

#-----How to run-----#

```
$ns filename.tcl
```

#----------Snapshot----------#



```
Time: 13.5 Pkt Transfer between Client(1) Receiver(2)..
Time: 14.25 Pkt Transfer between Client(3) Receiver(3)..
Time: 15 Pkt Transfer between Client(1) Receiver(3)..
```

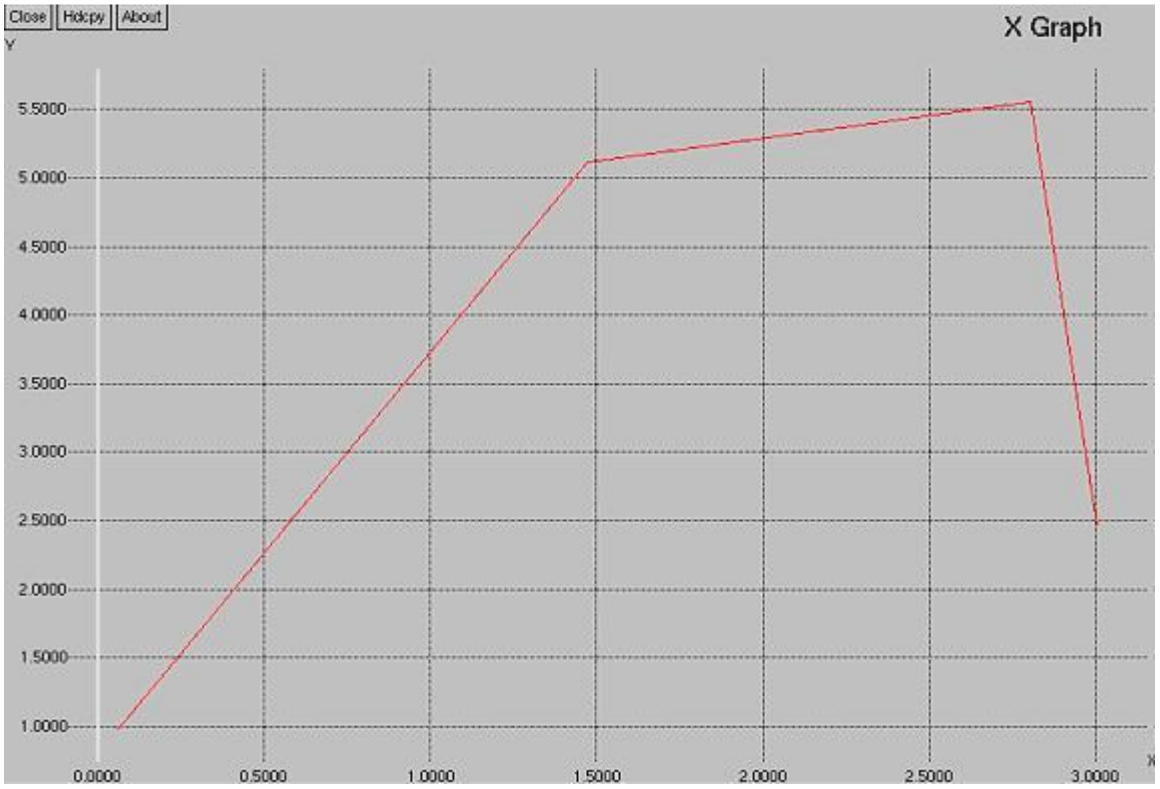
20.TCL script to generate graph.

Description:

X, Y coordinates points for the graph is generated randomly and put it in a trace file. The trace file is given as input file to xgraph to plot the graph.

File name: "graph1.tcl"

```
#-------Event scheduler object creation--------#

set ns [new Simulator]

#Creating nam file:
set nf [open graph1.nam w]
$ns namtrace-all $nf

#Open the trace file
set nt [open graph1.tr w]
$ns trace-all $nt



# graph procedure..
$ns at 1.0 "Graph"
set g [open graph.tr w]
proc Graph {} {
global ns g g1
set time 1.0
set now [$ns now]
puts $g "[expr rand()*8] [expr rand()*6]"

$ns at [expr $now+$time] "Graph"
}

#Finish Procedure..
proc finish {} {
      global ns nf nt
      exec xgraph -ng -geometry 600X500 graph.tr &
exec nam grahp1.nam &
      exit 0
      }

#Calling finish procedure
$ns at 5.0 "finish"
$ns run

# How to run
$ns filename.tcl

#snapshot
```



<u>21.TCL script to generata two graph in a single plot.</u>

Description:

X, Y coordinates points for the graph is generated randomly and put it in a trace file. The trace file is given as input file to xgraph to plot the graph. Here two trace files are plotted in the single graph.

<u>File name: "graph2.tcl"</u>

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

#Creating nam file:
set nf [open Tcpred.nam w]
$ns namtrace-all $nf

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



# graph procedure..
$ns at 1.0 "Graph"
set g [open graph.tr w]
set g1 [open graph1.tr w]

proc Graph {} {
global ns g g1
set time 1.0
set now [$ns now]
puts $g "[expr rand()*8] [expr rand()*6]"
puts $g1 "[expr rand()*8] [expr rand()*6]"
$ns at [expr $now+$time] "Graph"
}
```

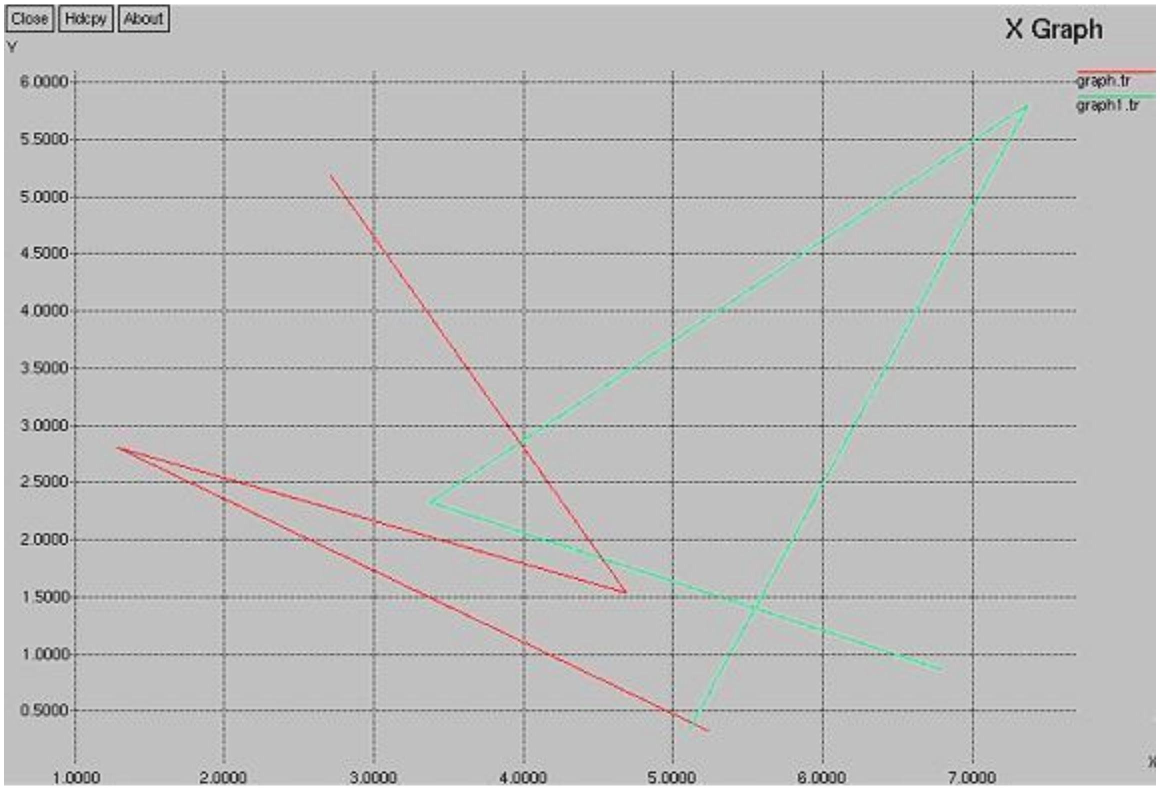
#Finish Procedure..

