Ex.No.1 21/Jan/2020	**SOCKET PROGRAMMING – TCP & UDP**

AIM:

To Write programs for

- Finding gst for 3 different products

using Socket Programming.

THEORY:

Socket:

A network socket is an internal endpoint for sending and receiving data within a node on a computer network. A socket is a software object that acts as an endpoint establishing a bidirectional network communication link between a server side and a client-side program.

Differences between TCP and UDP:

S.No	TCP	UDP
1	TCP stands for Transmission Control Protocol	UDP stands for User Datagram Protocol
2	It is Connection Oriented Protocol	It is a Connection less protocol.
3	TCP establishes a connection between the computers before transmitting the data.	UDP transmits the data directly to the destination without checking whether the system is ready to receive or not
4	Slow transmission – low speed	Fast transmission-high Speed
5	Highly reliable	Un reliable

Registered Ports and Examples (Any seven):

The range of port numbers from 1024 to 49151 are the registered ports. They are assigned by IANA for specific service upon application by a requesting entity. On most systems, registered ports can be used without superuser privileges.

1.1167 – Cisco IP SLA(Service Assurance Agent)

2.1029 - Microsoft DCOM services

3.1059 - nimreg, IBM AIX Network Installation Manager (NIM)

4.1080 – SOCKS proxy

5.1085 – WebObjects

6.1098 – rmiactivation, Java remote method invocation(RMI) activation

7.1099 – rmiregistry, Java remote method invocation(RMI) registry

IANA:

Internet Assigned Numbers Authority is a function of ICANN, a nonprofit American corporation that oversees a global IP address allocation, autonomous system number allocation, root zone management in Domain Name System (DNS), media types and other Internet Protocol related symbols and internet numbers.

Process for TCP Socket connection establishment:

1. Sender starts the process.
2. TCP is a full duplex protocol so both sender and receiver require a window for receiving messages from one another.
3. Sender makes the final reply for connection establishment.

Client side

- Create a socket using the socket() function;
- Connect the socket to the address of the server using the connect() function;
- Send and receive data by means of the read() and write() functions.

Server Side

- Create a socket with the socket() function;
- Bind the socket to an address using the bind() function;
- Listen for connections with the listen() function;
- Accept a connection with the accept() function system call. This call typically blocks until a client connects with the server.

Process for UDP Socket connection establishment:

Client Side

- Create a socket using the socket() function;
- Send and receive data by means of the recvfrom() and sendto() functions.

Server Side

- Create a socket with the socket() function;
- Bind the socket to an address using the bind() function;
- Send and receive data by means of recvfrom() and sendto().

ALGORITHM:

TCP

Client:

1. Start
2. Import required java packages.
3. In Tclient class , create the object with IP address and port number for the class Socket.
4. Create object for PrintWriter with socket.getOutputStream() method.
5. Create object for BufferedReader with InputStreamReader and socket.getInputStream().
6. Create object for Scanner class for getting input from the users.
7. Display the list of products with choices and get choice and product from users.
8. Send the choice and product name to the Server with PrintWriter object .
9. If choice equals to 1 then get the reply from the server by readLine() method with BufferedReader object and print the gst for product 1.
10. If choice equals to 2 then get the product 2 from server readLine() method with BufferedReader object and displayed.
11. If choice equals to 3 then read the starting index from user , send the index to the server and display gst for product 3.
12. If choice equals to 4 then get the reply from server and end the process.
13. Close the connection.
14. End

Server

1. Start
2. Import required java packages.
3. In TServer class,create object for the class ServerSocket with port number.
4. Accept the request from client by creating object for Socket and call the accept()method.
5. Create object for PrintWriter with socket.getOutputStream() method.

6. Create object for BufferedReader with InputStreamReader and socket.getInputStream().
7. Receive the string and choice from client with BufferedReader object and readLine() method.
8. If choice equals to 1 then find the gst for product 1 using user defined function and convert it into integer and send to the client.
9. If choice equals to 2 then product 2 and find the gst for the product 2 and send to the client.
10. If choice equals to 3 then product 3 and find the gst for the product 3 and send to the client.
11. If choice equals to 4 then display the total cost and end the connection
12. Close the connection
13. End

CODING:

Client

```
import java.net.*;
import java.io.*;
import java.util.*;

public class Client
{
private Socket socket                                                    =
null;
private DataInputStream input = null;
private DataOutputStream out                                          = null;
public Client(String address, int port)
{
Scanner sc=new Scanner(System.in);
try
{
socket = new Socket(address, port);
System.out.println("Connected");
input = new DataInputStream(System.in);
out = new DataOutputStream(socket.getOutputStream());
}
```

```
catch(UnknownHostException u)
{
System.out.println(u);
}
catch(IOException i)
{
System.out.println(i);
}
int a,n,money,l;
String line = "";
while (!line.equals("Over"))
{
try
{

//line=input.readUTF();
line=sc.nextLine();
out.writeUTF(line);
}
catch(IOException i)
{
System.out.println(i);
}
}

// close the connection
try
{
input.close();
out.close();
socket.close();
}
catch(IOException i)
{
System.out.println(i);
}
}
```

```
public static void main(String args[])
{
Client client = new Client("127.0.0.1", 4000);
}
}
```

Server

```
import java.net.*;
import java.io.*;

public class Server
{
private Socket
socket = null;
private ServerSocket server = null;
private DataInputStream in = null;
public static int gst(int l,int money)
{
int gstamount=(money*l)/100;
return gstamount;
}
public Server(int port)
{
int a,n,money,l;
try
{
server = new ServerSocket(port);
System.out.println("Server started");

System.out.println("Waiting for a client ...");
socket = server.accept();
System.out.println("Client accepted");

// takes input from the client socket
in = new DataInputStream(
new BufferedInputStream(socket.getInputStream()));
```

```
String line = "";

// reads message from client until "Over" is sent
while (!line.equals("Over"))
{
try
{
String list[]=new String[]{"a.Electronics","b.Furnitures","c.Garments"};
int[] gstper=new int[]{20,13,5};
String name="";
System.out.println("\nEnter The Name");
line=in.readUTF();
System.out.println(line);
name=line;
for(int i=0;i<3;i++)
{
System.out.println(list[i]);
}
line=in.readUTF();
System.out.println(line);
if(line.equals("a"))
{
n=0;
}
else if(line.equals('b'))
{
n=1;
}
else
{
n=2;
}
System.out.println(n);
System.out.println("\nEnter the amount");
line=in.readUTF();
money=10000;
System.out.println(money);
l=10;
```

```
if(n==0)
{
a=gst(l,money);
System.out.println(a);
line="Over";
}
if(n==1)
{
a=gst(l,money);
System.out.println(a);
line="Over";
}
if(n==2)
{
a=gst(l,money);
System.out.println(a);
line="Over";
}

}
catch(IOException i)
{
System.out.println(i);
}
}
System.out.println("Closing connection");
socket.close();
in.close();
}
catch(IOException i)
{
System.out.println(i);
}
}

public static void main(String args[])
{
```

```
Server server = new Server(4000);
}
}
```

SCREEN SHOTS:

```
--------------------Configu
Server started
Waiting for a client ...
Client accepted

Enter The Name
yajith
a.Electronics
b.Furnitures
c.Garments

2

Enter the amount
10000
1000
Closing connection

Process completed.
```

```
-------------.
Connected
yajith
a
10000
```

ALGORITHM:

UDP

Client

1. Start
2. Import necessary java packages
3. In Clientu class Create objects for InetAddress,DatagramSocket and DatagramPacket.
4. Create byte array to send and receive the messages.
5. Get IP address with InetAddress.getByName()method.
6. Initialise DatagramSocket object .
7. Get the choice and string from user.
8. Convert user inputs to byte array by input.getBytes() method.
9. Send Choice and string to the server by the DatagramPacket object, DatagramSocket object with send(array,array length,IP address,port number) method.
10. If choice equals to one then
    i. Initialise DatagramPacket object for receiving packet.
    ii. It receives the packet from server by DatagramPacket and DatagramSocket object with object.receive(receiving object).
    iii. Get the Name of product and total amount with the help of object.getData() method.
    iv. Convert the gst and display it
11. If choice equals to two then ,
    i. Initialise DatagramPacket object for receiving packet.
    ii. It receives the packet from server by DatagramPacket and DatagramSocket object with object.receive(receiving object).
    iii. Get the product name and amount with the help of object.getData() method and display it.
12. If choice equals to three then,
    i. Get the product name and total amount of user.
    ii. Convert it into byte array and send to the server.
13. If choice equals to four then display the gst and total amount to be paid.
14. End

Server

1. Start
2. Import necessary java packages
3. In Serveru class Create objects for InetAddress,DatagramSocket and DatagramPacket.

4. Create byte array to send and receive the messages.
5. Initialise DatagramSocket object with Port number.
6. Receive the choice and string from client using receivepacket and receive () method.
7. If choice equals to one then
   i. find gst of the product with string.length() method
   ii. Convert it into string and then byte array
   iii. Send to the client by DatagramSocket and DatagramPacket object with send() method.
8. If choice equals to two then Convert it into byte array and send to the client.
9.If choice equals to three then,
   i. Receive the starting and end index.
   ii. Convert it into gst for the product
   iii. Convert the substring into byte array and send to the client.
10. If choice equals to four then total amount is displayed and process ends
11.Close Connection
12.End

CODING:

Client

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;
public class ClientEcho
{
public static void main(String args[]) throws IOException
{
Scanner sc = new Scanner(System.in);
DatagramSocket ds = new DatagramSocket();
InetAddress ip = InetAddress.getLocalHost();
byte buf[] = null;
System.out.println("Enter the name\nEnter the amount");
System.out.print("1.Electronics,2.Garments,3.Furnitures\n");
while (true)
{
String inp = sc.nextLine();
```

```
// convert the String input into the byte array.
buf = inp.getBytes();

// Step 2 : Create the datagramPacket for sending
// the data.
DatagramPacket DpSend =
new DatagramPacket(buf, buf.length, ip, 1234);

// Step 3 : invoke the send call to actually send
// the data.
ds.send(DpSend);

// break the loop if user enters "bye"
if (inp.equals("bye"))
break;
}
}
}
```

Server

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;

public class ServerEcho
{
public static int gst(int amount,int per)
{
int result=(amount*per)/100;
return result;
}
public static void main(String[] args) throws IOException
{
// Step 1 : Create a socket to listen at port 1234
DatagramSocket ds = new DatagramSocket(1234);
byte[] receive = new byte[65535];
```

```
byte[] receive1 = new byte[65535];
int[] per=new int[]{10,5,14};
DatagramPacket DpReceive = null;
DatagramPacket DpReceive1 = null;
int p=1;
while (true)
{

// Step 2 : create a DatgramPacket to receive the data.
DpReceive = new DatagramPacket(receive, receive.length);
// Step 3 : revieve the data in byte buffer.
ds.receive(DpReceive);
String inp = new String(receive, 0, receive.length);
String[] arr=inp.split(" ",6);
System.out.println("Welcome "+arr[0]);
int amount=Integer.parseInt(arr[1].trim());
int sel=Integer.parseInt(arr[2].trim());
if(sel==1)
{
p=per[0];
}
else if(sel==2)
{
p=per[1];
}
else
{
p=per[2];
}
int gs=gst(amount,p);
System.out.print("Your gst is "+gs);

// Exit the server if the client sends "bye"
if (data(receive).toString().equals("bye"))
{
System.out.println("Client sent bye.....EXITING");
break;
}
```

```
// Clear the buffer after every message.
receive = new byte[65535];
}
}

// A utility method to convert the byte array
// data into a string representation.
public static StringBuilder data(byte[] a)
{
if (a == null)
return null;
StringBuilder ret = new StringBuilder();
int i = 0;
while (a[i] != 0)
{
ret.append((char) a[i]);
i++;
}
return ret;
}
}
```

SCREEN SHOTS:

```
----------------------Configuration: <Defau
Welcome yajith
Your gst is 1000
```

```
Enter the name
Enter the amount
1.Electronics,2.Garments,3.Furnitures
yajith 10000 1
```

RESULT:

Thus the programs for find the gst for different products are implemented in Java and the results are verified.

Evaluation:

Parameter	Max Marks	Marks Obtained
Complexity of the Application chosen	3	
Uniqueness of the Code	10	
Use of Comment lines and standard coding practices	2	
Viva	5	
Sub Total	20	
Completion of experiment on time	3	
Documentation	7	
Sub Total	10	
Signature of the faculty with Date		

Ex.No.2	**APPLICATION DEVELOPMENT USING REMOTE METHOD INVOCATION**

AIM:

To implement array application using Remote Method Invocation (RMI)in Java.

THEORY:

RMI - Definition:

Remote method invocation (RMI) is a distributed object technology developed by Sun for the Java programming language. It is available as part of the core Java application programming interface (API) where the object interfaces are defined as Java interfaces and use object serialization.RMI permits Java methods to refer to a remote object and invoke methods of the remote object. The remote object may reside on another Java virtual machine, the same host or on completely different hosts across the network. RMI marshals and unmarshals method arguments through object serialization and supports dynamic downloading of class files across networks.

Differences between RMI and RPC:

BASIS FOR COMPARISON	**RPC**	**RMI**
Supports	Procedural programming	Object-oriented programming
Parameters	Ordinary data structures are passed to remote procedures.	Objects are passed to remote methods.

BASIS FOR COMPARISON	RPC	RMI
Efficiency	Lower than RMI	More than RPC and supported by modern programming approach (i.e. Object-oriented paradigms)
Overheads	More	Less comparatively
In-out parameters are mandatory.	Yes	Not necessarily
Provision of ease of programming	High	low

Stubs:

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

1. It initiates a connection with remote Virtual Machine (JVM),
2. It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),
3. It waits for the result
4. It reads (unmarshals) the return value or exception, and
5. It finally, returns the value to the caller.

Skeletons:

The skeleton is an object, acts as a gateway for the serverside object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

1. It reads the parameter for the remote method
2. It invokes the method on the actual remote object, and
3. It writes and transmits (marshals) the result to the caller.

ALGORITHM:

1. Start
2. A file with name Adder1 is created which contains the method sum().
3. Another file AdderRemote1 is created that implements Adder1.
4. In AdderRemote1, we define the method sum().
5. We get the array(arr[]), it's size(n) and sum(sum) to be found as input from user.
6. We try to find the pivot element.
7. For i=0 to n-1
8. If(arr[i]>arr[i+1]), then break
9. Let r= index of larger element
10. Let l=index of smaller element in a array
11. While(l!=r)
12. If(arr[l]+arr[r]==sum), then return true
13. Or else if(arr[l]+arr[r]<sum), then increment l
14. Else increment r
15. In MyServer1, we create a stub that connects MyClient1.
16. In MyClient1, the input is received from user and passed as argument in sum().

Procedure for execution on same machine:

1. In one command prompt Adder1, AdderRemote1, MyServer1, MyClient1 is compiled.
2. The stub is created by rmic AdderRemote1 command.
3. The rmi is started by rmiregistry command with the port number.
4. Then the MyServer1 and MyClient1 should be run in two different prompts.

Procedure for execution on different machine:

1. In MyServer2 program the ip address of server machine should be given.
2. In server machine Adder1, AdderRemote1, MyServer2 is compiled.
3. The stub is created by rmic AdderRemote1 command.
4. The rmi is started by rmiregistry command.
5. MyServer2 run in server machine
6. In MyClient2 program the ip address of server machine should be given.
7. In client machine MyClient2 is compiled.
8. The stub of server should be present in client machine
9. MyClient2 run in client machine.

CODING:

Adder1:

```
import java.rmi.*;
public interface Adder1 extends Remote{
   public int add(int a[],int n)throws RemoteException;
}
```

AdderRemote1:

```
import java.rmi.*;
import java.rmi.server.*;
public class AdderRemote1 extends UnicastRemoteObject implements Adder1{
   AdderRemote1()throwsRemoteException{
super();
   }
public int add(int a[],int n){
int i, j, min_idx;
for (i = 0; i < n-1; i++)
{
min_idx = i;
for (j = i+1; j < n; j++)
{
if (a[j] < a[min_idx])
{
min_idx = j;
}
```

```
}
int temp = a[min_idx];
a[min_idx] = a[i];
a[i] = temp;
}
for(i=0;i<n;i++)
{
System.out.println(a[i]+"\n");
}
return n;
}
}
```

RMI on same machine

MyServer1:

```
import java.rmi.*;
import java.rmi.registry.*;
public class MyServer1{
   public static void main(String args[]){
try{
        //Establishes connection to client
        Adder1 stub=new AdderRemote1();
Naming.rebind("rmi://localhost:5000/sonoo",stub);
      }
catch(Exception e)
      {
System.out.println(e);
      }
   }
}
```

MyClient1:

```
import java.rmi.*;
import java.util.*;
public class MyClient1
{
   public static void main(String args[]){
//Adder stub=(Adder)Naming.lookup("rmi://localhost:5000/sonoo");
```

```
System.out.println("GOT");
Registry reg = LocateRegistry.getRegistry("172.17.18.66", 5000);
Adder stub = (Adder) reg.lookup("Stub");
int[] arr=new int[100];
int n;
Scanner s=new Scanner(System.in);
System.out.println("Enter how many elements");
n=s.nextInt();
System.out.println("Enter the elements");
for(int i=0;i<n;i++)
{
arr[i]=s.nextInt();
}
System.out.println(stub.add(a,n));
}catch(Exception e){
System.out.println( e);}
}
}
```

RMI on different machine

MyServer2:

```
import java.rmi.*;
import java.rmi.registry.*;
public class MyServer2
{
   public static void main(String args[])
   {
try{
System.setProperty("java.rmi.server.hostname", "192.168.0.6");
        Registry reg = LocateRegistry.createRegistry(82);
        Adder1 stub=new AdderRemote1();
reg.rebind("Stub", stub);
        }
catch(Exception e)
{
System.out.println(e);
}
```

```
   }
}
```

MyClient2:

```
import java.rmi.*;
import java.util.*;
public class MyClient2
{
   public static void main(String args[])
      {
Registry reg = LocateRegistry.getRegistry("192.168.0.6", 82);
Adder1 stub = (Adder1) reg.lookup("Stub");
int[] arr=new int[100];
int n;
Scanner s=new Scanner(System.in);
System.out.println("Enter how many elements");
n=s.nextInt();
System.out.println("Enter the elements");
for(int i=0;i<n;i++)
{
arr[i]=s.nextInt();
}
System.out.println(stub.add(a,n));
}catch(Exception e){
System.out.println( e);}
}
}
```

SCREEN SHOTS:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\acer>d:

D:\>javac Adder1.java

D:\>javac AdderRemote1.java

D:\>javac MyServer1.java

D:\>javac MyClient1.java

D:\>rmic AdderRemote1

D:\>rmiregistry 5000
```

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\acer>d:

D:\>java MyServer1
```

```
D:\>java MyClient1
enter the array:
9
6
3
1
2
enter the sum:
7

True- Sum found!!
False- Sum not found!!

Answer:
true

D:\>java MyClient1
enter the array:
8
6
7
3
5
enter the sum:
22

True- Sum found!!
False- Sum not found!!

Answer:
false
```

RESULT:

Thus the application for array application using Remote Method Invocation isimplemented in Java and the results are verified.

Evaluation:

Parameter	Max Marks	Marks Obtained
Complexity of the Application	10	
Clarity of Algorithm	3	
Use of Comment lines and standard coding practices	2	
Viva	5	
Sub Total	20	
Completion of experiment on time	3	
Documentation	7	
Sub Total	10	
Signature of the faculty with Date		

Ex.No.3 /Feb/2018	FILE TRANSFER APPLICATION – TCP & UDP

AIM:

To develop a file transfer application using TCP/UDP.

THEORY:

FTP:

The File Transfer Protocol is a standard network protocol used for the transfer of computer files between a client and server on a computer network. FTP is built on a client-server model architecture using separate control and data connections between the client and the server.

ALGORITHM:

FileServer.java

1. Create a package called Server.
2. Import all the necessary packages including jaa.net and java.io.
3. Create a public class and create instance for socket and assign the port.
4. Create the socket.accept() function for the client to accept the request.
5. Create the method to saveFile called savefile with client socket as the parameter.
6. Create instances for Data Input stream and File Output Stream respectively.
7. Implement counter mechanism to calculate the amount of bytes sent and decrement counter to identify the remaining bytes and print it.
8. Close the file stream and socket.

FileClient.java

1. Create a package Client.
2. Import all the necessary files and create a private Socket instance.
3. Create a constructor FileClient with host port and filename as parameters.
4. Connect to the socket using Socket() constructor

5. Create sendfile method and create instances for DataInputStream and FileOutputStream.
6. Implement buffer to verify the receival of data and retransmit if not received.
7. Print the message if received.
8. Close the file stream and socket.
9. End.

CODING:

FileServer.java

```
package server;

import java.io.DataInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
public class FileServer extends Thread {
   private ServerSocket ss;
   public FileServer(int port) {
      try {
         ss = new ServerSocket(port);
      } catch (IOException e) {
         e.printStackTrace();
      }
   }
   public void run() {
      while (true) {
         try {
            Socket clientSock = ss.accept();
            saveFile(clientSock);
         } catch (IOException e) {
            e.printStackTrace();
         }
      }
   }
```

```
    private void saveFile(Socket clientSock) throws IOException {
        DataInputStream dis = new DataInputStream(clientSock.getInputStream());
        FileOutputStream fos = new
FileOutputStream("E:\\18IT116\\FTP\\ send.pdf");
        byte[] buffer = new byte[4096];
        int filesize = 15123; // Send file size in separate msg
        int read = 0;
        int totalRead = 0;
        int remaining = filesize;
        while((read = dis.read(buffer, 0, Math.min(buffer.length, remaining))) > 0) {
            totalRead += read;
            remaining -= read;
            System.out.println("read " + totalRead + " bytes.");
            fos.write(buffer, 0, read);
        }
        fos.close();
        dis.close();
    }
    public static void main(String[] args) {
        FileServer fs = new FileServer(1988);
        fs.start();
    }
}
```

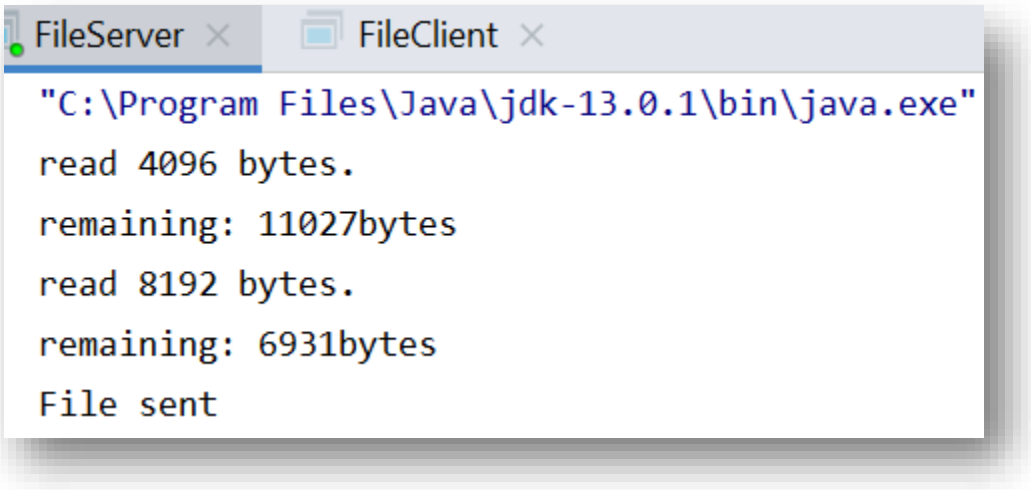
FileClient.java

```
package client;
import java.io.DataOutputStream;
import java.io.FileInputStream;
import java.io.IOException;
import java.net.Socket;
public class FileClient {
    private Socket s;
    public FileClient(String host, int port, String file) {
        try {
            s = new Socket(host, port);
            sendFile(file);
        } catch (Exception e) {
            e.printStackTrace();
```

```
    }
  }
  public void sendFile(String file) throws IOException {
    DataOutputStream dos = new DataOutputStream(s.getOutputStream());
    FileInputStream fis = new FileInputStream(file);
    byte[] buffer = new byte[4096];
    while (fis.read(buffer) > 0) {
      dos.write(buffer);
    }
    System.out.println("File sent...");
    fis.close();
    dos.close();
  }
  public static void main(String[] args) {


    FileClient fc = new FileClient("localhost", 1988,
"E:\\18IT116\\FTP\\send.pdf");
  }
}
```

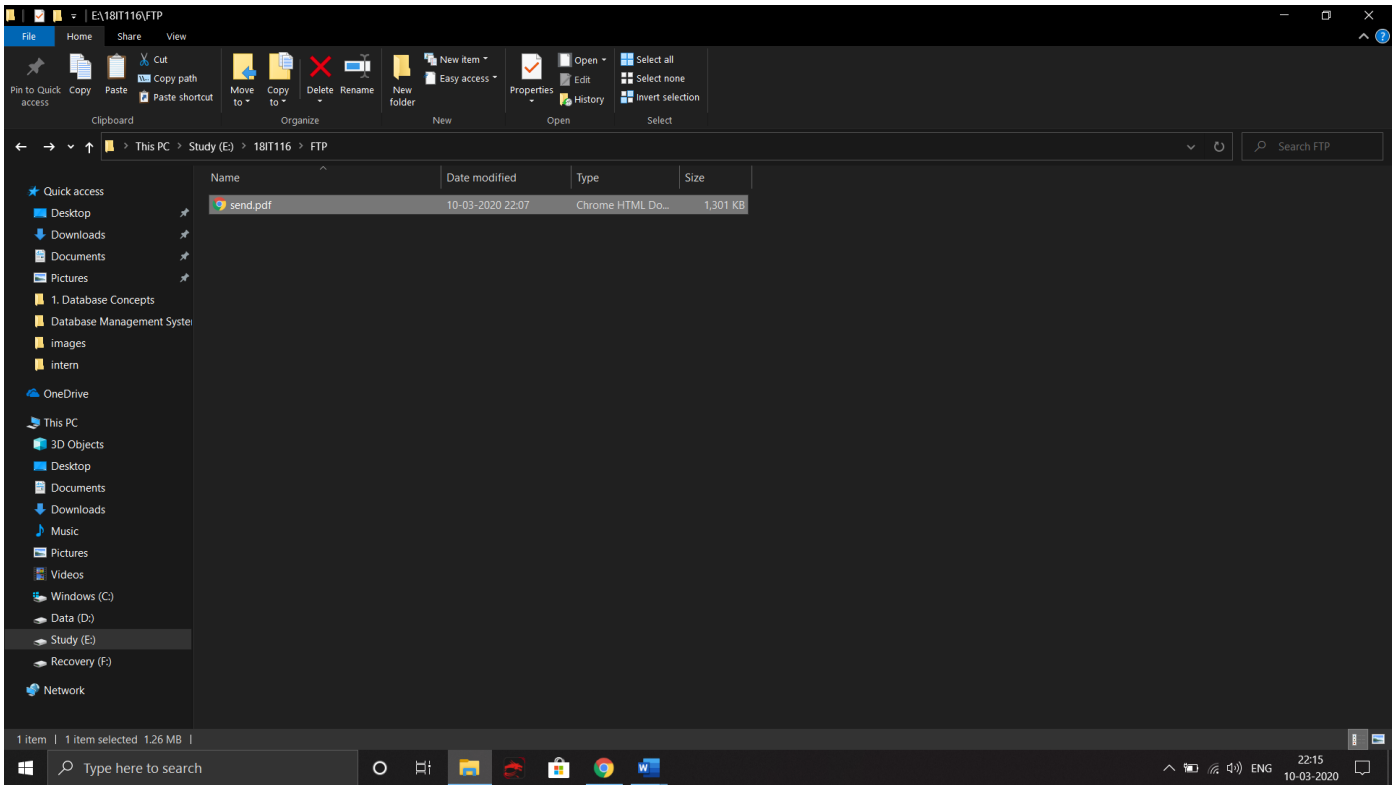
SCREEN SHOTS:



FileServer FileClient

```
"C:\Program Files\Java\jdk-13.0.1\bin\java.exe"
File received...

Process finished with exit code 0
```



Result:

Thus an application for the File transfer from the server to the client has been implemented and verified.

