**Program:**

include <wifi\_lte/wifi\_lte\_rtable.h>

struct r\_hist\_entry \*elm, \*elm2;

int num\_later = 1;

elm = STAILQ\_FIRST(&r\_hist\_);

while (elm != NULL && num\_later <= num\_dup\_acks\_){

num\_later;

elm = STAILQ\_NEXT(elm, linfo\_);

}

if (elm != NULL){

elm = findDataPacketInRecvHistory(STAILQ\_NEXT(elm,linfo\_));

if (elm != NULL){

elm2 = STAILQ\_NEXT(elm, linfo\_);

while(elm2 != NULL){

if (elm2->seq\_num\_ < seq\_num && elm2->t\_recv\_ < time){

STAILQ\_REMOVE(&r\_hist\_,elm2,r\_hist\_entry,linfo\_);

delete elm2;

} else

elm = elm2;

elm2 = STAILQ\_NEXT(elm, linfo\_);

}

}

}

}

void DCCPTFRCAgent::removeAcksRecvHistory(){

struct r\_hist\_entry \*elm1 = STAILQ\_FIRST(&r\_hist\_);

struct r\_hist\_entry \*elm2;

int num\_later = 1;

while (elm1 != NULL && num\_later <= num\_dup\_acks\_){

num\_later;

elm1 = STAILQ\_NEXT(elm1, linfo\_);

}

if(elm1 == NULL)

return;

elm2 = STAILQ\_NEXT(elm1, linfo\_);

while(elm2 != NULL){

if (elm2->type\_ == DCCP\_ACK){

STAILQ\_REMOVE(&r\_hist\_,elm2,r\_hist\_entry,linfo\_);

delete elm2;

} else {

elm1 = elm2;

}

elm2 = STAILQ\_NEXT(elm1, linfo\_);

}

}

inline r\_hist\_entry \*DCCPTFRCAgent::findDataPacketInRecvHistory(r\_hist\_entry \*start){

while(start != NULL && start->type\_ == DCCP\_ACK)

start = STAILQ\_NEXT(start,linfo\_);

return start;

}

**Program:**

set ns [new Simulator]

#Define different colors for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red

#Open the Trace files

set file1 [open out.tr w]

set winfile [open WinFile w]

$ns trace-all $file1

#Open the NAM trace file

set file2 [open out.nam w]

$ns namtrace-all $file2

#Define a 'finish' procedure

proc finish {} {

global ns file1 file2

$ns flush-trace

close $file1

close $file2

exec nam out.nam &

exit 0

}

#Create six nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

$n1 color red

$n1 shape box

#Create links between the nodes

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

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

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

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

set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Ca Channel]

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"

$ns at 124.0 "$ftp stop"

$ns at 124.5 "$cbr stop"

# next procedure gets two arguments: the name of the

# tcp source node, will be called here "tcp",

# and the name of output file.

proc plotWindow {tcpSource file} {

global ns

set time 0.1

set now [$ns now]

set cwnd [$tcpSource set cwnd\_]

set wnd [$tcpSource set window\_]

puts $file "$now $cwnd"

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

$ns at 0.1 "plotWindow $tcp $winfile"

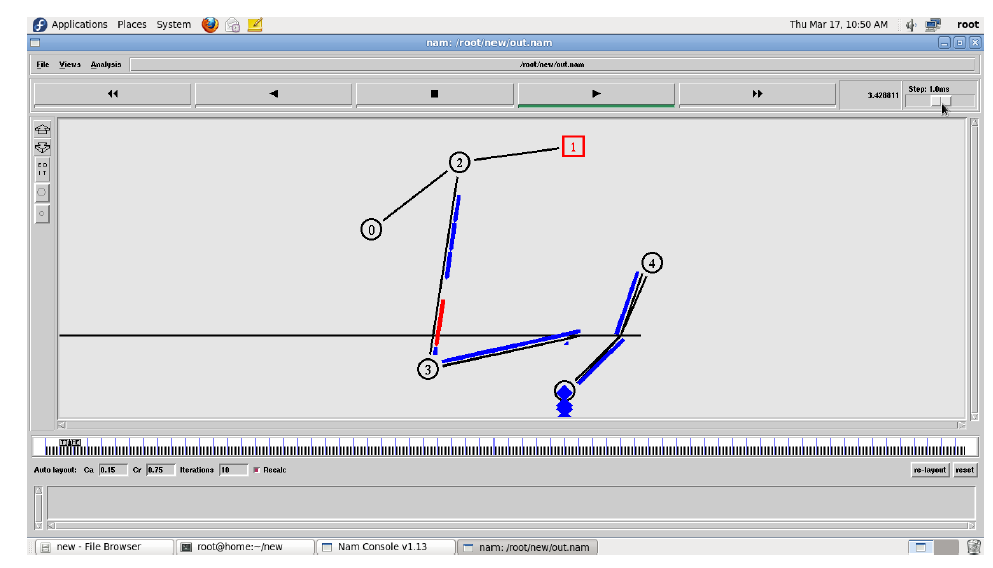
$ns at 5 "$ns trace-annotate \"packet drop\""

# PPP

$ns at 125.0 "finish"

$ns run

**Output :**

****