```
proc finish {} {
      global ns nf nt
      exec xgraph -brb -geometry 600X500 graph.tr graph1.tr &
      exit 0
      }
```

```
#Calling finish procedure
$ns at 5.0 "finish"
$ns run
```

# How to run
$ns filename.tcl

#snapshot



22.TCL script to draw graph using more trace input file.

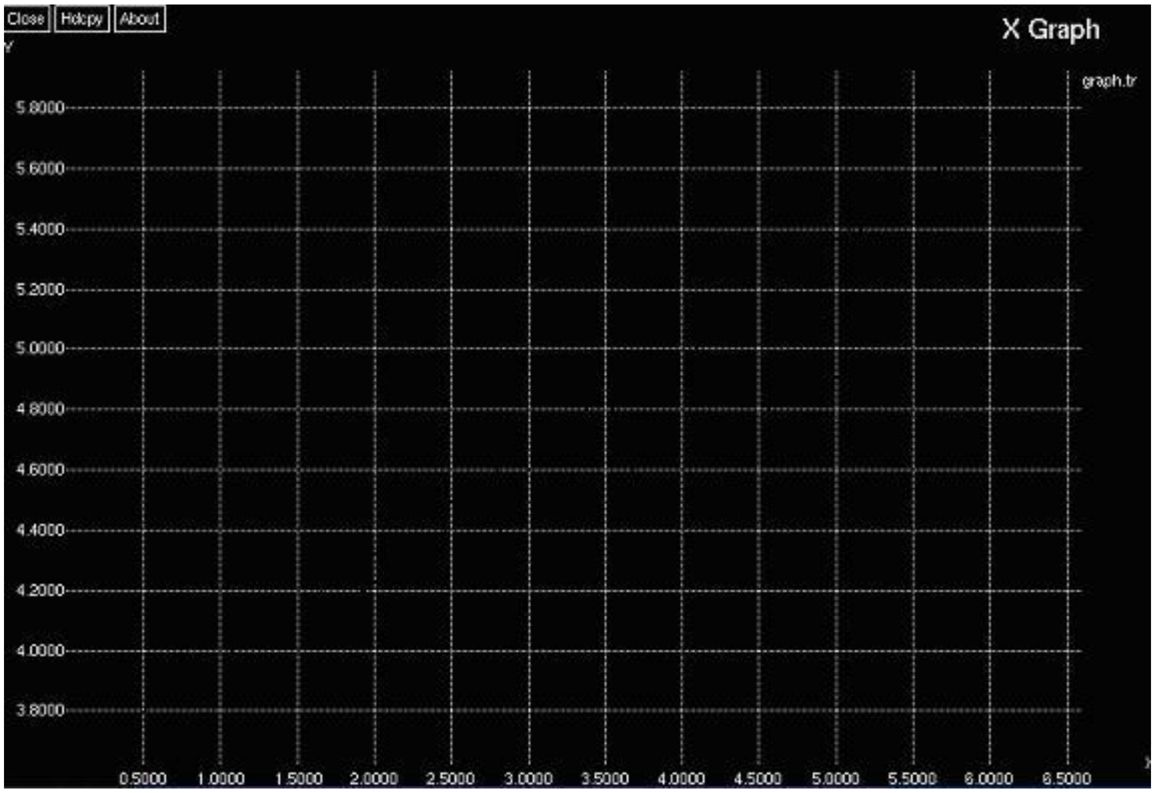
Description:

X, Y coordinates points for more than one graph is generated

randomly and put it in the trace files. All the trace files are given as input file to xgraph to plot all the graphs in the same plot.

<u>File name: "graph2.tcl"</u>

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

#Creating nam file:
set nf [open Tcpred.nam w]
$ns namtrace-all $nf

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



# graph procedure..
$ns at 1.0 "Graph"
set g [open graph.tr w]
set g1 [open graph1.tr w]

proc Graph {} {
global ns g g1
set time 1.0
set now [$ns now]
puts $g "[expr rand()*8] [expr rand()*6]"
puts $g1 "[expr rand()*8] [expr rand()*6]"
$ns at [expr $now+$time] "Graph"
}


#Finish Procedure..
proc finish {} {
      global ns nf nt
      exec xgraph -brb -geometry 600X500 graph.tr graph1.tr &
      exit 0
      }

#Calling finish procedure
$ns at 5.0 "finish"
$ns run

# How to run
$ns filename.tcl

#snapshot
```



23.Tcl script to construct complex wired network

Description:

This network consists of 26 nodes . The duplex link construction is very complex.Set the labels for every node. Setting orientation between nodes in very different way.