Evaluation:

Parameter	Max Marks	Marks Obtained
Complexity of the Application chosen	3	
Uniqueness of the Code	10	
Use of Comment lines and standard coding practices	2	
Viva	5	
Sub Total	20	
Completion of experiment on time	3	
Documentation	7	
Sub Total	10	
Signature of the faculty with Date		

Ex.No.4 09/FEB/2020	**IMPLEMENTATION OF ARP AND RARP**

AIM:

To implement

- Address Resolution Protocol (ARP)
- Reverse Address Resolution Protocol (RARP)

in Java.

THEORY:

ARP:

Address Resolution Protocol is a communication protocol used for discovering physical address associated with given network address. Typically, ARP is a network layer to data link layer mapping process, which is used to discover MAC address for given Internet Protocol Address.

Example:

Two computers in an office (C1 and C2) are connected to each other in a local area network by Ethernet cables and network switches, C1 has a packet to send to C2. Through DNS, it determines that C2 has the IP address some IP address. C1 has to send a broadcast ARP request message (Mac Address), which is accepted by all computers on the local network, requesting an answer for C2 IP Address

RARP :

Reverse ARP is a networking protocol used by a client machine in a local area network to request its Internet Protocol address (IPv4) from the gateway-router's ARP table. The network administrator creates a table in gateway-router, which is used to map the MAC address to corresponding IP address.

Example:

When a new machine is setup or any machine which don't have memory to store IP address, needs an IP address for its own use. So the machine sends a RARP broadcast packet which contains its own MAC address in both sender and receiver hardware address field.

ALGORITHM:

ARP:

1. Initialize the string of ip addresses
2. For every ip address, assign an ethernet address
3. To Perform ARP, enter the ip address
4. The equivalent ethernet address is displayedStep
5. If the ethernet address is not found, then “No Address Found” message is displayed

RARP :

1. To Perform RARP, enter the ethernet address
2. The equivalent ip address is displayed
3. If the ip address is not found, then “No Address Found “ message is displayed
4. Terminate the program and Print the message.

CODING:

ARP :

ARPCLIENT:

```
import java.io.*;

import java.net.*;

import java.util.*;

public class ARP {

   public static void main(String args[]){

   try{

      DatagramSocket client = new DatagramSocket();

      InetAddress addr = InetAddress.getByName("127.0.0.1");

      byte[] sendByte = new byte[1204];

      byte[] receiveByte = new byte[1024];
```

```
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter the Logical Address ");
    String str = in.readLine();
    sendByte = str.getBytes();
    DatagramPacket sender = new DatagramPacket(sendByte,sendByte.length,addr,1309);
    client.send(sender);
    DatagramPacket receiver = new DatagramPacket(receiveByte,receiveByte.length);
    client.receive(receiver);
    String s = new String(receiver.getData());
    System.out.println("The Physical Address is :" + s.trim());
    client.close();
  }
  catch(Exception e){
  System.out.println(e);
  }
  }
}
```

ARPSERVER :

```
import java.io.*;
import java.net.*;
import java.util.*;
public class ARPS{
  public static void main(String args[])  {
  try{
```

```
   DatagramSocket server = new DatagramSocket(1309);
      while(true){
            byte[] sendByte = new byte[1204];
            byte[] receiveByte = new byte[1204];
            DatagramPacket receiver = new DatagramPacket(receiveByte,receiveByte.length);
            server.receive(receiver);
            String str = new String(receiver.getData());
            String s = str.trim();
            InetAddress addr = receiver.getAddress();
            int port = receiver.getPort();
            String ip[] =
{"10.0.3.186","10.0.3.187","10.0.3.188","10.0.3.189","10.0.3.190","192.28.11.1"};
            String mac[] =
{"D4:3D:7E:12:A3:D9","00:1B:44:11:3A:B7","00:A0:C9:14:C8:29","B3:2C:00:4E:D2:A0","3D:7E:12:
C9:14:C8","Y0:G1:TH:GT:28:N0"};
for (int i = 0; i < ip.length; i++) {
if(s.equals(ip[i]))
{  sendByte = mac[i].getBytes();
    DatagramPacket sender = new DatagramPacket(sendByte,sendByte.length,addr,port);
server.send(sender);
break;}
 break;
      }
   }
catch(Exception e)
   {
```

```
      System.out.println(e);
   }
   }
}
```

RARP :

RARP_SERVER:

```
import java.io.*;
import java.net.*;
import java.util.*;
public class RARPServer{
   public static void main(String args[])    {
   try{
      DatagramSocket server = new DatagramSocket(1309);
      while(true){
            byte[] sendByte = new byte[1204];
            byte[] receiveByte = new byte[1204];
            DatagramPacket receiver = new
DatagramPacket(receiveByte,receiveByte.length);
            server.receive(receiver);
            String str = new String(receiver.getData());
            String s = str.trim();
            InetAddress addr = receiver.getAddress();
            int port = receiver.getPort();
String ip[] = {"10.0.3.186","10.0.3.187","10.0.3.188","10.0.3.189","10.0.3.190","192.28.11.1"};
```

```
String mac[] =
{"D4:3D:7E:12:A3:D9","00:1B:44:11:3A:B7","00:A0:C9:14:C8:29","B3:2C:00:4E:D2:A0","3D:7E:12:
C9:14:C8","Y0:G1:TH:GT:28:N0"};

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

 if(s.equals(mac[i])

{

sendByte = ip[i].getBytes();

DatagramPacket sender = new DatagramPacket(sendByte,sendByte.length,addr,port);

server.send(sender);

break;                                                                    }       }

      break;

      }

   }catch(Exception e)

   {

      System.out.println(e);

   }}

}
```

RARP_CLIENT:

```
import java.io.*;

import java.net.*;

import java.util.*;

public class RARP {

   public static void main(String args[]){

   try{

      DatagramSocket client = new DatagramSocket();
```

```
    InetAddress addr = InetAddress.getByName("127.0.0.1");
    byte[] sendByte = new byte[1204];
    byte[] receiveByte = new byte[1024];
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter the Physical Address ");
    String str = in.readLine();
    sendByte = str.getBytes();
    DatagramPacket sender = new DatagramPacket(sendByte,sendByte.length,addr,1309);
    client.send(sender);
    DatagramPacket receiver = new DatagramPacket(receiveByte,receiveByte.length);
    client.receive(receiver);
    String s = new String(receiver.getData());
    System.out.println("The Logical Address is :" + s.trim());
    client.close();
  }
  catch(Exception e){
  System.out.println(e);
  }
  }
}
```

SCREEN SHOTS:

```
Enter the Logical Address
192.28.11.1
The Physical Address is :Y0:G1:TH:GT:28:N0
PS C:\Users\yogit\OneDrive\Desktop\College\Sem - 4\Computer Networks\Lab\Ex-4> java RARPClient
Enter the Physical Address
Y0:G1:TH:GT28:N0
```

```
PS C:\Users\yogit\OneDrive\Desktop\College\Sem - 4\Computer Networks\Lab\Ex-4> java ARPClient
Enter the Logical Address
192.28.11.1
The Physical Address is :Y0:G1:TH:GT:28:N0
PS C:\Users\yogit\OneDrive\Desktop\College\Sem - 4\Computer Networks\Lab\Ex-4>
```

RESULT:

Thus the programs for

- Address Resolution Protocol (ARP)
- Reverse Address Resolution Protocol (RARP)

are implemented in Java and the results are verified.

Evaluation:

Parameter	Max Marks	Marks Obtained
Uniqueness of the Code	10	
Clarity of Algorithm	3	
Use of Comment lines and standard coding practices	2	
Viva	5	
Sub Total	20	
Completion of experiment on time	3	
Documentation	7	
Sub Total	10	
Signature of the faculty with Date		

Ex.No.5 28/FEB/2020	PACKET CAPTURE AND ANALYSIS USING WIRE SHARK

AIM:

To demonstrate

1. Live packet capture and analysis
2. Perform password sniffing

using wire shark.

THEORY:

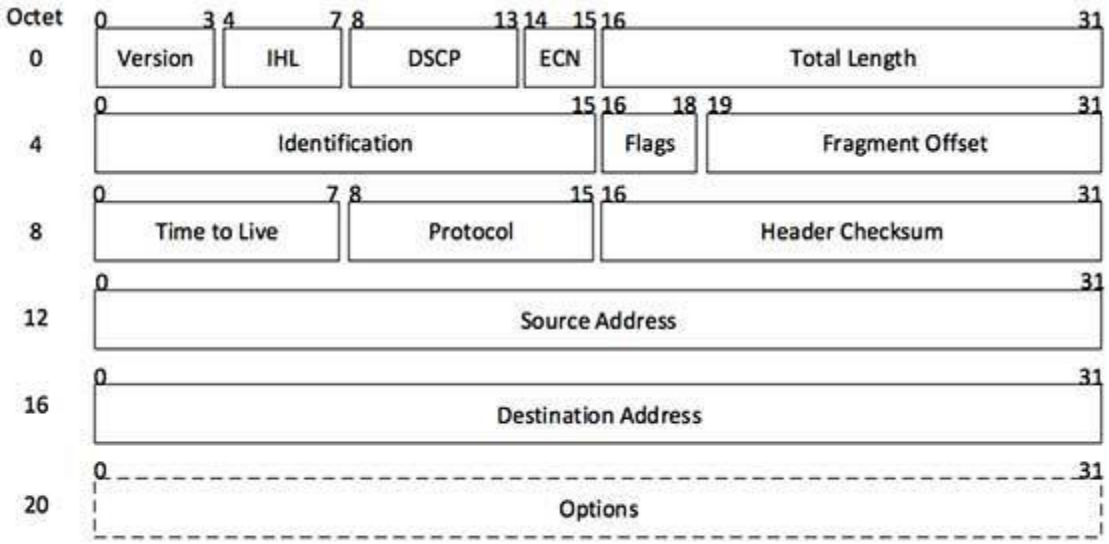
WireShark:

Wireshark is the world's foremost and widely open-source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education.

Frame format of Internet Protocol:

Internet Protocol being a layer-3 protocol (OSI) takes data Segments from layer-4 (Transport) and divides it into packets. IP packet encapsulates data unit received from above layer and add to its own header information.

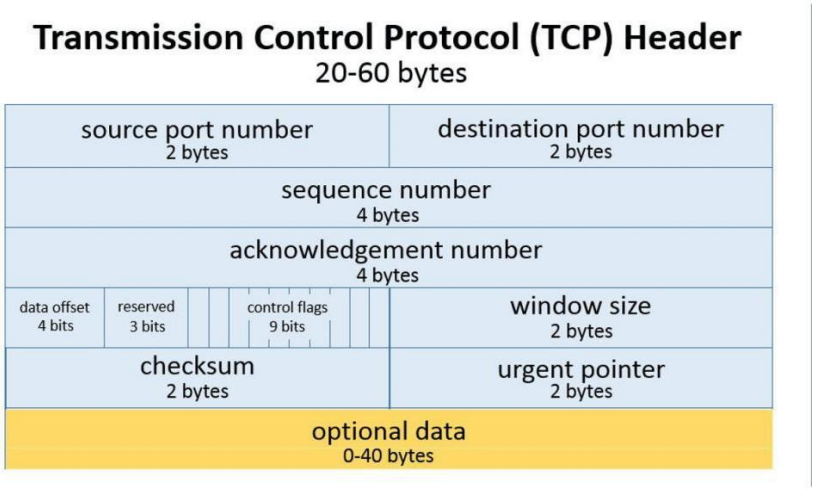
(IP Encapsulation)



[Image: IP Header]

Frame format of transmission control protocol

Transmission Control Protocol is a **transport** layer **protocol**. It continuously receives data from the application layer. It divides the data into chunks where each chunk is a collection of bytes. It then creates **TCP** segments by adding a **TCP header** to the data chunks

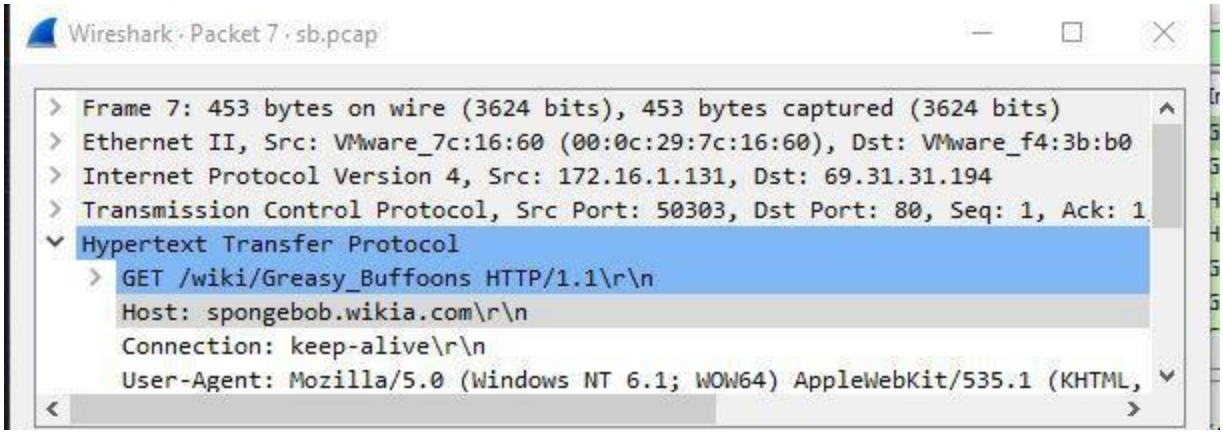


Password Sniffing

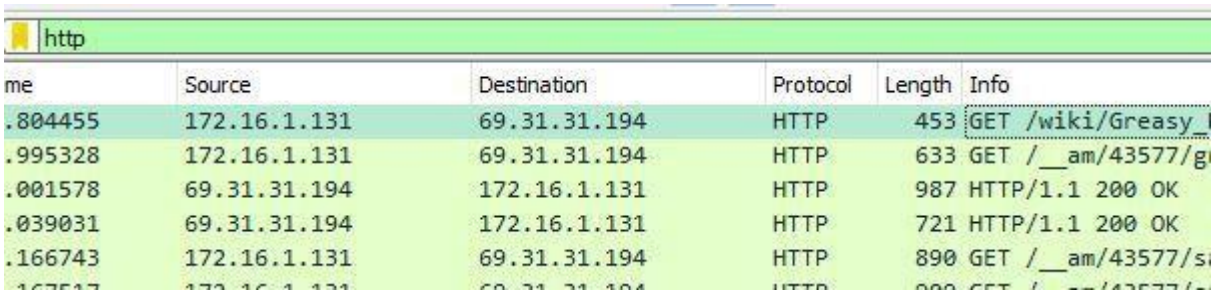
Password sniffing is a technique used to gain knowledge of passwords that involves monitoring traffic on a network to pull out information.

Part –A

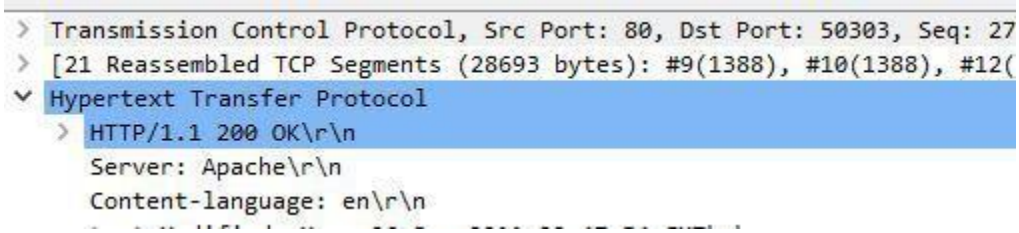
1. What webpage was visited in the above 2 packets? Webpage: https://spongebob.wikia.com



2. What version of HTTP was used? Version: 1.1

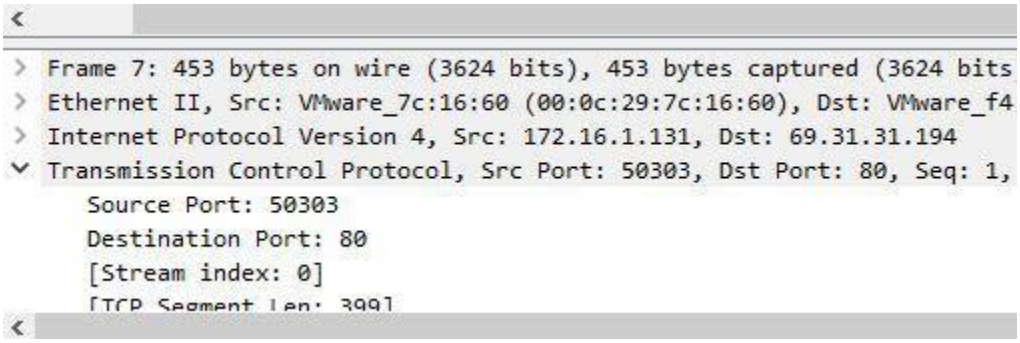


3. What is the destination IP address in the above packets? Destination: 172.16.1.131



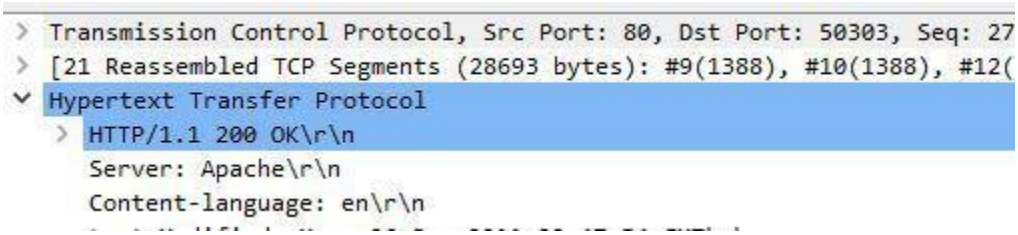
4. List the source and destination ports of the packets travelling from the client to the serverin the above packets?

Source: 80 Destination: 50317



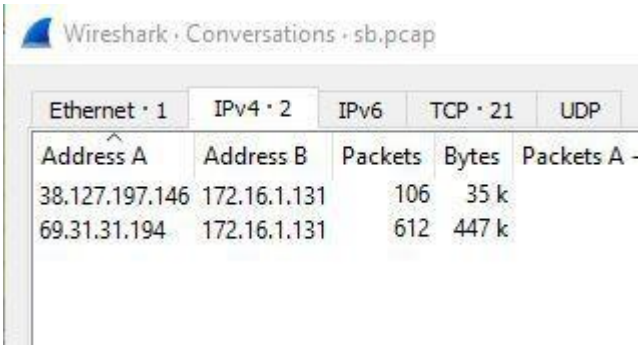
5. In the HTTP server's response, look at the information sent about the server. What server software was used?

Software: Apache



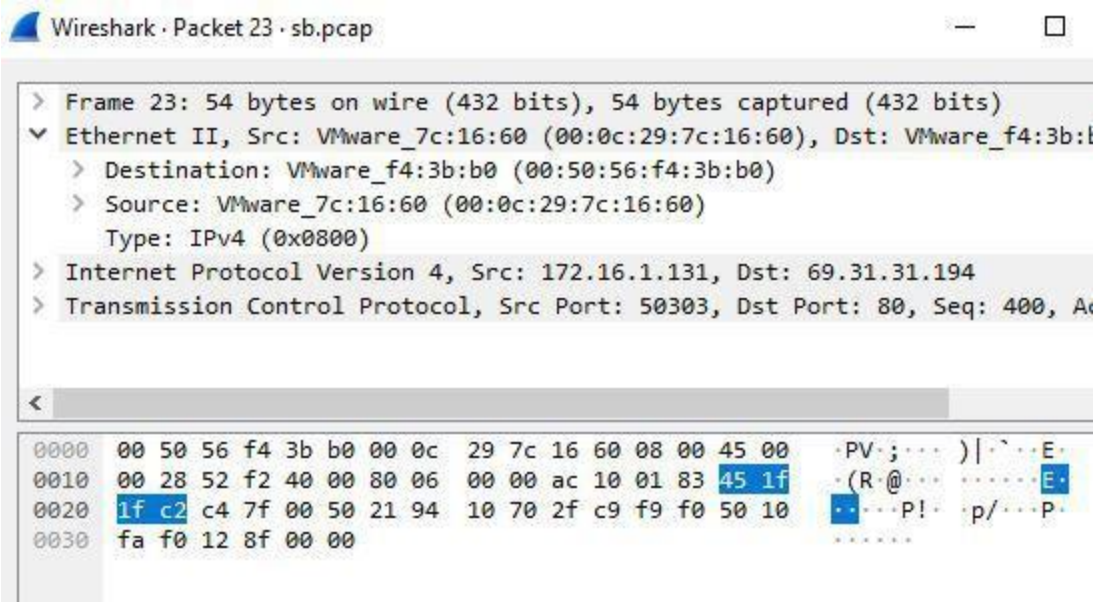
6. What are the IP addresses of the server? a. 38.127.197.146

b. 69.31.31.194



7. What are the MAC addresses of the client and server? Client: 00:0c:29:7c:16:60

Server: 00:50:56:f4:3b:b0



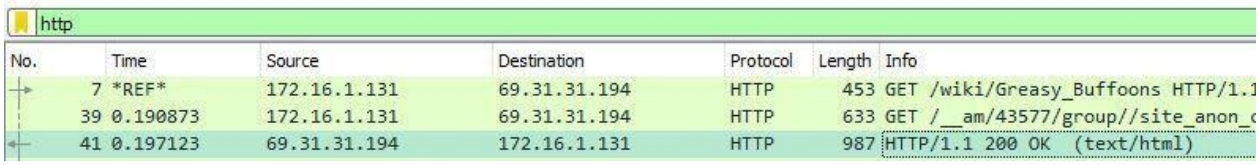
8. Identify the first 2 packets (i.e. their packet numbers) containing HTTP GET request. Ans: 649,626

9. How many webpages (not websites) have been opened?

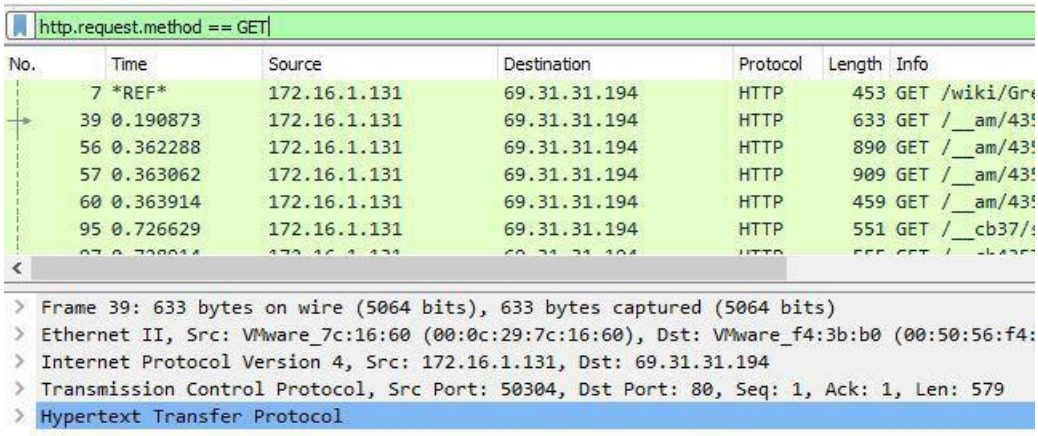
Count: 51

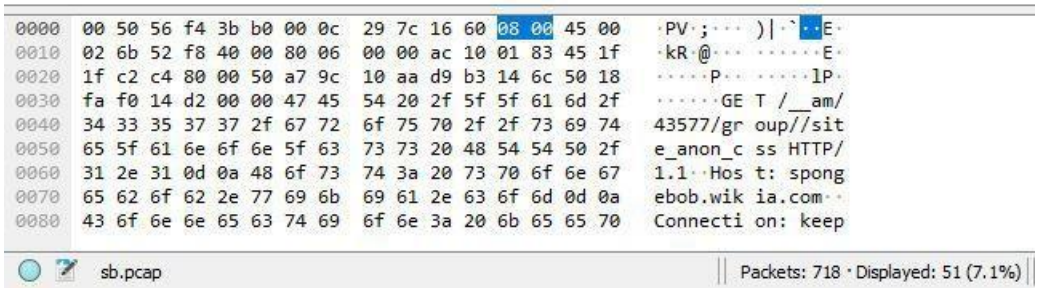
10. What is the time difference between first HTTP GET and the first HTTP

response (OK)? Time Difference: 0.197123



11. Count the total number of HTTP GET requests. Length: 718





12. What is the time difference between the first and last HTTP GET requests? Time Difference: 13.442726

http

No.	Time	Source	Destina
620	10.384495	172.16.1.131	69.31
623	10.524045	69.31.31.194	172.1
626	10.630086	172.16.1.131	69.31
632	10.903869	69.31.31.194	172.1
649	13.349456	172.16.1.131	69.31
652	13.442726	69.31.31.194	172.1

13. How many packets were exchanged between the server (corresponding to the both IP addresses) and the client? (Note: Their sum must be equal to the total no. of packets)

Total: 718 Packets.

Ethernet · 1 | IPv4 · 2 | IPv6 | TCP · 21 | UDP

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A
38.127.197.146	172.16.1.131	106	35 k	55	24 k	51	10 k
69.31.31.194	172.16.1.131	612	447 k	381	377 k	231	69 k

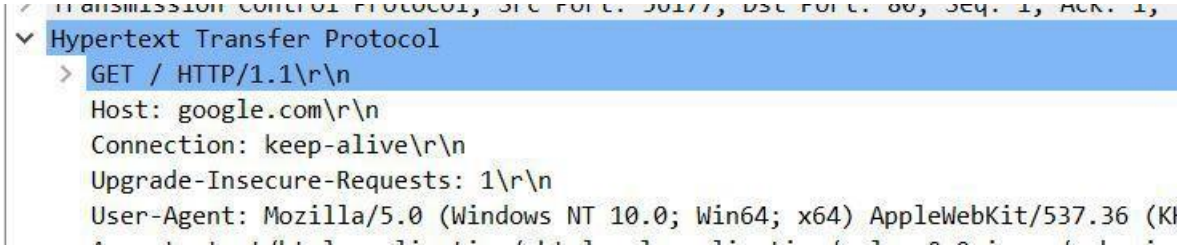
14. Find the total no. of HTTP requests sent by the host spongebob.wikia.com. Total: 25

HTTP Requests by HTTP Host	51
spongebob.wikia.com	25

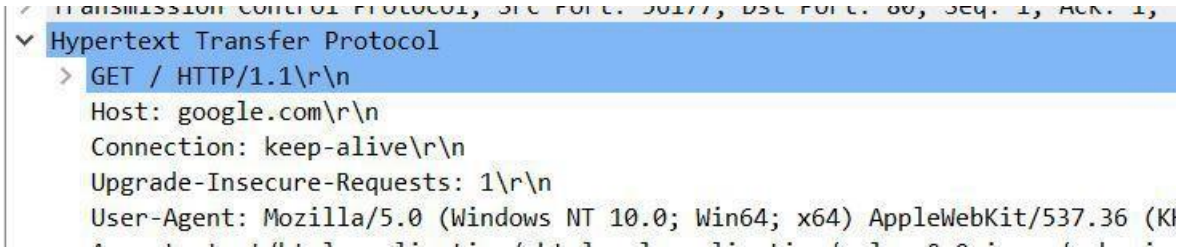
PART – B

1. What webpage was visited in the above 2 packets?

   Ans: www.google.com



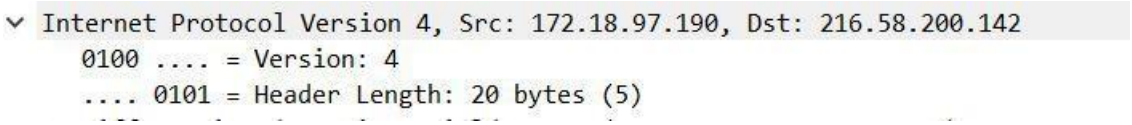
2. What version of HTTP was used?



