**CODE-1:**

**SERVER:**

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class SimpleServerGUI {

private static ServerSocket serverSocket;

private static Socket clientSocket;

private static BufferedReader in;

private static PrintWriter out;

public static void main(String[] args) {

JFrame frame = new JFrame("Server");

JTextArea textArea = new JTextArea(10, 30);

JScrollPane scrollPane = new JScrollPane(textArea);

JTextField textField = new JTextField(20);

JButton sendButton = new JButton("Send");

frame.setLayout(new java.awt.BorderLayout());

frame.add(scrollPane, java.awt.BorderLayout.CENTER);

JPanel panel = new JPanel();

panel.add(textField);

panel.add(sendButton);

frame.add(panel, java.awt.BorderLayout.SOUTH);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

try {

serverSocket = new ServerSocket(12345);

textArea.append("Server started. Waiting for a client...\n");

clientSocket = serverSocket.accept();

textArea.append("Client connected.\n");

in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));

out = new PrintWriter(clientSocket.getOutputStream(), true);

sendButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String message = textField.getText();

out.println("Server: " + message);

textArea.append("Server: " + message + "\n");

textField.setText("");

}

});

String message;

while ((message = in.readLine()) != null) {

textArea.append(message + "\n");

}

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

if (clientSocket != null) clientSocket.close();

if (serverSocket != null) serverSocket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

**CLIENT:**

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class SimpleClientGUI {

private static Socket socket;

private static BufferedReader in;

private static PrintWriter out;

public static void main(String[] args) {

JFrame frame = new JFrame("Client");

JTextArea textArea = new JTextArea(10, 30);

JScrollPane scrollPane = new JScrollPane(textArea);

JTextField textField = new JTextField(20);

JButton sendButton = new JButton("Send");

frame.setLayout(new java.awt.BorderLayout());

frame.add(scrollPane, java.awt.BorderLayout.CENTER);

JPanel panel = new JPanel();

panel.add(textField);

panel.add(sendButton);

frame.add(panel, java.awt.BorderLayout.SOUTH);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.pack();

frame.setVisible(true);

try {

socket = new Socket("localhost", 12345);

in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out = new PrintWriter(socket.getOutputStream(), true);

sendButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

String message = textField.getText();

out.println("Client: " + message);

textArea.append("Client: " + message + "\n");

textField.setText("");

}

});

String response;

while ((response = in.readLine()) != null) {

textArea.append(response + "\n");

}

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

if (socket != null) socket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

**CODE-2:**

import java.io.\*;

import java.net.\*;

public class TCPServer {

public static void main(String[] args) {

try (ServerSocket ss = new ServerSocket(4567)) {

System.out.println("Server is running and waiting for clients...");

while (true) {

Socket s = ss.accept();

System.out.println("Client connected.");

new ClientHandler(s).start();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

class ClientHandler extends Thread {

private Socket socket;

ClientHandler(Socket socket) {

this.socket = socket;

}

public void run() {

try {

DataInputStream dis = new DataInputStream(socket.getInputStream());

DataOutputStream dos = new DataOutputStream(socket.getOutputStream());

String command = dis.readUTF();

if (command.equalsIgnoreCase("upload")) {

String filename = dis.readUTF();

int len = dis.readInt();

byte[] data = new byte[len];

dis.readFully(data);

try (FileOutputStream fos = new FileOutputStream(filename)) {

fos.write(data);

System.out.println("File uploaded: " + filename);

}

} else if (command.equalsIgnoreCase("download")) {

String filename = dis.readUTF();

File file = new File(filename);

if (file.exists()) {

byte[] data = new byte[(int) file.length()];

try (FileInputStream fis = new FileInputStream(file)) {

fis.read(data);

}

dos.writeUTF("OK");

dos.writeInt(data.length);

dos.write(data);

} else {

dos.writeUTF("ERROR: File not found");

}

}

socket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

import java.io.\*;

import java.net.\*;

import java.util.Scanner;

public class TCPClient {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

while (true) {

System.out.println("Enter 'upload' to upload a file, 'download' to download a file, or 'exit' to quit:");