Code:

```
# Create a simulator object

set ns [new Simulator -multicast on]

# Open the nam trace file, associated with nf, log everything as nam
output in nf

set nf [open out.nam w]

$ns namtrace-all $nf

set node_(n0) [$ns node]

set node_(n1) [$ns node]

set node_(n2) [$ns node]

set node_(n3) [$ns node]

set node_(n4) [$ns node]
set node_(n5) [$ns node]

set node_(n6) [$ns node]

set node_(n7) [$ns node]

set node_(n8) [$ns node]

set node_(n9) [$ns node]

set node_(n10) [$ns node]

set node_(n11) [$ns node]

set node_(n12) [$ns node]

set node_(n13) [$ns node]

set node_(n14) [$ns node]

set node_(n15) [$ns node]

set node_(n16) [$ns node]

set node_(n17) [$ns node]

set node_(n18) [$ns node]

set node_(n19) [$ns node]

set node_(n20) [$ns node]

set node_(n21) [$ns node]

set node_(n22) [$ns node]

set node_(n23) [$ns node]

set node_(n24) [$ns node]

set node_(n25) [$ns node]

#Create a duplex link between the nodes

$ns duplex-link $node_(n0) $node_(n2) 3.0Mb 10ms DropTail

$ns duplex-link $node_(n0) $node_(n1) 3.0Mb 10ms DropTail

$ns duplex-link $node_(n1) $node_(n3) 2.0Mb 10ms DropTail

$ns duplex-link $node_(n3) $node_(n4) 1.5Mb 10ms DropTail

$ns duplex-link $node_(n3) $node_(n7) 1.5Mb 10ms DropTail
```

```
$ns duplex-link $node_(n2) $node_(n5) 1.0Mb 10ms DropTail

$ns duplex-link $node_(n2) $node_(n6) 1.5Mb 10ms DropTail

$ns duplex-link $node_(n0) $node_(n3) 1.5Mb 10ms DropTail

$ns duplex-link $node_(n8) $node_(n9) 2.0Mb 10ms DropTail

$ns duplex-link $node_(n8) $node_(n10) 3.0Mb 10ms DropTail

$ns duplex-link $node_(n9) $node_(n11) 2.0Mb 10ms DropTail

$ns duplex-link $node_(n11) $node_(n12) 1.5Mb 10ms DropTail

$ns duplex-link $node_(n11) $node_(n15) 1.5Mb 10ms DropTail

$ns duplex-link $node_(n10) $node_(n13) 1.0Mb 10ms DropTail

$ns duplex-link $node_(n10) $node_(n14) 1.5Mb 10ms DropTail

$ns duplex-link $node_(n8) $node_(n11) 1.5Mb 10ms DropTail

$ns duplex-link $node_(n16) $node_(n18) 3.0Mb 10ms DropTail

$ns duplex-link $node_(n16) $node_(n17) 3.0Mb 10ms DropTail

$ns duplex-link $node_(n17) $node_(n19) 2.0Mb 10ms DropTail

$ns duplex-link $node_(n19) $node_(n20) 1.5Mb 10ms DropTail

$ns duplex-link $node_(n19) $node_(n23) 1.5Mb 10ms DropTail

$ns duplex-link $node_(n18) $node_(n21) 1.0Mb 10ms DropTail

$ns duplex-link $node_(n18) $node_(n22) 1.5Mb 10ms DropTail

$ns duplex-link $node_(n16) $node_(n19) 1.5Mb 10ms DropTail

$ns duplex-link $node_(n24) $node_(n0) 3.0Mb 10ms DropTail

$ns duplex-link $node_(n24) $node_(n8) 3.0Mb 10ms DropTail

$ns duplex-link $node_(n25) $node_(n8) 3.0Mb 10ms DropTail

$ns duplex-link $node_(n16) $node_(n25) 3.0Mb 10ms DropTail

#setting orientation of links

$ns duplex-link-op $node_(n0) $node_(n3) orient down

$ns duplex-link-op $node_(n0) $node_(n1) orient left-down

$ns duplex-link-op $node_(n0) $node_(n2) orient right-down
```

```
$ns duplex-link-op $node_(n1) $node_(n3) orient right-down

$ns duplex-link-op $node_(n3) $node_(n4) orient left-down

$ns duplex-link-op $node_(n3) $node_(n7) orient right-down

$ns duplex-link-op $node_(n2) $node_(n5) orient right-down

$ns duplex-link-op $node_(n2) $node_(n6) orient right

$ns duplex-link-op $node_(n8) $node_(n11) orient down

$ns duplex-link-op $node_(n8) $node_(n9) orient left-down

$ns duplex-link-op $node_(n8) $node_(n10) orient right-down

$ns duplex-link-op $node_(n9) $node_(n11) orient right-down

$ns duplex-link-op $node_(n11) $node_(n12) orient left-down

$ns duplex-link-op $node_(n11) $node_(n15) orient right-down

$ns duplex-link-op $node_(n10) $node_(n13) orient right-down

$ns duplex-link-op $node_(n10) $node_(n14) orient right

$ns duplex-link-op $node_(n16) $node_(n19) orient down

$ns duplex-link-op $node_(n16) $node_(n17) orient left-down

$ns duplex-link-op $node_(n16) $node_(n18) orient right-down

$ns duplex-link-op $node_(n17) $node_(n19) orient right-down

$ns duplex-link-op $node_(n19) $node_(n20) orient left-down

$ns duplex-link-op $node_(n19) $node_(n23) orient right-down

$ns duplex-link-op $node_(n18) $node_(n21) orient right-down

$ns duplex-link-op $node_(n18) $node_(n22) orient right

$ns duplex-link-op $node_(n24) $node_(n0) orient left-down

$ns duplex-link-op $node_(n24) $node_(n8) orient right-down

$ns duplex-link-op $node_(n25) $node_(n8) orient left-down

$ns duplex-link-op $node_(n25) $node_(n16) orient right-down

$node_(n0) set X_ -57.6704

$node_(n0) set Y_ 74.9762
```

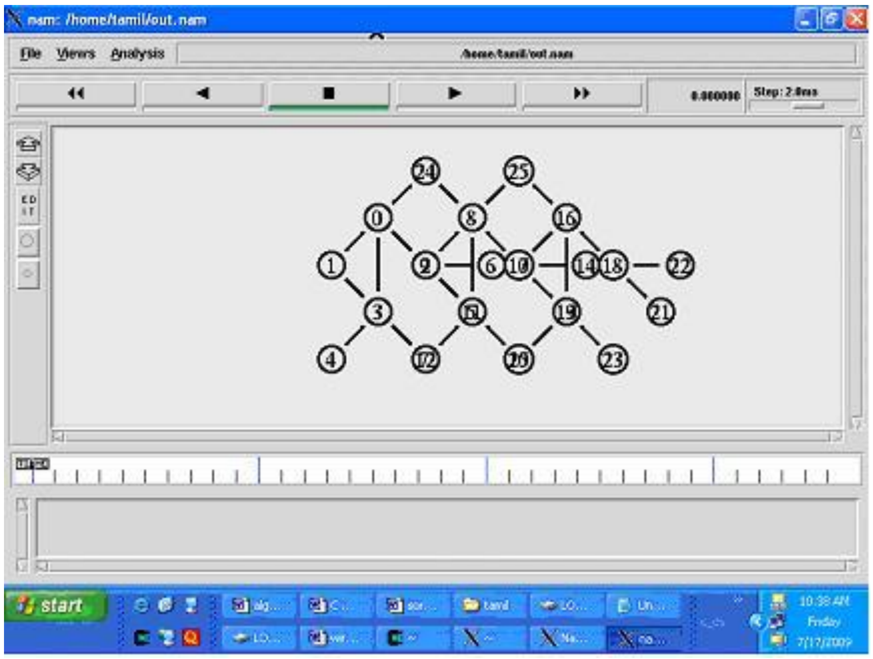
```
$node_(n0) set Z_ 0.0

$node_(n1) set X_ -74.9762

$node_(n1) set Y_ 50.9936

$node_(n1) set Z_ 0.0

$node_(n2) set X_ -36.0815

$node_(n2) set Y_ 48.2768

$node_(n2) set Z_ 0.0

$node_(n3) set X_ -56.8873

$node_(n3) set Y_ 38.8308

$node_(n3) set Z_ 0.0

$node_(n4) set X_ -76.5445

$node_(n4) set Y_ 10.1674

$node_(n4) set Z_ 0.0

$node_(n5) set X_ -38.0313

$node_(n5) set Y_ 7.9817

$node_(n5) set Z_ 0.0

$node_(n6) set X_ -19.8069

$node_(n6) set Y_ 7.9817

$node_(n6) set Z_ 0.0

$node_(n7) set X_ -57.8945

$node_(n7) set Y_ 10.2669

$node_(n7) set Z_ 0.0

$node_(n8) set X_ 39.0212

$node_(n8) set Y_ 74.9762

$node_(n8) set Z_ 0.0

$node_(n9) set X_ 8.07317

$node_(n9) set Y_ 50.9936
```

```
$node_(n9) set Z_ 0.0

$node_(n10) set X_ 88.6017

$node_(n10) set Y_ 50.9936

$node_(n10) set Z_ 0.0

$node_(n11) set X_ 37.3668

$node_(n11) set Y_ 50.9936

$node_(n11) set Z_ 0.0

$node_(n12) set X_ 16.88

$node_(n12) set Y_ 10.1674

$node_(n12) set Z_ 0.0

$node_(n13) set X_ 64.635

$node_(n13) set Y_ 27.4717

$node_(n13) set Z_ 0.0

$node_(n14) set X_ 86.0753

$node_(n14) set Y_ 7.9817

$node_(n14) set Z_ 0.0

$node_(n15) set X_ 45.6967

$node_(n15) set Y_ 29.2669

$node_(n15) set Z_ 0.0

$node_(n16) set X_ 148.967

$node_(n16) set Y_ 72.6919

$node_(n16) set Z_ 0.0

$node_(n17) set X_ 125.115

$node_(n17) set Y_ 49.8054

$node_(n17) set Z_ 0.0

$node_(n18) set X_ 183.037

$node_(n18) set Y_ 49.8054
```

```
$node_(n18) set Z_ 0.0

$node_(n19) set X_ 152.962

$node_(n19) set Y_ 50.868

$node_(n19) set Z_ 0.0

$node_(n20) set X_ 133.147

$node_(n20) set Y_ 10.096

$node_(n20) set Z_ 0.0

$node_(n21) set X_ 208.519

$node_(n21) set Y_ 11.5933

$node_(n21) set Z_ 0.0

$node_(n22) set X_ 183.386

$node_(n22) set Y_ 10.0766

$node_(n22) set Z_ 0.0

$node_(n23) set X_ 161.768

$node_(n23) set Y_ 10.4865

$node_(n23) set Z_ 0.0

$node_(n24) set X_ -10.6704

$node_(n24) set Y_ 100.9762

$node_(n24) set Z_ 0.0

$node_(n25) set X_ 100.00

$node_(n25) set Y_ 100.9762

$node_(n25) set Z_ 0.0

$ns at 0.0 "$node_(n0) label S"

$ns at 0.0 "$node_(n1) label N1"

$ns at 0.0 "$node_(n2) label N3"

$ns at 0.0 "$node_(n3) label N2"

$ns at 0.0 "$node_(n4) label R1"
```

```
$ns at 0.0 "$node_(n7) label R2"

$ns at 0.0 "$node_(n5) label R3"

$ns at 0.0 "$node_(n6) label R4"

$ns at 0.0 "$node_(n8) label S"

$ns at 0.0 "$node_(n9) label N1"

$ns at 0.0 "$node_(n10) label N3"

$ns at 0.0 "$node_(n11) label N2"

$ns at 0.0 "$node_(n12) label R1"

$ns at 0.0 "$node_(n13) label R2"

$ns at 0.0 "$node_(n14) label R3"

$ns at 0.0 "$node_(n15) label R4"

$ns at 0.0 "$node_(n16) label S"

$ns at 0.0 "$node_(n17) label N1"

$ns at 0.0 "$node_(n18) label N3"

$ns at 0.0 "$node_(n19) label N2"

$ns at 0.0 "$node_(n20) label R1"

$ns at 0.0 "$node_(n21) label R2"

$ns at 0.0 "$node_(n22) label R3"

$ns at 0.0 "$node_(n23) label R4"

$ns at 0.0 "$node_(n24) label Overlay"

$ns at 0.0 "$node_(n25) label Overlay"

$ns color 1 blue

# 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

}

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

# Run the simulation

$ns run
```

