CN Lab CIA II

CONTENTS

Wireless Network

```
#Example of Wireless networks
#Step 1 initialize variables
#Step 2 - Create a Simulator object
#step 3 - Create Tracing and animation file
#step 4 - topography
#step 5 - GOD - General Operations Director
#step 6 - Create nodes
#Step 7 - Create Channel (Communication PATH)
#step 8 - Position of the nodes (Wireless nodes needs a location)
#step 9 - Any mobility codes (if the nodes are moving)
#step 10 - TCP, UDP Traffic
#run the simulation

#initialize the variables
set val(chan)         Channel/WirelessChannel    ;#Channel Type
set val(prop)         Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)        Phy/WirelessPhy            ;# network interface type WAVELAN DSSS 2.4GHz
set val(mac)          Mac/802_11                 ;# MAC type
set val(ifq)          Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)           LL                         ;# link layer type
set val(ant)          Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)       50                         ;# max packet in ifq
set val(nn)           6                          ;# number of mobilenodes
set val(rp)           AODV                       ;# routing protocol
set val(x)  500   ;# in metres
set val(y)  500   ;# in metres
#Adhoc OnDemand Distance Vector

#creation of Simulator
set ns [new Simulator]

#creation of Trace and namfile
set tracefile [open wireless.tr w]
$ns trace-all $tracefile

#Creation of Network Animation file
set namfile [open wireless.nam w]
$ns namtrace-all-wireless $namfile $val(x) $val(y)

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

#GOD Creation - General Operations Director
create-god $val(nn)

set channel1 [new $val(chan)]
set channel2 [new $val(chan)]
set channel3 [new $val(chan)]

#configure the node
$ns node-config -adhocRouting $val(rp) \
  -llType $val(ll) \
  -macType $val(mac) \
  -ifqType $val(ifq) \
  -ifqLen $val(ifqlen) \
  -antType $val(ant) \
  -propType $val(prop) \
  -phyType $val(netif) \
  -topoInstance $topo \
  -agentTrace ON \
  -macTrace ON \
  -routerTrace ON \
  -movementTrace ON \
  -channel $channel1

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

$n0 random-motion 0
$n1 random-motion 0
$n2 random-motion 0
$n3 random-motion 0
$n4 random-motion 0
$n5 random-motion 0

$ns initial_node_pos $n0 20
$ns initial_node_pos $n1 20
$ns initial_node_pos $n2 20
$ns initial_node_pos $n3 20
$ns initial_node_pos $n4 20
$ns initial_node_pos $n5 50

#initial coordinates of the nodes
$n0 set X_ 10.0
$n0 set Y_ 20.0
$n0 set Z_ 0.0

$n1 set X_ 210.0
$n1 set Y_ 230.0
$n1 set Z_ 0.0

$n2 set X_ 100.0
$n2 set Y_ 200.0
$n2 set Z_ 0.0

$n3 set X_ 150.0
$n3 set Y_ 230.0
$n3 set Z_ 0.0

$n4 set X_ 430.0
$n4 set Y_ 320.0
$n4 set Z_ 0.0

$n5 set X_ 270.0
$n5 set Y_ 120.0
$n5 set Z_ 0.0

#Dont mention any values above than 500 because in this example, we use X and Y as 500,500

#mobility of the nodes
#At what Time? Which node? Where to? at What Speed?
