PROCEDURE TO SET UP WIRED NETWORK IN NETWORK SIMULATOR-2

Step 1 : Create Simulator Class' Object

Step 2: Store results in File

Step 3: Create Nodes

Step 4: Connect Nodes

Step 5: Create Agent (TCP or UDP)

Step 6: Connect Agents on Both Nodes

Step 7: Setup Application over Agent

Step 8: Attach Application with Agent

Step 9: Create finish procedure Flush Buffer and Start NAM

Step 10: Schedule events

Step 11: Start Simulation

Step 12: Save the Simulation program as <filename>.tcl

Step 13: Run the Simulation using ns command

Ex: $ns filename.tcl

Step 14: Check for trace file

Ex: gedit filename.tr

Step 15: Measure the required performance using suitable filters.

1. **Simulate a point-to-point network with duplex link as follows: n0-n2, n1-n2 and n2- n3. Apply TCP agent between n0-n3 and UDP agent between n1-n3. Apply relevant applications over TCP and UDP agents. Set the queue size to 5 and vary the bandwidth to find the number of packets dropped and received by TCP and UDP agents using awk script and grep command.**

CODE:

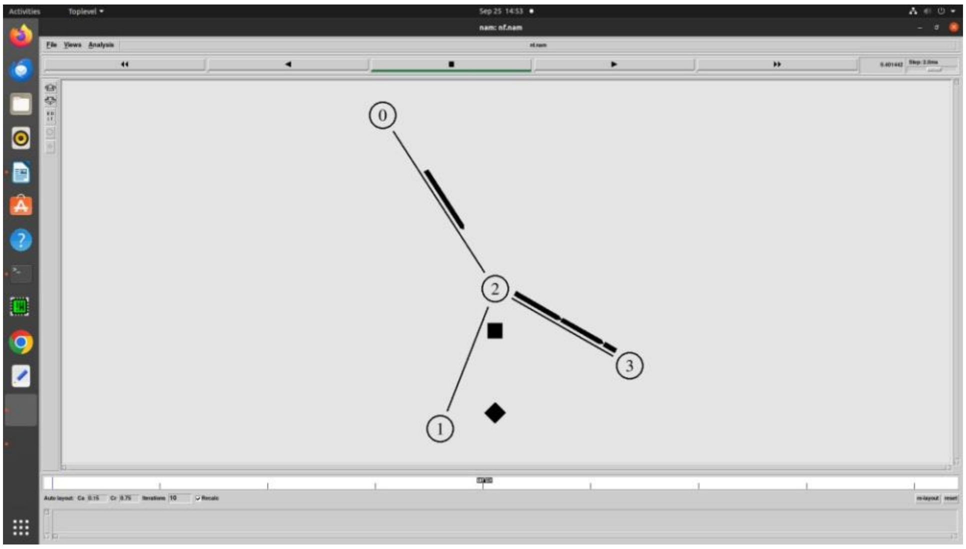
```
set ns [new Simulator]

# Create tracefile
set tf [open tf.tr w]
$ns trace-all $tf

# Create namtrace file
set nf [open nf.nam w]
$ns namtrace-all $nf
# Creating 4 nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
# Creating links between nodes
$ns duplex-link $n0 $n2 2Mb 2ms DropTail
$ns duplex-link $n1 $n2 2Mb 2ms DropTail
$ns duplex-link $n2 $n3 0.4Mb 10ms DropTail
$ns queue-limit $n2 $n3 5

# Create UDP source agent
set udp1 [new Agent/UDP]
$ns attach-agent $n0 $udp1

# Create UDP destination
source set null1 [new
Agent/Null]
$ns attach-agent $n3 $null1

# Connect source agent to destination agent
$ns connect $udp1 $null1
# Creating traffic
set cbr1 [new Application/Traffic/CBR]
```

```
$cbr1 attach-agent $udp1

# Starting and stopping traffic
$ns at 0.1 "$cbr1 start"
$ns at 0.4 "$cbr1 stop"

# Create TCP source
agent set tcp1 [new
Agent/TCP]
$ns attach-agent $n1 $tcp1
# Create TCP destination agen
set tcpsink [new Agent/TCPSink]
$ns attach-agent $n2 $tcpsink
# Attach source agent to destination agent
$ns connect $tcp1 $tcpsink
# Creating traffic
set ftp1 [new Application/FTP]
$ftp1 attach-agent $tcp1

# Starting and Stopping traffic
$ns at 0.5 "$ftp1 start"
$ns at 0.7 "$ftp1 stop"
# Ending the simulation
$ns at 0.9 "finish"

proc finish {} {
global ns tf nf
$ns flush-
trace close
$tf
close $nf
puts "Running nam..."
exec nam nf.nam & exit 0
}
$ns run
```

