**COMPUTER NETWORKS**

**PROJECT REPORT**

Submitted by:

**Aakash Jain (9913103653)**

**Kapil Sharma (9913103514)**

**Naman Sharma (9913103546)**

Under the supervision of:

**Prof. Sanjeev Patel**

**(Department of CSE/IT)**

****

**Department of CSE/IT**

**Jaypee Institute of Information Technology , Noida**

# MAY 2016

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

#Aim : To monitor traffic for Star, Ring and mesh topology using NS2

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#

// mesh.tcl

#Create a simulator object

set ns [new Simulator]

#Define different colors for data flows (for NAM)

$ns color 1 Blue

$ns color 2 Red

$ns color 3 green

#Open the NAM trace file

set nf [open out.nam w]

$ns namtrace-all $nf

#Open the Trace file

set tf [open out.tr w]

$ns trace-all $tf

#Define a 'finish' procedure

proc finish {} {

global ns nf tf

$ns flush-trace

#Close the NAM trace file

close $nf

#Close the Trace file

close $tf

#Execute NAM on the trace file

exec nam out.nam &

exit 0

}

#Create four nodes

set n0 [$ns node]

set n1 [$ns node]

set n2 [$ns node]

set n3 [$ns node]

set n4 [$ns node]

set n5 [$ns node]

set n6 [$ns node]

set n7 [$ns node]

set n8 [$ns node]

set n9 [$ns node]

set n10 [$ns node]

set n11 [$ns node]

set n12 [$ns node]

set n13 [$ns node]

set n14 [$ns node]

set n15 [$ns node]

#Create links between the nodes

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

$ns duplex-link $n0 $n2 1MB 10ms DropTail

$ns duplex-link $n0 $n3 1MB 10ms DropTail

$ns duplex-link $n1 $n2 1MB 10ms DropTail

$ns duplex-link $n1 $n3 1MB 10ms DropTail

$ns duplex-link $n2 $n3 1MB 10ms DropTail

$ns duplex-link $n3 $n4 1MB 10ms DropTail

$ns duplex-link $n4 $n5 1MB 10ms DropTail

$ns duplex-link $n4 $n6 1MB 10ms DropTail

$ns duplex-link $n7 $n8 1MB 10ms DropTail

$ns duplex-link $n4 $n8 1MB 10ms DropTail

$ns duplex-link $n4 $n9 1MB 10ms DropTail

$ns duplex-link $n7 $n10 1MB 10ms DropTail

$ns duplex-link $n10 $n11 1MB 10ms DropTail

$ns duplex-link $n11 $n12 1MB 10ms DropTail

$ns duplex-link $n12 $n13 1MB 10ms DropTail

$ns duplex-link $n13 $n14 1MB 10ms DropTail

$ns duplex-link $n14 $n15 1MB 10ms DropTail

$ns duplex-link $n15 $n10 1MB 10ms DropTail

#Create a TCP agent and attach it to node n0

set tcp0 [new Agent/TCP]

$tcp0 set class\_ 1

$ns attach-agent $n1 $tcp0

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3

set sink0 [new Agent/TCPSink]

$ns attach-agent $n13 $sink0

#Connet the traffic sources with the traffic sink

$ns connect $tcp0 $sink0

#Create a TCP agent and attach it to node n0

set tcp1 [new Agent/TCP]

$tcp1 set class\_ 2

$ns attach-agent $n9 $tcp1

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3

set sink1 [new Agent/TCPSink]

$ns attach-agent $n5 $sink1

#Connet the traffic sources with the traffic sink

$ns connect $tcp1 $sink1

#Create a TCP agent and attach it to node n0

set tcp2 [new Agent/TCP]

$tcp2 set class\_ 3

$ns attach-agent $n6 $tcp2

#Create a TCP Sink agent (a traffic sink) for TCP and attach it to node n3

set sink2 [new Agent/TCPSink]

$ns attach-agent $n2 $sink2

#Connet the traffic sources with the traffic sink

$ns connect $tcp2 $sink2

# Create a FTP traffic source and attach it to tcp0

set ftp0 [new Application/FTP]

$ftp0 set packetSize\_ 1500

$ftp0 set interval\_ 0.07

$ftp0 attach-agent $tcp0

# Create a FTP traffic source and attach it to tcp0

set ftp1 [new Application/FTP]