$ns at 1.0 "$n1 setdest 490.0 340.0 25.0"
$ns at 1.0 "$n4 setdest 300.0 130.0 5.0"
$ns at 1.0 "$n5 setdest 190.0 440.0 15.0"
#the nodes can move any number of times at any location during the simulation (runtime)
$ns at 20.0 "$n5 setdest 100.0 200.0 30.0"

#creation of agents
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 1.0 "$ftp start"

set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n2 $udp
$ns attach-agent $n3 $null
$ns connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$ns at 1.0 "$cbr start"

$ns at 30.0 "finish"

proc finish {} {
 global ns tracefile namfile
 $ns flush-trace
 close $tracefile
 close $namfile
 exit 0
}

puts "Starting Simulation"
$ns run
```

AWK

```
BEGIN {
    seqno = -1;
    droppedPackets = 0;
    receivedPackets = 0;
    count = 0;
}

{
    # Packet delivery ratio
    if ($4 == "AGT" && $1 == "s" && seqno < $6) {
        seqno = $6;
    } else if ($4 == "AGT" && $1 == "r") {
        receivedPackets++;
    } else if ($1 == "D" && $7 == "tcp" && $8 > 512) {
        droppedPackets++;
    }

    # End-to-end delay
    if ($4 == "AGT" && $1 == "s") {
        start_time[$6] = $2;
    } else if ($7 == "tcp" && $1 == "r") {
        end_time[$6] = $2;
    } else if ($1 == "D" && $7 == "tcp") {
        end_time[$6] = -1;
    }
}

END {
    for (i = 0; i <= seqno; i++) {
        if (end_time[i] > 0) {
            delay[i] = end_time[i] - start_time[i];
            count++;
        } else {
            delay[i] = -1;
        }
    }

    for (i = 0; i < count; i++) {
        if (delay[i] > 0) {
            n_to_n_delay = n_to_n_delay + delay[i];
        }
    }

    n_to_n_delay = n_to_n_delay / count;
    print "\n";
    print "GeneratedPackets = " seqno + 1;
    print "ReceivedPackets = " receivedPackets;
    printf("Packet Delivery Ratio = %.2f%%\n", (receivedPackets / (seqno + 1)) * 100);
    print "Total Dropped Packets = " droppedPackets;
    printf("Average End-to-End Delay = %.2f ms\n", n_to_n_delay * 1000);
    print "\n";
}
```

TCP Flow

```
#creating a simulator object
set ns [ new Simulator ]
#creating trace file
set tf [open trace1.tr w]
$ns trace-all $tf
#creating nam file
set nf [open opnam.nam w]
$ns namtrace-all $nf

#creating variables for throughput files
set ft1 [open "Sender1_throughput" "w"]
set ft2 [open "Sender2_throughput" "w"]
set ft3 [open "Sender3_throughput" "w"]
set ft4 [open "Total_throughput" "w"]

#creating variables for bandwidth files
set fb1 [open "Bandwidth1" "w"]
set fb2 [open "Bandwidth2" "w"]
set fb3 [open "Bandwidth3" "w"]
set fb4 [open "TotalBandwidth" "w"]

#finish procedure to call nam and xgraph
proc finish {} {
 global ns nf ft1 ft2 ft3 ft4 fb1 fb2 fb3 fb4
 $ns flush-trace
 #closing all files
 close $nf
 close $ft1
 close $ft2
 close $ft3
 close $ft4
 close $fb1
 close $fb2
 close $fb3
 close $fb4
 #executing graphs
 exec xgraph Sender1_throughput Sender2_throughput Sender3_throughput Total_throughput &
 exec xgraph Bandwidth1 Bandwidth2 Bandwidth3 TotalBandwidth &
 puts "running nam..."