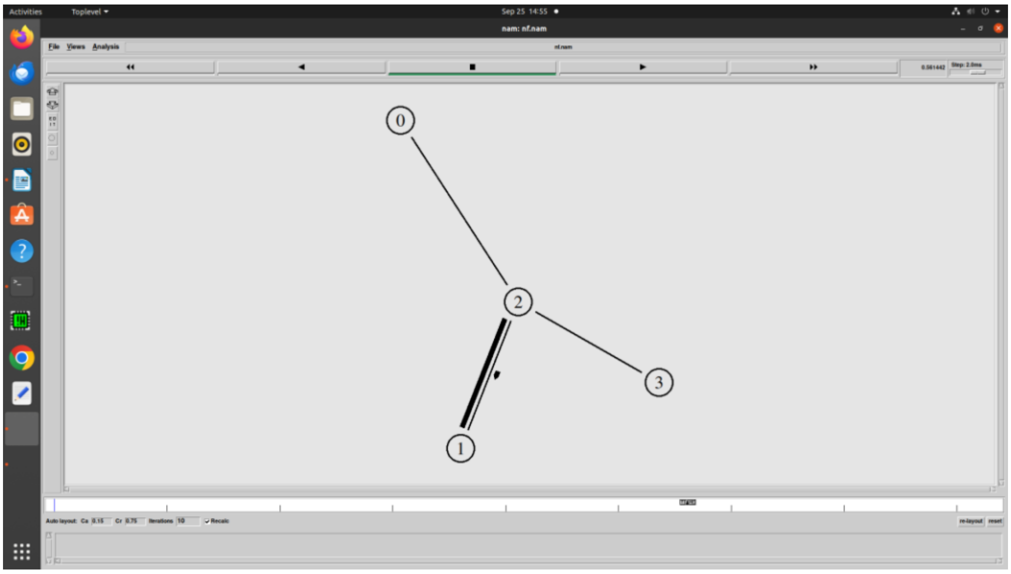
AWK Script:

```
BEGIN {
tcp_count=0;
udo_count=0;
}{
if($1 == "d" && $5 == "tcp")
tcp_count++;
if($1 == "d" && $5 == "cbr")
udp_count++;
} END {
printf("TCP %d\n",tcp_count);
printf("UDP %d\n",udp_count);
}
```

OUTPUT: UDP
(n0 – n3)



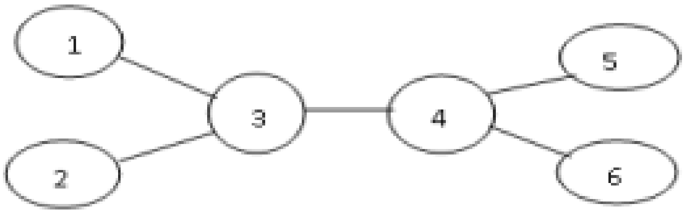
TCP (n1 – n2)



0.561442 Step: 2.0ms

Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ☑ Recalc

2. **Set up the network topology as shown in fig 1. Simulate different type of internet traffic Such as traffic using FTP between the nodes n1 – n6 and Telnet between the nodes n2-n5. Plot congestion window for FTP and Telnet and analyze the throughput.**