Http version: 1.1

3. What is the destination IP address in the above packets?

Ans: 216.58.200.142

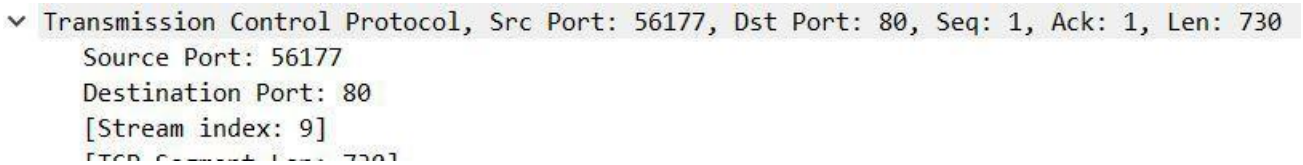


4. List the source and destination ports of the packets travelling from the client to the serverin the above packets?

Ans: Source port:

56177 Destination

port: 80



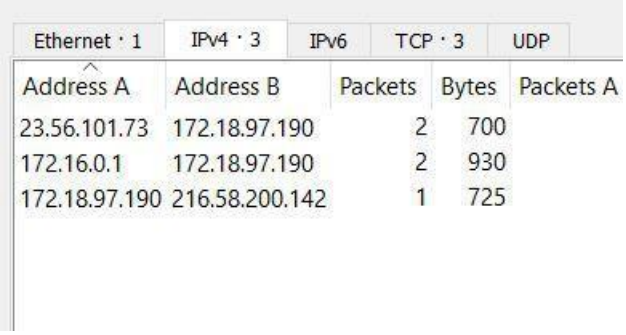
5. In the HTTP server's response, look at the information sent about the server. What server software was used?

Ans: gws

```
Hypertext Transfer Protocol
  HTTP/1.1 301 Moved Permanently\r\n
  Location: http://www.google.com/\r\n
  Content-Type: text/html; charset=UTF-8\r\n
  Date: Fri, 28 Feb 2020 04:31:41 GMT\r\n
  Expires: Sun, 29 Mar 2020 04:31:41 GMT\r\n
  Cache-Control: public, max-age=2592000\r\n
  Server: gws\r\n
```

6. What are the IP addresses of the server?

   Ans: 23.56.101.73 and 172.18.98.190



Ethernet · 1 | IPv4 · 3 | IPv6 | TCP · 3 | UDP

Address A	Address B	Packets	Bytes	Packets A
23.56.101.73	172.18.97.190	2	700	
172.16.0.1	172.18.97.190	2	930	
172.18.97.190	216.58.200.142	1	725	

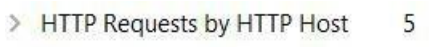
7. What are the MAC addresses of the client and server?

   Ans: Src: 98:f2:b3:8f:b8:79 and Dst: d8:9c:67:3b:9c:27

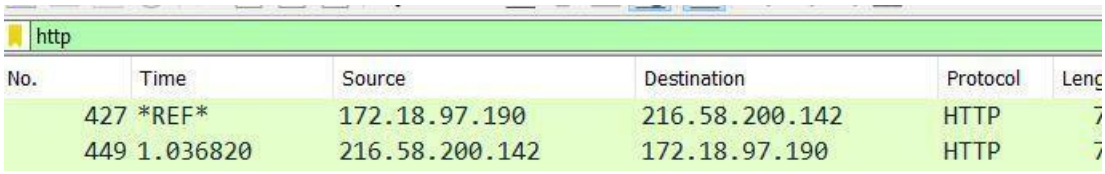
```
Ethernet II, Src: HonHaiPr_3b:9c:27 (d8:9c:67:3b:9c:27), Dst: HewlettP_8f:b8:79 (98:f2:b3:8f
  Destination: HewlettP_8f:b8:79 (98:f2:b3:8f:b8:79)
    Address: HewlettP_8f:b8:79 (98:f2:b3:8f:b8:79)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
    .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
  Source: HonHaiPr_3b:9c:27 (d8:9c:67:3b:9c:27)
    Address: HonHaiPr_3b:9c:27 (d8:9c:67:3b:9c:27)
    .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
           0                       = IG bit: Individual address (unicast)
```

8. Identify the first 2 packets (i.e. their packet numbers) containing HTTP GET request. Ans: 427,903

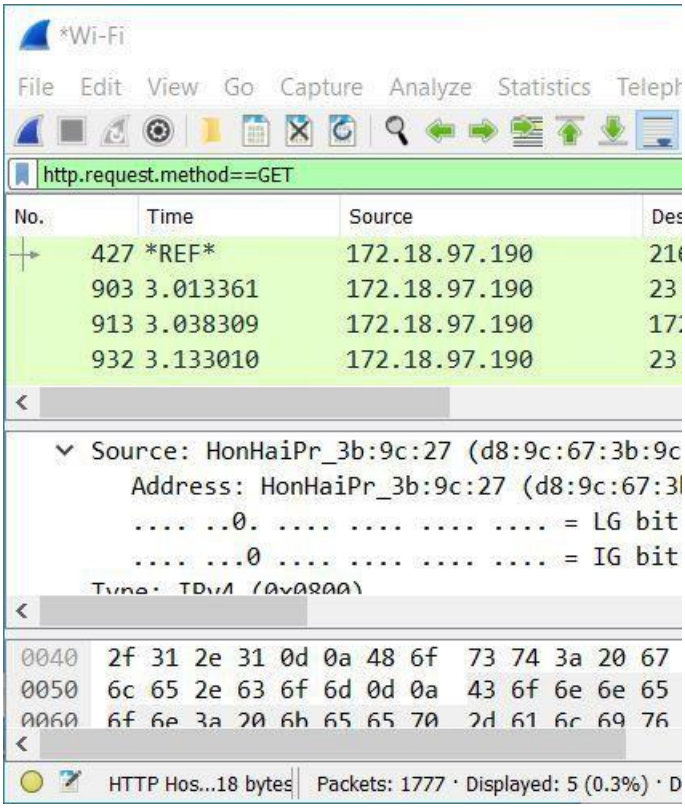
9. How many webpages (not websites) have been opened? Ans: 5



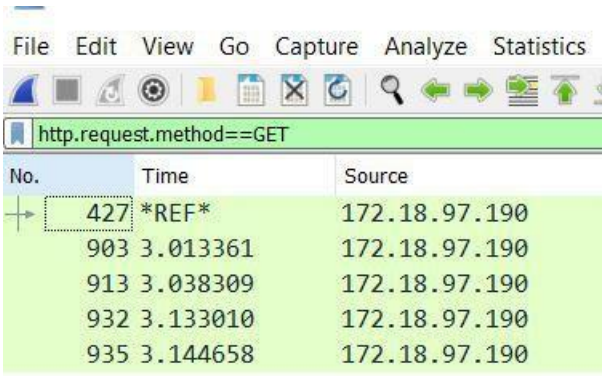
10. What is the time difference between first HTTP GET and the first HTTP response (OK)? Ans: 1.03



11. Count the total number of HTTP GET requests. Ans: 1777

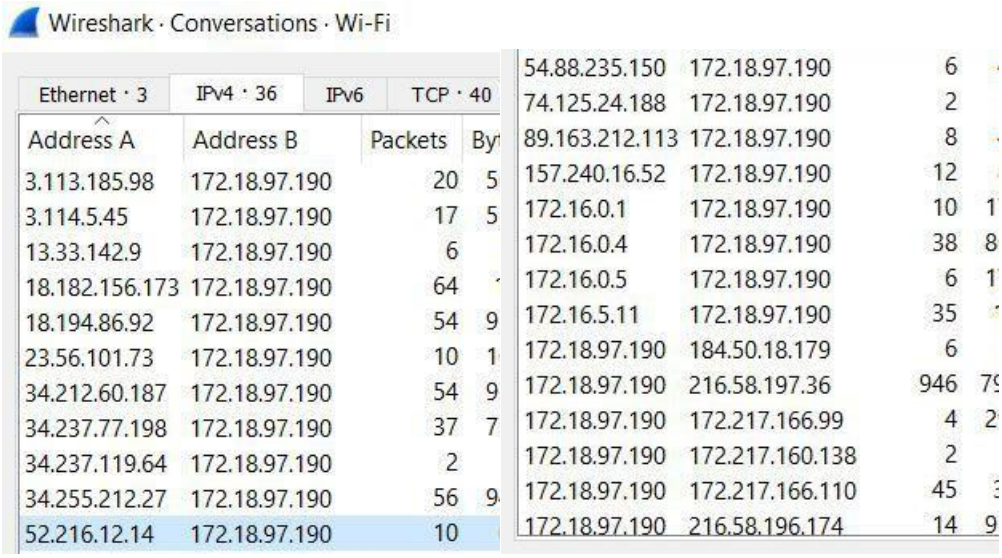


12. What is the time difference between the first and last HTTP GET requests? Ans: 3.144 seconds



13. How many packets were exchanged between the server (corresponding to the both IP addresses) and the client? (Note: Their sum must be equal to the total no. of packets)

Ans: 1777packets



14. Find the total no. of HTTP requests sent by the host google.com and total HTTP requests count Ans: 8

Values in the field of Internet Protocol:

```
Internet Protocol Version 4, Src: 172.18.97.190, Dst: 216.58.200.142
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 770
    Identification: 0x0d02 (3330)
  > Flags: 0x4000, Don't fragment
    ...0 0000 0000 0000 = Fragment offset: 0
    Time to live: 128
    Protocol: TCP (6)
    Header checksum: 0x3c5a [validation disabled]
    [Header checksum status: Unverified]
    Source: 172.18.97.190
    Destination: 216.58.200.142
```

Values of Transmission Control Protocol

```
Transmission Control Protocol, Src Port: 56177, Dst Port: 80, Seq: 1, Ack: 1, Len: 730
    Source Port: 56177
    Destination Port: 80
    [Stream index: 9]
    [TCP Segment Len: 730]
    Sequence number: 1    (relative sequence number)
    Sequence number (raw): 1962586332
    [Next sequence number: 731    (relative sequence number)]
    Acknowledgment number: 1    (relative ack number)
    Acknowledgment number (raw): 503598829
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    Window size value: 256
    [Calculated window size: 65536]
    [Window size scaling factor: 256]
    Checksum: 0x80ed [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (730 bytes)
```

PART – C

What is network sniffing?

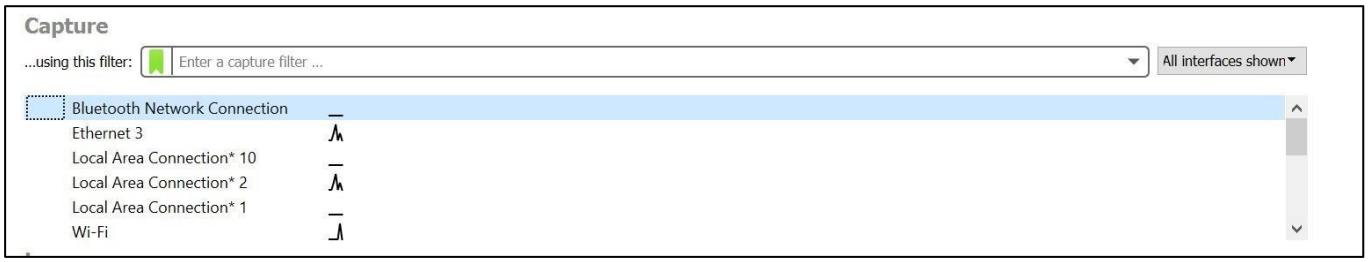
Computers communicate by broadcasting messages on a network using IP addresses. Once a message has been sent on a network, the recipient computer with the matching IP address responds with its MAC address.

Network sniffing is the process of intercepting data packets sent over a network. This can be done by the specialized software program or hardware equipment. Sniffing can be used to;

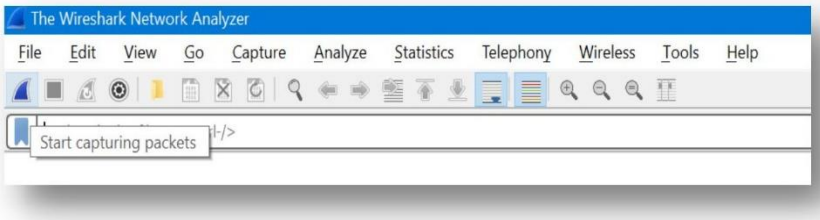
- Capture sensitive data such as login credentials
- Eavesdrop on chat messages
- Capture files have been transmitted over a network

Steps Involved:

Step 1. Open Wireshark



Step 1. Select the network interface you want to sniff. Step 2. Click on start button as shown

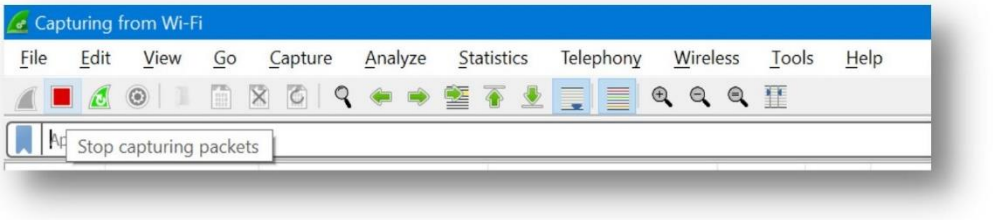


Step 3. Open a website of your choice (consider selecting http website

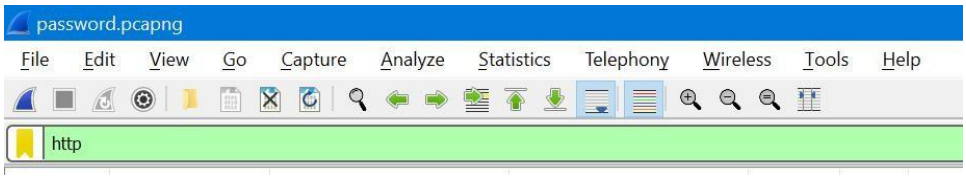
with login facilities) example:
http://aakashveera.herokuapp.com/campgrounds

Step 4. A successful login shall be required

Step 5. Go back to Wireshark and stop the live capture

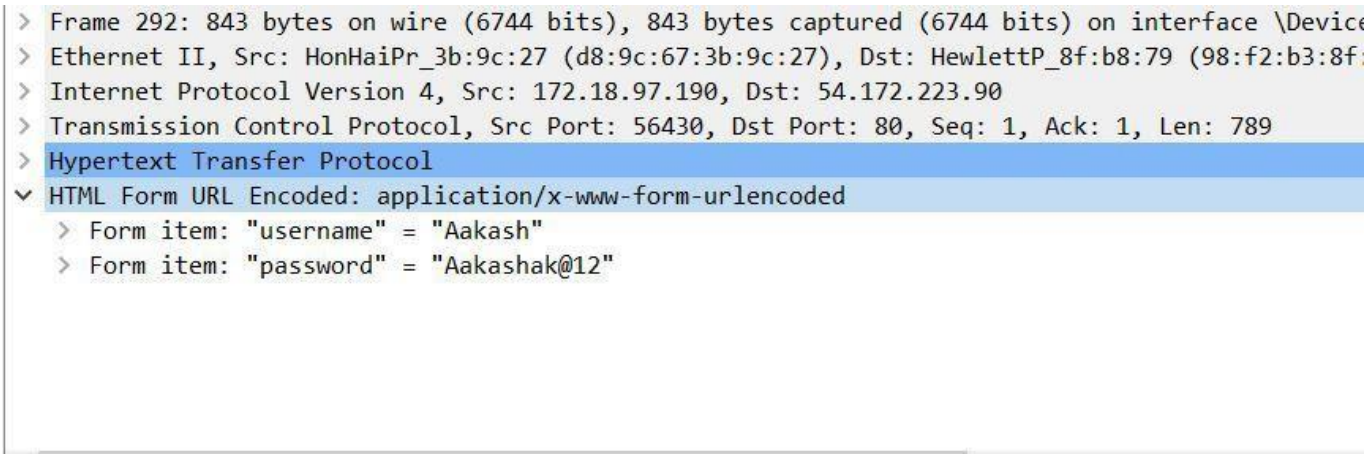


Step 6. Filter for HTTP protocol results only using the filter textbox



Step 7. Locate the Info column and look for entries with the HTTP verb POST and click on it

Step 8. Just below the log entries, there is a panel with a summary of captured data. Look for the summary that says Line-based text data. You should be able to view the plaintext values of all the POST variables submitted to the server via HTTP protocol. You'll be able to view the username and password as shown below.



RESULT:

Thus the live capture and analysis of packet and Password Sniffing has been performed using wire shark

Evaluation:

Parameter	Max Marks	Marks Obtained
Uniqueness of the Code	10	
Clarity of Algorithm	3	
Use of Comment lines and standard coding practices	2	
Viva	5	
Sub Total	20	
Completion of experiment on time	3	
Documentation	7	
Sub Total	10	

Ex.No.6 Date:27/2/2020	**STOP AND WAIT ARQ PROTOCOL**

AIM

To implement Stop and Wait ARQ Protocol using TCP in Java.

THEORY

Stop and Wait ARQ Protocol

Stop-and-wait ARQ, also referred to as alternating bit protocol, is a method in telecommunications to send information between two connected devices. It ensures that information is not lost due to dropped packets and that packets are received in the correct order.

ALGORITHM:

Sender

Step1: sequence <- 0

Step2: Accept new packet and assign sequence to it.

Step3: Send packet sequence with sequence number sequence.

Step4: Set timer for recently sent packets.

Step5: If error free acknowledgment from receiver and NextFrameExpected -> sequence then sequence<- NextFrameExpected.

Step6: If time out then go to step3.

Step7: Stop.

Receiver

Step1: Start.

Step2: NextFrameExpected<-0, repeat steps 3 forever.

Step3: If error-free frame received and sequence= NextFrameExpected, then pass packet to higher layer and NextFrameExpected<- NextFrameExpected+1(modulo 2).

Step4: Stop.

CODING:

Sender.java

```
import java.io.*;
import java.net.*;
public class Sender{
   // declaring the Socket
   Socket sender;
ObjectOutputStream out;
ObjectInputStream in;
   String packet,ack,str, msg;
   int n,i=0,sequence=0;
   Sender(){}
   public void run(){
      try{
BufferedReaderbr=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Waiting for Connection....");
         // getting the connection
```

```
        sender = new Socket("localhost",2005);
         //defining the sequence
        sequence=0;
        out=new ObjectOutputStream(sender.getOutputStream());
out.flush();
        in=new ObjectInputStream(sender.getInputStream());
        str=(String)in.readObject();
System.out.println("receiver > "+str);
System.out.println("Enter the data to send....");
        packet=br.readLine();
        n=packet.length();
        do{
          try{
            if(i<n){
                 // getiing message
              msg=String.valueOf(sequence);
              msg=msg.concat(packet.substring(i,i+1));
            }else if(i==n){
                 //sending message
              msg="end";out.writeObject(msg);break;
            }out.writeObject(msg);
            sequence=(sequence==0)?1:0;
out.flush();
System.out.println("Data Sent>"+msg);
            ack=(String)in.readObject();
```

```
System.out.println("Waiting for ack.....");
            if(ack.equals(String.valueOf(sequence))){
i++;
System.out.println("receiver   >  "+" packet received");
            }else{
                // Handling timeout
System.out.println("Time out resending data....");
              sequence=(sequence==0)?1:0;
            }}catch(Exception e){}
        }while(i<n+1);
System.out.println("All data sent. Exiting.");
      }catch(Exception e){}
      finally{
        try{
in.close();
out.close();
sender.close();
        }
        catch(Exception e){}
      }}
   public static void main(String args[]){
      Sender s=new Sender();
s.run();
   }}
```

Receiver.java

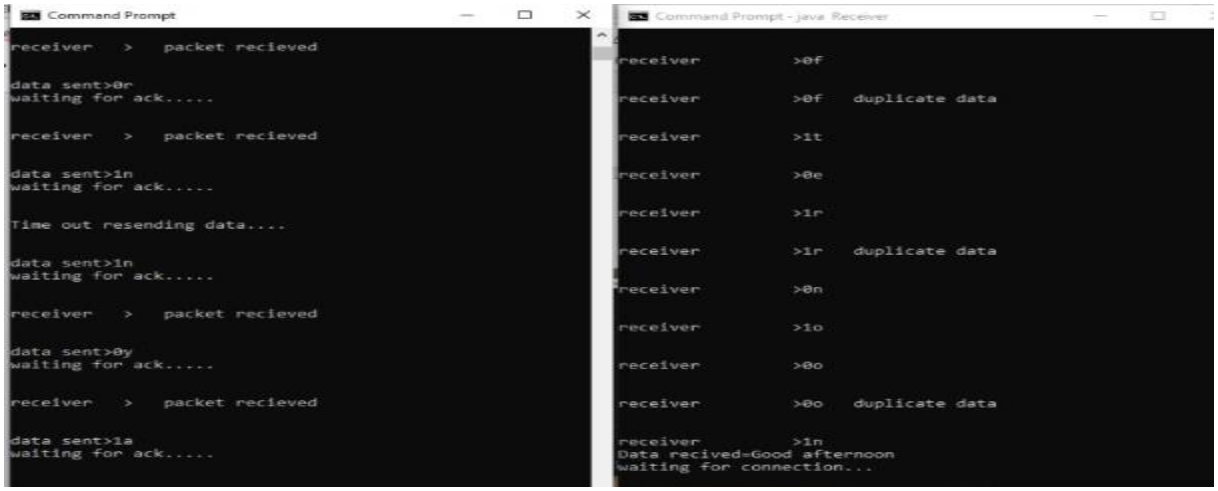
```
import java.io.*;
import java.net.*;
public class Receiver{
//declaring Socket
ServerSocketreciever;
   Socket connection=null;
ObjectOutputStream out;
ObjectInputStream in;
   String packet,ack,data="";
   int i=0,sequence=0;
   Receiver(){}
   public void run(){
      try{
BufferedReaderbr=new BufferedReader(new InputStreamReader(System.in));
reciever = new ServerSocket(2005,10);
System.out.println("Waiting for connection...");
         connection=reciever.accept();
          //defining the sequence
         sequence=0;
System.out.println("Connection established:");
         out=new ObjectOutputStream(connection.getOutputStream());
out.flush();
         in=new ObjectInputStream(connection.getInputStream());
out.writeObject("Connected.");
```

```
        do{
          try{
            packet=(String)in.readObject();
            if(Integer.valueOf(packet.substring(0,1))==sequence){
              data+=packet.substring(1);
                //parsing the sequence
              sequence=(sequence==0)?1:0;
System.out.println("receiver  >"+" "+packet);
            }
            else
            {
System.out.println("receiver  >"+packet +"  duplicate data");
            }if(i<3){
out.writeObject(String.valueOf(sequence));i++;
            }else{
out.writeObject(String.valueOf((sequence+1)%2));
i=0;
            }}
          catch(Exception e){}
        }while(!packet.equals("end"));
System.out.println("Data received = "+data);
out.writeObject("Closing Connection.");
      }catch(Exception e){}
      finally{
        try{in.close();
```

```
out.close();
reciever.close();
        }
        catch(Exception e){}
    }}

  public static void main(String args[]){
    Receiver s=new Receiver();
    while(true){
s.run();
    }
  }
}
```

SCREEN SHOTS:



RESULT:

Thus Stop and Wait ARQ Protocol has been implemented using TCP in Java.

Evaluation:

Parameter	Max Marks	Marks Obtained
	3	
Uniqueness of the Code	10	
Use of Comment lines and standard coding practices	2	
Viva	5	
Sub Total	20	
Completion of experiment on time	3	
Documentation	7	
Sub Total	10	
Signature of the faculty with Date		

Ex.No.7	**BASICS OF NETWORK CONFIGURATION USING OPNET 17.5(NETWORK DESIGN,ICMP PING)**

AIM

To implement Stop and Wait ARQ Protocol using TCP in Java.

THEORY

Network design refers to the planning of the implementation of a computer network infrastructure. Network design is generally performed by network designers, engineers, IT administrators and other related staff. It is done before the implementation of a network infrastructure.

Usage of Network Design in Network Management:

The whole network design is usually represented as a network diagram that serves as the blueprint for implementing the network physically.

What is the functionality of Network Design?

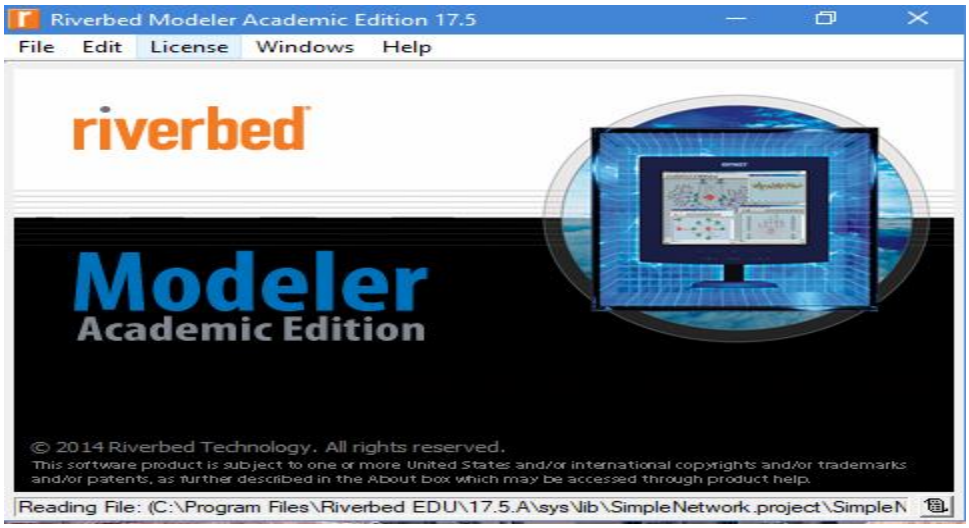
The main network functionalities includes the following:- - forecasts of how the new network/service will operate; - the economic information concerning costs; and - the technical details of the network's capabilities

SCREEN SHOTS:

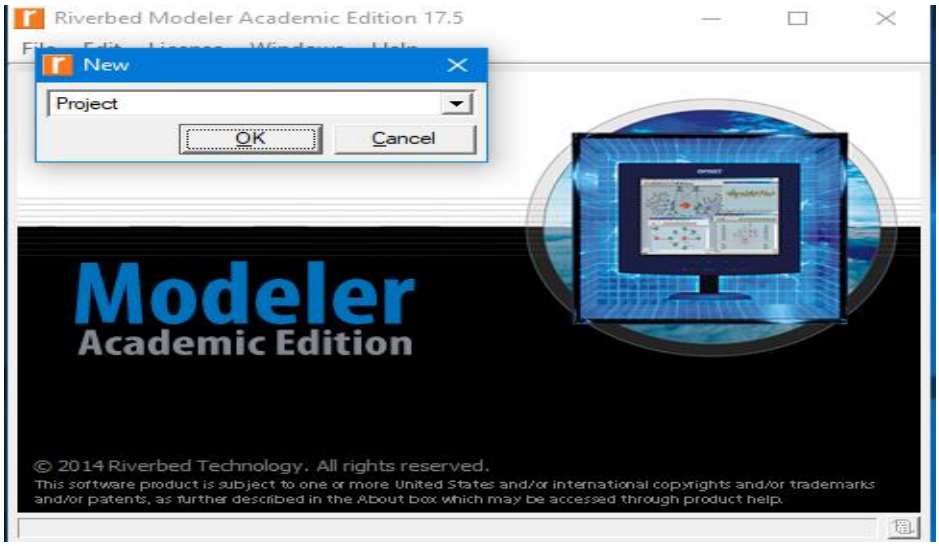
Implementation of Network Designin OPNET 17.5

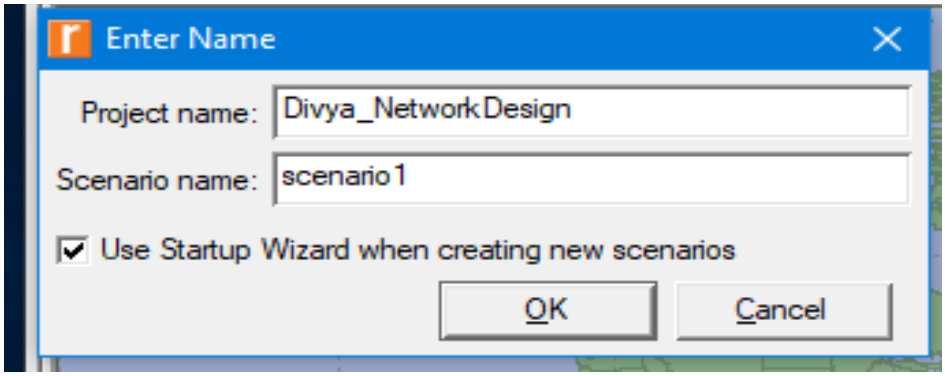
Create a New Project

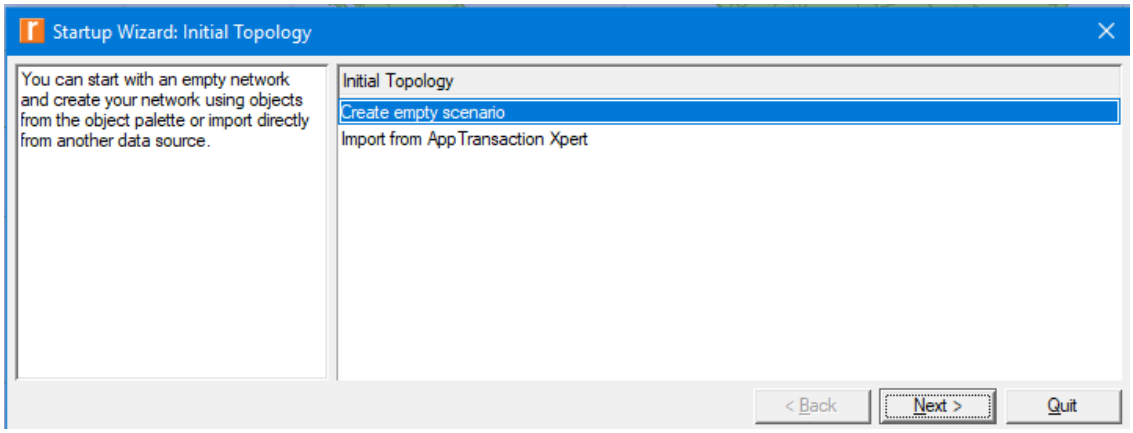
1. Start Riverbed Modeler Academic Edition ⇒ Choose New from the File menu.