 exec nam opnam.nam &
 #exec awk -f analysis.awk trace1.tr
 exit 0
}

#record procedure to calculate total bandwidth and throughput
proc record {} {
 global null1 null2 null3 ft1 ft2 ft3 ft4 fb1 fb2 fb3 fb4
 global ftp1 smtp1 http1

 set ns [Simulator instance]
 set time 0.1
 set now [$ns now]

 set bw0 [$null1 set bytes_]
 set bw1 [$null2 set bytes_]
 set bw2 [$null3 set bytes_]

 set totbw [expr $bw0 + $bw1 + $bw2]
 puts $ft4 "$now [expr $totbw/$time*8/1000000]"

 puts $ft1 "$now [expr $bw0/$time*8/1000000]"
 puts $ft2 "$now [expr $bw1/$time*8/1000000]"
 puts $ft3 "$now [expr $bw2/$time*8/1000000]"

 puts $fb1 "$now [expr $bw0]"
 puts $fb2 "$now [expr $bw1]"
 puts $fb3 "$now [expr $bw2]"
 puts $fb4 "$now [expr $totbw]"

 $null1 set bytes_ 0
 $null2 set bytes_ 0
 $null3 set bytes_ 0

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

#creating 10 nodes
for {set i 0} {$i < 10} {incr i} {
 set n($i) [$ns node]
}

#creating duplex links
$ns duplex-link $n(0) $n(1) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(3) 1.5Mb 10ms RED
$ns duplex-link $n(1) $n(2) 1Mb 10ms DropTail
$ns duplex-link $n(2) $n(7) 2Mb 10ms RED
$ns duplex-link $n(7) $n(8) 2Mb 10ms DropTail
$ns duplex-link $n(8) $n(9) 2Mb 10ms RED
$ns duplex-link $n(3) $n(5) 1Mb 10ms DropTail
$ns duplex-link $n(5) $n(6) 1Mb 10ms RED
$ns duplex-link $n(6) $n(4) 1Mb 10ms DropTail
$ns duplex-link $n(4) $n(7) 1Mb 10ms RED

#orienting links
$ns duplex-link-op $n(0) $n(1) orient right-up
$ns duplex-link-op $n(1) $n(2) orient right
$ns duplex-link-op $n(0) $n(3) orient right-down
$ns duplex-link-op $n(2) $n(7) orient right-down
$ns duplex-link-op $n(7) $n(8) orient right-up
$ns duplex-link-op $n(5) $n(6) orient right
$ns duplex-link-op $n(6) $n(4) orient left-up
$ns duplex-link-op $n(3) $n(5) orient right-down
$ns duplex-link-op $n(4) $n(7) orient right-up
```

```
$ns duplex-link-op $n(8) $n(9) orient right-down

proc ftp_traffic {node0 node9 } {
 global ns null1 tcp1 ftp1
 set tcp1 [new Agent/TCP]
 set null1 [new Agent/TCPSink]
 $ns attach-agent $node0 $tcp1
 $ns attach-agent $node9 $null1
 $ns connect $tcp1 $null1
 set ftp1 [new Application/FTP]
 $ftp1 attach-agent $tcp1
 $ns at 1.0 "$ftp1 start"
 $ns at 3.2 "$ftp1 stop"
 }
ftp_traffic $n(0) $n(8)

proc smtp_traffic {node0 node3 } {
 global ns null2 tcp2 smtp1
 set tcp2 [new Agent/TCP]
 set null2 [new Agent/TCPSink]
 $ns attach-agent $node0 $tcp2
 $ns attach-agent $node3 $null2
 $ns connect $tcp2 $null2
 set smtp1 [new
Application/Traffic/Exponential]
 $smtp1 attach-agent $tcp2
 $ns at 2.0 "$smtp1 start"
 $ns at 3.8 "$smtp1 stop"
 }
smtp_traffic $n(3) $n(6)

proc http_traffic {node1 node7 } {
 global ns null3 tcp3 http1
 set tcp3 [new Agent/TCP]
 set null3 [new Agent/TCPSink]
 $ns attach-agent $node1 $tcp3
 $ns attach-agent $node7 $null3
 $ns connect $tcp3 $null3
 set http1 [new
Application/Traffic/Exponential]
 $http1 attach-agent $tcp3
 $ns at 0.2 "$http1 start"
 $ns at 3.2 "$http1 stop"  }
http_traffic $n(0) $n(7)

#scheduling events
$ns at 0.5 "record"
$ns at 0.2 "$ns trace-annotate \"Starting HTTP
from 0 to 7\""