- Fig. 1: Network Topology

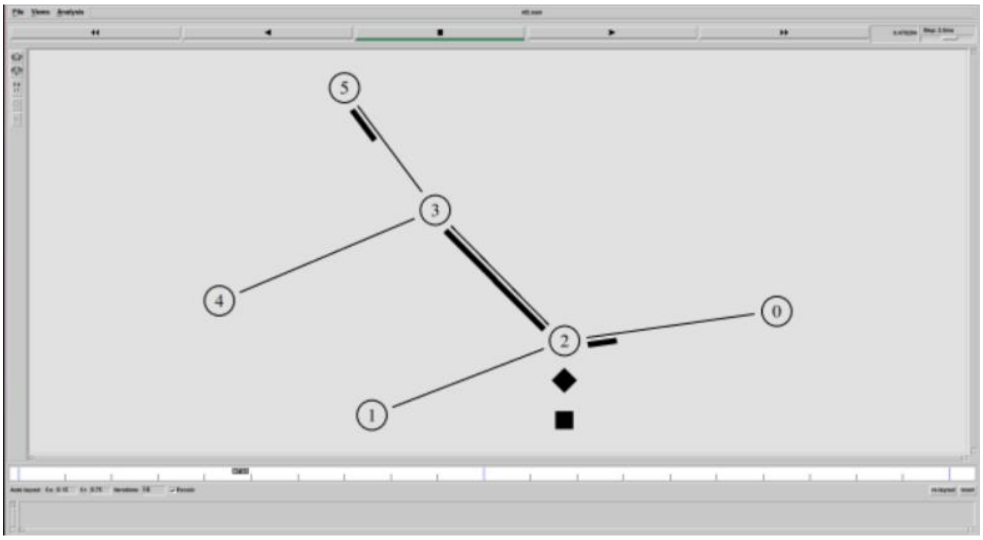
CODE:

```
set ns [new Simulator]
set tf [open tf2.tr w]
set nf [open nf2.nam w]
$ns trace-all $tf
$ns namtrace-all $nf
set cwind [open win2.tr w]
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$ns duplex-link $n0 $n2 2Mb 2ms DropTail
$ns duplex-link $n1 $n2 2Mb 2ms DropTail
$ns duplex-link $n2 $n3 0.4Mb 5ms DropTail
$ns duplex-link $n3 $n4 2Mb 2ms DropTail
$ns duplex-link $n3 $n5 2Mb 2ms DropTail
$ns queue-limit $n2 $n3 10
set tcp1 [new Agent/TCP]
set sink1 [new Agent/TCPSink]
set ftp1 [new Application/FTP]
$ns attach-agent $n0 $tcp1
$ns attach-agent $n5 $sink1
$ns connect $tcp1 $sink1
$ftp1 attach-agent $tcp1
```

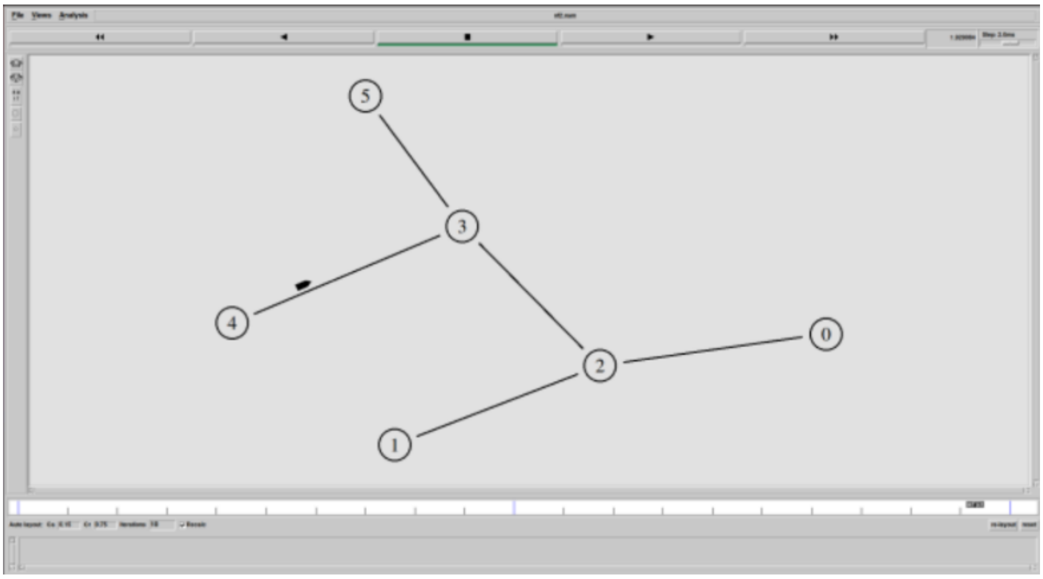
```
$ns at 0.1 "$ftp1 start"

set tcp2 [new Agent/TCP]
set sink2 [new Agent/TCPSink]
set telnet1 [new Application/Telnet]
$ns attach-agent $n1 $tcp2
$ns attach-agent $n4 $sink2
$ns connect $tcp2 $sink2
$telnet1 attach-agent $tcp2
$ns at 1.1 "$telnet1 start"
$ns at 1.0 "$ftp1 stop"
#$ns at 4.0 "$telnet1 stop"
$ns at 2.0 "finish"
proc plotWindow {tcpsource file} {
global ns
set time 0.01
set now [$ns now]
set cwind [$tcpsource set cwnd_]
puts $file "$now $cwind"
$ns at [expr $now + $time] "plotWindow $tcpsource $file"
}
$ns at 0.2 "plotWindow $tcp1 $cwind"
$ns at 0.5 "plotWindow $tcp2 $cwind"
proc finish {} {
global ns tf nf
$ns flush-trace
close $tf
close $nf
puts "Running nam..."
exec nam nf2.nam &
exec xgraph win2.tr &
exit 0
}
$ns run
```