String command = scanner.nextLine().trim().toLowerCase();

if (command.equals("exit")) {

System.out.println("Exiting the client.");

break;

}

System.out.print("Enter file name: ");

String filename = scanner.nextLine().trim();

if (command.equals("upload")) {

uploadFile(filename);

} else if (command.equals("download")) {

downloadFile(filename);

} else {

System.out.println("Invalid command. Please enter 'upload' or 'download'.");

}

}

scanner.close();

}

private static void uploadFile(String filename) {

try (Socket s = new Socket("localhost", 4567);

DataOutputStream dos = new DataOutputStream(s.getOutputStream())) {

dos.writeUTF("upload");

dos.writeUTF(filename);

File file = new File(filename);

if (!file.exists()) {

System.out.println("File not found: " + filename);

return;

}

byte[] data = new byte[(int) file.length()];

try (FileInputStream fis = new FileInputStream(file)) {

fis.read(data);

}

// Encrypt data using Caesar cipher

for (int i = 0; i < data.length; i++) {

data[i] = (byte) ((data[i] + 3) % 256);

}

dos.writeInt(data.length);

dos.write(data);

System.out.println("File uploaded: " + filename);

} catch (IOException e) {

e.printStackTrace();

}

}

private static void downloadFile(String filename) {

try (Socket s = new Socket("localhost", 4567);

DataOutputStream dos = new DataOutputStream(s.getOutputStream());

DataInputStream dis = new DataInputStream(s.getInputStream())) {

dos.writeUTF("download");

dos.writeUTF(filename);

String response = dis.readUTF();

if (response.equalsIgnoreCase("OK")) {

int len = dis.readInt();

byte[] data = new byte[len];

dis.readFully(data);

// Decrypt data using Caesar cipher

for (int i = 0; i < data.length; i++) {

data[i] = (byte) ((data[i] - 3 + 256) % 256);

}

try (FileOutputStream fos = new FileOutputStream(filename)) {

fos.write(data);

}

System.out.println("File downloaded: " + filename);

} else {

System.out.println(response);

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

**CODE-3:**

**INTF:**

//function prototype

import java.rmi.\*;

public interface MyServerIntf extends Remote //remote interface

{ int i=0;

double add(double a, double b) throws RemoteException;

}

**IMPL:**

import java.rmi.\*;

import java.rmi.server.\*;

// UnicastRemoteObject supports for point-to-point active object references (invocations, parameters, and // results) using TCP streams.

public class MyServerImpl extends UnicastRemoteObject implements MyServerIntf

{

MyServerImpl() throws RemoteException

{}

public double add(double a, double b) throws RemoteException

{

return(a+b);

}

}

**SERVER:**

import java.net.\*;

import java.rmi.\*;

public class MyServer

{

public static void main(String[] arg)

{

try

{

MyServerImpl asi = new MyServerImpl();

Naming.rebind("RMServer",asi);

System.out.println("\nServer Started...");

}

catch(Exception e)

{

System.out.println("Exception: "+e);

}

}

}

**CLIENT:**

import java.net.\*;

import java.rmi.\*;

public class MyClient

{

public static void main(String[] arg)

{

try

{

String sName = "rmi://"+arg[0]+"/RMServer";

MyServerIntf asif = (MyServerIntf)Naming.lookup(sName); // requesting remote objects on // the server

double d1=2000,d2=500;

System.out.println("Addition: "+asif.add(d1,d2));

}

catch(Exception e)

{

System.out.println("Exception: "+e);

}

}

}

**CODE-4:**

**NS2(WIRED):**

set ns [new Simulator]

$ns color 0 blue

$ns color 1 black

set tr [open droptail-queue-out.tr w]

$ns trace-all $tr

set ftr [open droptail-queue-out.nam w]

$ns namtrace-all $ftr

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 5Mb 2ms DropTail

$ns duplex-link $n1 $n2 10Mb 5ms DropTail

$ns duplex-link $n2 $n3 4Mb 3ms DropTail

$ns duplex-link $n3 $n4 100Mb 2ms DropTail

$ns duplex-link $n3 $n5 15Mb 4ms DropTail

set udp [new Agent/UDP]