$ns at 1.0 "$ns trace-annotate \"Starting FTP
from 0 to 8\""
$ns at 2.0 "$ns trace-annotate \"Starting
SMTP from 3 to 6\""
$ns at 5.0 "finish"
$ns run
AWK
BEGIN {
    st1 = 0
    ft1 = 0
    throughput1 = 0
    delay1 = 0
    data1 = 0

    st2 = 0
    ft2 = 0
    throughput2 = 0
    delay2 = 0
    data2 = 0

    st3 = 0
    ft3 = 0
    throughput3 = 0
    delay3 = 0
    data3 = 0

    total_delay = 0
    total_th = 0
}

{
    if ($1 == "r" && $4 == 7) {  # HTTP
        data1 += $6
        if (flag1 == 0) {
            st1 = $2
            flag1 = 1
        }
        ft1 = $2
    }

    if ($1 == "r" && $4 == 8) {  # FTP
        data2 += $6
        if (flag2 == 0) {
            st2 = $2
            flag2 = 1
        }
        ft2 = $2
    }

    if ($1 == "r" && $4 == 6) {  # SMTP
        data3 += $6
        if (flag3 == 0) {
            st3 = $2
            flag3 = 1
        }
        ft3 = $2
    }
}

END {
    printf("**********HTTP***********\n")
    printf("start time %f\n", st1)
    printf("end time %f\n", ft1)
    printf("data %f\n", data1)
    delay1 = ft1 - st1
    throughput1 = data1 / delay1
    printf("throughput %f\n", throughput1)
    printf("delay %f\n", delay1)

    printf("**********SMTP***********\n")
    printf("start time %f\n", st3)
    printf("end time %f\n", ft3)
    printf("data %f\n", data3)
    delay3 = ft3 - st3
    throughput3 = data3 / delay3
    printf("throughput %f\n", throughput3)
    printf("delay %f\n", delay3)

    printf("**********FTP***********\n")
    printf("start time %f\n", st2)
    printf("end time %f\n", ft2)
    printf("data %f\n", data2)
    delay2 = ft2 - st2
    throughput2 = data2 / delay2
    printf("throughput %f\n", throughput2)
    printf("delay %f\n", delay2)

    total_th = throughput1 + throughput2 +
throughput3
    total_delay = delay1 + delay2 + delay3
    printf("Avg throughput %f\n", total_th / 3)
    printf("Avg delay %f\n", total_delay / 3)
}
```

TCP Congestion Control

```
#creating a simulator object
set ns [ new Simulator ]
$ns color 3 Green
#creating trace file
set tf [open reno.tr w]
$ns trace-all $tf

#creating nam file
set nf [open reno.nam w]
$ns namtrace-all $nf

set ft3 [open reno_Sender_throughput w]

#finish procedure to call nam and xgraph
proc finish {} {
 global ns nf ft3
 $ns flush-trace
 #closing all files
 close $nf
 close $ft3
#executing graphs
 exec xgraph reno_Sender_throughput &
 puts "running nam..."
 exec nam reno.nam &
#exec awk -f analysis.awk trace1.tr &
 exit 0
}
#record procedure to calculate total
bandwidth and throughput
proc record {} {
 global null3 ft3
 global http1
 set ns [Simulator instance]
 set time 0.1
 set now [$ns now]
 set bw2 [$null3 set bytes_]
 puts $ft3 "$now [expr $bw2/$time*8/1000000]"
 $null3 set bytes_ 0
 $ns at [expr $now+$time] "record"
 }
#creating 10 nodes
for {set i 0} {$i < 6} {incr i} {
 set n($i) [$ns node]
}
#creating duplex links
$ns duplex-link $n(0) $n(1) 10Kb 10ms DropTail
$ns duplex-link $n(0) $n(3) 100Kb 10ms RED
$ns duplex-link $n(1) $n(2) 50Kb 10ms DropTail
$ns duplex-link $n(2) $n(5) 200Kb 10ms RED