OUTPUT:

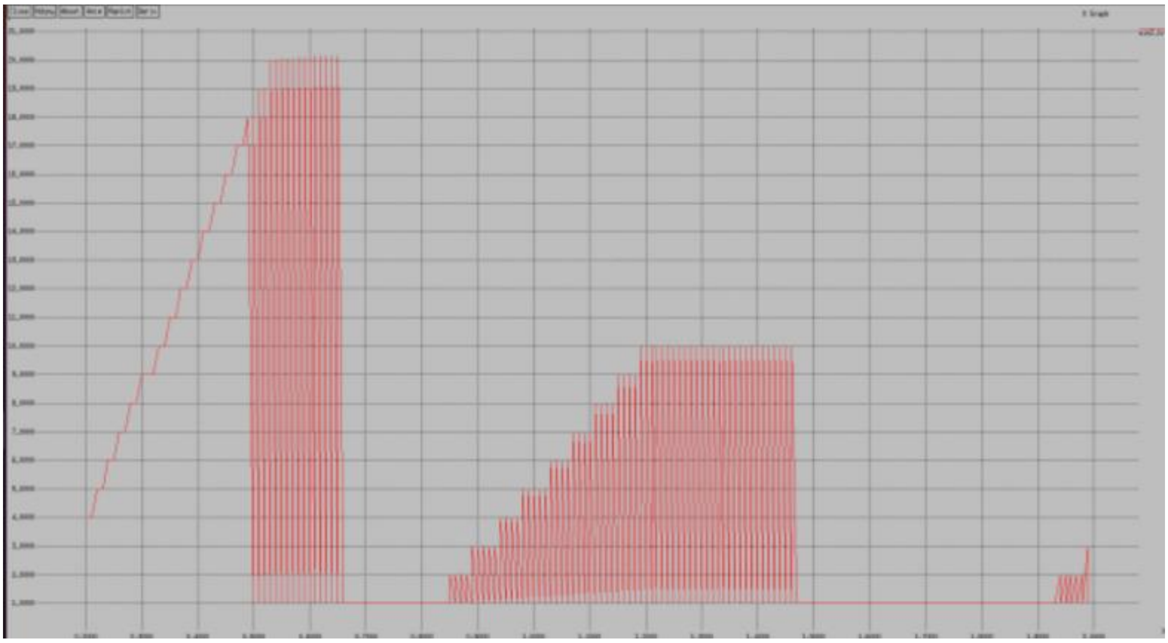
FTP (n0 – n5)



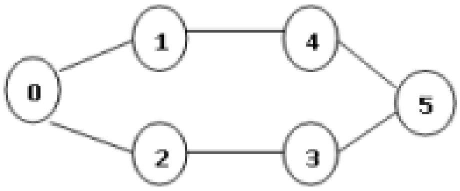
TELNET (n1 – n4)



CONGESTION WINDOW:



Design networks as shown in figure 2 that demonstrate the working of Distance vector routing protocol. The link between node 1 and 4 breaks at 1.0 ms and comes up at 3.0ms. Assume that the source node 0 transmits packets to node 4. Plot the congestion window when TCP sends packets via other nodes. Assume your own parameters for bandwidth and delay.