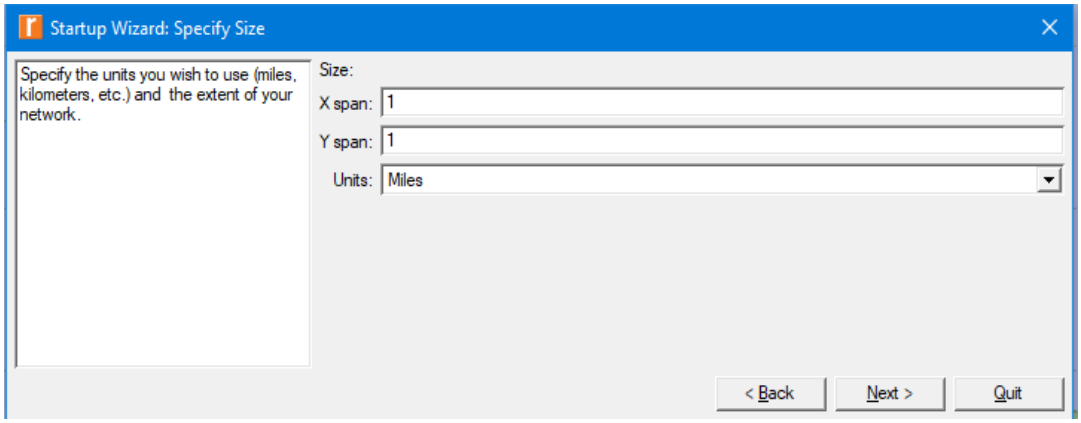


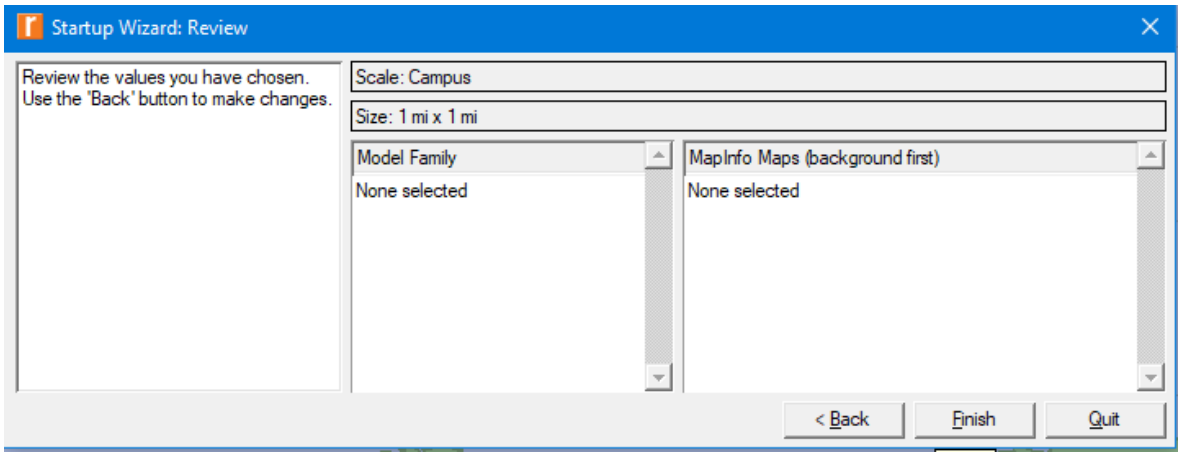
D890.A2FD.6F01.5AF2.54C3.83A0






Startup Wizard: Initial Topology
You can start with an empty network and create your network using objects from the object palette or import directly from another data source.
Initial Topology
Create empty scenario
Import from AppTransaction Xpert
< Back
Next >
Quit


Startup Wizard: Specify Size
Specify the units you wish to use (miles, kilometers, etc.) and the extent of your network.
Size:
X span: 1
Y span: 1
Units: Miles
< Back
Next >
Quit


Startup Wizard: Review
Review the values you have chosen. Use the 'Back' button to make changes.
Scale: Campus
Size: 1 mi x 1 mi
Model Family
None selected
MapInfo Maps (background first)
None selected
< Back
Finish
Quit

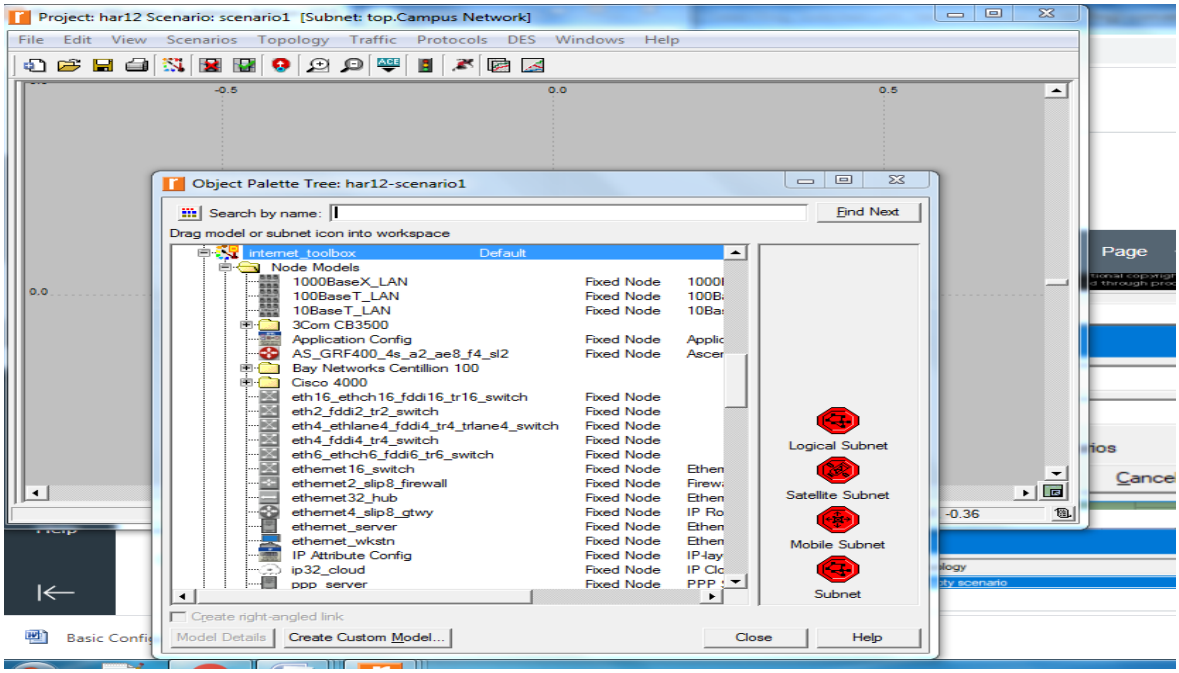
2. Select Project and click OK ⇒ Name the project <your initials>_NetDesign, and the scenario SimpleNetwork⇒ Make sure that the Use Startup Wizard is checked ⇒ Click OK.

3. In the Startup Wizard: Initial Topology dialog box, make sure that Create Empty Scenario is selected ⇒ Click Next ⇒ Choose Campus from the Network Scale list ⇒ Click Next ⇒ Choose Miles from the Size drop-down menu and assign 1 for both X Span and Y Span ⇒ Click Next twice ⇒ Click Finish.

Create and Configure the Network

Initialize the Network:

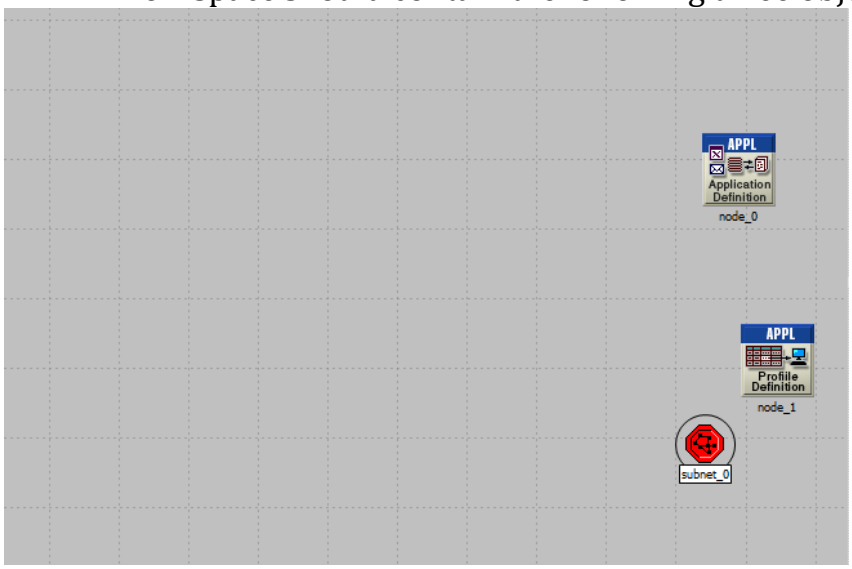
1. The Object Palette dialog box should be now on the top of your project space. If it is not there, open it by clicking . Make sure that the internet_toolbox is selected from the pull-down menu on the object palette.



2. Add to the project workspace the following objects from the palette: Application

Config, Profile Config, and a subnet.

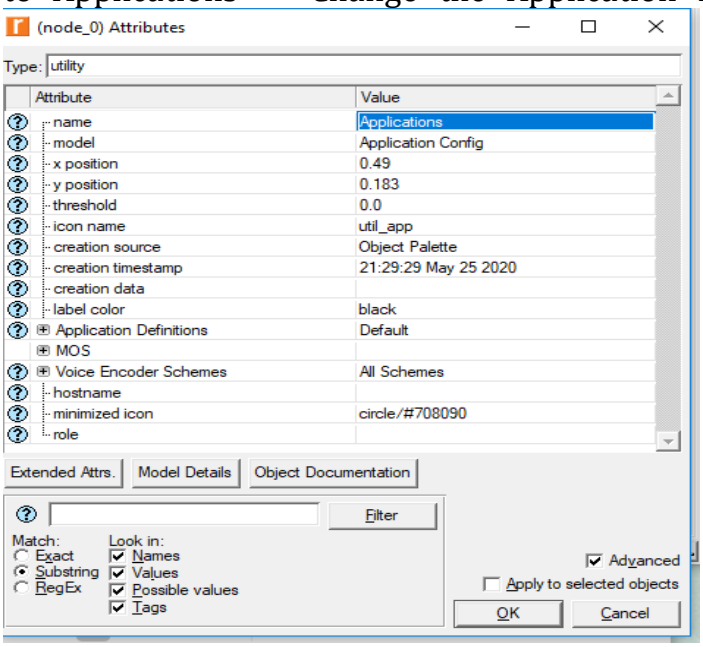
a. To add an object from a palette, click its icon in the object palette ⇒ Move your mouse to the workspace ⇒ Left-click to place the object. Right-click when finished. The workspace should contain the following three objects:



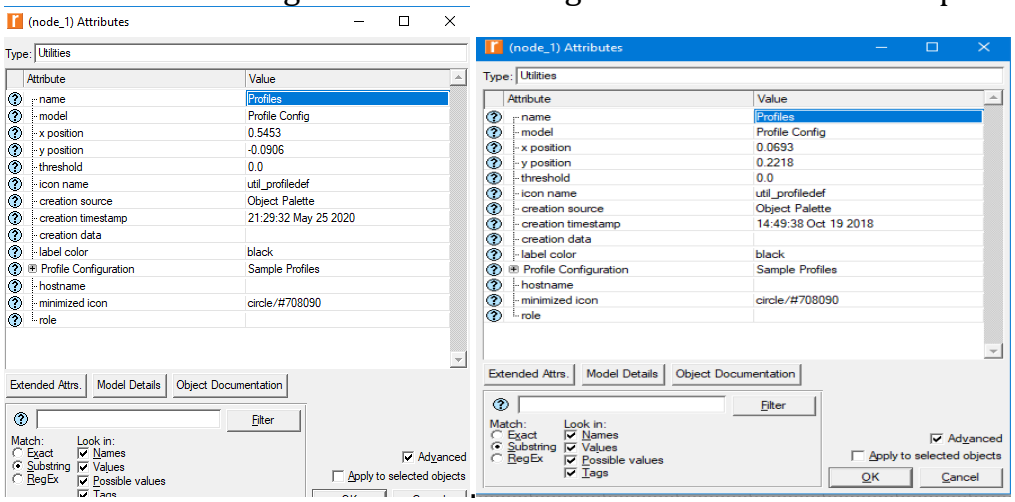
3. Close the Object Palette dialog box and save your project.

Configure the Services:

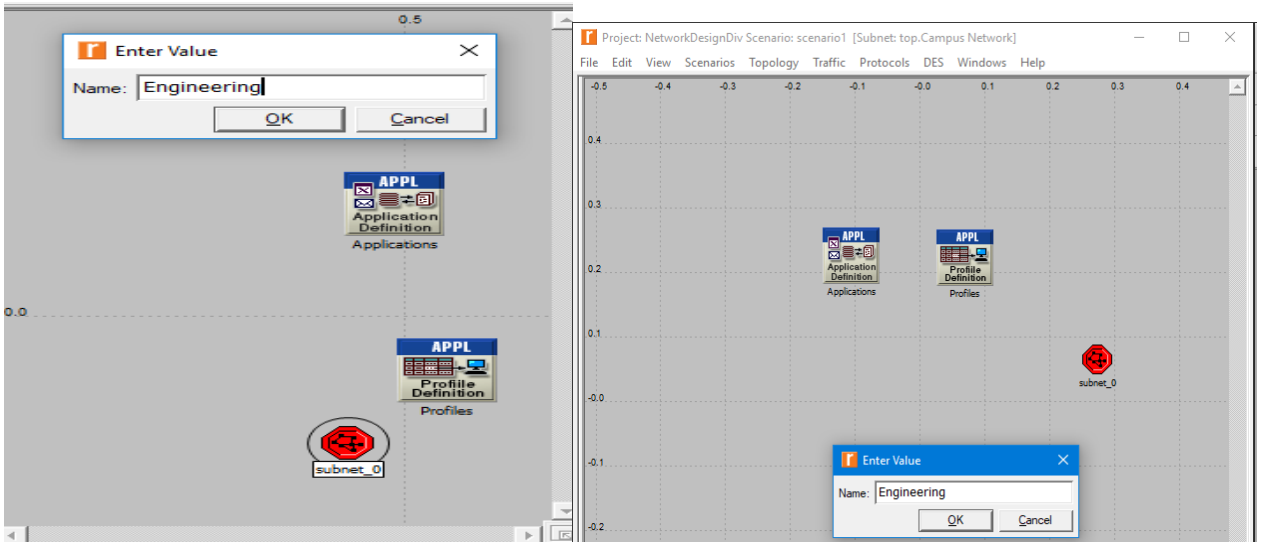
1. Right-click on the Application Config node ⇒ Edit Attributes ⇒ Change the name attribute to Applications ⇒ Change the Application Definitions attribute to Default ⇒ OK.



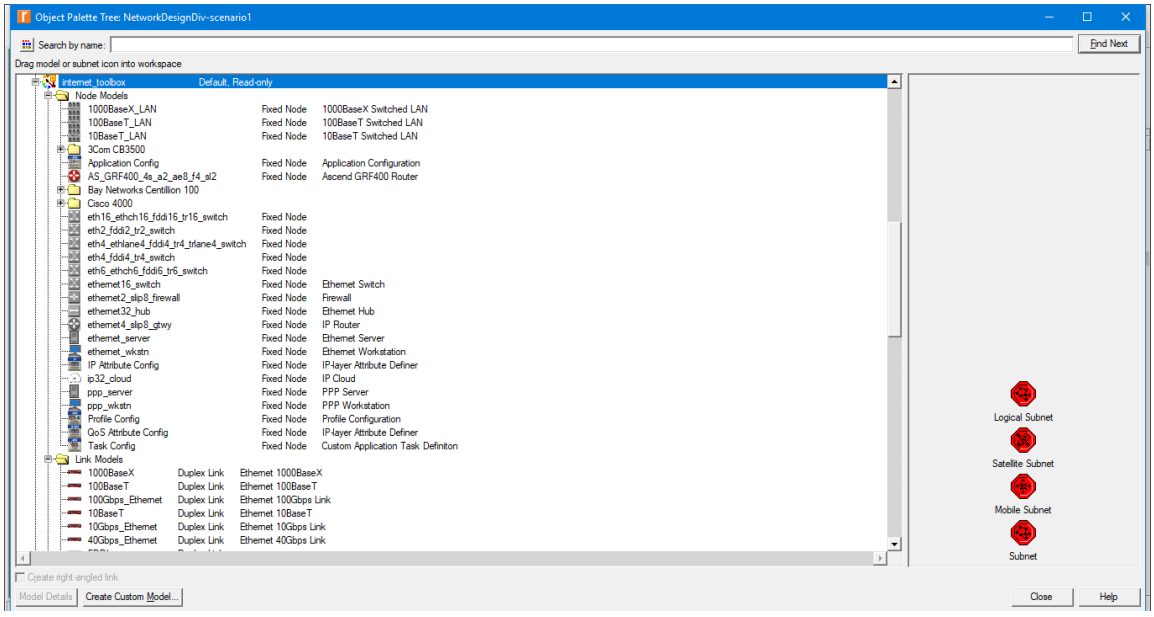
2. Right-click on the Profile Config node ⇒ Edit Attributes ⇒ Change the name attribute to Profiles ⇒ Change the Profile Configuration attribute to Sample Profiles ⇒ Click OK



Configure a Subnet:

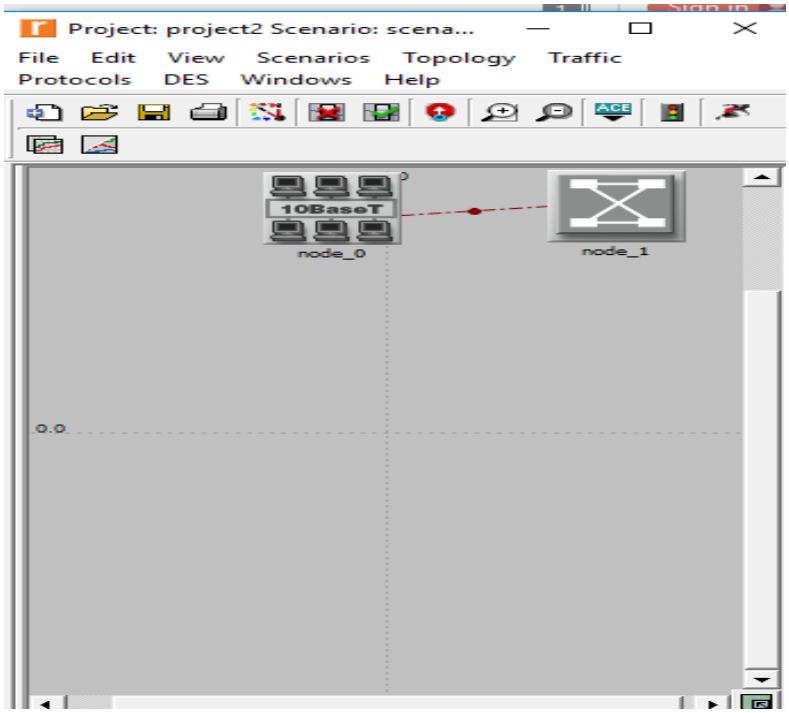
1. Right-click on the subnet node ⇒ Set Name ⇒ Change the name attribute to Engineering and click OK.
2. 

3. Double-click on the Engineering node. You get an empty workspace, indicating that the subnet contains no objects.
3. Open the object palette and make sure it is still set to internet_toolbox.

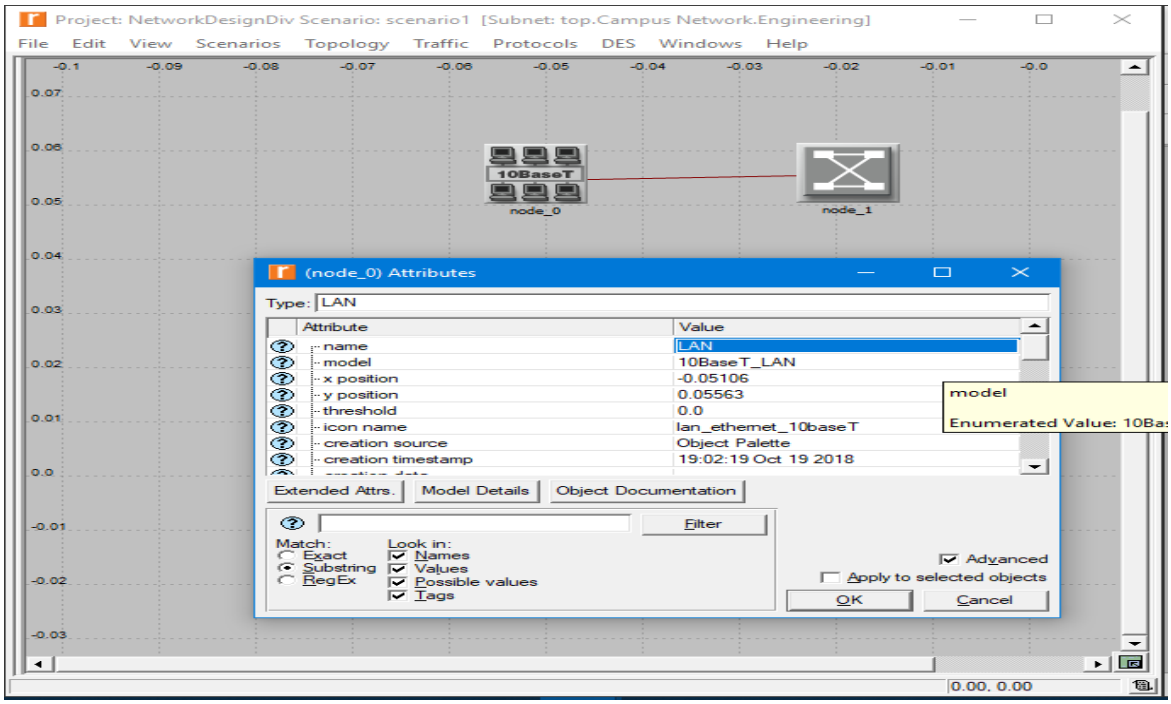


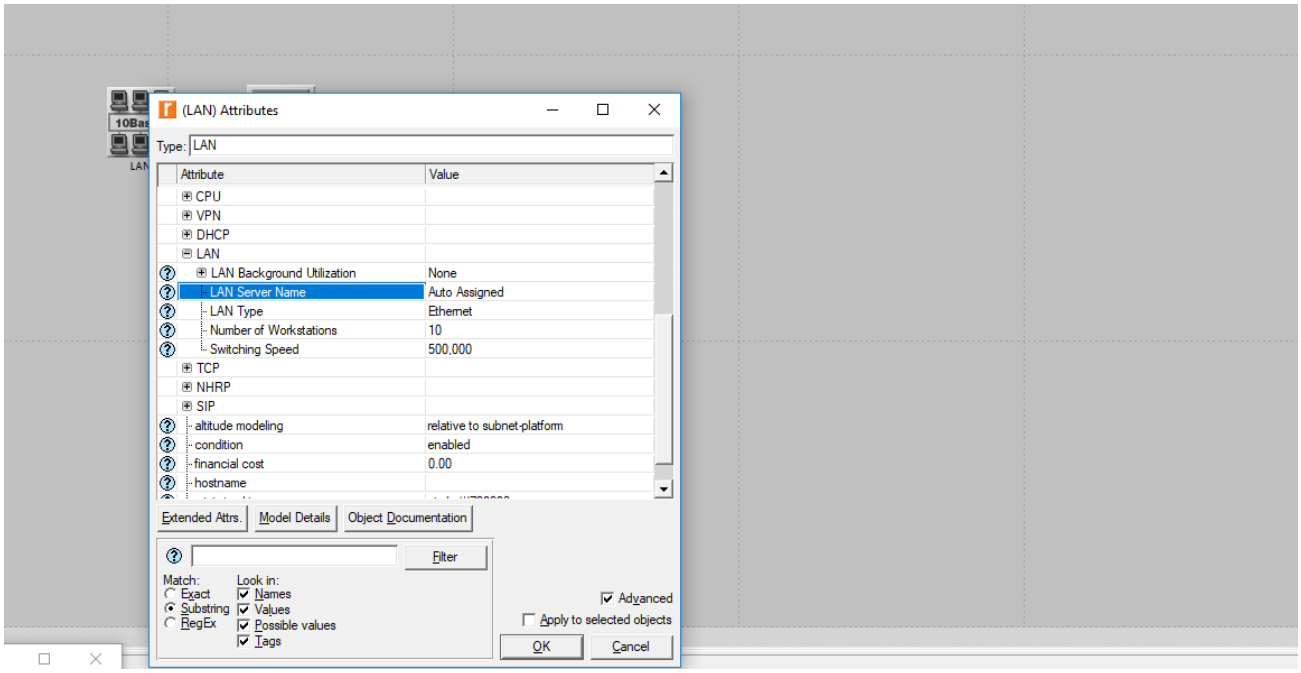
4. Add the following items to the subnet workspace: 10BaseT_LAN,

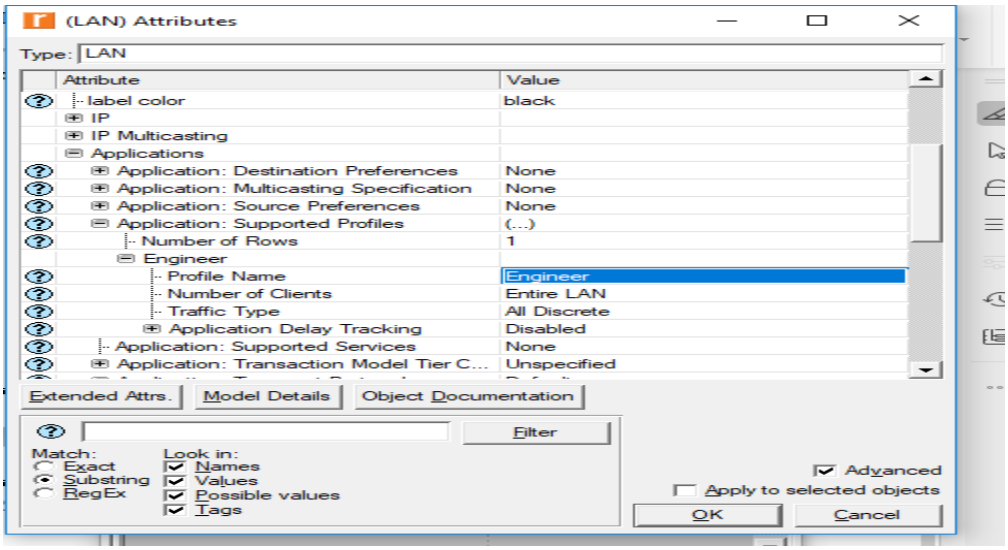
ethernet16_switch, and a 10BaseT_link to connect the LAN with the Switch ⇒ Close the palette.