$udp set fid\_ 1

set null [new Agent/Null]

$ns attach-agent $n0 $udp

$ns attach-agent $n4 $null

$ns connect $udp $null

set tcp [new Agent/TCP]

$tcp set fid\_ 0

set sink [new Agent/TCPSink]

$ns attach-agent $n1 $tcp

$ns attach-agent $n5 $sink

$ns connect $tcp $sink

$ns connect $tcp $sink

set cbr [new Application/Traffic/CBR]

$cbr attach-agent $udp

$cbr set interval 0.020

set ftp [new Application/FTP]

$ftp attach-agent $tcp

$ftp set interval 0.020

proc finish {} {

global ns tr ftr

$ns flush-trace

close $tr

close $ftr

exec nam droptail-queue-out.nam &

exec gawk -f analysis.awk droptail-queue-out.tr &

exit

}

$ns at 0.1 "$cbr start"

$ns at 2.0 "$cbr stop"

$ns at 0.1 "$ftp start"

$ns at 2.0 "$ftp stop"

$ns at 2.1 "finish"

$ns run

**THROUGHPUT FILE:**

BEGIN {

rec= 0

drp=0

tot=0

rat=0.0

sum=0

sum1=0

throughput=0.0

throughput1=0.0

}

{

if($1== "r" && $4== 4)

{

rec++

}

if($1== "d" && $4 ==4 )

{

drp++

}

if($2<1.00 && $4==4)

{

sum=sum+$6

}

if($2<1.00 && $4==5)

{

sum1=sum1+$6

}

}

END {

tot = rec + drp

rat = (rec/tot) \*100

throughput= (sum\*8)/1000000

throughput1=(sum1\*8)/1000000

printf(" \n Packets received %d ", rec)

printf(" \n Packets dropped %d ", drp)

printf("\n Packets delivery ratio %f",rat)

printf("\n Throughput for udp is %f",throughput)

printf("\n Throughput for tcp is %f",throughput1)

}

EXECUTION:

~$ cd Desktop/

ns wired-net.tcl

nam wired.nam

awk -f sample.awk wired.tr

CONGESTION.tcl

set ns [new Simulator]

set f [ open congestion.tr w ]

$ns trace-all $f

set nf [ open congestion.nam w ]

$ns namtrace-all $nf

$ns color 1 Red

$ns color 2 Blue

$ns color 3 White

$ns color 4 Green

#to 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]

# to create the link between the nodes with bandwidth, delay and queue

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

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

$ns duplex-link $n2 $n3 0.3Mb 200ms DropTail

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

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

# Sending node with agent as Reno Agent

set tcp1 [new Agent/TCP/Reno]

$ns attach-agent $n0 $tcp1

set tcp2 [new Agent/TCP/Reno]

$ns attach-agent $n1 $tcp2

set tcp3 [new Agent/TCP/Reno]

$ns attach-agent $n2 $tcp3

set tcp4 [new Agent/TCP/Reno]

$ns attach-agent $n1 $tcp4

$tcp1 set fid\_ 1

$tcp2 set fid\_ 2

$tcp3 set fid\_ 3

$tcp4 set fid\_ 4

# receiving (sink) node

set sink1 [new Agent/TCPSink]

$ns attach-agent $n4 $sink1

set sink2 [new Agent/TCPSink]

$ns attach-agent $n5 $sink2

set sink3 [new Agent/TCPSink]

$ns attach-agent $n3 $sink3

set sink4 [new Agent/TCPSink]

$ns attach-agent $n4 $sink4

# establish the traffic between the source and sink

$ns connect $tcp1 $sink1

$ns connect $tcp2 $sink2

$ns connect $tcp3 $sink3

$ns connect $tcp4 $sink4

# Setup a FTP traffic generator on "tcp"

set ftp1 [new Application/FTP]