Screen shows the critical node connection:



24.TCL script for merger two Tcl script source files.

Description:

This sample script creates only three wired nodes namely client, router and end server. The main objective of this program is to understand include one tcl source file into another one.

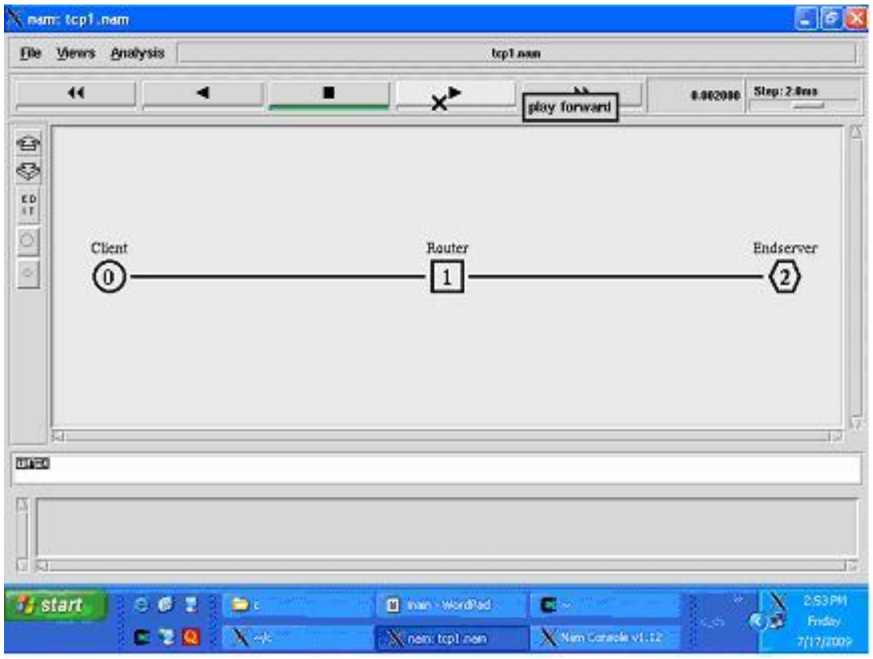
Source file 1: (main.tcl)

```
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 Client"

$ns at 0.0 "$Router1 label Router"

$ns at 0.0 "$Endserver1 label Endserver"

#-----------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

#--------Include finish.tcl ----------------#

source "finish.tcl"

source file 2: (finish.tcl)

#---------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
```

Sample screen show's three different kind of nodes.



25.Tcl script to create CBR traffic object

Description:

This tcl script generates three wired nodes n0, n1 and n2.It uses the duplex link for node connection. Bandwidth between the nodes n(0) and n(1) is 1 Mbps and network delay is 10ms and it use Drop tail queue. Bandwidth between the nodes n (1) and n (2) as 512Kbps and network delay is 10ms and it use Drop tail queue. Here create the CBR traffic between the nodes n (0), n (1) and also between the nodes n (1), n (2).

Code:

```
#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]

set n2 [$ns node]

#Create a duplex link between the nodes

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

$ns duplex-link $n1 $n2 512Kbps 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

#Create a Null agent (a traffic sink) and attach it to node n1

set null0 [new Agent/Null]

$ns attach-agent $n2 $null0

#Connect the traffic source with the traffic sink

$ns connect $udp0 $null0

#Schedule events for the CBR agent

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

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

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"

#Run the simulation

$ns run
```

26.Tcl script to create bottleneck network.

Description:

This tcl script generate the three wired nodes n0,n1 and n2.It use the duplex link for node connection. Band width between the nodes n(0) and n(1) is 2 MBs and network delay is 10ms and it use Drop tail queue. Bandwidth between the nodes n(1) and n(2) as 128kbs and network delay is 10ms and it use Drop tail queue .Here create the CBR traffic Between the nodes n(0),n(1) and also between the nodes n(1),n(2). In this script we connect the n(0) with n(1) very high Bandwidth and n(1) with n(2) very low bandwidth. Here we make the communication from n(0) to n(2) through the intermediate node n(1).