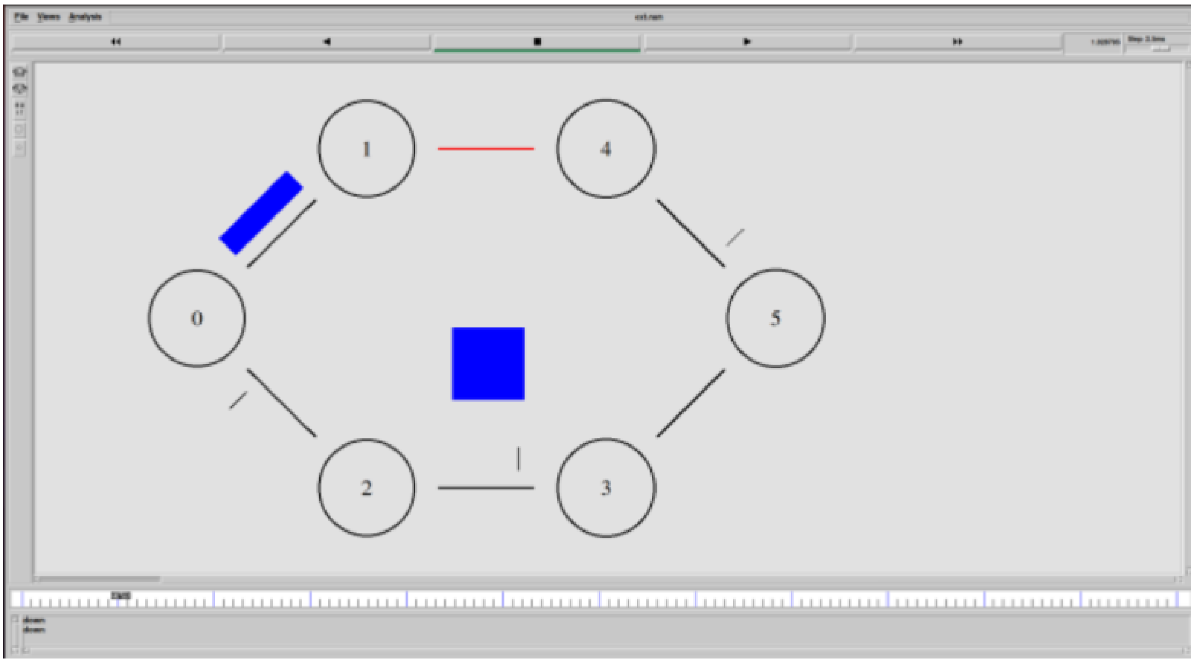
- Fig 2: Network Topology

CODE:

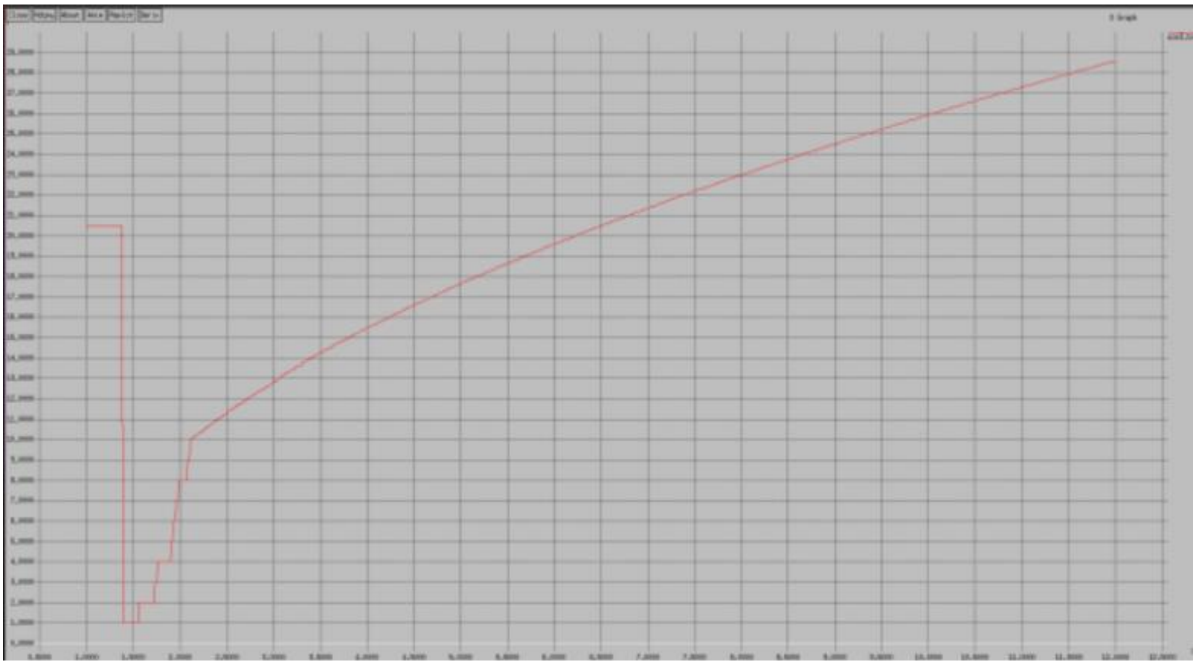
```
set ns [new Simulator]
set tf [open ex3.tr w]
$ns trace-all $tf
set nf [open ex3.nam w]
$ns namtrace-all $nf
set cwind [open win3.tr w]

$ns color 1 Blue
$ns color 2 Red

$ns rtproto DV

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

$ns duplex-link $n0 $n1 0.3Mb 10ms DropTail
$ns duplex-link $n0 $n2 0.3Mb 10ms DropTail
$ns duplex-link $n2 $n3 0.3Mb 10ms DropTail
$ns duplex-link $n1 $n4 0.3Mb 10ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 10ms DropTail
$ns duplex-link $n4 $n5 0.5Mb 10ms DropTail

$ns duplex-link-op $n0 $n1 orient right-up
$ns duplex-link-op $n0 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
$ns duplex-link-op $n1 $n4 orient right
```