$ftp1 set packetSize\_ 1500

$ftp1 set interval\_ 0.1

$ftp1 attach-agent $tcp1

# Create a FTP traffic source and attach it to tcp0

set ftp2 [new Application/FTP]

$ftp2 set packetSize\_ 1500

$ftp2 set interval\_ 0.2

$ftp2 attach-agent $tcp2

#Schedule events for the ftp agents

$ns at 0.5 "$ftp0 start"

$ns at 1.0 "$ftp1 start"

$ns at 1.0 "$ftp2 start"

$ns at 6.0 "$ftp1 stop"

$ns at 6.5 "$ftp2 stop"

$ns at 7.0 "$ftp0 stop"

#Call the finish procedure after 5 seconds of simulation time

$ns at 7.5 "finish"

#Run the simulation

$ns run

/// throughput.awk

BEGIN

{

fromNode=3; toNode=4;

lineCount = 0;totalBits = 0;

}

/^r/&&$3==fromNode&&$4==toNode {

totalBits += 8\*$6;

if ( lineCount==0 )

{

timeBegin = $2; lineCount++;

}

Else

{

timeEnd = $2;

};

};

END

{

duration = timeEnd-timeBegin;

print "Number of records is " NR;

print "Output: ";

print "Transmission: N" fromNode "->N" toNode;

print " - Total transmitted bits = " totalBits " bits";

print " - duration = " duration " s";

print " - Thoughput = " totalBits/duration/1e3 " kbps.";

};

//endtoenddelay.awk

BEGIN {

highest\_packet\_id = 0;

}

{

action = $1;

time = $2;

from = $3;

to = $4;

type = $5;

pktsize = $6;

flow\_id = $8;

src = $9;

dst = $10;

seq\_no = $11;

packet\_id = $12;

if ( packet\_id > highest\_packet\_id )

highest\_packet\_id = packet\_id;

if ( start\_time[packet\_id] == 0 )

start\_time[packet\_id] = time;

if ( flow\_id == 2 && action != "d" ) {

if ( action == "r" ) {

end\_time[packet\_id] = time;

}

} else {

end\_time[packet\_id] = -1;

}

}

END {

for ( packet\_id = 0; packet\_id <= highest\_packet\_id; packet\_id++ ) {

start = start\_time[packet\_id];

end = end\_time[packet\_id];

packet\_duration = end - start;

if ( start < end ) printf("%f %f\n", start, packet\_duration);

else printf("kuch nhi hua");

}

}

//jitters.awk

#--------- Formula ------------:

#Packet Loss = GeneratedPackets - ReceivedPackets

# jitter =((recvtime(j)-sendtime(j))-(recvtime(i)-sendtime(i)))/(j-i), j > i

BEGIN {

# Initialization

highest\_packet\_id = 0;

}

{

action = $1;

time = $2;

from = $3;

to = $4;

type = $5;

pktsize = $6;

flow\_id = $8;

src = $9;

dst = $10;

seq\_no = $11;

packet\_id = $12;

if ( packet\_id > highest\_packet\_id ) {

highest\_packet\_id = packet\_id;

}

#Record the transmission time

if ( start\_time[packet\_id] == 0 ) {

# Record the sequence number

pkt\_seqno[packet\_id] = seq\_no;

start\_time[packet\_id] = time;

}

#Record the receiving time for CBR (flow\_id=2)

if ( flow\_id == 2 && action != "d" ) {

if ( action == "r" ) {

end\_time[packet\_id] = time;

}

} else {

end\_time[packet\_id] = -1;

}

}

END {

last\_seqno = 0;

last\_delay = 0;

seqno\_diff = 0;

for ( packet\_id = 0; packet\_id <= highest\_packet\_id; packet\_id++ ) {

start = start\_time[packet\_id];

end = end\_time[packet\_id];

packet\_duration = end - start;

if ( start < end ) {

seqno\_diff = pkt\_seqno[packet\_id] - last\_seqno;

delay\_diff = packet\_duration -d last\_delay;

if (seqno\_diff == 0) {

jitter =0;

} else {

jitter = delay\_diff/seqno\_diff;

}

printf("Start: %f Jitter: %f\n", start, jitter);

last\_seqno = pkt\_seqno[packet\_id];

last\_delay = packet\_duration;

}}}