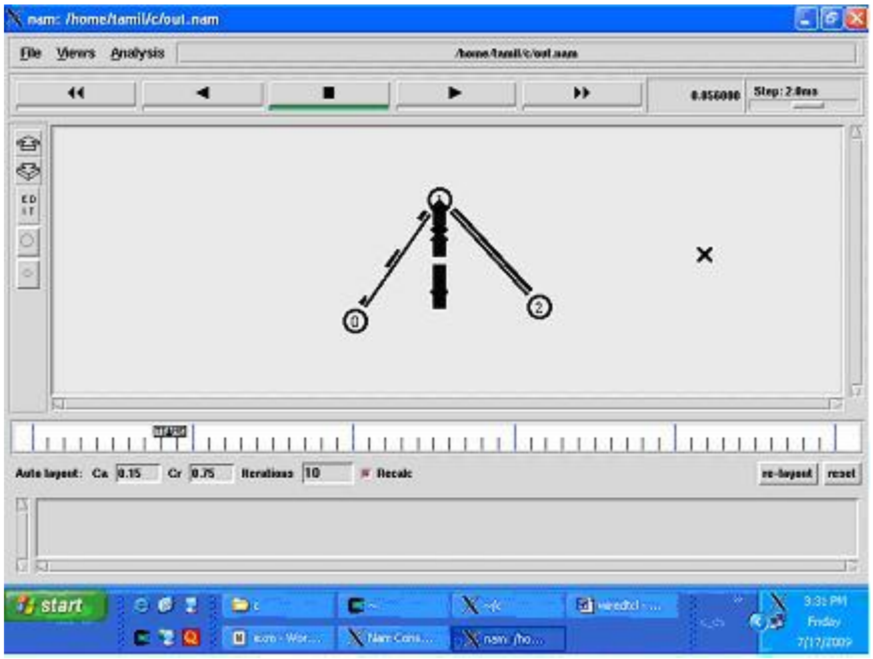
So Bottle neck situation is created at the node n(1).

Code:

```
#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]

set n2 [$ns node]

#Create a duplex link between the nodes

$ns duplex-link $n0 $n1 2Mb 10ms DropTail

$ns duplex-link $n1 $n2 128Kbps 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

#Create a Null agent (a traffic sink) and attach it to node n1

set null0 [new Agent/Null]

$ns attach-agent $n2 $null0

#Connect the traffic source with the traffic sink

$ns connect $udp0 $null0

#Schedule events for the CBR agent

$ns at 0.5 "$cbr0 start"

$ns at 4.5 "$cbr0 stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 5.0 "finish"
```

```
#Run the simulation

$ns run
```

Screen for bottle neck node.



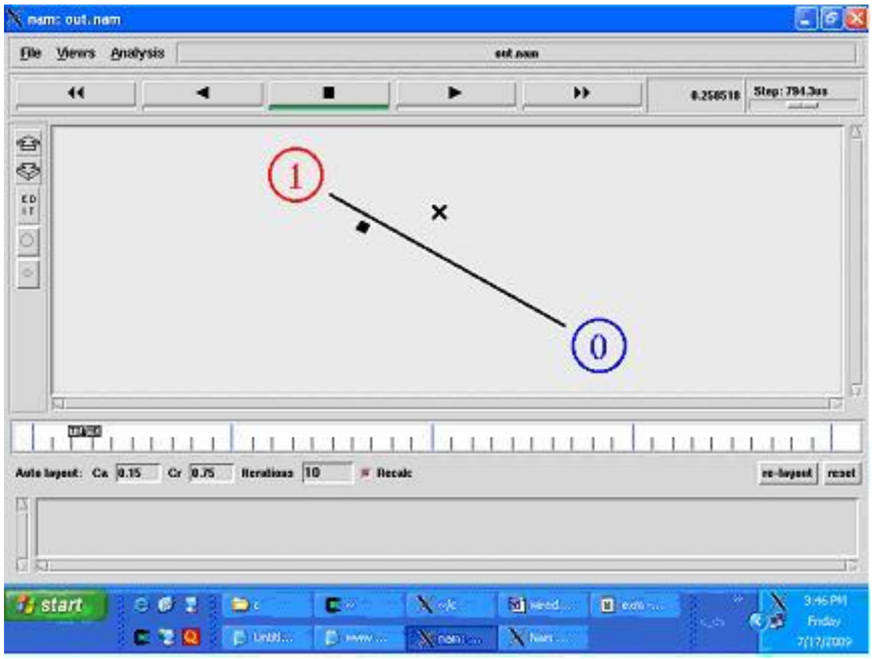
27.TCL script to create WWW traffic .

Description:

This tcl script generates the two wired nodes n0 and .It use the duplex link for node connection. Band width between the nodes n(0) and n(1) is 1 MBs and network delay is 10ms and it use Drop tail queue. .Here create the WWW traffic Between the nodes n(0),n(1) . In this script we connect the n(0) with n(1) very high Bandwidth . Here we make the communication from n(0) to n(1) 0.2 secs.

```
#Create a simulator object

set ns [new Simulator]

#Open a 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 $nf

exec nam out.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

$n0 color blue

$n1 color red

#Connect the nodes with two links

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

proc www_traffic { node0 node1 } {

global ns

set www_UDP_agent [new Agent/UDP]

set www_UDP_sink [new Agent/Null]

$ns attach-agent $node0 $www_UDP_agent

$ns attach-agent $node1 $www_UDP_sink

$ns connect $www_UDP_agent $www_UDP_sink

set www_CBR_source [new Application/Traffic/CBR]

$www_CBR_source attach-agent $www_UDP_agent

$www_CBR_source set packetSize_ 48

$www_CBR_source set interval_ 50ms

$ns at 0.2 "$www_CBR_source start"

#$ns at 0.2 "$cbr0 start"

$ns at 4.5 "$www_CBR_source stop"

}

www_traffic $n0 $n1

$ns at 4.0 "finish"
```

```
$ns run
```



<u>28.TCL script to create SMTP traffic.</u>

Description:

This tcl script generates the two wired nodes n0 and n(1) .It uses the duplex link for node connection. Bandwidth between the nodes n(0) and n(1) is 1 Mbps and network delay is 10ms and it uses Drop tail queue. Here we create the SMTP traffic between the nodes n(0) and n(1) . In this script we connect the n(0) and n(1) with very high bandwidth . Here we start the communication from n(0) to n(1) at 0.2 second.

```
#Create a simulator object

set ns [new Simulator]

#Open a 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 $nf
exec nam out.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

$n0 color blue

$n1 color red

#Connect the nodes with two links

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

proc smtp_traffic {node0 node1 } {

global ns

set smtp_UDP_agent [new Agent/UDP]

set smtp_UDP_sink [new Agent/UDP]

$ns attach-agent $node0 $smtp_UDP_agent

$ns attach-agent $node1 $smtp_UDP_sink

$ns connect $smtp_UDP_agent $smtp_UDP_sink

set smtp_UDP_source [new Application/Traffic/Exponential]

$smtp_UDP_source attach-agent $smtp_UDP_agent

$smtp_UDP_source set packetSize_ 210

$smtp_UDP_source set burst_time_ 50ms

$smtp_UDP_source set idle_time_ 50ms

$smtp_UDP_source set rate_ 100k

$ns at 0.2 "$smtp_UDP_source start"

$ns at 3.2 "$smtp_UDP_source stop"

}

smtp_traffic $n0 $n1

$ns at 4.0 "finish"
```

```
$ns run
```

29.TCL script to create exponential traffic.