$ftp1 attach-agent $tcp1

$ftp1 set type\_ FTP

set ftp2 [new Application/FTP]

$ftp2 attach-agent $tcp2

$ftp2 set type\_ FTP

set ftp3 [new Application/FTP]

$ftp3 attach-agent $tcp3

$ftp3 set type\_ FTP

set ftp4 [new Application/FTP]

$ftp4 attach-agent $tcp4

$ftp4 set type\_ FTP

# RTT Calculation Using Ping ------------------------------------------------

set p0 [new Agent/Ping]

$ns attach-agent $n0 $p0

set p1 [new Agent/Ping]

$ns attach-agent $n4 $p1

#Connect the two agents

$ns connect $p0 $p1

# Method call from ping.cc file

Agent/Ping instproc recv {from rtt} {

$self instvar node\_

puts "node [$node\_ id] received ping answer from \

$from with round-trip-time $rtt ms."

}

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

# start/stop the traffic

$ns at 0.2 "$p0 send"

$ns at 0.3 "$p1 send"

$ns at 0.5 "$ftp1 start"

$ns at 0.6 "$ftp2 start"

$ns at 0.7 "$ftp3 start"

$ns at 0.8 "$ftp4 start"

$ns at 66.0 "$ftp4 stop"

$ns at 67.0 "$ftp3 stop"

$ns at 68.0 "$ftp2 stop"

$ns at 70.0 "$ftp1 stop"

$ns at 70.1 "$p0 send"

$ns at 70.2 "$p1 send"

# Set simulation end time

$ns at 80.0 "finish"

# procedure to plot the congestion window

# cwnd\_ used from tcp-reno.cc file

proc plotWindow {tcpSource outfile} {

global ns

set now [$ns now]

set cwnd\_ [$tcpSource set cwnd\_]

# the data is recorded in a file called congestion.xg.

puts $outfile "$now $cwnd\_"

$ns at [expr $now+0.1] "plotWindow $tcpSource $outfile"

}

set outfile [open "congestion.xg" w]

$ns at 0.0 "plotWindow $tcp1 $outfile"

proc finish {} {

exec nam congestion.nam &

exec xgraph congestion.xg -geometry 300x300 &

exit 0

}

# Run simulation

$ns run

FLOW.tcl

#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

analysis.awk

BEGIN{

st1=0

ft1=0

throughput1=0

delay1=0

flag1=0

data1=0

st2=0

ft2=0

throughput2=0

delay2=0

flag2=0

data2=0

st3=0

ft3=0

throughput3=0

delay3=0

flag3=0

data3=0

total\_delay=0

total\_th=0

}

{

if($1=="r"&&$4==7)#http

{

data1+=$6

if(flag1==0)

{

st1=$2

flag1=1

}

if(flag1==1)

{

ft1=$2

}

}

if($1=="r"&&$4==8)#ftp

{

data2+=$6

if(flag2==0)

{

st2=$2

flag2=1

}

if(flag2==1)

{

ft2=$2

}

}

if($1=="r"&&$4==6)#smtp

{

data3+=$6

if(flag3==0)

{

st3=$2

flag3=1

}

if(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("start time %f\n",st3)

printf("end time %f\n",ft3)

printf("throughput %f\n",throughput3)

printf("delay %f\n",delay3)

printf("data %f\n",data3)

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)

}

cc-analysis.awk

BEGIN{

st1=0

ft1=0

throughput1=0

delay1=0

flag1=0

data1=0

}

{

if($1=="r"&&$4==5)#http

{

data1+=$6

if(flag1==0)

{

st1=$2

flag1=1

}

if(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

throughput1=data1/delay1

printf("Throughput %f\n",throughput1)

printf("Delay %f\n",delay1)

}

Commands.txt:

ns flow.tcl

ns congestion.tcl

awk -f exp6\_analysis.awk trace1.tr

awk -f exp6-cc-analysis.awk congestion.tr

LINK STATE:

set ns [new Simulator]

$ns rtproto 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\_ls.tr w]

$ns trace-all $tf

set nf [open out\_ls.nam w]