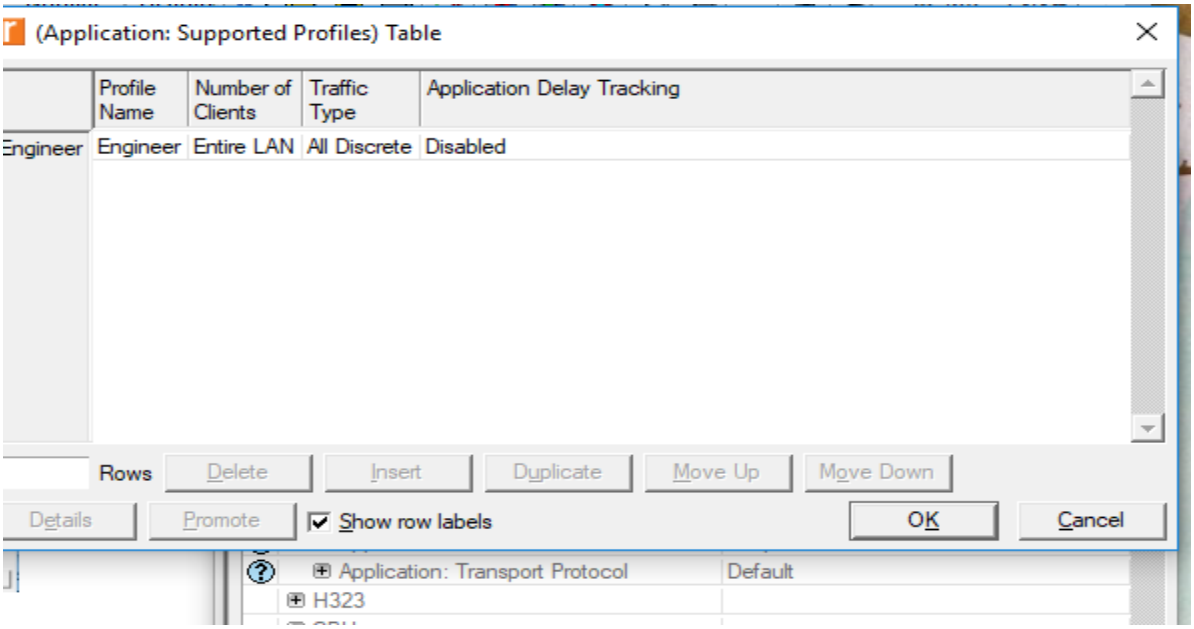


5. Right-click on the 10BaseT_LAN node ⇒ Edit Attributes ⇒ Change the name attribute to LAN ⇒ Expand LAN ⇒ Observe that the Number of Workstations attribute has a value of 10 ⇒ Expand Application ⇒ Expand Application: Supported Profiles ⇒ Set the number of rows to 1 ⇒ Expand None ⇒ Set the Profile Name to Engineer.


Project: NetworkDesignDiv Scenario: scenario1 [Subnet: top.Campus Network.Engineering]
File Edit View Scenarios Topology Traffic Protocols DES Windows Help
10BaseT
node_0
node_1
(node_0) Attributes
Type: LAN
Attribute
Value
name
LAN
model
10BaseT_LAN
x position
-0.05106
y position
0.05563
threshold
0.0
icon name
lan_ethernet_10baseT
creation source
Object Palette
creation timestamp
19:02:19 Oct 19 2018
model
Enumerated Value: 10Ba
Extended Attrs.
Model Details
Object Documentation
Filter
Match:
Exact
Substring
RegEx
Look in:
Names
Values
Possible values
Tags
Advanced
Apply to selected objects
OK
Cancel
0.00, 0.00


(LAN) Attributes
Type: LAN
Attribute
Value
CPU
VPN
DHCP
LAN
LAN Background Utilization
None
LAN Server Name
Auto Assigned
LAN Type
Ethernet
Number of Workstations
10
Switching Speed
500,000
TCP
NHRP
SIP
altitude modeling
relative to subnet-platform
condition
enabled
financial cost
0.00
hostname
Extended Attrs.
Model Details
Object Documentation
Filter
Match:
Exact
Substring
RegEx
Look in:
Names
Values
Possible values
Tags
Advanced
Apply to selected objects
OK
Cancel





6. Click OK.

The object we just created is equivalent to a 10-workstation star topology LAN. The traffic generated from the users of this LAN resembles that generated by “engineers.”

7.Rename the ethernet16 Switch to Switch.

8. The subnet should look like the shown one.

9.Save your project.

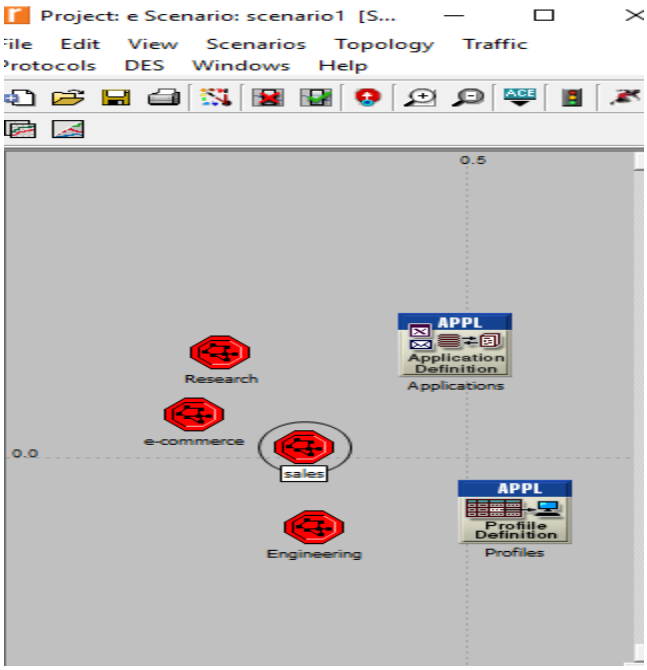
Configure All Departments:

1. Now you have completed the configuration of the Engineering department subnet. To go back to the main project space, click the Go to the Parent Subnet button.

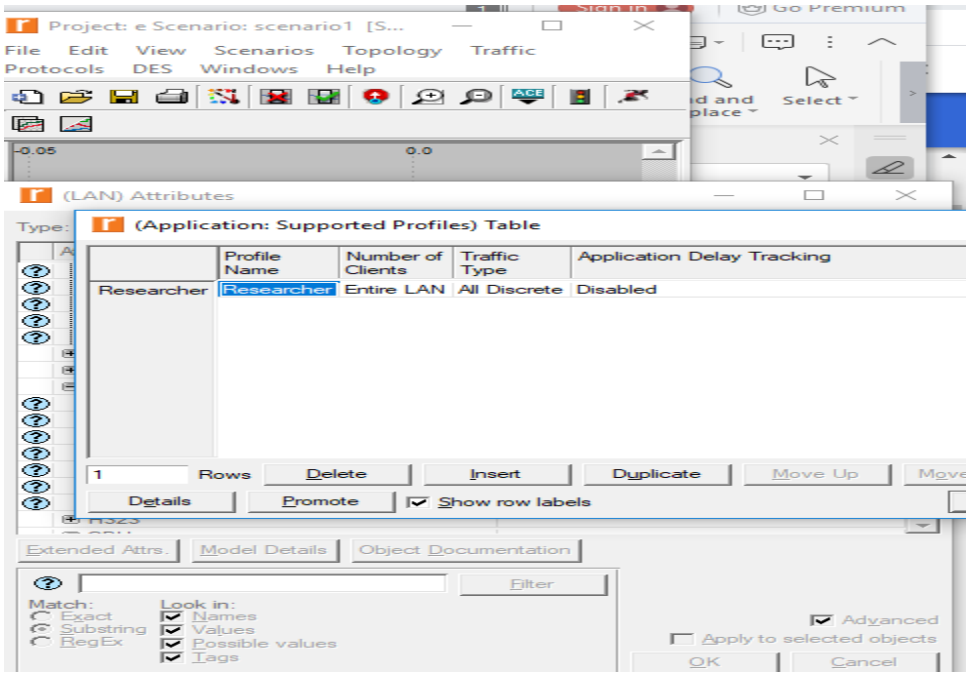
The subnets of the other departments in the company should be similar to the engineering one except for the supported profiles.

2. Make three copies of the Engineering subnet we just created: Click on the Engineering node ⇒ From the Edit menu, select Copy ⇒ From the Edit menu, select Paste three times, placing the subnet in the workspace after each, to create the new subnets.

3. Rename (right-click on the subnet and select Set Name) and arrange the subnets as shown below.

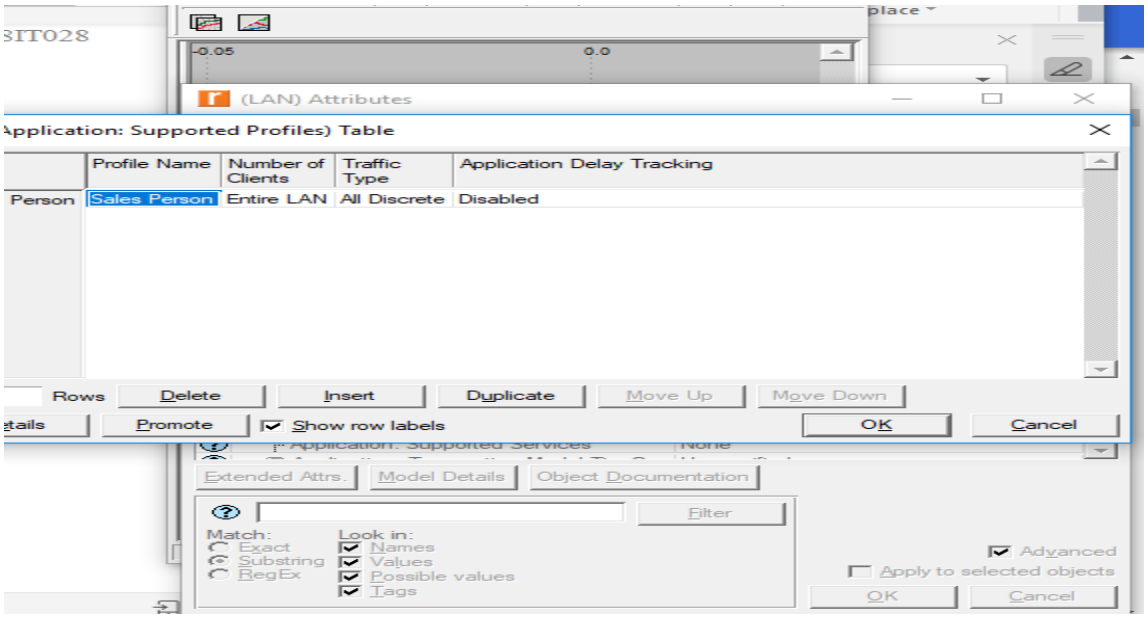


4. Double-click the Research node ⇒ Edit the attributes of its LAN ⇒ Edit the value of the Application: Supported Profiles attribute ⇒ Change the value of the Profile Name from Engineer to Researcher ⇒ Click OK ⇒ Go to the parent subnet by clicking the button.

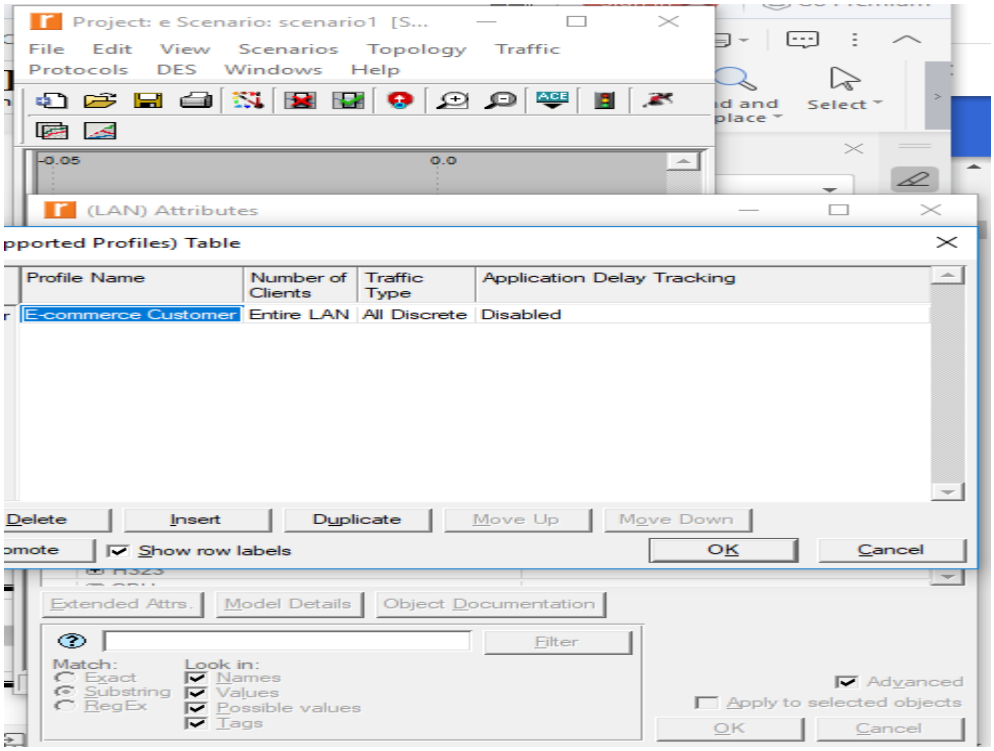


5. Repeat step 4 with the Sales node and assign to its Profile Name the profile

Sales Person.



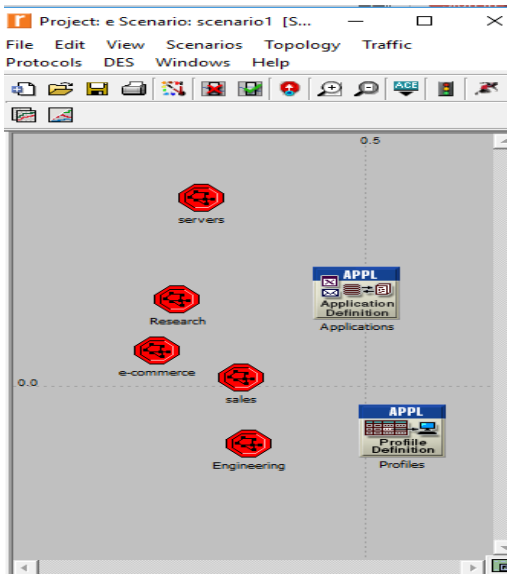
4. Repeat step 4 with the E-Commerce node and assign to its Profile Name the profile E-commerce Customer.



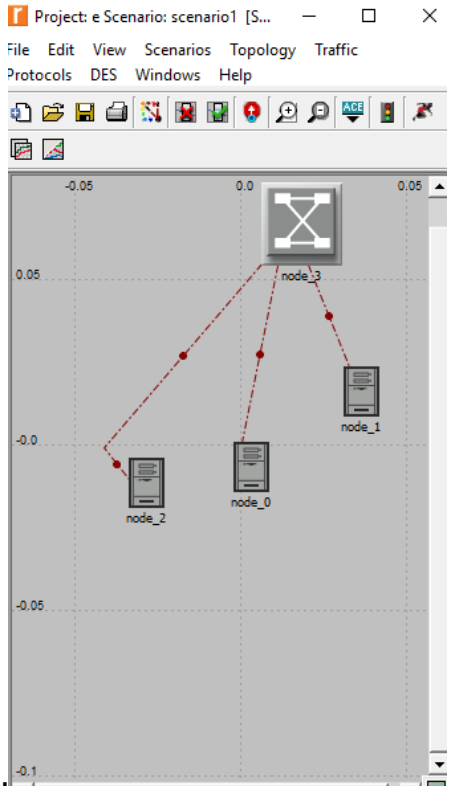
7. Save your project.

Configure the Servers:

1. Open the Object Palette and add a new subnet ⇒ Rename the new subnet to Servers ⇒ Double-click the Servers node to enter its workspace.

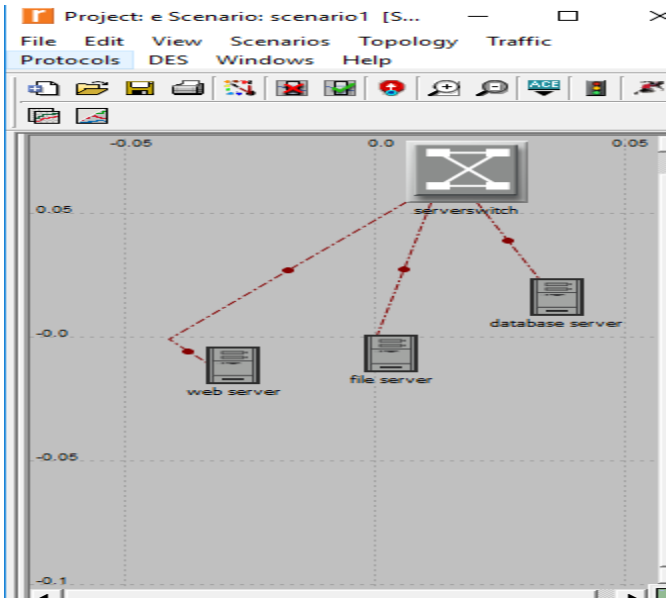
2. 

3. From the Object Palette, add three ethernet_servers, one ethernet16_switch, and three 10BaseT links to connect the servers with the switch.



3. Close the Object Palette.

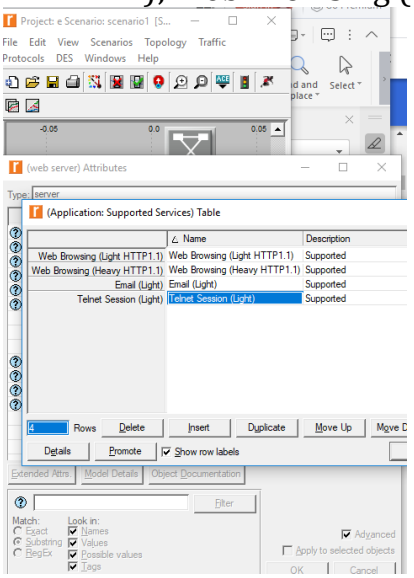
4. Rename the servers and the switch as follows:



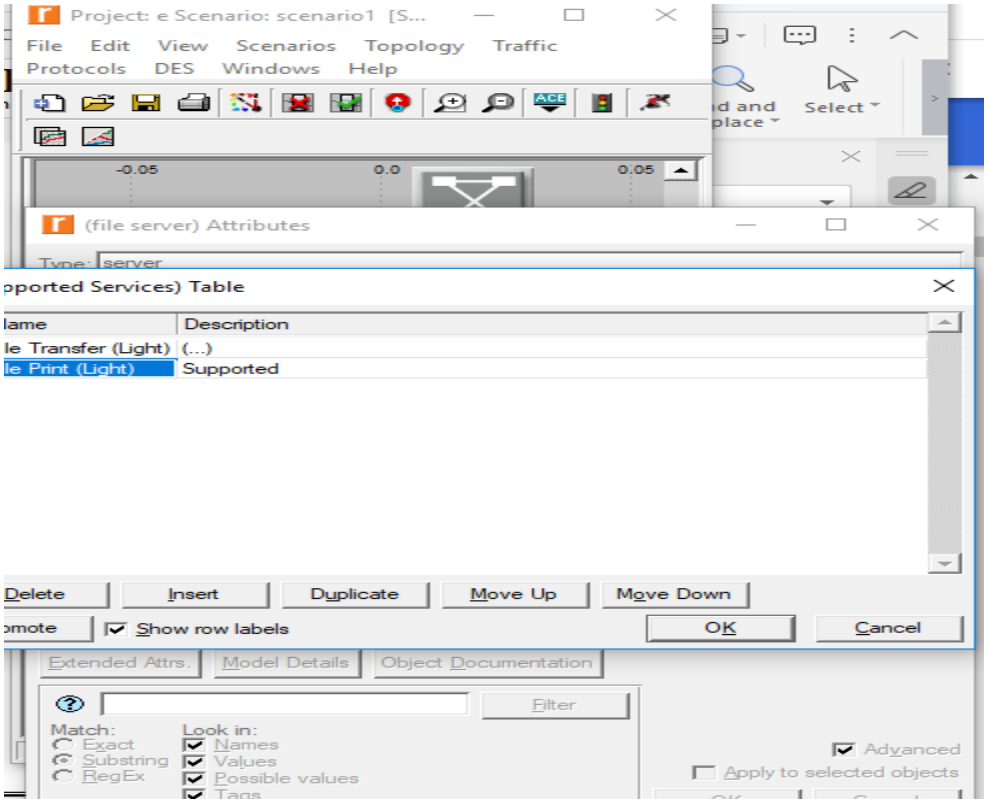
5. Right-click on each one of the above servers and Edit the value of the

Application: Supported Services attribute under Application.

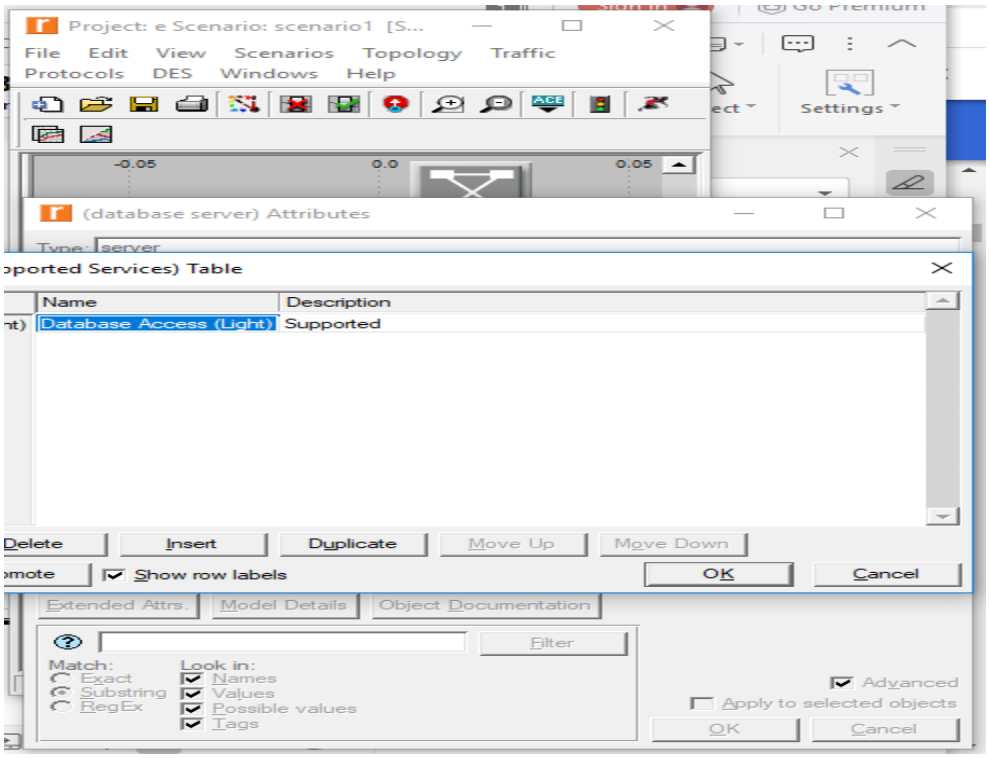
i. For the Web Server add 4 rows to support the following services: Web Browsing (Light HTTP1.1), Web Browsing (Heavy HTTP1.1), Email (Light), and Telnet Session (Light).

ii. 

ii. For the File Server add two rows to support the following services: File Transfer (Light) and File Print (Light).



iii. For the Database Server add one row to support the following service: Database Access (Light).