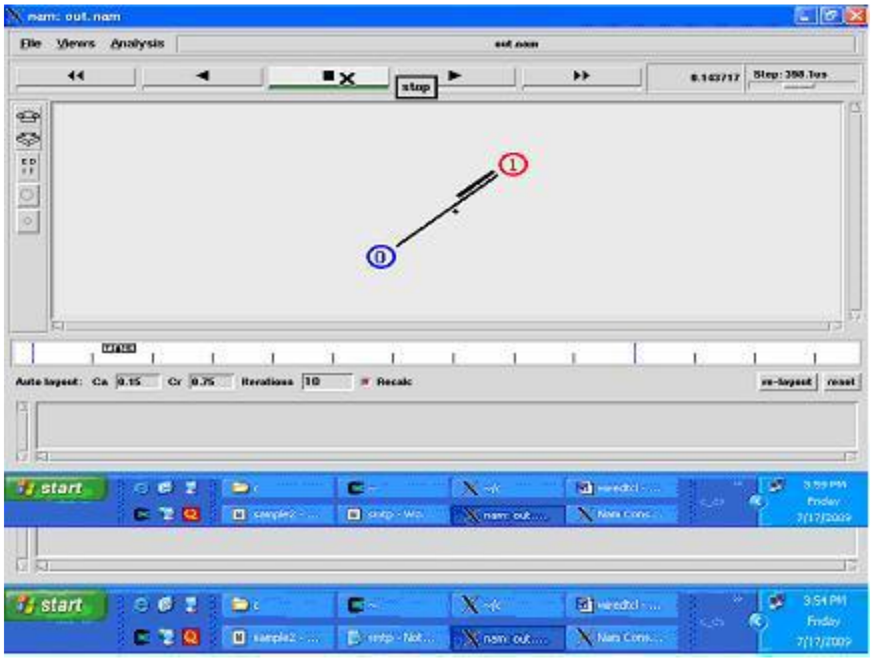
Description:

This tcl script generates the two wired nodes n0 and .It use the duplex link for node connection. Link Bandwidth between the nodes n(0) and n(1) is 1 Mbps and network delay is 10ms and it uses Drop tail queue. Here create the exponential traffic between the nodes n(0) and n(1) . In this script we connect the n(0) and n(1) with very high bandwidth . Here we start the communication from n(0) to n(1) at 0.2 second.

```
#Create a simulator object

set ns [new Simulator]

#Open a 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 $nf

exec nam out.nam &

exit 0

}

set n0 [$ns node]

set n1 [$ns node]

$n0 color blue

$n1 color red

#Connect the nodes with two links

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

proc smtp_traffic {node0 node1 } {
```

```
global ns

set smtp_UDP_agent [new Agent/UDP]

set smtp_UDP_sink [new Agent/UDP]

$ns attach-agent $node0 $smtp_UDP_agent

$ns attach-agent $node1 $smtp_UDP_sink

$ns connect $smtp_UDP_agent $smtp_UDP_sink

set smtp_UDP_source [new Application/Traffic/Exponential]

$smtp_UDP_source attach-agent $smtp_UDP_agent

$smtp_UDP_source set packetSize_ 210

$smtp_UDP_source set burst_time_ 50ms

$smtp_UDP_source set idle_time_ 50ms

$smtp_UDP_source set rate_ 100k

$ns at 0.2 "$smtp_UDP_source start"

$ns at 3.2 "$smtp_UDP_source stop"

}

smtp_traffic $n0 $n1

$ns at 4.0 "finish"

$ns run
```

Screens for FTP_Traffic:



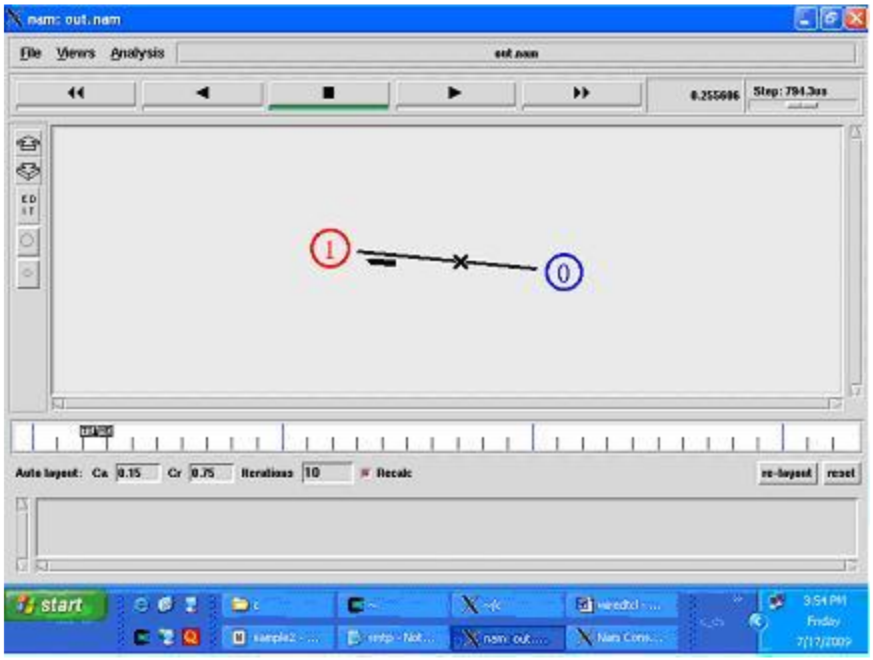
30. TCL script to create telnet traffic.

Description:

This tcl script generates two wired nodes n0 and .It uses the duplex link for the node connection. Bandwidth between the nodes n(0) and n(1) is 1 Mbps and network delay is 10ms and it uses Drop tail queue. .Here we create the telnet traffic between the nodes n(0) and n(1) . In this script we connect the n(0) n(1) with very high bandwidth . Here we start the communication from n(0) to n(1) at 0.2 second.

```
#Create a simulator object

set ns [new Simulator]

#Open a 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 $nf

exec nam out.nam &
exit 0

}

set n0 [$ns node]

set n1 [$ns node]

$n0 color blue

$n1 color red

#Connect the nodes with two links

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

proc telnet_traffic {node0 node1 } {

global ns

set telnet_TCP_agent [new Agent/TCP]

set telnet_TCP_sink [new Agent/TCPSink]

$ns attach-agent $node0 $telnet_TCP_agent

$ns attach-agent $node1 $telnet_TCP_sink

$ns connect $telnet_TCP_agent $telnet_TCP_sink

set telnet_TELNET_source [new Application/Telnet]

$telnet_TELNET_source attach-agent $telnet_TCP_agent

$telnet_TELNET_source set interval_ 20

$ns at 0.2 "$telnet_TELNET_source start"

$ns at 4.0 "$telnet_TELNET_source stop"

}

telnet_traffic $n0 $n1

$ns at 7.0 "finish"

$ns run
```

out put screen



31.Tcl script to understand more trace techniques

Description:

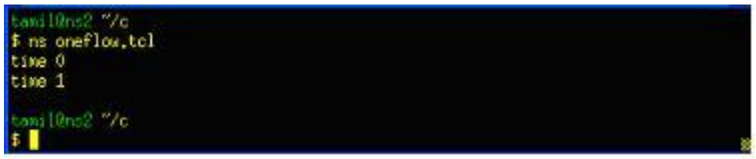
Demonstrates setting up a simple dumbell network and sending

```
# one-way long-lived TCP traffic

proc main {} {

set end 2.0

# setup simulator

remove-all-packet-headers; # removes all packet headers - saves memory

add-packet-header IP TCP; # adds TCP/IP headers

set ns_ [new Simulator]; # instantiate the simulator

#-----------------------------------------------------------------------
----

# SETUP NETWORK

#-----------------------------------------------------------------------
----

# create nodes

for {set i 0} {$i < 2} {incr i} {

set n($i) [$ns_ node]; # end nodes
set r($i) [$ns_ node]; # router nodes

}

# create links

$ns_ duplex-link $r(0) $r(1) 10Mbps 1ms DropTail; # between routers


# links between end nodes and routers (order of nodes doesn't matter)

$ns_ duplex-link $n(0) $r(0) 10Mbps 1ms DropTail

$ns_ duplex-link $n(1) $r(1) 10Mbps 1ms DropTail

#----------------------------------------------------------------------
----

# SETUP END NODES

#----------------------------------------------------------------------
----

set tcpsrc [new Agent/TCP/Reno]; # create TCP sending Agent

$tcpsrc set fid_ 1

$tcpsrc set packetSize_ 1460; # all packets have same size

$tcpsrc set window_ 64; # maximum congestion window size (pckts)

$ns_ attach-agent $n(0) $tcpsrc; # all Agents must be attached to a
node

set tcpsink [new Agent/TCPSink]; # TCP receiver

$ns_ attach-agent $n(1) $tcpsink

#----------------------------------------------------------------------
----

# SETUP TRACING

#----------------------------------------------------------------------
----

# tracing TCP variables

set tracevar_chan_ [open "|gzip > tracevar.out.gz" w]; # trace file

$tcpsrc attach $tracevar_chan_; # attach to TCP Agent

$tcpsrc tracevar cwnd_; # trace cwnd
```

```
# can trace anything in tcp.h defined with Traced*

# (t_seqno_, ssthresh_, t_rtt_, dupacks_, ...)

# tracing links

# Trace set show_tcphdr_ 1; # displays extra TCP header info

# trace all packet events between src and 1st router

set trq_src0_ [open "|gzip > TCP-src0.trq.gz" w]

$ns_ trace-queue $n(0) $r(0) $trq_src0_

$ns_ trace-queue $r(0) $n(0) $trq_src0_; # order matters here

# trace all packet events between dst and 2nd router

set trq_dst1_ [open "|gzip > TCP-dst1.trq.gz" w]

$ns_ trace-queue $n(1) $r(1) $trq_dst1_

$ns_ trace-queue $r(1) $n(1) $trq_dst1_

# trace all packet events between the two routers

set trq_01_ [open "|gzip > TCP-q01.trq.gz" w]

$ns_ trace-queue $r(0) $r(1) $trq_01_

$ns_ trace-queue $r(1) $r(0) $trq_01_

#-----------------------------------------------------------------------
----

# MAKE THE CONNECTION

#-----------------------------------------------------------------------
----

$ns_ connect $tcpsrc $tcpsink

$tcpsink listen

#-----------------------------------------------------------------------
----

# SETUP TRAFFIC

#-----------------------------------------------------------------------
----

# FTP application
```

```
set ftp [new Application/FTP]

$ftp attach-agent $tcpsrc

#----------------------------------------------------------------------
----

# MAKE SCHEDULE

#----------------------------------------------------------------------
----

# schedule this FTP flow

$ns_ at 0.0 "$ftp start"; # send packets forever

$ns_ at $end "$ftp stop"

# The above could be done without an FTP Application with this line:

# $ns_ at 0.0 "$tcpsrc send -1"; # send packets forever

$ns_ at $end "finish"; # end of simulation

# output progress

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

$ns_ at $i "puts stderr \"time $i\""

}

$ns_ run; # GO!

}

proc finish {} {

# insert post-processing code here

#----------------------------------------------------------------------
----

# EXAMPLES

#----------------------------------------------------------------------
----

# printing output:

# puts "Finished!"

# puts stderr "Finished!"
# puts [format "%f" [expr ($timeouts / $segs)]]

# running shell commands:

# exec zcat TCP-q01.trq.gz | awk {{if ($1=="+" && $5=="tcp" && $6>40)
print $6}}

exit 0;

}

main
```

output:



```
tamil@ns2 ~/c
$ ns oneflow.tcl
time 0
time 1

tamil@ns2 ~/c
$
```

32. Tcl script for create Multicast network

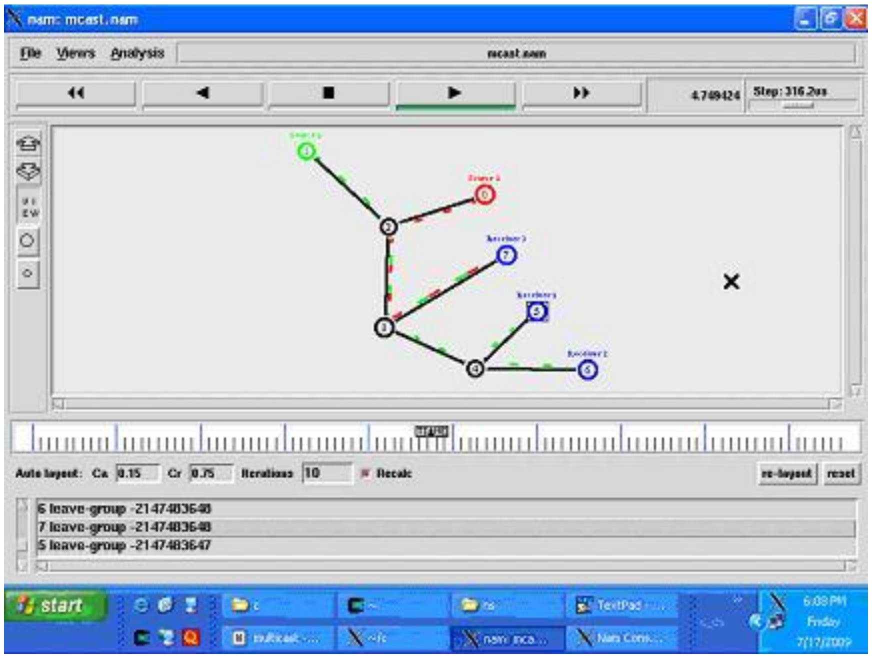
```
# Create scheduler

#Create an event scheduler wit multicast turned on

set ns [new Simulator -multicast on]

#$ns multicast

#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

$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

set mproto DM

set mrthandle [$ns mrtproto $mproto {}]

# Allocate group addresses

set group1 [Node allocaddr]

set group2 [Node allocaddr]

# UDP Transport agent for the traffic source

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

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

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

$ns at 3.5 "$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

}

# For nam

#Colors for packets from two mcast groups

$ns color 10 red

$ns color 11 green

$ns color 30 purple

$ns color 31 green

# Manual layout: order of the link is significant!

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

#$ns duplex-link-op $n0 $n2 orient right-up

#$ns duplex-link-op $n0 $n3 orient right-down
```

```
# Show queue on simplex link n0->n1

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

# Group 0 source

$udp0 set fid_ 10

$n0 color red

$n0 label "Source 1"

# Group 1 source

$udp1 set fid_ 11

$n1 color green

$n1 label "Source 2"

$n5 label "Receiver 1"

$n5 color blue

$n6 label "Receiver 2"

$n6 color blue

$n7 label "Receiver 3"

$n7 color blue

#$n2 add-mark m0 red

#$n2 delete-mark m0"

# Animation rate

$ns set-animation-rate 3.0ms

$ns run
```

Sample screen for multicasting structure.



```
6 leave-group -2147483648
7 leave-group -2147483648
5 leave-group -2147483647
```

33. Tcl script for Router handles ftp and cbr traffic simultaneously

Description:

This network consists of 4 nodes (n0, n1, n2 and n3). 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 link uses the DropTail queue. 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 "TCPSink" 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 1000bytes packets. 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.