$ns namtrace-all $nf

set ft [open "lsr\_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\_ls.nam &

exec xgraph lsr\_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 node3\""

$ns at 0.5 "$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

routing\_throughput.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\n StartTime=%.2f\nStopTime=%.2f\n",(recvdSize/(stopTime-startTime))\*(8/1000),startTime,stopTime)

}

Command.txt:

ns LS.tr

awk -f routing\_throughput out\_ls.tr

Distance Vector.tcl:

set ns [new Simulator]

$ns rtproto DV

$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

Routing\_throughput.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\n StartTime=%.2f\nStopTime=%.2f\n",(recvdSize/(stopTime-startTime))\*(8/1000),startTime,stopTime)

}

Commands.txt:

ns DV.tcl

awk -f routing\_throughput out\_dv.tr

EXP-8:

multicast\_routing.tcl:

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 $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

#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 &amp;

exit 0

}

$ns set-animation-rate 3.0ms

$ns run

analysis.awk:

BEGIN {

recvdSize = 0

startTime = 0.5

stopTime = 0.0

}

{

event = $1

time = $2

node\_id = $3

pkt\_size = $6

# Update startTime and stopTime based on packet reception

if (event == "s") {

if (time < startTime) {

startTime = time

}

}

if (event == "r") {

if (time > stopTime) {

stopTime = time

}

recvdSize += pkt\_size

}

}

END {

if (stopTime > startTime) {

avgThroughput = (recvdSize / (stopTime - startTime)) \* (8 / 1000); # Convert to kbps

printf("Average Throughput [kbps] = %.2f\nStart Time = %.2f\nStop Time = %.2f\n", avgThroughput, startTime, stopTime);

} else {

printf("No packets received during the specified time interval.\n");

}

}

Commands.txt:

awk -f analysis.awk output.tr

nam mcast.nam

EXP-9:

DHCPClient.java:

import java.net.\*;

import java.util.Arrays;

public class DHCPClient {

private static final int SERVER\_PORT = 4900;

private static final String SERVER\_IP = "127.0.0.1"; // Change to your server's IP

public static void main(String[] args) {

try {

DatagramSocket socket = new DatagramSocket();

InetAddress serverAddress = InetAddress.getByName(SERVER\_IP);

// Create and send DHCP request

byte[] requestData = createDHCPRequest("00:11:22:33:44:55"); // Replace with your MAC address

DatagramPacket requestPacket = new DatagramPacket(requestData, requestData.length, serverAddress, SERVER\_PORT);

socket.send(requestPacket);

// Receive DHCP response

byte[] receiveData = new byte[1024];

DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

socket.receive(receivePacket);

// Process and print DHCP response

String response = new String(receivePacket.getData()).trim();

System.out.println("Received DHCP Response: " + response);

} catch (Exception e) {

e.printStackTrace();

}

}

private static byte[] createDHCPRequest(String macAddress) {

// Simulate creating a DHCP request packet with the MAC address

// In a real implementation, you'd construct a proper DHCP packet

String request = "DHCP Request with MAC: " + macAddress;

return request.getBytes();

}

}

DHCPServer.java:

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class DHCPServer {

private static final int SERVER\_PORT = 4900;

private static final String SERVER\_IP = "127.0.0.1"; // Change to your server's IP

private static final String IP\_ALLOCATIONS\_FILE = "ip\_allocations.txt";

private static List<String> availableIpAddresses = new ArrayList<>();

private static Map<String, String> ipAllocations = new HashMap<>();

public static void main(String[] args) {

loadIpAllocations(); // Load IP allocations from file (if available)

initializeIpAddresses();

try {

DatagramSocket socket = new DatagramSocket(SERVER\_PORT);

while (true) {

byte[] receiveData = new byte[1024];

DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);

socket.receive(receivePacket);

// Simulate IP address allocation

InetAddress clientAddress = receivePacket.getAddress();

String macAddress = extractMacAddress(receiveData);

String allocatedIp = allocateIpAddress(macAddress);