6. Go back to the project space by clicking the Go to the Parent Subnet button.

7. Save your project.

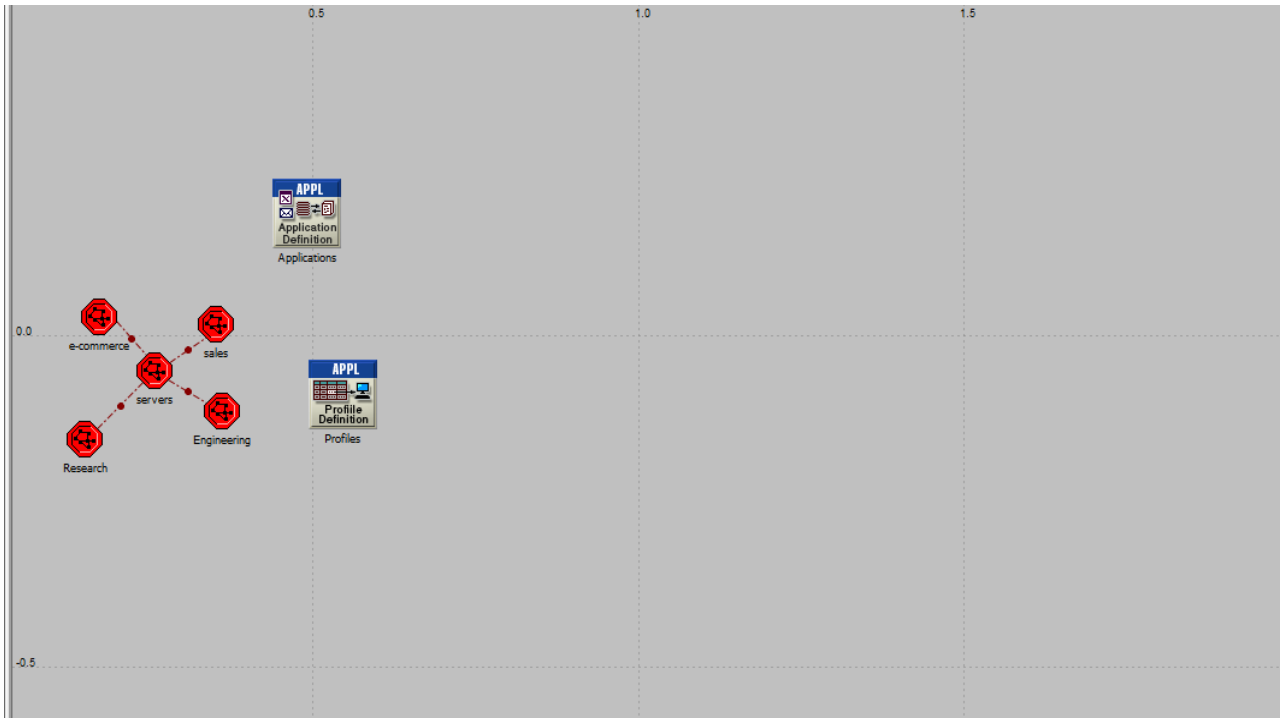
Connect the Subnets:

1. Open the Object Palette and add four 100BaseT links to connect the subnets of the departments to the Servers subnet.

As you create each link, make sure that it is configured to connect the "switches" in both subnets to each other. Do this by choosing them from the drop-down menus as follows:

2. Close the Object Palette.

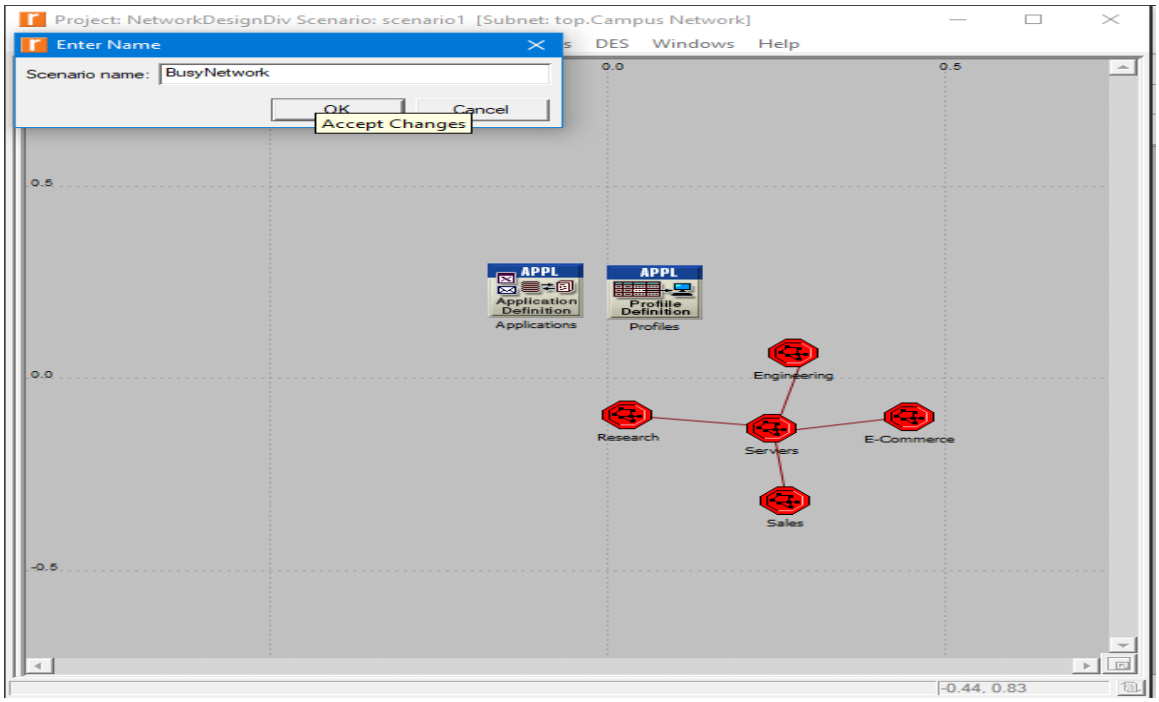
3. Now your network should resemble the following one:



4. Save your project.

Duplicate the Scenario

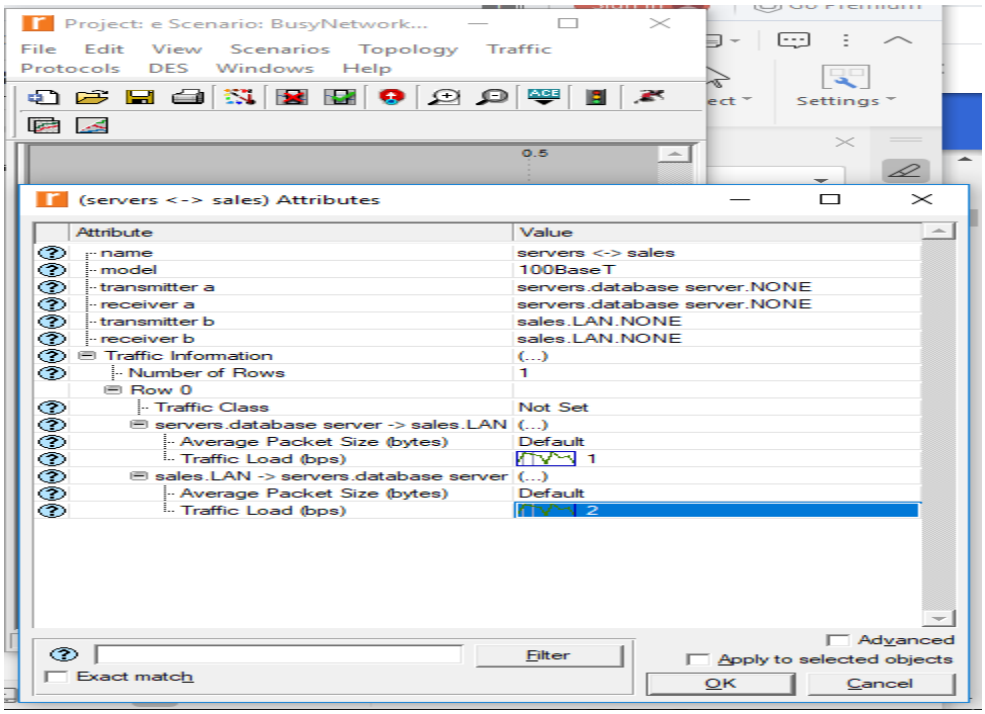
1. Select Duplicate Scenario from the Scenarios menu and give it the name BusyNetwork⇒ Click OK.



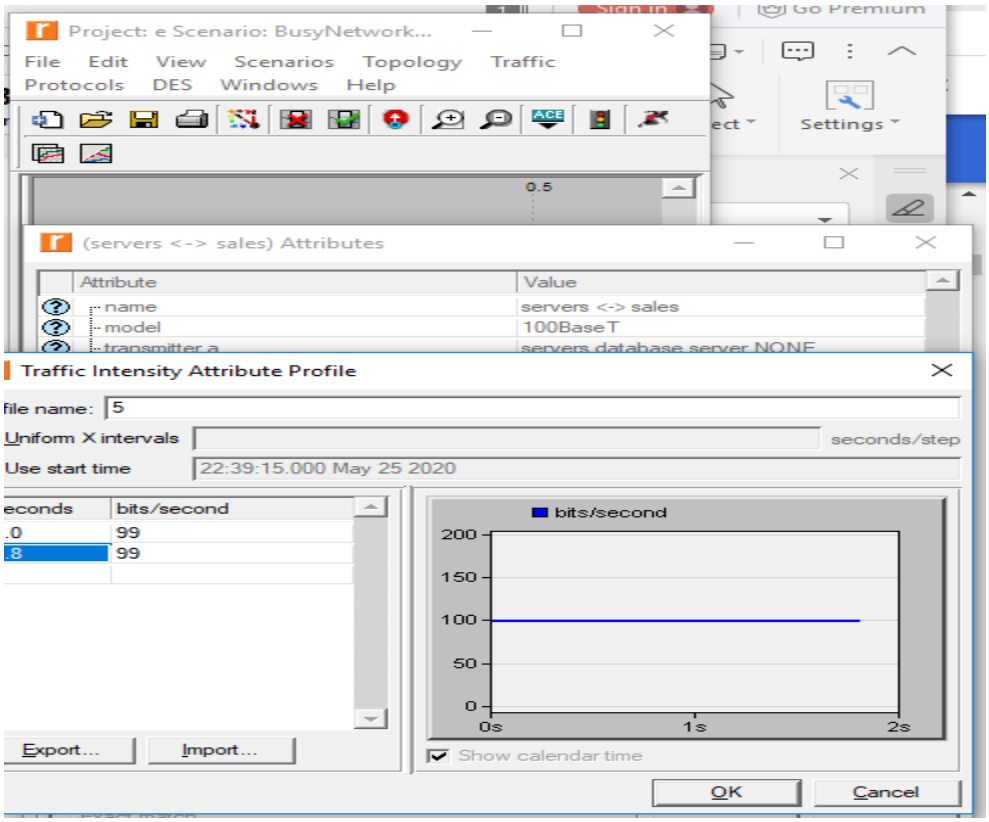
2. Select all the 100BaseT links simultaneously (click on all of them while holding the Shift key) ⇒ Right-click on anyone of them ⇒ Edit Attributes ⇒ Check the Apply Changes to Selected Objects check box.

3. Expand the hierarchy of the Traffic Information attribute ⇒ Click on the value of Number of Rows and assign 1 ⇒ Click enter.

4. Expand the hierarchy of the Row 0 attribute ⇒ Expand the hierarchy of the 2 Campus Network.*** attribute ⇒ Click on the value of Traffic Load of the first and choose 1 ⇒ Click enter.



We now wish to add a background utilization of 99%. That means that we want the Link Load (bps) to be 99000000 bps and since our simulation will last 30 minutes we will do the following:



5. Click OK.

6. Save your project.

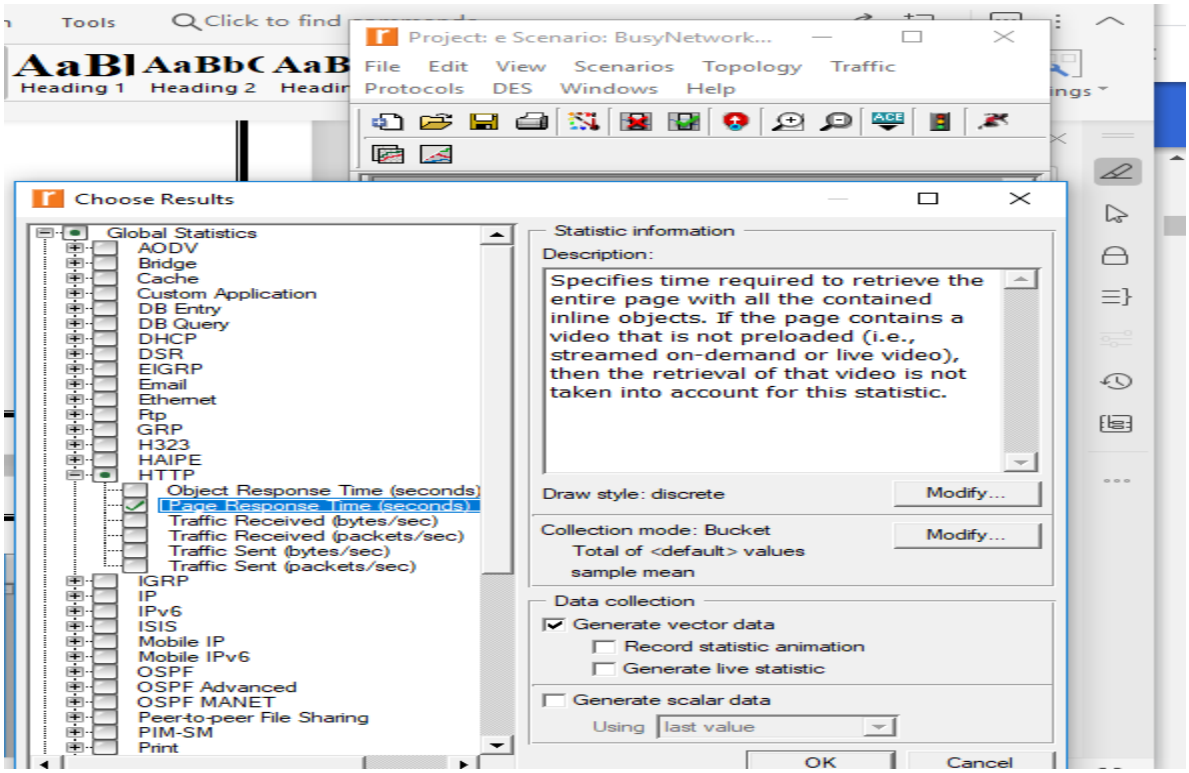
Choose the Statistics

1. In each scenario, right-click anywhere in the project workspace and select

Choose Individual DES Statistics from the pop-up menu.

2. In the Choose Results dialog box, choose the following statistic:

Page Response Time is the required time to retrieve the entire page.

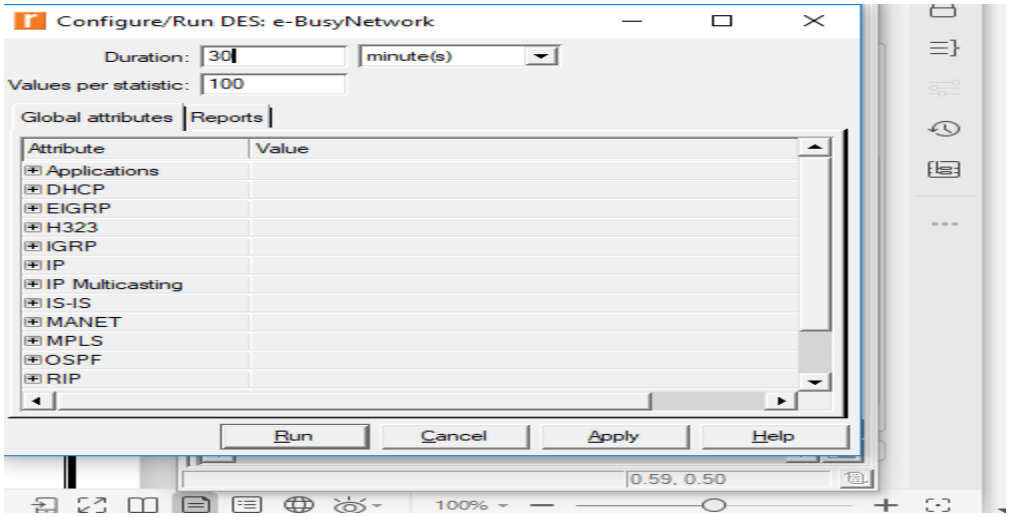


3. Click OK.

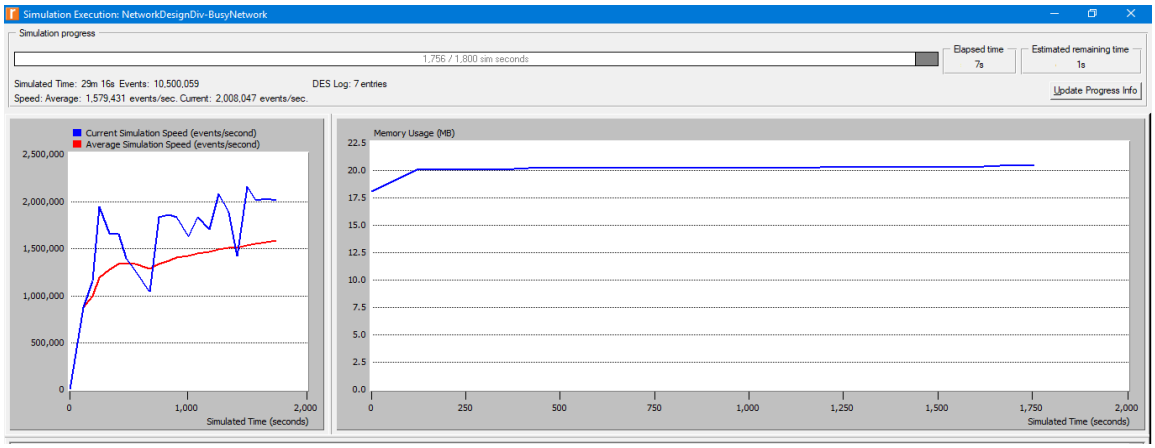
Configure and Run the Simulation

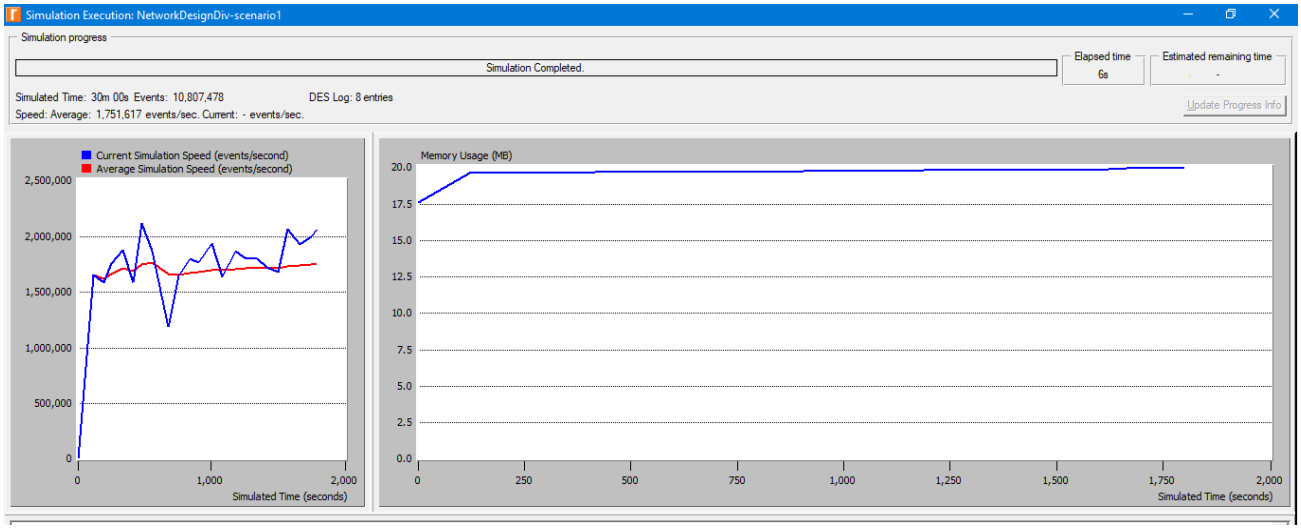
1. Click on the Configure/Run Simulation button.

2. Set the duration to be 30.0 minutes.



3. Press Run. Repeat for both simulations.





View the Results

1. Select Compare Results from Results in the DES menu.

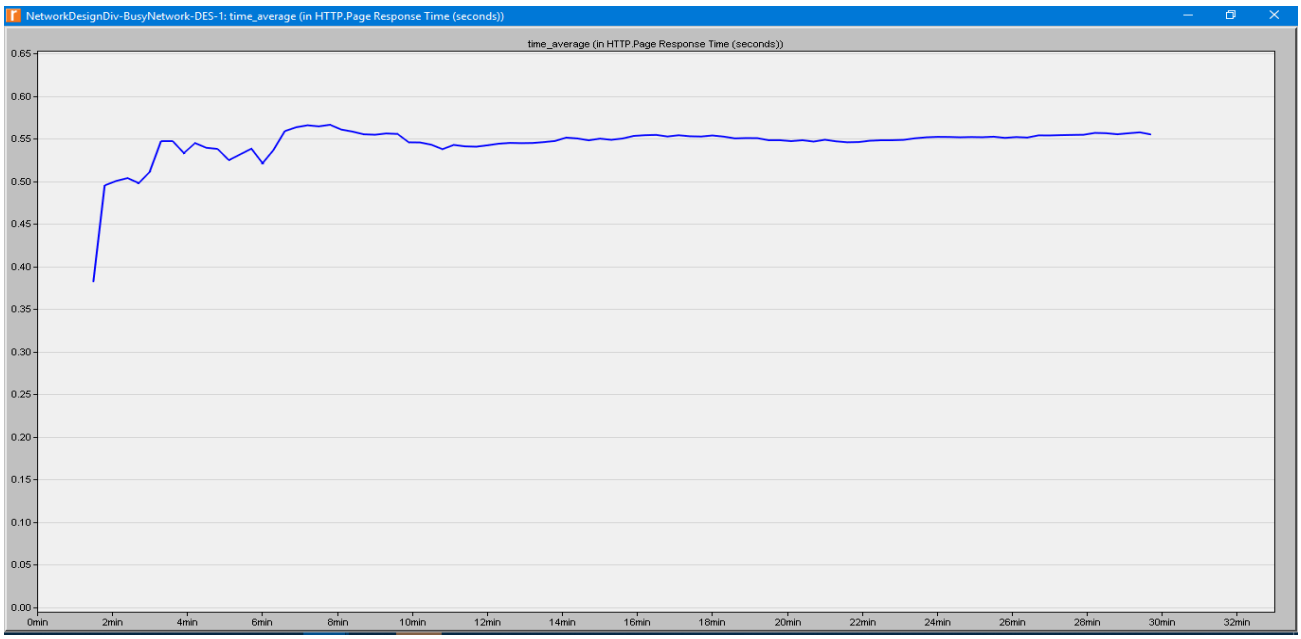
2. Change the drop-down menu in the lower-right part of the Compare Results dialog box from As Is to time_average, as shown.

3. From the Results for drop-down menu, select Current Project.

4. Check SimpleNetwork and BusyNetwork.

5. Expand Global Statistics ⇒ expand HTTP

6. Select the Page Response Time (seconds) statistic and click Show.



Explain the OPNET 17.5 Output for Network Design:

The output specifies the time required to retrieve the entire page with all the contained inline objects. If the page contains a video that is not preloaded, that is streamed on-demand or live video), then the retrieval of that video is not taken into account for this statistic.

4. ICMP Ping:

Definition ofICMP Ping:

ICMP (Internet Control Message Protocol) is an error-reporting protocol network devices like routers use to generate error messages to the source IP address when network problems prevent delivery of IP packets. ICMP creates and sends messages to the source IP address indicating that a gateway to the Internet that a router, service or host cannot be reached for packet delivery.

ping. Ping is a computer network administration software utility used to test the reachability of a host on an Internet Protocol (IP) network.

Usage of ICMP Ping in Network Management:

Ping works by sending an Internet Control Message Protocol (ICMP) Echo Request to a specified interface on the network and waiting for a reply. Ping can be used for troubleshooting to test connectivity and determine response time

Functionality of ICMP Ping:

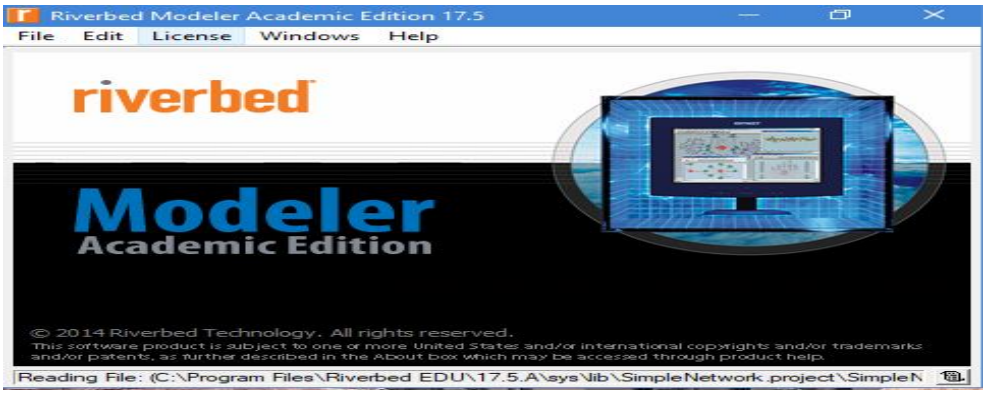
The Ping utility is essentially a system administrator's tool that is used to see if a computer is operating and also to see if network connections are intact. Ping uses the Internet Control Message Protocol (ICMP) Echo function which is detailed in RFC 792.

Implementation of ICMP Ping in OPNET 17.5:

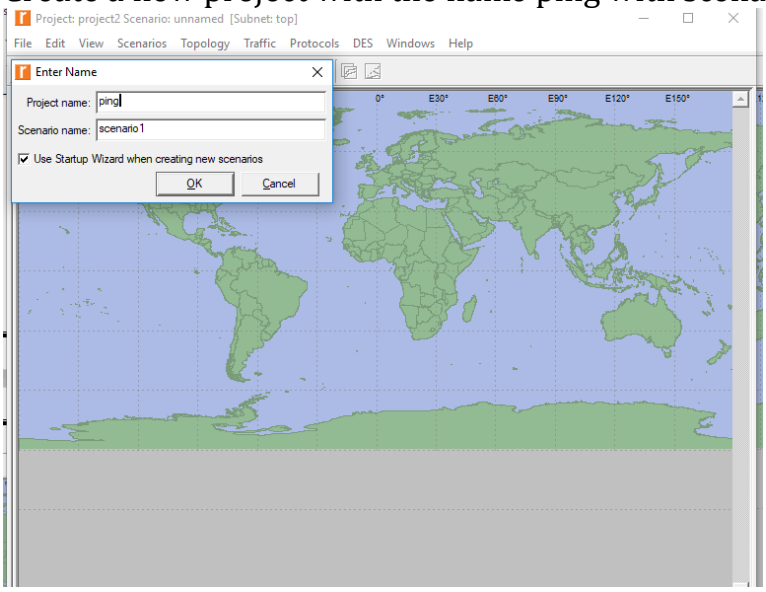
Project Name: Ping

Scenario: 1

1. Start Riverbed Modeler Academic Edition

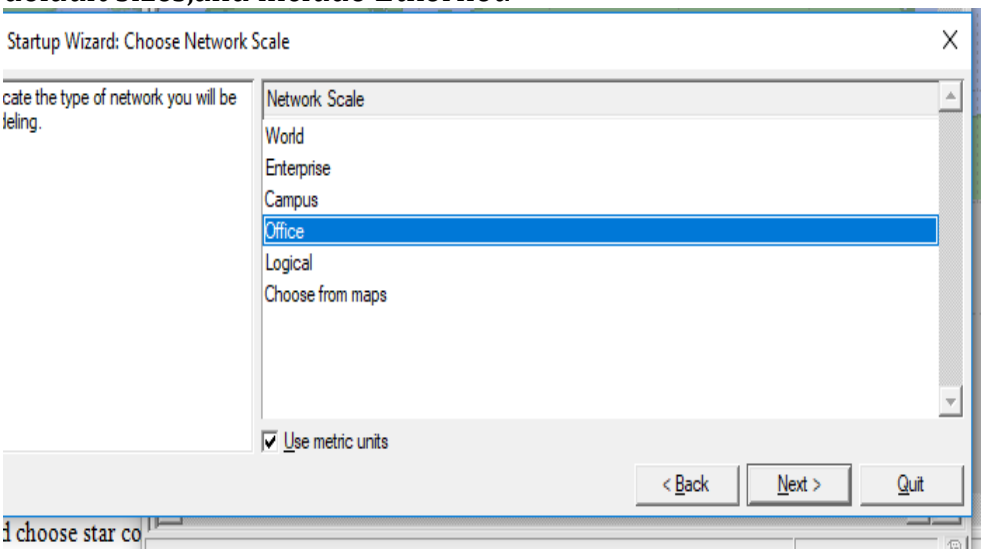


2. Create a new project with the name ping with scenario number as 1



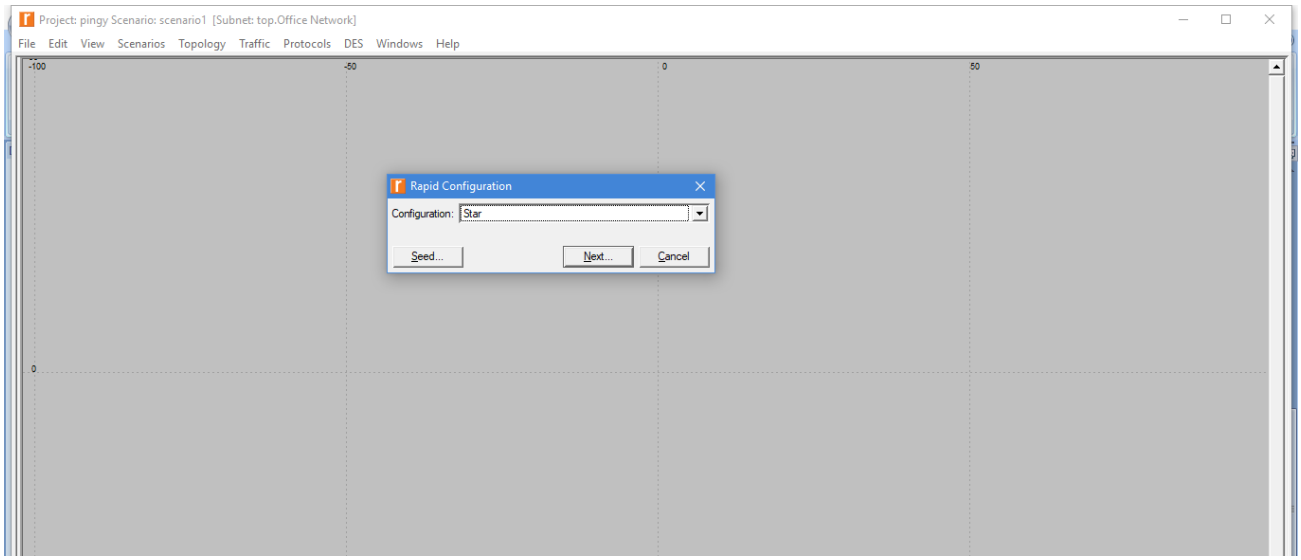
3.