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

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
set ftp [new Application/FTP]
$ftp attach-agent $tcp

$ns rtmodel-at 1.0 down $n1 $n4
$ns rtmodel-at 3.0 up $n1 $n4
$ns at 0.1 "$ftp start"
$ns at 12.0 "finish"

proc plotWindow {tcpSource file} {
global ns
set time 0.01
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now + $time] "plotWindow $tcpSource $file"
}

$ns at 1.0 "plotWindow $tcp $cwind"

proc finish {} {
global ns tf nf cwind
$ns flush-trace
close $tf
close $nf
exec nam ex3.nam &
exec xgraph win3.tr &
exit 0
}
$ns run
```

OUTPUT:



CONGESTION WINDOW



4. **Consider a client and a server. The server is running an FTP application over TCP. The client sends a request to download a file of size 10 MB from the server. Write a TCL script to simulate this scenario. Let node n0 be the server and node n1 be the client. TCP packet size is 1500 Bytes.**

CODE:

```
set ns [new Simulator]
set tf [open 4.tr w]
$ns trace-all $tf
set nf [open 4.nam w]
$ns namtrace-all $nf

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

$ns color 1 Blue

$n0 label "Server"
$n1 label "Client"

$ns duplex-link $n0 $n1 10Mb 22ms DropTail
$ns duplex-link-op $n0 $n1 orient right

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
$tcp set packetSize_ 1500
set sink [new Agent/TCPSink]
$ns attach-agent $n1 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$tcp set fid_ 1

proc finish {} {
global ns tf nf
$ns flush-trace
close $tf
close $nf
exec nam 4.nam &
exec awk -f p4transfer.awk 4.tr &
exec awk -f p4convert.awk 4.tr > convert.tr
exec xgraph convert.tr -geometry 800*400 -t
"Bytes_received_at_Client" -x "Time _in_secs" -y "Bytes_in_bps" &
}

$ns at 0.01 "$ftp start"
```

```
$ns at 15.0 "$ftp stop"
$ns at 15.1 "finish"
$ns run
```

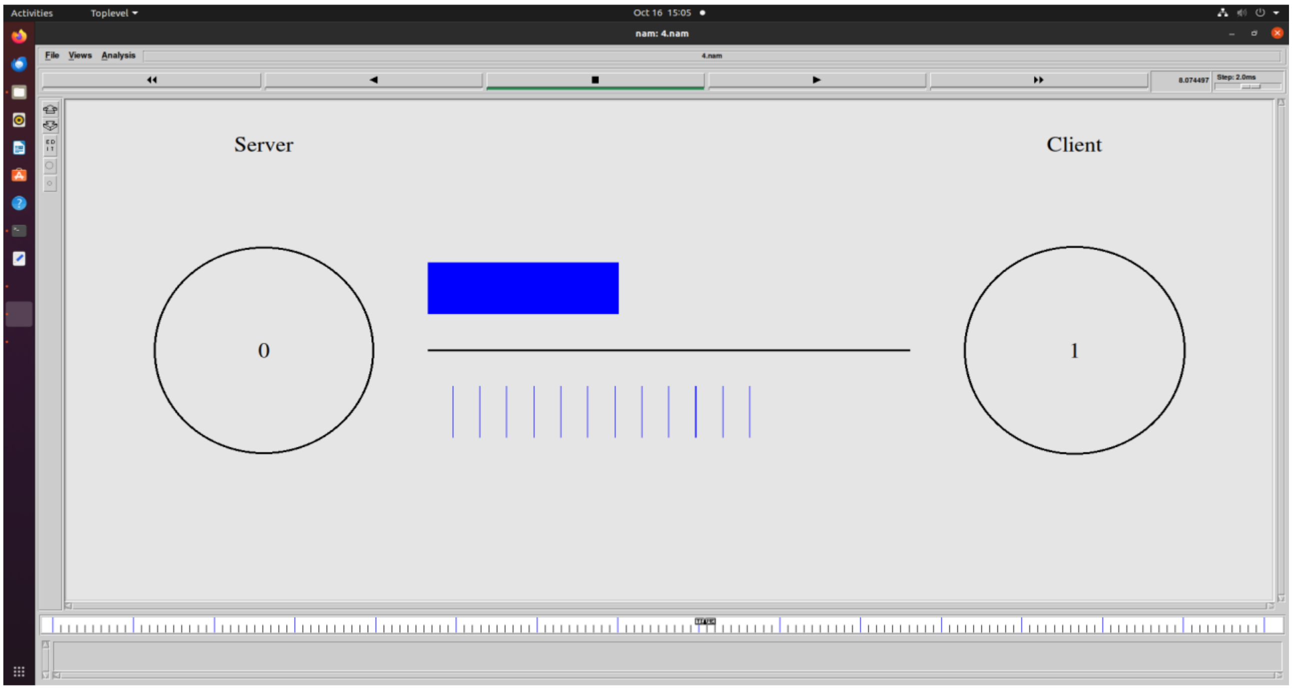
AWK Script to calculate time required:

```
BEGIN{
count=0;
time=0;
total_bytes_received=0;
total_bytes_sent=0;
}
{
if($1=="r" && $4==1 && $5=="tcp")
total_bytes_received+=$6;
if($1=="+" && $3==0 && $5=="tcp")
total_bytes_sent+=$6;
}
END{
system("clear");
printf("\nTransmission time required to transfer the file is %f",$2);
printf("\nActual data sent from the server is %f Mbps",
(total_bytes_sent)/1000000);
printf("\nData received by the client is %f Mbps\n",
(total_bytes_received)/1000000);
}
```

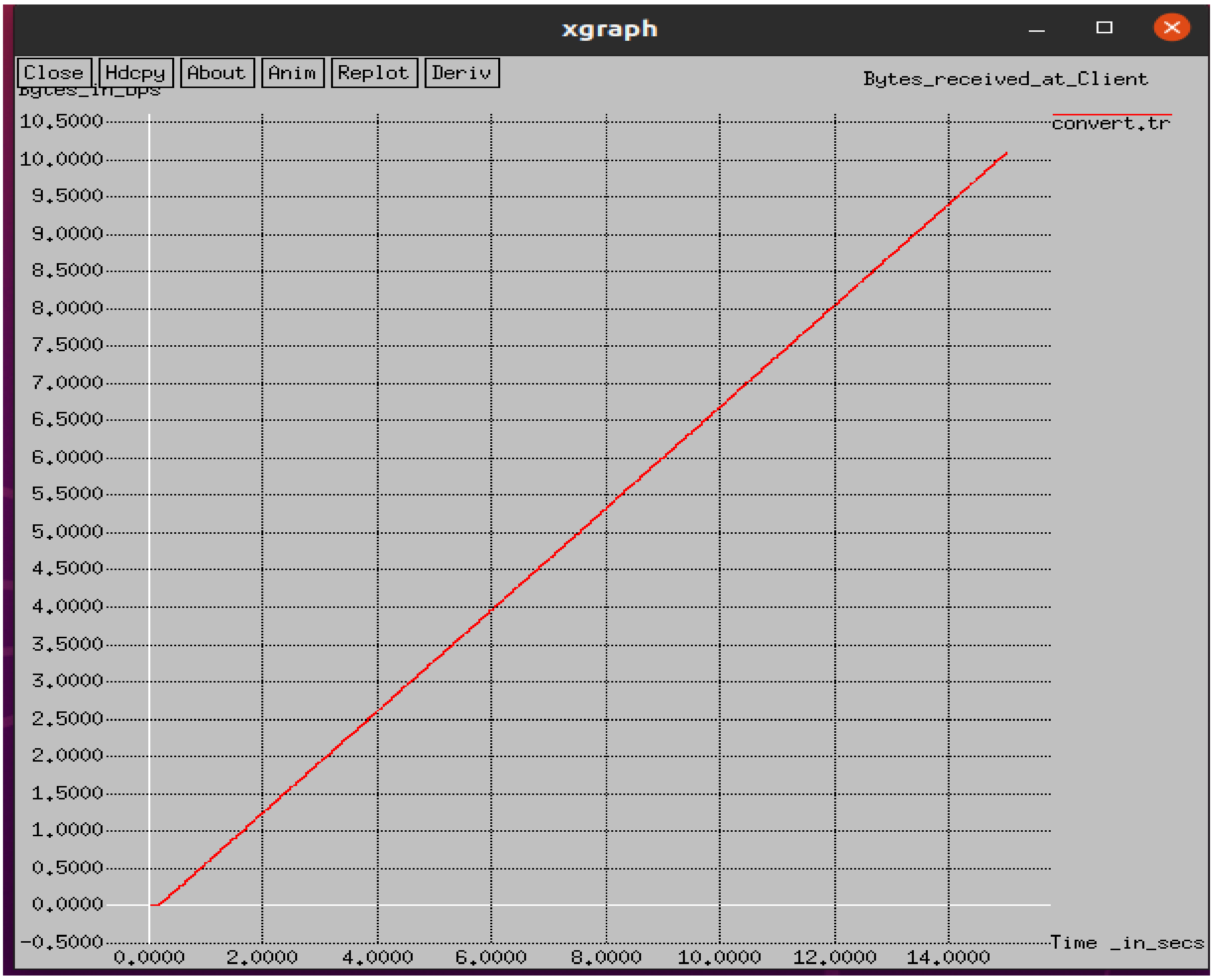
AWK Script to convert file into graph values