// Create and send DHCP response

byte[] responseData = createDHCPResponse(macAddress, allocatedIp);

DatagramPacket responsePacket = new DatagramPacket(responseData, responseData.length,

clientAddress, receivePacket.getPort());

socket.send(responsePacket);

System.out.println("Allocated IP " + allocatedIp + " to client with MAC " + macAddress);

// Save IP allocations to file

saveIpAllocations();

}

} catch (Exception e) {

e.printStackTrace();

}

}

private static void initializeIpAddresses() {

// Simulate a pool of available IP addresses

for (int i = 2; i <= 254; i++) {

availableIpAddresses.add("192.168.1." + i);

}

}

private static String extractMacAddress(byte[] data) {

// Simulate extracting MAC address from DHCP request

// In a real implementation, you'd parse the DHCP request to get the MAC address

return "00:11:22:33:44:55"; // Placeholder

}

private static String allocateIpAddress(String macAddress) {

if (availableIpAddresses.isEmpty()) {

return "No available IP addresses";

}

Random random = new Random();

int index = random.nextInt(availableIpAddresses.size());

String allocatedIp = availableIpAddresses.remove(index);

// Bind MAC address with the allocated IP address

ipAllocations.put(macAddress, allocatedIp);

return allocatedIp;

}

private static byte[] createDHCPResponse(String macAddress, String allocatedIp) {

// Simulate creating a DHCP response with the allocated IP address

// In a real implementation, you'd construct a proper DHCP packet

return ("Allocated IP: " + allocatedIp).getBytes();

}

private static void saveIpAllocations() {

try (ObjectOutputStream outputStream = new ObjectOutputStream(new FileOutputStream(IP\_ALLOCATIONS\_FILE))) {

outputStream.writeObject(ipAllocations);

System.out.println("Saved IP allocations to " + IP\_ALLOCATIONS\_FILE);

} catch (IOException e) {

e.printStackTrace();

}

}

private static void loadIpAllocations() {

try (ObjectInputStream inputStream = new ObjectInputStream(new FileInputStream(IP\_ALLOCATIONS\_FILE))) {

ipAllocations = (HashMap<String, String>) inputStream.readObject();

System.out.println("Loaded IP allocations from " + IP\_ALLOCATIONS\_FILE);

} catch (FileNotFoundException e) {

System.out.println(IP\_ALLOCATIONS\_FILE + " not found. Starting with an empty IP allocations map.");

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

}

}

EXP-10:

LAN.tcl:

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 analysis\_BW-Delay.awk LAN.tr &

exec xgraph file3.xg &

exit 0

}

$ns at 10 "finish"

$ns run

analysis\_BW-Delay.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("\n\n\n Final Values..")

print("\n\nfilesize : ",filesize1)

latency=endtime1-starttime1

print("\nlatency :",latency)

bandwidth1=filesize1/latency

print("\n Throughput (Mbps):",bandwidth1/10^6)

}

Commands.txt:

XGraph v4.38

awk -f analysis\_BW-Delay.awk LAN.tr

EXP-11.tcl:

Mac/802\_11 set dataRate\_ 1Mb

set val(chan) Channel/WirelessChannel ;# channel type

set val(prop) Propagation/TwoRayGround ;# radio-propagation model

set val(ant) Antenna/OmniAntenna ;# Antenna type

set val(ll) LL ;# Link layer type

set val(ifq) Queue/DropTail/PriQueue ;# Interface queue type

set val(ifqlen) 50 ;# max packet in ifq

set val(netif) Phy/WirelessPhy ;# network interface type

set val(mac) Mac/802\_11 ;# MAC type

set val(nn) 15 ;# number of mobilenodes

set val(rp) AODV ;# routing protocol

set val(x) 800

set val(y) 800

# Creating simulation object

set ns [new Simulator]

#creating Output trace files

set f [open complexdcf.tr w]

$ns trace-all $f

set namtrace [open complexdcf.nam w]

$ns namtrace-all-wireless $namtrace $val(x) $val(y)

set f0 [open C\_DCF\_AT.tr w]

set topo [new Topography]