4. Select Network Scale as office in the dialog box that appears, and maintain the default sizes,and include Ethernet.

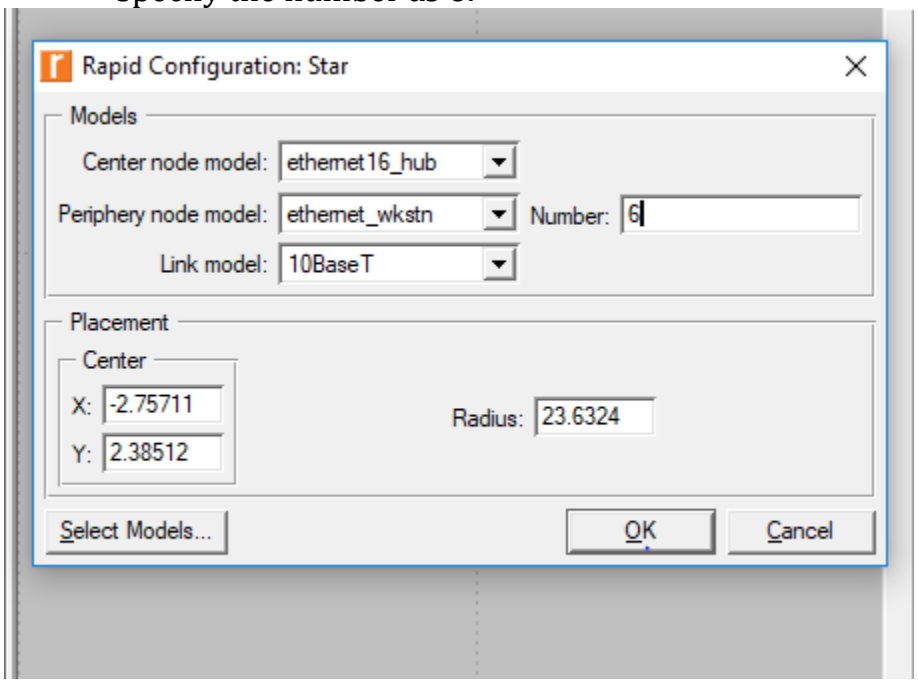


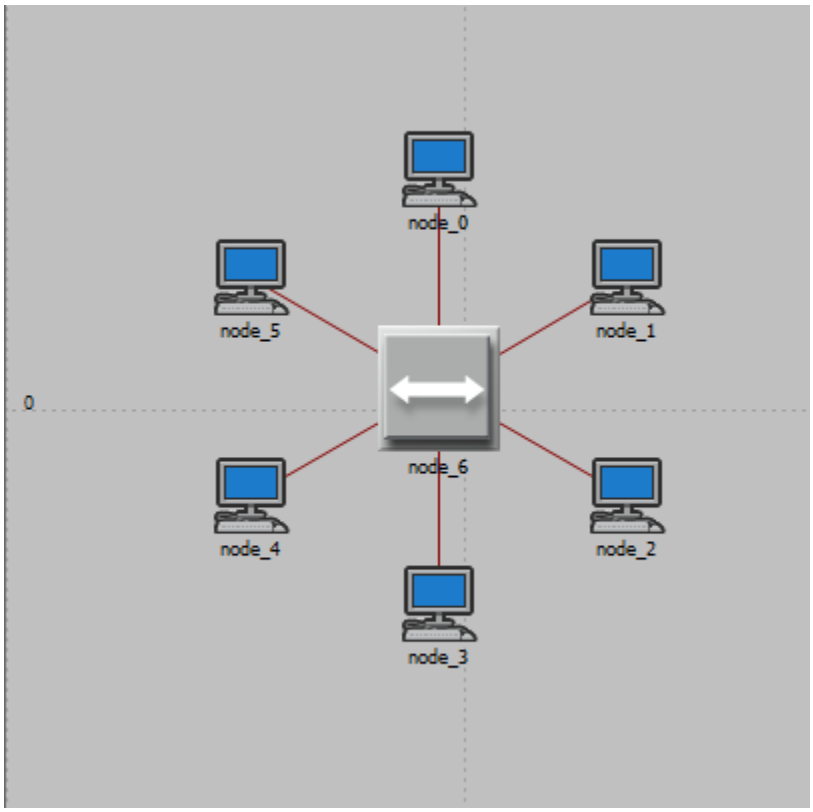
5.

6. Then in Traffic menu, choose the Rapid Configuration option and choose star configuration.

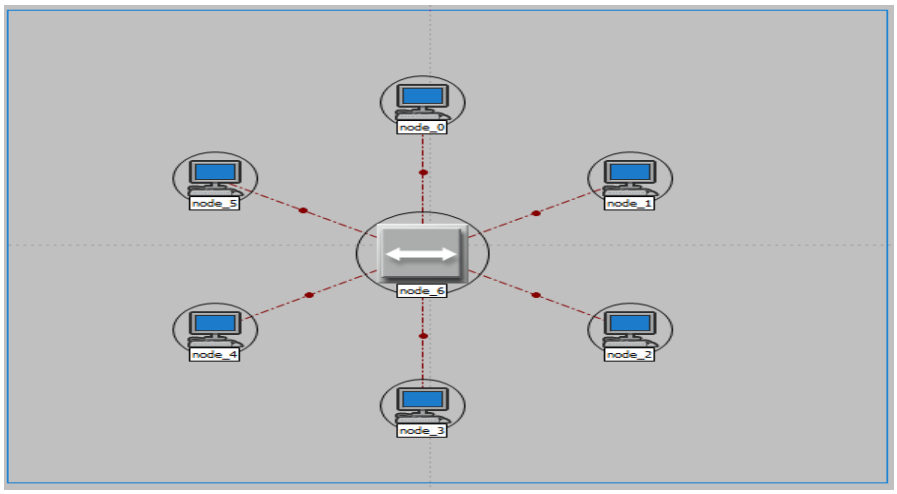


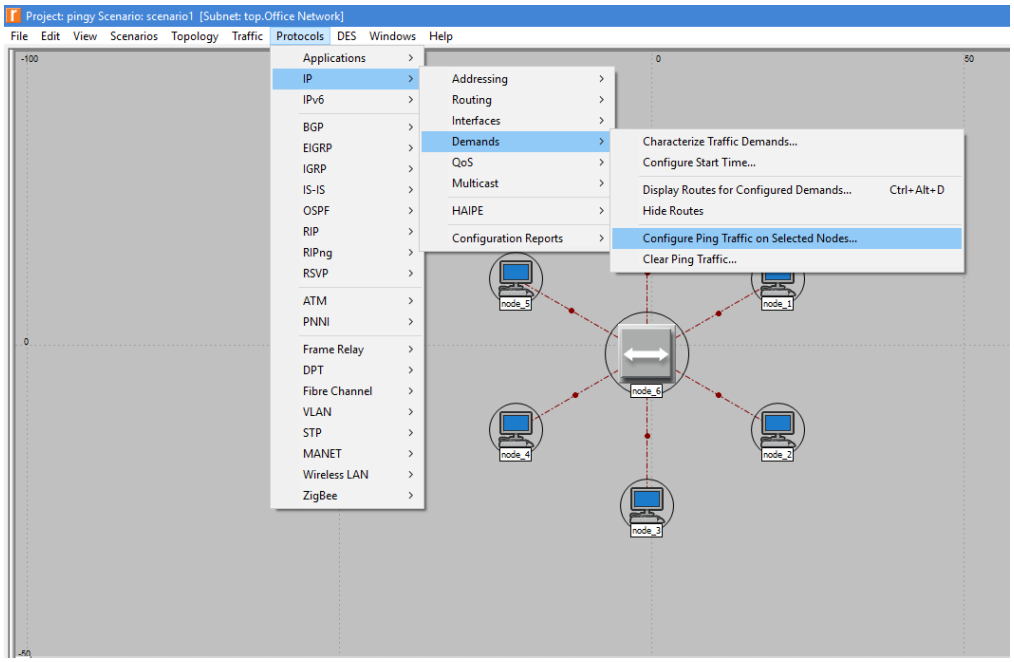
3. After that step, mention the center node model as ethernet16_hub and peripheral node model as Ethernet workstation. Choose link model type as 10BASET. Next, specify the number as 6.



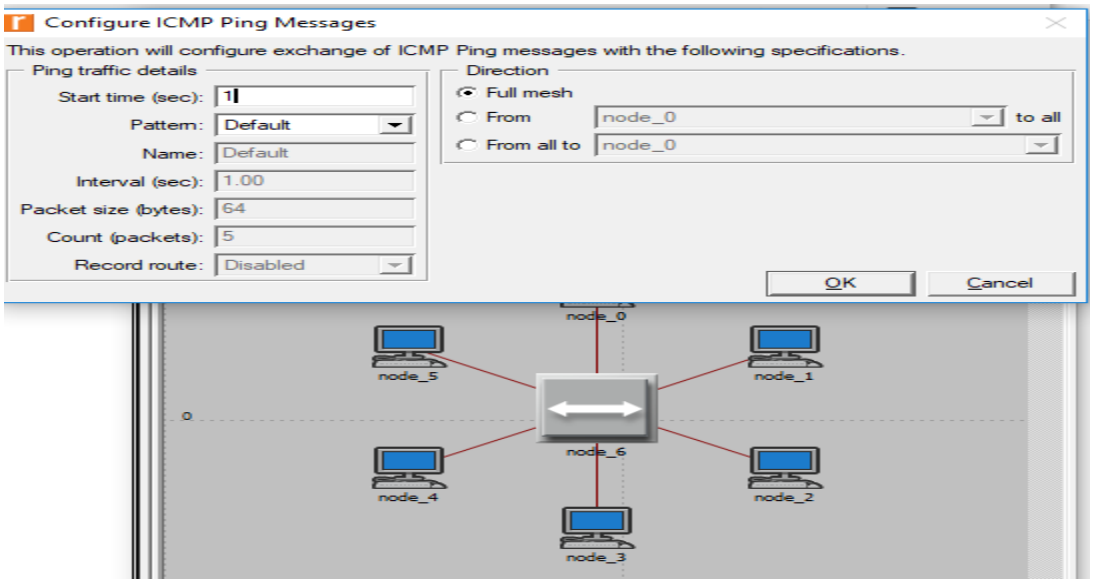


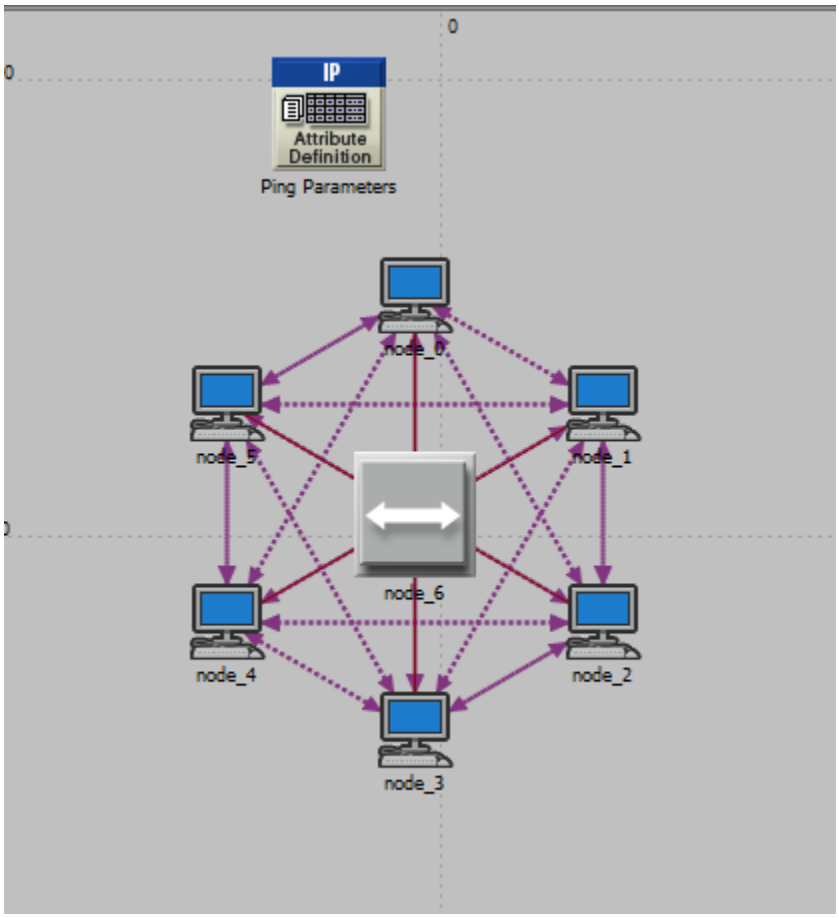
6. In the protocols menu, choose IP option, demands option and Configure ICMP Ping Messages option in that order.



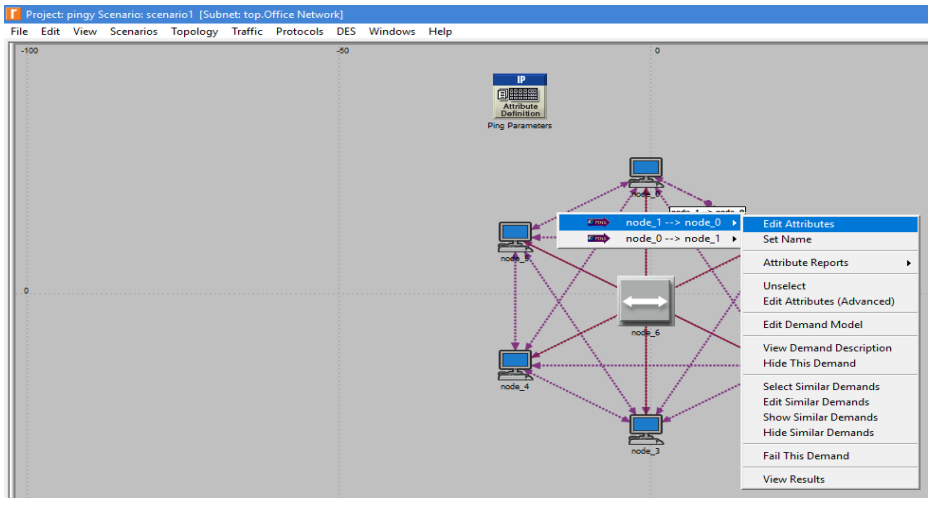


4. There specify the Start time as 1 and click ok.

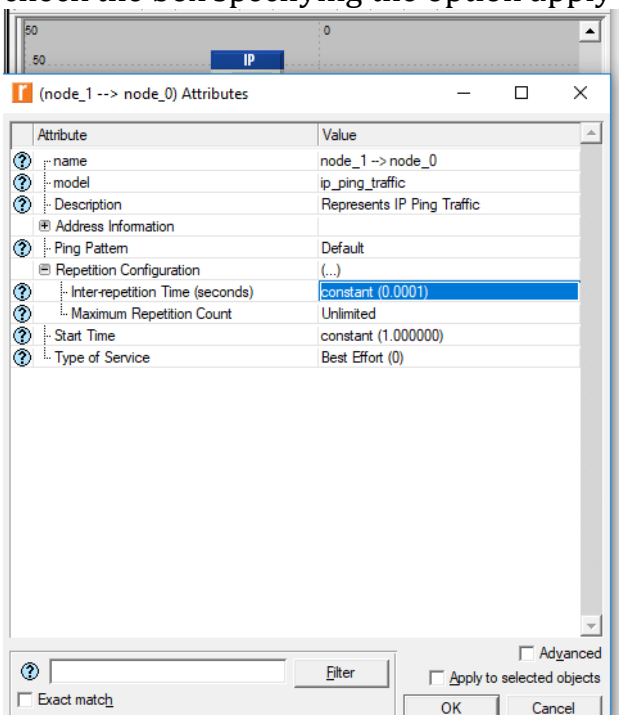




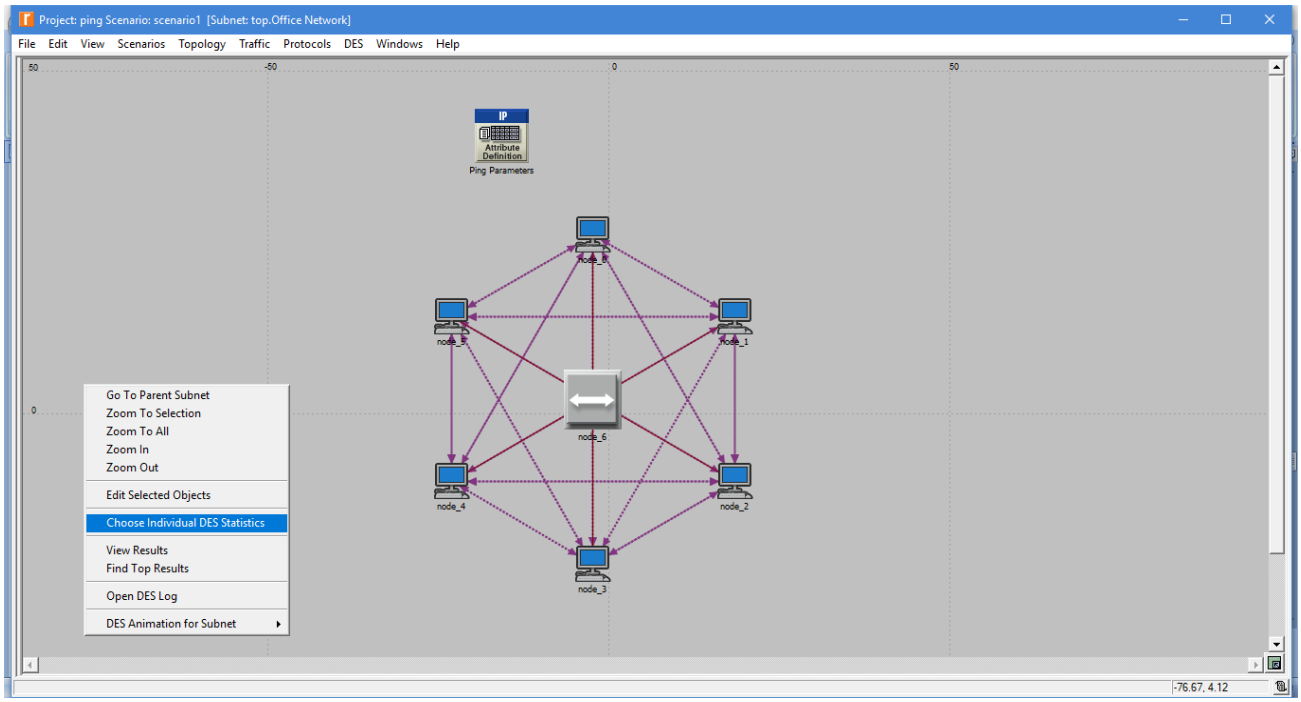
7. Then choose any link and right click to edit its attributes.



9. In the attributes section, go to the Ping pattern, repetition configuration and fix the mean outcome to 0.0001 and set the maximum repetition count to Unlimited and check the box specifying the option apply to selected objects.

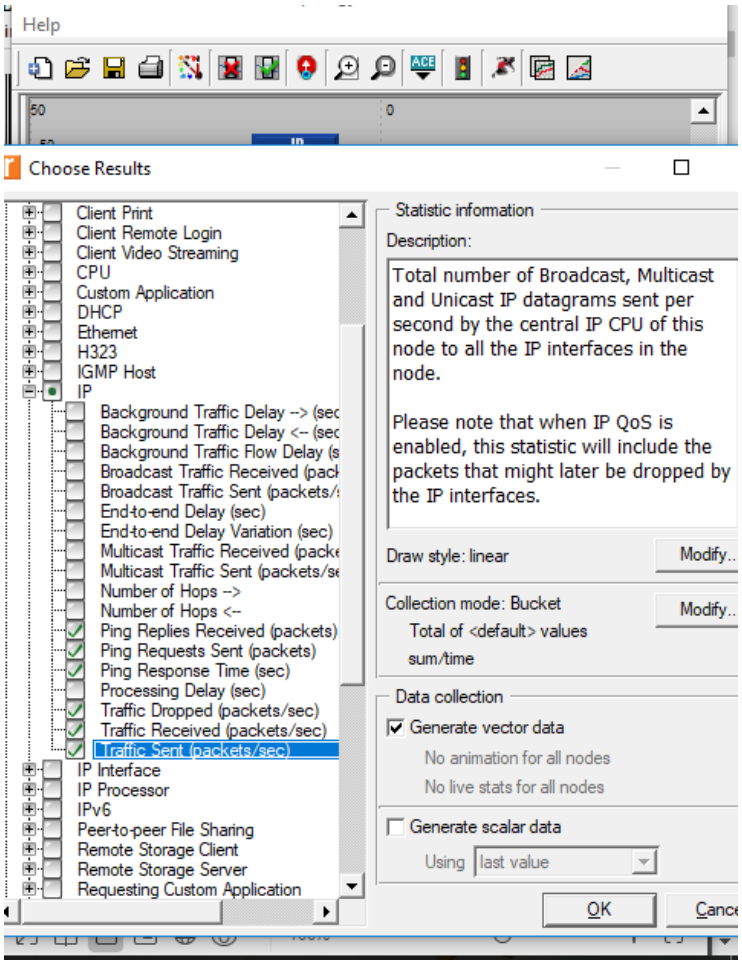


10. Right click on the blank background of simulator and choose the option "Choose the individual DES statistics"

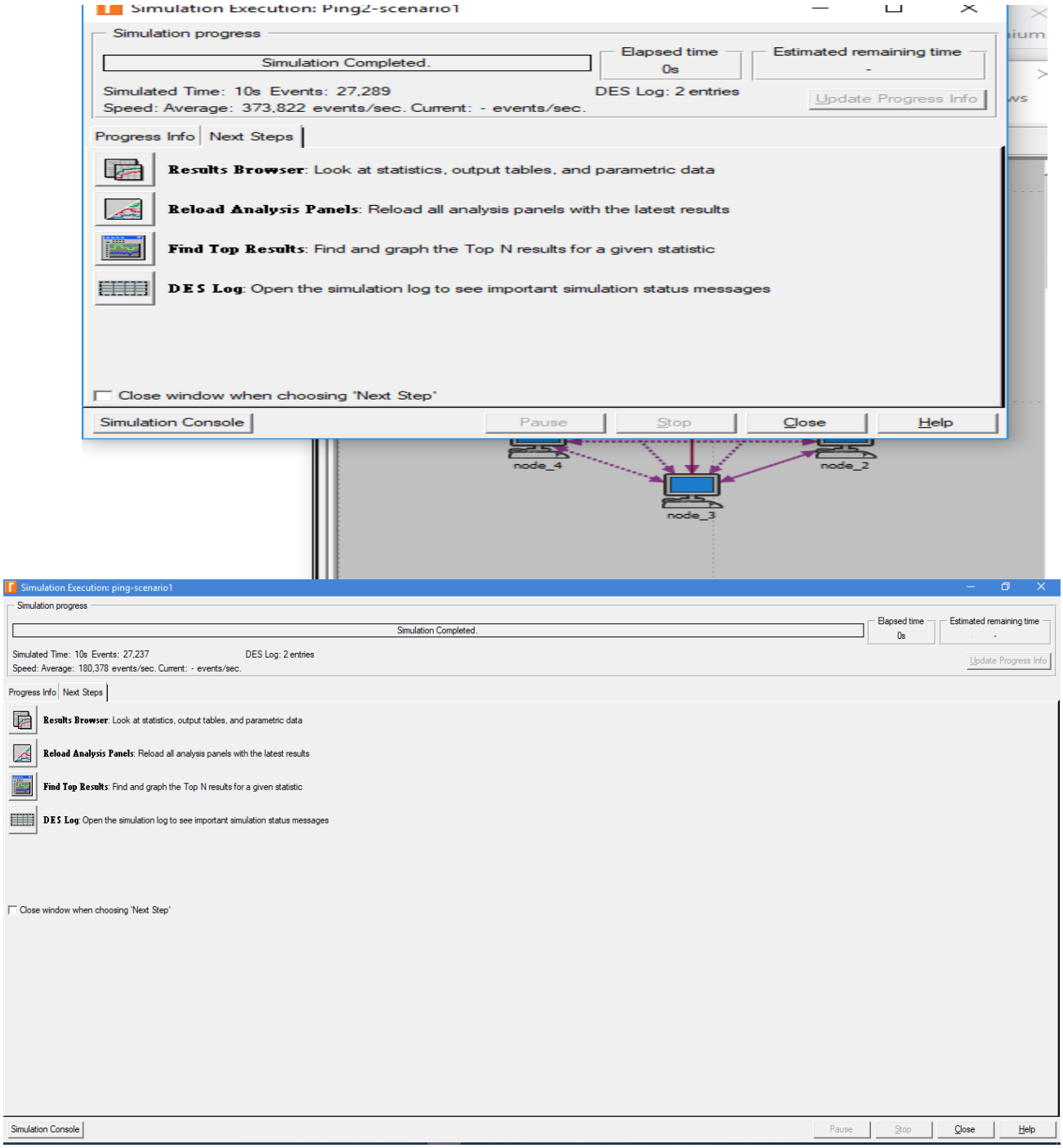


11. In the node statistics option, go to the IP section and select all the options related to Ping and Traffic.

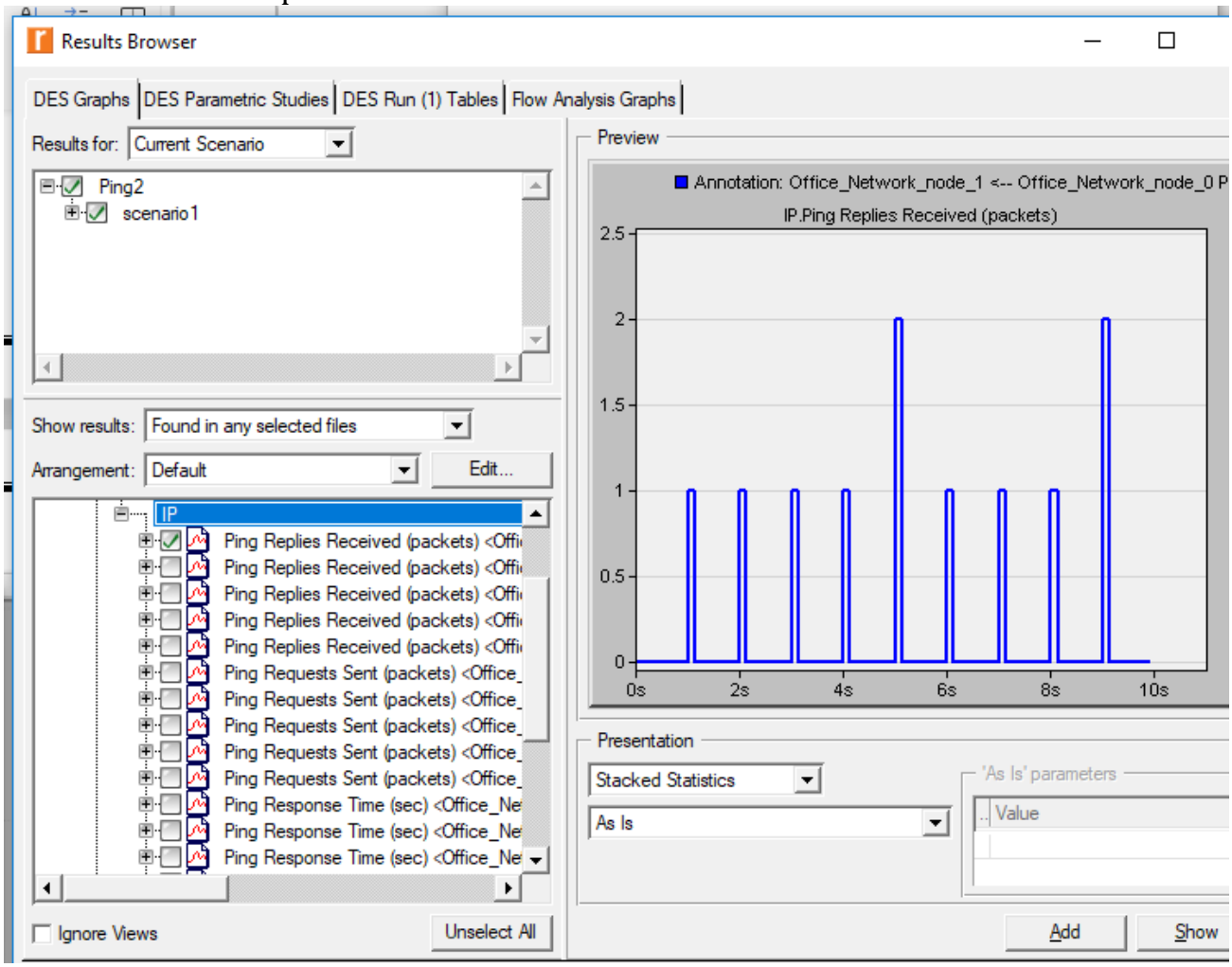
12.