$ns duplex-link $n(3) $n(4) 70Kb 10ms DropTail
$ns duplex-link $n(4) $n(5) 100Kb 10ms DropTail

#orienting links
$ns duplex-link-op $n(0) $n(1) orient right
$ns duplex-link-op $n(1) $n(2) orient right-down
$ns duplex-link-op $n(0) $n(3) orient left-down
$ns duplex-link-op $n(3) $n(4) orient right-down
$ns duplex-link-op $n(4) $n(5) orient right
$ns duplex-link-op $n(2) $n(5) orient left-down


set tcp3 [new Agent/TCP/Reno]  //Newreno, TCP ie Tahoe, Sack1, Vegas
set null3 [new Agent/TCPSink]
$ns attach-agent $n(0) $tcp3
$ns attach-agent $n(5) $null3
$ns connect $tcp3 $null3
set http1 [new Application/Traffic/Exponential]
$http1 attach-agent $tcp3
```

```
#scheduling events
$ns at 0.5 "record"
$ns at 0.2 "$ns trace-annotate \"Starting HTTP from 0 to 5\""
$ns at 0.2 "$n(0) color \"green\""
$ns at 0.2 "$n(5) color \"green\""
$ns at 0.2 "$http1 start"
$ns at 3.2 "$http1 stop"
$ns at 5.0 "finish"
$ns run
AWK
BEGIN {
    st1 = 0
    ft1 = 0
    throughput1 = 0
    delay1 = 0
    flag1 = 0
    data1 = 0
}

{
    if ($1 == "r" && $4 == 5) { # Check if it's an HTTP packet (assuming HTTP packets have "r" in $1 and 5 in $4)
        data1 += $6
        if (flag1 == 0) {
            st1 = $2
            flag1 = 1
        }
        ft1 = $2
    }
}

END {
    printf("**********HTTP***********\n")
    printf("Start time: %f\n", st1)
    printf("End time: %f\n", ft1)
    printf("Data: %f\n", data1)
    delay1 = ft1 - st1
    if (delay1 > 0) { # Check if delay1 is greater than 0 to avoid division by zero
        throughput1 = data1 / delay1
        printf("Throughput: %f\n", throughput1)
    } else {
        printf("Throughput: N/A (Zero delay)\n")
    }
    printf("Delay: %f\n", delay1)
}
```

Distance Vector & Link State

```
set ns [new Simulator]
$ns rtproto DV //LS
$ns color 1 green
set node0 [$ns node]
set node1 [$ns node]
set node2 [$ns node]
set node3 [$ns node]
set node4 [$ns node]

set node5 [$ns node]
set node6 [$ns node]
set tf [open out_dv.tr w]
$ns trace-all $tf
set nf [open out_dv.nam w]
$ns namtrace-all $nf

set ft [open "dvr_th" "w"]
$node0 label "node 0"
$node1 label "node 1"
$node2 label "node 2"
$node3 label "node 3"
$node4 label "node 4"

$node5 label "node 5"
$node6 label "node 6"
$ns duplex-link $node0 $node1 1.5Mb 10ms DropTail
$ns duplex-link $node1 $node2 1.5Mb 10ms DropTail
$ns duplex-link $node2 $node3 1.5Mb 10ms DropTail
$ns duplex-link $node3 $node4 1.5Mb 10ms DropTail

$ns duplex-link $node4 $node5 1.5Mb 10ms DropTail
$ns duplex-link $node5 $node6 1.5Mb 10ms DropTail
$ns duplex-link $node6 $node0 1.5Mb 10ms DropTail
$ns duplex-link-op $node0 $node1 orient left-down
$ns duplex-link-op $node1 $node2 orient left-down
$ns duplex-link-op $node2 $node3 orient right-down
$ns duplex-link-op $node3 $node4 orient right
$ns duplex-link-op $node4 $node5 orient right-up
$ns duplex-link-op $node5 $node6 orient left-up
$ns duplex-link-op $node6 $node0 orient left-up

set tcp2 [new Agent/TCP]
$tcp2 set class_ 1
$ns attach-agent $node0 $tcp2
set sink2 [new Agent/TCPSink]
$ns attach-agent $node3 $sink2