$topo load\_flatgrid 800 800

# Defining Global Variables

create-god $val(nn)

set chan\_1 [new $val(chan)]

# setting the wireless nodes parameters

$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 OFF \

-routerTrace ON \

-macTrace ON \

-movementTrace OFF \

-channel $chan\_1

proc finish {} {

global ns f f0 namtrace# global variables

# Closing the trace files

$ns flush-trace

#close $namtrace

close $f0

exec nam -r 5m complexdcf.nam & # Running the animator

exit 0

}

# Defining a procedure to calculate the througpout

proc record {} {

global sink1 sink3 sink7 sink10 sink11 f0

set ns [Simulator instance]

set time 0.5

set bw0 [$sink3 set bytes\_]

set bw3 [$sink3 set bytes\_]

set bw7 [$sink7 set bytes\_]

set bw10 [$sink10 set bytes\_]

set bw11 [$sink11 set bytes\_]

set now [$ns now]

puts $f0 "$now [expr ($bw0+$bw3+$bw7+$bw10+$bw11)/$time\*8/1000000]"

# Calculating the average throughput

$sink1 set bytes\_ 0

$sink3 set bytes\_ 0

$sink7 set bytes\_ 0

$sink10 set bytes\_ 0

$sink11 set bytes\_ 0

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

}

#Creating the wireless Nodes

for {set i 0} {$i < $val(nn) } {incr i} {

set n($i) [$ns node]

$n($i) random-motion 0 ;

}

#setting the initial position for the nodes

for {set i 0} {$i < $val(nn)} {incr i} {

$ns initial\_node\_pos $n($i) 30+i\*100

}

for {set i 0} {$i < $val(nn)} {incr i} {

$n($i) set X\_ 0.0

$n($i) set Y\_ 0.0

$n($i) set Z\_ 0.0

}

# making some nodes move in the topography

$ns at 0.0 "$n(0) setdest 100.0 100.0 3000.0"

$ns at 0.0 "$n(1) setdest 200.0 200.0 3000.0"

$ns at 0.0 "$n(2) setdest 300.0 200.0 3000.0"

$ns at 0.0 "$n(3) setdest 400.0 300.0 3000.0"

$ns at 0.0 "$n(4) setdest 500.0 300.0 3000.0"

$ns at 0.0 "$n(5) setdest 600.0 400.0 3000.0"

$ns at 0.0 "$n(6) setdest 600.0 100.0 3000.0"

$ns at 0.0 "$n(7) setdest 600.0 200.0 3000.0"

$ns at 0.0 "$n(8) setdest 600.0 300.0 3000.0"

$ns at 0.0 "$n(9) setdest 600.0 350.0 3000.0"

$ns at 0.0 "$n(10) setdest 700.0 100.0 3000.0"

$ns at 0.0 "$n(11) setdest 700.0 200.0 3000.0"

$ns at 0.0 "$n(12) setdest 700.0 300.0 3000.0"

$ns at 0.0 "$n(13) setdest 700.0 350.0 3000.0"

$ns at 0.0 "$n(14) setdest 700.0 400.0 3000.0"

$ns at 2.0 "$n(5) setdest 100.0 400.0 500.0"

$ns at 1.5 "$n(3) setdest 450.0 150.0 500.0"

$ns at 50.0 "$n(7) setdest 300.0 400.0 500.0"

$ns at 2.0 "$n(10) setdest 200.0 400.0 500.0"

$ns at 2.0 "$n(11) setdest 650.0 400.0 500.0"

#Creating receiving sinks with monitoring ability to monitor the incoming bytes

# LossMonitor objects are a subclass of agent objects that implement a traffic sink.

set sink1 [new Agent/LossMonitor]

set sink3 [new Agent/LossMonitor]

set sink7 [new Agent/LossMonitor]

set sink10 [new Agent/LossMonitor]

set sink11 [new Agent/LossMonitor]