```
BEGIN{
count=0;
time=0;
}
{
if($1=="r" && $4==1 && $5=="tcp")
{
count+=$6;
time=$2;
printf("\n%f\t%f",time,(count)/1000000);
}
}
END{
}
```

OUTPUT:



GRAPH:



5. Demonstrate the working of multicast routing protocol. Assume your own parameters for bandwidth and delay.

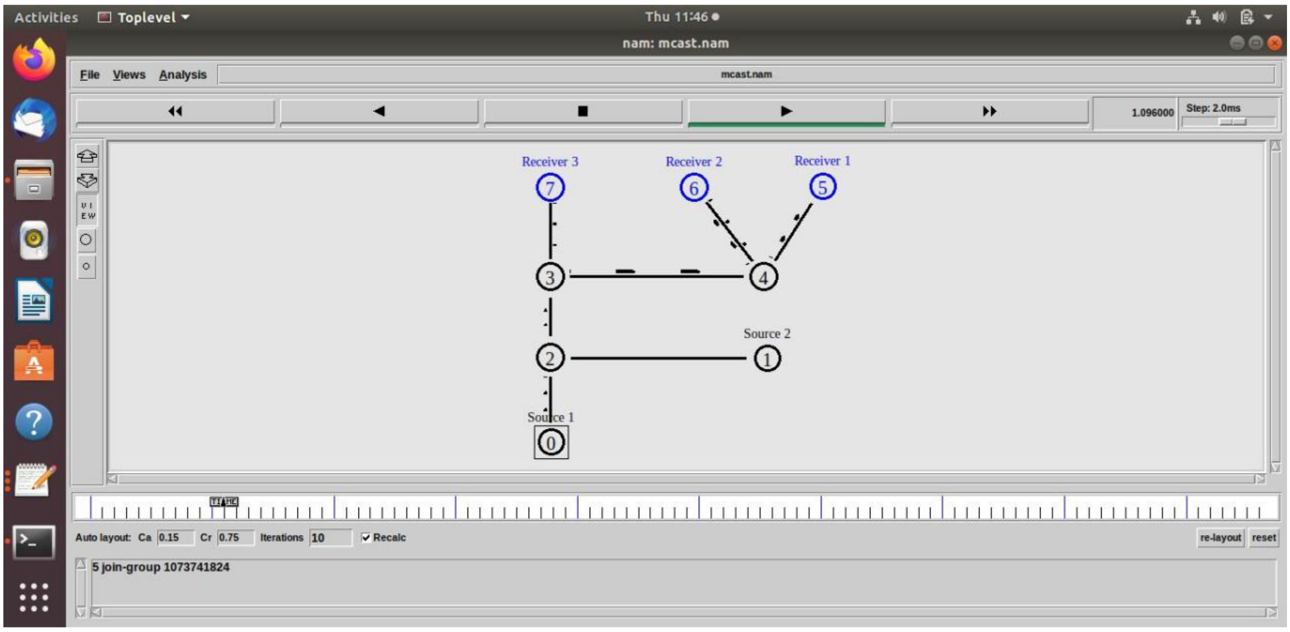
CODE:

```
#Create an event scheduler wit multicast turned on
set ns [new Simulator -multicast on]

$ns multicast

#Turn on Tracing
set tf [open mcast.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"
$ns at 10.0 "finish"

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

# For nam
# Group 0 source
#$udp0 set fid_ 1
#$n0 color red
$n0 label "Source 1"

# Group 1 source
#$udp1 set fid_ 2
#$n1 color green
$n1 label "Source 2"

#Colors for packets from two mcast groups
$ns color 1 red
$ns color 2 green

$n5 label "Receiver 1"
$n5 color blue
$n6 label "Receiver 2"
$n6 color blue
$n7 label "Receiver 3"
$n7 color blue
$ns run
```

OUTPUT:



Auto layout: Ca 0.15 Cr 0.75 Iterations 10 ☑ Recalc

```
5 join-group 1073741824
```