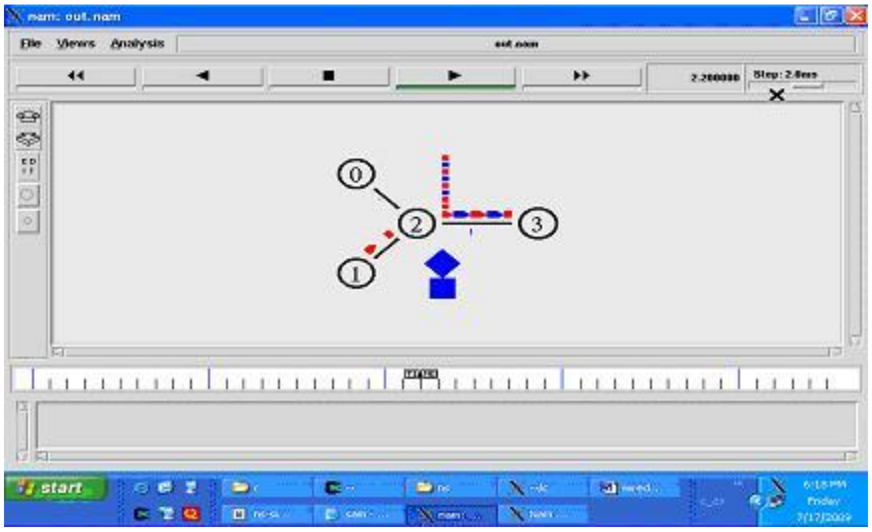
Code:

```
#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
```



34. TCL script for tracing fulltcp flow in zip files

Description:

Demonstrates setting up a simple dumbell network and sending

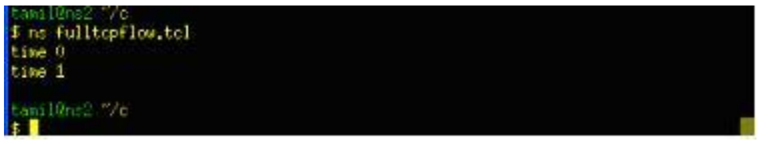
```
# one-way long-lived FullTCP traffic

proc main {} {

set end 2.0

# setup simulator

remove-all-packet-headers; # removes all packet headers - saves memory

add-packet-header IP TCP; # adds TCP/IP headers

set ns_ [new Simulator]; # instantiate the simulator

#-----------------------------------------------------------------------
----

# SETUP NETWORK

#-----------------------------------------------------------------------
----

# create nodes

for {set i 0} {$i < 2} {incr i} {

set n($i) [$ns_ node]; # end nodes

set r($i) [$ns_ node]; # router nodes

}

# create links

$ns_ duplex-link $r(0) $r(1) 10Mbps 1ms DropTail; # between routers

# links between end nodes and routers (order of nodes doesn't matter)

$ns_ duplex-link $n(0) $r(0) 10Mbps 1ms DropTail

$ns_ duplex-link $n(1) $r(1) 10Mbps 1ms DropTail

#-----------------------------------------------------------------------
----

# SETUP END NODES

#-----------------------------------------------------------------------
----

set tcpsrc [new Agent/TCP/FullTcp]; # create TCP sending Agent

$tcpsrc set fid_ 1
```

```
$tcpsrc set segsize_ 1460; # default packet size

$tcpsrc set window_ 64; # maximum congestion window size (pckts)

$ns_ attach-agent $n(0) $tcpsrc; # all Agents must be attached to a
node

set tcpsink [new Agent/TCP/FullTcp]; # TCP receiver

$ns_ attach-agent $n(1) $tcpsink

#----------------------------------------------------------------------
----

# SETUP TRACING

#----------------------------------------------------------------------
----

Trace set show_tcphdr_ 1

# tracing TCP variables

set tracevar_chan_ [open "|gzip > full-tracevar.out.gz" w]; # trace
file

$tcpsrc attach $tracevar_chan_; # attach to TCP Agent

$tcpsrc tracevar cwnd_; # trace cwnd

# can trace anything in tcp.h defined with Traced*

# (t_seqno_, ssthresh_, t_rtt_, dupacks_, ...)

# tracing links

# Trace set show_tcphdr_ 1; # displays extra TCP header info

# trace all packet events between src and 1st router

set trq_src0_ [open "|gzip > full-TCP-src0.trq.gz" w]

$ns_ trace-queue $n(0) $r(0) $trq_src0_

$ns_ trace-queue $r(0) $n(0) $trq_src0_; # order matters here

# trace all packet events between dst and 2nd router

set trq_dst1_ [open "|gzip > full-TCP-dst1.trq.gz" w]

$ns_ trace-queue $n(1) $r(1) $trq_dst1_

$ns_ trace-queue $r(1) $n(1) $trq_dst1_
```

```
# trace all packet events between the two routers

set trq_01_ [open "|gzip > full-TCP-q01.trq.gz" w]

$ns_ trace-queue $r(0) $r(1) $trq_01_

$ns_ trace-queue $r(1) $r(0) $trq_01_

#----------------------------------------------------------------------
----

# MAKE THE CONNECTION

#----------------------------------------------------------------------
----

$ns_ connect $tcpsrc $tcpsink

$tcpsink listen

#----------------------------------------------------------------------
----

# SETUP TRAFFIC

#----------------------------------------------------------------------
----

# FTP application

set ftp [new Application/FTP]

$ftp attach-agent $tcpsrc

#----------------------------------------------------------------------
----

# MAKE SCHEDULE

#----------------------------------------------------------------------
----

# schedule this FTP flow

$ns_ at 0.0 "$ftp start"; # send packets forever

$ns_ at $end "$ftp stop"

# The above could be done without an FTP Application with this line:

# $ns_ at 0.0 "$tcpsrc send -1"; # send packets forever

$ns_ at $end "finish"; # end of simulation
```

```
# output progress

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

$ns_ at $i "puts stderr \"time $i\""

}

$ns_ run; # GO!

}

proc finish {} {

# insert post-processing code here

#----------------------------------------------------------------------
----

# EXAMPLES

#----------------------------------------------------------------------
----

# printing output:

# puts "Finished!"

# puts stderr "Finished!"

# puts [format "%f" [expr ($timeouts / $segs)]]

# running shell commands:

# exec zcat TCP-q01.trq.gz | awk {{if ($1=="+" && $5=="tcp" && $6>40)
print $6}}

exit 0;

}

main
```

sample screen



```
tamil@ns2 ~/c
$ ns fulltcpflow.tcl
time 0
time 1

tamil@ns2 ~/c
$
```