$ns connect $tcp2 $sink2
set traffic_ftp2 [new Application/FTP]

$traffic_ftp2 attach-agent $tcp2
proc record {} {
global sink2 tf ft
global ftp
set ns [Simulator instance]
set time 0.1
set now [$ns now]
set bw0 [$sink2 set bytes_]
puts $ft "$now [expr $bw0/$time*8/1000000]"
$sink2 set bytes_ 0
$ns at [expr $now+$time] "record"
}

proc finish {} {
global ns nf
$ns flush-trace
close $nf
exec nam out_dv.nam &
exec xgraph dvr_th &
exit 0
}
```

```
$ns at 0.55 "record"
#Schedule events for the CBR agents
$ns at 0.5 "$node0 color \"Green\""
$ns at 0.5 "$node3 color \"Green\""
$ns at 0.5 "$ns trace-annotate \"Starting FTP node0 to node6\""
$ns at 0.5 "$node0 label-color green"
$ns at 0.5 "$node3 label-color green"
$ns at 0.5 "$traffic_ftp2 start"
$ns at 0.5 "$node1 label-color green"

$ns at 0.5 "$node2 label-color green"
$ns at 0.5 "$node4 label-color blue"
$ns at 0.5 "$node5 label-color blue"
$ns at 0.5 "$node6 label-color blue"
$ns rtmodel-at 2.0 down $node2 $node3

$ns at 2.0 "$node4 label-color green"
$ns at 2.0 "$node5 label-color green"
$ns at 2.0 "$node6 label-color green"
$ns at 2.0 "$node1 label-color blue"
$ns at 2.0 "$node2 label-color blue"

$ns rtmodel-at 3.0 up $node2 $node3
$ns at 3.0 "$traffic_ftp2 start"
$ns at 4.9 "$traffic_ftp2 stop"
$ns at 5.0 "finish"
$ns run
AWK
BEGIN {
    recvdSize = 0
    startTime = 0.5
    stopTime = 5.0
}

{
    event = $1
    time = $2
    node_id = $3
    pkt_size = $6
    level = $4

    if (event == "s") {
        if (time < startTime) {
            startTime = time
        }
    }

    if (event == "r") {
        if (time > stopTime) {
            stopTime = time
        }
        recvdSize += pkt_size
    }
}

END {
    printf("Average Throughput[kbps] = %.2f\nStartTime=%.2f\nStopTime=%.2f\n",(recvdSize/(stopTime-startTime))*(8/1000),startTime,stopTime)
}
```

Multicast

```
set ns [new Simulator -multicast on]
# Turn on Tracing
set tf [open output.tr w]
$ns trace-all $tf

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

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

# Create links with DropTail Queues
$ns duplex-link $n0 $n2 1.5Mb 10ms DropTail
$ns duplex-link $n1 $n2 1.5Mb 10ms DropTail
$ns duplex-link $n2 $n3 1.5Mb 10ms DropTail
$ns duplex-link $n3 $n4 1.5Mb 10ms DropTail
$ns duplex-link $n3 $n7 1.5Mb 10ms DropTail
$ns duplex-link $n4 $n5 1.5Mb 10ms DropTail
$ns duplex-link $n4 $n6 1.5Mb 10ms DropTail

# DM: dense-mode; SM: sparse-mode
set mproto DM
set mrthandle [$ns mrtproto $mproto {}]

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

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

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

# Create receiver to accept the packets
set rcvr1 [new Agent/Null]
$ns attach-agent $n5 $rcvr1
$ns at 1.0 "$n5 join-group $rcvr1 $group1"
set rcvr2 [new Agent/Null]
$ns attach-agent $n6 $rcvr2
$ns at 1.5 "$n6 join-group $rcvr2 $group1"
set rcvr3 [new Agent/Null]
$ns attach-agent $n7 $rcvr3
$ns at 2.0 "$n7 join-group $rcvr3 $group1"
set rcvr4 [new Agent/Null]
$ns attach-agent $n5 $rcvr4
$ns at 2.5 "$n5 join-group $rcvr4 $group2"