$ns attach-agent $n(1) $sink1

$ns attach-agent $n(3) $sink3

$ns attach-agent $n(7) $sink7

$ns attach-agent $n(10) $sink10

$ns attach-agent $n(11) $sink11

# setting TCP as the transmission protocol over the connections

set tcp0 [new Agent/TCP]

$ns attach-agent $n(0) $tcp0

set tcp2 [new Agent/TCP]

$ns attach-agent $n(2) $tcp2

set tcp4 [new Agent/TCP]

$ns attach-agent $n(4) $tcp4

set tcp5 [new Agent/TCP]

$ns attach-agent $n(5) $tcp5

set tcp9 [new Agent/TCP]

$ns attach-agent $n(9) $tcp9

set tcp13 [new Agent/TCP]

$ns attach-agent $n(13) $tcp13

set tcp6 [new Agent/TCP]

$ns attach-agent $n(6) $tcp6

set tcp14 [new Agent/TCP]

$ns attach-agent $n(14) $tcp14

set tcp8 [new Agent/TCP]

$ns attach-agent $n(8) $tcp8

set tcp12 [new Agent/TCP]

$ns attach-agent $n(12) $tcp12

# Setting FTP connections

set ftp9 [new Application/FTP]

$ftp9 attach-agent $tcp9

$ftp9 set type\_ FTP

set ftp13 [new Application/FTP]

$ftp13 attach-agent $tcp13

$ftp13 set type\_ FTP

set ftp6 [new Application/FTP]

$ftp6 attach-agent $tcp6

$ftp6 set type\_ FTP

set ftp14 [new Application/FTP]

$ftp14 attach-agent $tcp14

$ftp14 set type\_ FTP

set ftp8 [new Application/FTP]

$ftp8 attach-agent $tcp8

$ftp8 set type\_ FTP

set ftp12 [new Application/FTP]

$ftp12 attach-agent $tcp12

$ftp12 set type\_ FTP

#connecting the nodes

$ns connect $tcp0 $sink3

$ns connect $tcp5 $sink3

$ns connect $tcp2 $sink1

$ns connect $tcp4 $sink1

$ns connect $tcp9 $sink7

$ns connect $tcp13 $sink7

$ns connect $tcp6 $sink10

$ns connect $tcp14 $sink10

$ns connect $tcp8 $sink11

$ns connect $tcp12 $sink11

# Defining CBR procedure with the required parametes

proc attach-CBR-traffic { node sink size interval } {

set ns [Simulator instance]

set cbr [new Agent/CBR]

$ns attach-agent $node $cbr

$cbr set packetSize\_ $size

$cbr set interval\_ $interval

$ns connect $cbr $sink

return $cbr

}

set cbr0 [attach-CBR-traffic $n(0) $sink3 1000 .015]

set cbr1 [attach-CBR-traffic $n(5) $sink3 1000 .015]

set cbr2 [attach-CBR-traffic $n(2) $sink1 1000 .015]

set cbr3 [attach-CBR-traffic $n(4) $sink1 1000 .015]

# Setting the begining and ending time of each connection

$ns at 0.0 "record"

$ns at 20.0 "$cbr0 start"

$ns at 20.0 "$cbr2 start"

$ns at 800.0 "$cbr0 stop"

$ns at 850.0 "$cbr2 stop"

$ns at 30.0 "$cbr1 start"

$ns at 30.0 "$cbr3 start"

$ns at 850.0 "$cbr1 stop"

$ns at 870.0 "$cbr3 stop"

$ns at 25.0 "$ftp6 start"

$ns at 25.0 "$ftp14 start"

$ns at 810.0 "$ftp6 stop"

$ns at 860.0 "$ftp14 stop"

$ns at 35.0 "$ftp9 start"

$ns at 35.0 "$ftp13 start"

$ns at 830.0 "$ftp9 stop"

$ns at 889.0 "$ftp13 stop"

$ns at 40.0 "$ftp8 start"

$ns at 40.0 "$ftp12 start"

$ns at 820.0 "$ftp8 stop"

$ns at 890.0 "$ftp12 stop"

$ns at 900.0 "finish"

# Runnning the simulation

puts "Start of simulation.."

$ns run