set rcvr5 [new Agent/Null]
$ns attach-agent $n6 $rcvr5
$ns at 3.0 "$n6 join-group $rcvr5 $group2"
set rcvr6 [new Agent/Null]
$ns attach-agent $n7 $rcvr6

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

# Schedule events
$ns at 0.5 "$cbr1 start"
$ns at 9.5 "$cbr1 stop"
$ns at 0.5 "$cbr2 start"
$ns at 9.5 "$cbr2 stop"

# Post-processing
$ns at 10.0 "finish"

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

$ns set-animation-rate 3.0ms
$ns run
AWK
# Initialize variables
BEGIN {
  rec = 0
  drp = 0
  sum = 0
  sum1 = 0
}

# Process each line of the trace file
{
  # Check if the line contains a "r" (received) event with packet size 4
  if ($1 == "r" && $4 == 4) {
    rec++
    sum += $6
  }

  # Check if the line contains a "d" (dropped) event with packet size 4
  if ($1 == "d" && $4 == 4) {
    drp++
  }

  # Check if the line contains a packet sent with size 5 and destination address $group1
  if ($2 > 1.00 && $4 == 5) {
    sum1 += $6
  }
}

# Calculate packet delivery ratio
END {
  tot = rec + drp
  if (tot == 0) {
    rat = 0
  } else {
    rat = (rec / tot) * 100
  }

  throughput = (sum * 8) / 1000000
  throughput1 = (sum1 * 8) / 1000000

  printf("\nPackets received: %d\n", rec)
  printf("Packets dropped: %d\n", drp)
  printf("Packets delivery ratio: %.2f%%\n", rat)
  printf("Throughput for UDP: %.2f Mbps\n", throughput)
  printf("Throughput for TCP: %.2f Mbps\n", throughput1)
}
```

Wired Ethernet

```
set ns [new Simulator]
set tr [open "LAN.tr" w]
$ns trace-all $tr

set nam [open "LAN.nam" w]
$ns namtrace-all $nam

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

$ns make-lan "$n1 $n2 $n3 $n4 $n5 $n6" 0.2Mb 20ms LL Queue/DropTail Mac/802_3

set tcpsendagent1 [new Agent/TCP]
set tcpsendagent2 [new Agent/TCP]

set tcprecvagent1 [new Agent/TCPSink]
set tcprecvagent2 [new Agent/TCPSink]

$ns attach-agent $n1 $tcpsendagent1
$ns attach-agent $n2 $tcpsendagent2

$ns attach-agent $n6 $tcprecvagent1
$ns attach-agent $n6 $tcprecvagent2

set app1 [new Application/FTP]
set app2 [new Application/FTP]

$app1 attach-agent $tcpsendagent1
$app2 attach-agent $tcpsendagent2

#As soon as you create agents make sure i connect them

$ns connect $tcpsendagent1 $tcprecvagent1
$ns connect $tcpsendagent2 $tcprecvagent2

$ns at 0.1 "$app1 start"
$ns at 0.4 "$app2 start"



proc finish { } {
global ns tr nam
$ns flush-trace
close $tr
close $nam
#exec nam namfile_tcp_ls.nam &
exec gawk -f anal.awk LAN.tr &
exit 0
}

$ns at 10 "finish"

$ns run
```

AWK

```
BEGIN{
drop=0
recv=0
starttime1=0
endtime1=0
latency1=0
filesize1=0
starttime2=0
endtime2=0
latency2=0
filesize2=0
flag0=0
flag1=0
bandwidth1=0
bandwidth2=0
}

{

if($1=="r" && $3==6)
{
if(flag1=0)
{
flag1=1
starttime1=$2
}
filesize1+=$6
endtime1=$2
latency=endtime1-starttime1
bandwidth1=filesize1/latency
printf "%f %f\n", endtime1, bandwidth1 >> "file3.xg"

}

}
END{
print("Final Values\n")
print("Filesize : ",filesize1)
latency=endtime1-starttime1
print("Latency :",latency)
bandwidth1=filesize1/latency
print("Throughput (Mbps):",bandwidth1/10^6)
}
```