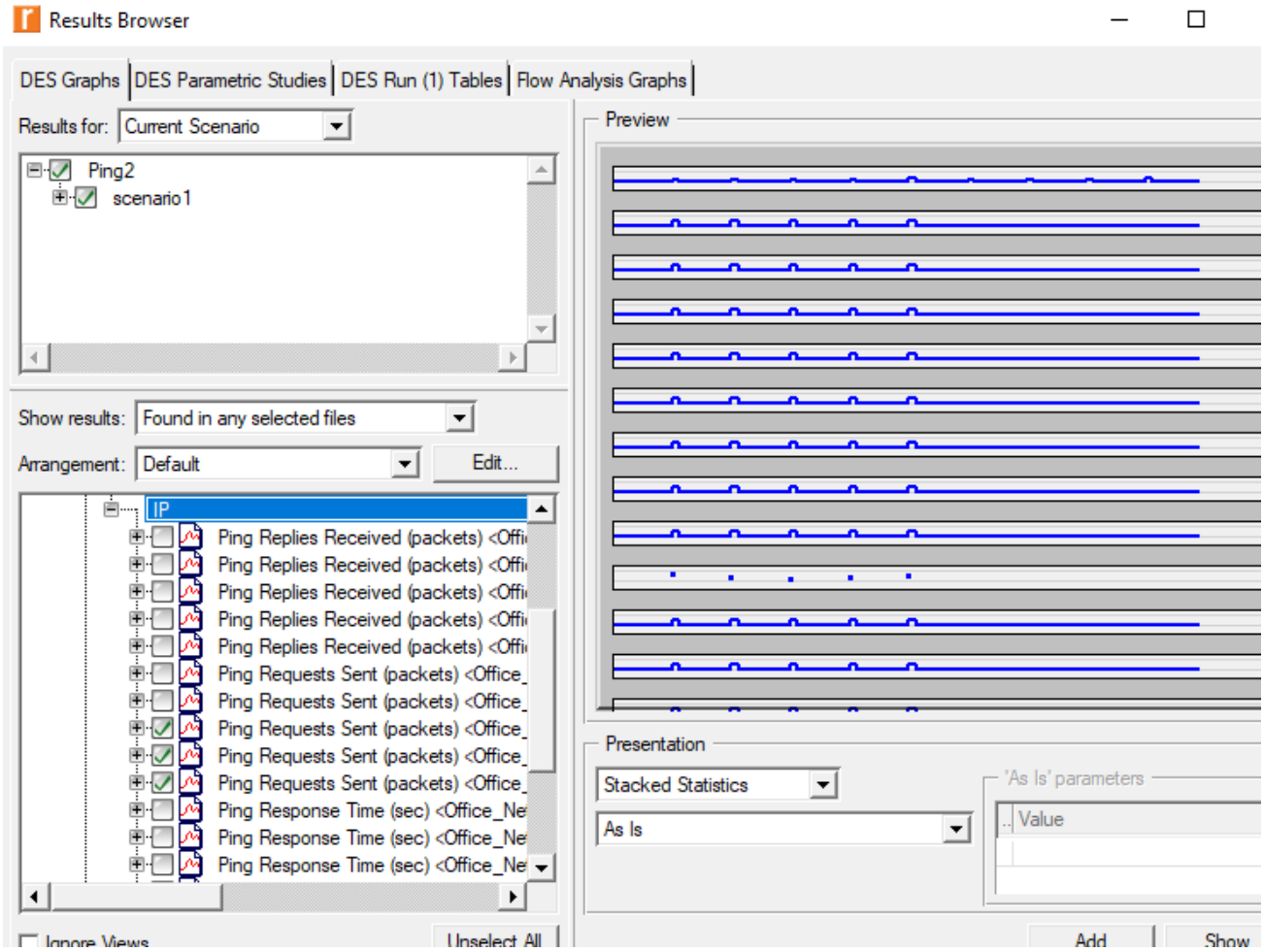

Help
Choose Results
Client Print
Client Remote Login
Client Video Streaming
CPU
Custom Application
DHCP
Ethernet
H323
IGMP Host
IP
Background Traffic Delay --> (sec
Background Traffic Delay <-- (sec
Background Traffic Flow Delay (s
Broadcast Traffic Received (pack
Broadcast Traffic Sent (packets/s
End-to-end Delay (sec)
End-to-end Delay Variation (sec)
Multicast Traffic Received (packe
Multicast Traffic Sent (packets/se
Number of Hops -->
Number of Hops <--
Ping Replies Received (packets)
Ping Requests Sent (packets)
Ping Response Time (sec)
Processing Delay (sec)
Traffic Dropped (packets/sec)
Traffic Received (packets/sec)
Traffic Sent (packets/sec)
IP Interface
IP Processor
IPv6
Peer-to-peer File Sharing
Remote Storage Client
Remote Storage Server
Requesting Custom Application
Statistic information
Description:
Total number of Broadcast, Multicast and Unicast IP datagrams sent per second by the central IP CPU of this node to all the IP interfaces in the node.
Please note that when IP QoS is enabled, this statistic will include the packets that might later be dropped by the IP interfaces.
Draw style: linear
Modify..
Collection mode: Bucket
Total of <default> values
sum/time
Modify..
Data collection
Generate vector data
No animation for all nodes
No live stats for all nodes
Generate scalar data
Using
last value
OK

13. Click on the Run button, specify the time duration as 10 seconds, click ok and we'll see the status of the simulation.



13. Then we will be redirected to a new dialog box with many options. Choose the Results Browser option and choose any random node and check the results which you want to see stacked up.



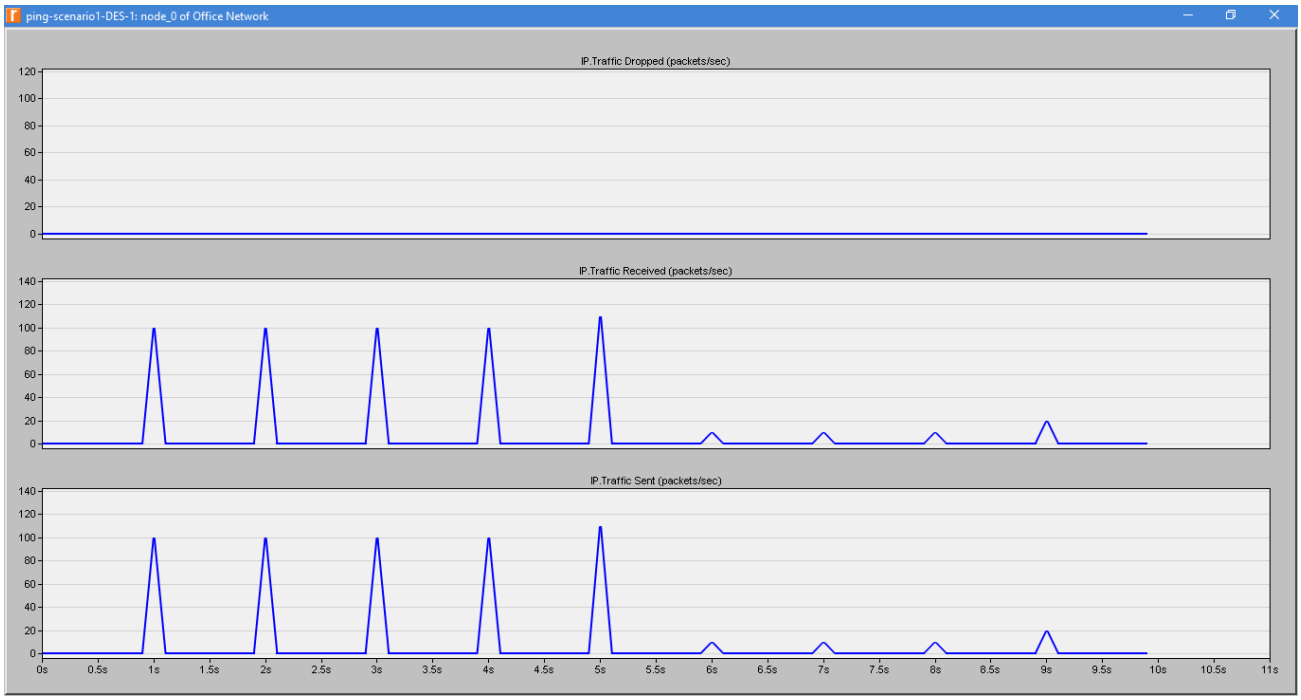


Explanation of the OPNET 17.5 Output for ICMP Ping:

The first graph shows the number of IP datagrams dropped per second by this node across all IP interfaces.

The next graph shows the total number of Broadcast, Multicast and Unicast IP datagrams received per second by this node from the network across all IP interfaces in this node.

The last graph shows the total number of Broadcast, Multicast and Unicast IP datagrams sent per second by the central IP CPU of this node to all the IP interfaces in the node.



RESULT:

Thus the procedure for basic network configurations with Ethernet, Switched LAN, Network Design, and ICMP Ping has been practiced using OPNET 17.5 and the output has been verified successfully.

Evaluation:

Parameter	Max Marks	Marks Obtained
	3	
Uniqueness of the Code	10	
Use of Comment lines and standard coding practices	2	
Viva	5	

Sub Total	20	
Completion of experiment on time	3	
Documentation	7	
Sub Total	10	
Signature of the faculty with Date		

Ex.No.8	IMPLEMENTATION OF CONCURRENT SERVERS USING MULTITHREADING

AIM:

To develop Concurrent server/Client application for Student Query System in Java

THEORY:

Concurrent Server:

A concurrent server has to be able to serve more than one client at a time. i.e. it iterates through each client and serves one request at a time. Alternatively, a server can handle multiple clients at the same time in parallel, and this type of a server is called a concurrent server.

Differences between Concurrent Server and Iterative Server:

Concurrent Server	Iterative Server
Handles multiple client at a time	Handles one at a time
It waits for client through accept() and it closes the connection when done	It waits for client through accept() and it goes back to listen for further upcoming clients

APIs and methods required to implement Concurrent Servers:

ServerSocket() – To get server up and running

Socket() – To accept clients and create clients

Thread() – To allot each client the server's functionality

PrintStream() – To write to socket stream

InputStream() – Socket's input stream

OutputStream() – Socket's Output stream

InetAddress() – Used to get the internet address

Adding Comments to Summation Server and Summation Client:

SummationServer.java:

```
import java.io.*;
import java.net.*;
class summationThread extends Thread{
Socket clientSocket;
summationThread(Socket cs){ clientSocket = cs; } //Copying incoming client connection
public void run( ){
try{
BufferedReader br = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream( ))); //Getting socket input stream
int count = Integer.parseInt(br.readLine( ));
int sum = 0;
for (int ctr = 1; ctr <= count; ctr++){
sum += ctr;
Thread.sleep(200);
}
PrintStream ps = new PrintStream(clientSocket.getOutputStream( )); //getting socket
                                                                    //output stream and prints the result
ps.println(sum);
ps.flush( );
clientSocket.close( );  //Closing connection
}
catch(Exception e){e.printStackTrace( );}
}
}
class summationServer{
public static void main(String[ ] args){
try{
int serverPort = Integer.parseInt(args[0]);
ServerSocket calcServer = new ServerSocket(serverPort); //Initializing server at port
```

```
while (true){
Socket clientSocket = calcServer.accept( ); //waiting for incoming connection
summationThread thread = new summationThread(clientSocket);
thread.start( ); //new thread process for client
}
}
catch(Exception e){e.printStackTrace( );}
}
}
```

SummationClient.java:

```
import java.io.*;
import java.net.*;
class summationClient{
public static void main(String[ ] args){
try{
InetAddress serverHost = InetAddress.getByName(args[0]);
int serverPort = Integer.parseInt(args[1]);
long startTime = System.currentTimeMillis( );
int count = Integer.parseInt(args[2]);
Socket clientSocket = new Socket(serverHost, serverPort); //creating new client with given
                                                          //ip and port
PrintStream ps = new PrintStream(clientSocket.getOutputStream()); //getting out stream
ps.println(count);
BufferedReader br = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream())); //getting input stream
int sum = Integer.parseInt(br.readLine());
System.out.println(" sum = "+sum);
long endTime = System.currentTimeMillis();
System.out.println(" Time consumed for receiving the feedback from the server:
"+(endTime-startTime)+" milliseconds");
clientSocket.close( );                                    //close connection
}
catch(Exception e){e.printStackTrace( );}
}
}
```

ALGORITHM:

Server:

1. Setting port address and running server in localhost
2. Storing it in a non primitive data type variable
3. Establishing the Connection
4. Server socket object is initialized and inside a while loop a socket object continuously accepts incoming connection.
5. Obtaining the Streams
6. The inputstream object and outputstream object is extracted from the current requests' socket object
7. Invoking the start()
8. Accepting further connections
9. Displaying served connection names

Client:

1. Creating a client socket with ip and port number
2. Obtaining the input and output stream of the socket
3. Query the server
4. Print the result
5. Close connection if needed

CODING:

Server Side Code:

```
import java.io.*;
import java.net.*;
import java.util.*;
class summationThread extends Thread{
Socket clientSocket;
summationThread(Socket cs){ clientSocket = cs; }
public void run( ){
try{
BufferedReader br = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream( )));
```

```
String name=br.readLine();
String i1=br.readLine();
String i2=br.readLine();
int gst=Integer.parseInt(i1);
int total=Integer.parseInt(i2);
float a=(gst*total)/100;
float sum=a+total;
Thread.sleep(200);
PrintStream ps = new PrintStream(clientSocket.getOutputStream( ));
ps.println(sum);
System.out.println("Served : "+name);
ps.flush( );
clientSocket.close( );
}
catch(Exception e){e.printStackTrace( );}
}
}
class summationServer{
public static void main(String[ ] args){
try{
System.out.print("Enter the port address");
Scanner s=new Scanner(System.in);
int serverPort =s.nextInt();
ServerSocket calcServer = new ServerSocket(serverPort);
while (true){
Socket clientSocket = calcServer.accept( );
summationThread thread = new summationThread(clientSocket);
thread.start( );
}
}
catch(Exception e){e.printStackTrace( );}
}
}
```

Client Side Code:

```
import java.io.*;
import java.util.*;
import java.net.*;
class summationClient{
public static void main(String[ ] args){
try{
InetAddress serverHost = InetAddress.getByName("localhost");
System.out.print("Enter the port address");
Scanner s=new Scanner(System.in);
int serverPort =s.nextInt();
long startTime = System.currentTimeMillis( );
System.out.println("Enter the name");
String name=s.next();
System.out.println("Enter the product id:\n1.Electronics\n2.Furnitures\n3.Garments");
String types=s.next();
System.out.print("Enter the gst");
int gst=s.nextInt();
System.out.println("Enter the total amount");
int total=s.nextInt();
Socket clientSocket = new Socket(serverHost, serverPort);
PrintStream ps = new PrintStream(clientSocket.getOutputStream());
ps.println(name);
ps.println(String.valueOf(gst));
ps.println(String.valueOf(total));
BufferedReader br = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
String sum = br.readLine();
System.out.println(" sum = "+sum);
long endTime = System.currentTimeMillis();
System.out.println(" Time consumed for receiving the feedback from the
server:"+(endTime-startTime)+" milliseconds");
clientSocket.close( );
}
catch(Exception e){e.printStackTrace( );}
```

```
}
}
```

SCREEN SHOTS:

Clients:

```
Enter the port address6000
Enter the name
vishwa
Enter the product id:
1.Electronics
2.Furnitures
3.Garments
1
Enter the gst20
Enter the total amount
2000
 sum = 2400.0
 Time consumed for receiving the feedback from the server:11493 milliseconds

Process completed.
```

Server:

```
----------------------Configuratic
Enter the port address6000
Served : yajith
Served : vishwa
```

RESULT:

Thus, the application for Student Query System is deployed as a Concurrent Server/Client in Java and the results are verified.

Evaluation:

Parameter	Max Marks	Marks Obtained
Complexity of the Application	10	
Clarity of Algorithm	3	
Use of Comment lines and standard coding practices	2	
Viva	5	
Sub Total	20	
Completion of experiment on time	3	
Documentation	7	
Sub Total	10	
Signature of the faculty with Date		

Ex.No.9	IMPLEMENTATION OF GROUPCHAT APPLICATION USING MULTICAST SOCKETS

AIM:

To develop group chat application using multicast sockets and multithreading

THEORY:

Multicasting:

Multicasting in computer network is a group communication, where a sender(s) send data to multiple receivers simultaneously. It supports one – to – many and many – to – many data transmission across LANs or WANs. Through the process of multicasting, the communication and processing overhead of sending the same data packet or data frame in minimized.

Addressing in Multicasting:

A multicast address is a logical identifier for a group of hosts in a computer network that are available to process datagrams or frames intended to be multicast for a designated network service. An IP multicast group address is used by sources and the receivers to send and receive multicast messages. Sources use the group address as the IP destination address in their data packets. Receivers use this group address to inform the network that they are interested in receiving packets sent to that group. Multicast addressing can be used in the link layer (layer 2 in the OSI model), such as Ethernet multicast, and at the internet layer (layer 3 for OSI) for Internet Protocol Version 4 (IPv4) or Version 6 (IPv6) multicast. The IPv4 and IPv6 are known as class D addressing used mainly for multicasting.

APIs and methods required to implement Multicast Sockets:

API: class java .net.datagram, class java.util.*

Connection establishment Methods: Inetaddress, getByName(),MulticastSocket, joinGroup(),leaveGroup()

Thread: Thread(),start(),run()

Chat establishment: getBytes(),DatagramPacket,send(object),receive(object)

Differences between Group chat and Chat in Client/Server Mode

Group chat	Chat in client/server mode
Group chat is based on UDP	Chat in client/server mode is based on either TCP or UDP
Multicast addressing	Unicast addressing
Multicast can be of either many – many or one -many	Unicast in purely one-one(server-client)
Datagram packets are sent and received from source address to destination address	Unicast packets are encapsulated and delivered using TCP via the Transport layer.

ALGORITHM:

1.Chat using Client/Server Mode

Client side:

- Class NormalChatClient is created.
- A socket is initilised with host name and port number.
- In try block,
- DataInputStream ,DataOutputStream ,BufferedReader are initialised.
- Two strings servermessage and clientmessage are initilised.
- While loop is enabled until clientmessage reads 'bye'.
- In while loop, the string entered by the user is stored in clientmessage and written using outstream.
- The string is read by servermessage using instream.readUTF() and the servermessage is printed.
- Socket,instream,outstream are closed followed by catch block.

Server side:

- Class NormalChatServer in initialised.
- In main(),try block is initialised.

- A serversocket is created with port number.
- Incoming requests are accepted by accept() passed to client object.
- Loop is enabled.
- DataInputStream ,DataOutputStream ,BufferedReader are initialised.
- Data is read in clientmessage through instream and printed.
- The user entered string is written in servermessage.
- Socket,instream,outstream are closed followed by catch block.

2.Group Chat

- start
- Class GroupChat is initialised.
- Variables are initialised.
- In main function, the arguments passed during execution is checked by if loop.
- In else block, a try block is initialised.
- In try block,Inetaddress method is initialised with getByname().
- The argument is passed to 'port' of type int.
- Scanner is initialised and the user entered string is stored under name.
- Multicast socket is initialised with port.
- Clients are added using joingroup().
- A new thread is initialised with start() method.
- In a while loop, the messages are read and stored in message string.
- If the string equals exit, leavegroup() is initialised and the socket is closed.
- Else, the string is converted into byte array and is stored in buffer for sending.
- Datagram packet is initialised with buffer, buffer length,inetaddress,port number.
- The datagram packet is sent to the port number followed by catch block and the class is closed.
- A new class readthread implementing runnable is created.
- A constructor is initialised with socket,inetaddress object and port.
- The values are inherited from server class using this keyword.
- Run() Is initialised.
- In while loop with condition finished=true, the datagram packet is initialised for receiving datagram from the user, a string is initialised.

- In a try block,the datagram is received and converted into string from byte array by string() .
- In an if block ,if the string!= name from groupchat class ,it is printed.
- Catch block is initialised and exceptions are handled.
- End

CODING:

1. **Chat using Client/Server Mode**

CLIENT:

```
import java.net.*;
import java.io.*;
public class NormalChatClient
{
 public static void main(String args[])
 {
   try
    {
     Socket client = new Socket("127.0.0.1",8888);
     System.out.println("Connecting..");
     System.out.println("Entered Waiting room");
     System.out.println("Queue:100+");
     System.out.println("Queue:1");
     System.out.println("Connected.");
     System.out.println("Start Typing");
    DataInputStream in=new DataInputStream(client.getInputStream());
    DataOutputStream out=new DataOutputStream(client.getOutputStream());
    BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
    String clientmessage= "",servermessage="";
    while(!clientmessage.equals("bye"))
      {
       clientmessage=br.readLine();
       out.writeUTF(clientmessage);
```

```
        out.flush();
        servermessage=in.readUTF();
        System.out.println(servermessage);
       }
     client.close();
     in.close();
     out.close();
    }
    catch(Exception e)
   {
    System.out.println("Connection lost");
   }
 }
}
```

SERVER:

```
import java.net.*;
import java.io.*;
public class NormalChatServer
{
 public static void main(String args[])
  {
   try
    {
     ServerSocket server = new ServerSocket(8888);
     Socket client=server.accept();
     System.out.println("Connected");
     DataInputStream in=new DataInputStream(client.getInputStream());
     DataOutputStream out=new DataOutputStream(client.getOutputStream());
     BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
     String clientmessage="",servermessage="";
     while(!clientmessage.equals("bye"))
      {
        clientmessage=in.readUTF();
        System.out.println(clientmessage);
        servermessage=br.readLine();
```

```
        out.writeUTF(servermessage);
      }
     out.close();
     in.close();
     server.close();
   }
   catch(Exception e)
   {
    System.out.println(e);
   }
  }
}
```

2.Group Chat

```
import java.net.*;
import java.io.*;
import java.util.*;
public class GroupChat
{
   private static final String TERMINATE = "Exit";
   static String name;
   static volatile boolean finished = false;

   public static void main(String[] args)
   {

         try
         {
            InetAddress group = InetAddress.getByName("224.45.0.2");
            int port = 8888;
            Scanner sc = new Scanner(System.in);
            System.out.print("Enter your name: ");
            name = sc.nextLine();
            MulticastSocket socket = new MulticastSocket(port);
            socket.joinGroup(group);
```

```
            Thread t = new Thread(new  ReadThread(socket,group,port));
            t.start();
            System.out.println("Start typing messages...\n");
            while(true)
            {
               String message;
               message = sc.nextLine();
               if(message.equalsIgnoreCase(GroupChat.TERMINATE))
               {
                  finished = true;
                  socket.leaveGroup(group);
                  socket.close();
                  break;
               }
               message = name + ": " + message;
               byte[] buffer = message.getBytes();
               DatagramPacket datagram = new
DatagramPacket(buffer,buffer.length,group,port);
               socket.send(datagram);
            }
         }
         catch(SocketException se)
         {
            System.out.println("Error creating socket");
            se.printStackTrace();
         }
         catch(IOException ie)
         {
            System.out.println("Error reading/writing from/to socket");
            ie.printStackTrace();
         }
      }
}
class ReadThread implements Runnable
{
```

```
   private MulticastSocket socket;
   private InetAddress group;
   private int port;
   private static final int MAX_LEN = 1000;
   ReadThread(MulticastSocket socket,InetAddress group,int port)
   {
      this.socket = socket;
      this.group = group;
      this.port = port;
   }

   @Override
   public void run()
   {
      while(!GroupChat.finished)
      {
            byte[] buffer = new byte[ReadThread.MAX_LEN];
            DatagramPacket datagram = new
DatagramPacket(buffer,buffer.length,group,port);
            String message;
         try
         {
            socket.receive(datagram);
            message = new String(buffer,0,datagram.getLength(),"UTF-8");
            if(!message.startsWith(GroupChat.name))
               System.out.println(message);
         }
         catch(IOException e)
         {
            System.out.println("Socket closed!");
         }
      }
   }
}
```

SCREEN SHOTS:

1. Chat using Client/Server Mode

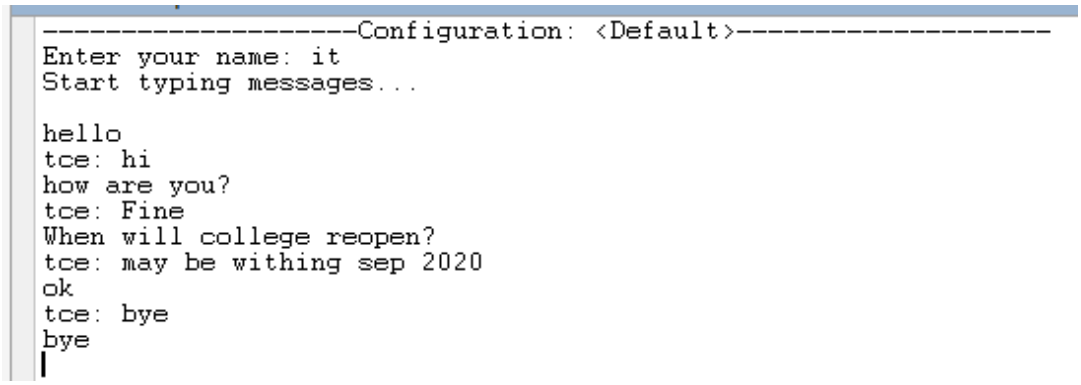
```
--------------------Configuration: <Default>--------------------
Connected
Hello I am From Tce
How may i help you?
Can you navigate me to IT dept?
yes
Navigate me
Go straight untill you reach cs dept and turn right.walk 10m and your destination on your left.
ok thank you
welcome
```

```
--------------------Configuration: <Default>--------------------
Connecting..
Entered Waiting room
Queue:100+
Queue:1
Connected.
Start Typing
Hello I am From Tce
How may i help you?
Can you navigate me to IT dept?
yes
Navigate me
Go straight untill you reach cs dept and turn right.walk 10m and your destination on your left.
ok thank you
welcome
```

2.Group Chat

```
----------------------Configuration: <Default>--------------
Enter your name: tce
Start typing messages...

it: hello
hi
it: how are you?
Fine
it: When will college reopen?
may be withing sep 2020
it: ok
bye
it: bye
```



RESULT:

Thus, the programs for Group chat and Chat in Client/Server Mode are implemented in Java and the results are verified.

Evaluation:

Parameter	Max Marks	Marks Obtained
Comparison between Chat in Client /Server mode and Group chat	10	
Uniqueness in Coding	10	
Sub Total	20	
Completion of experiment on time	3	
Documentation	7	
Sub Total	10	
Signature of the faculty with Date		

Ex.No.10	SIMULATION OF RIP PROTOCOL USING OPNET RIVERBED

AIM:

To simulate the Routing Information Protocol using Opnet Riverbed 17.5

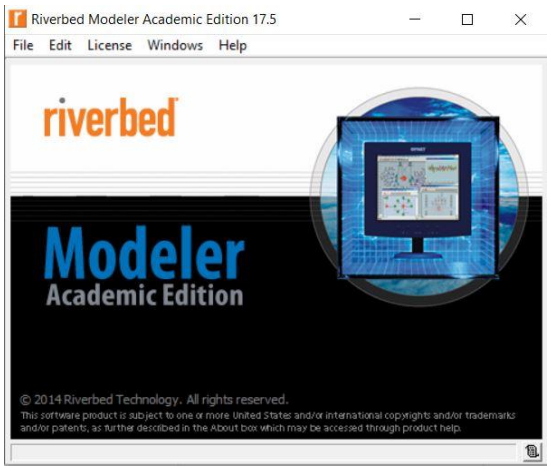
THEORY:

RIP:

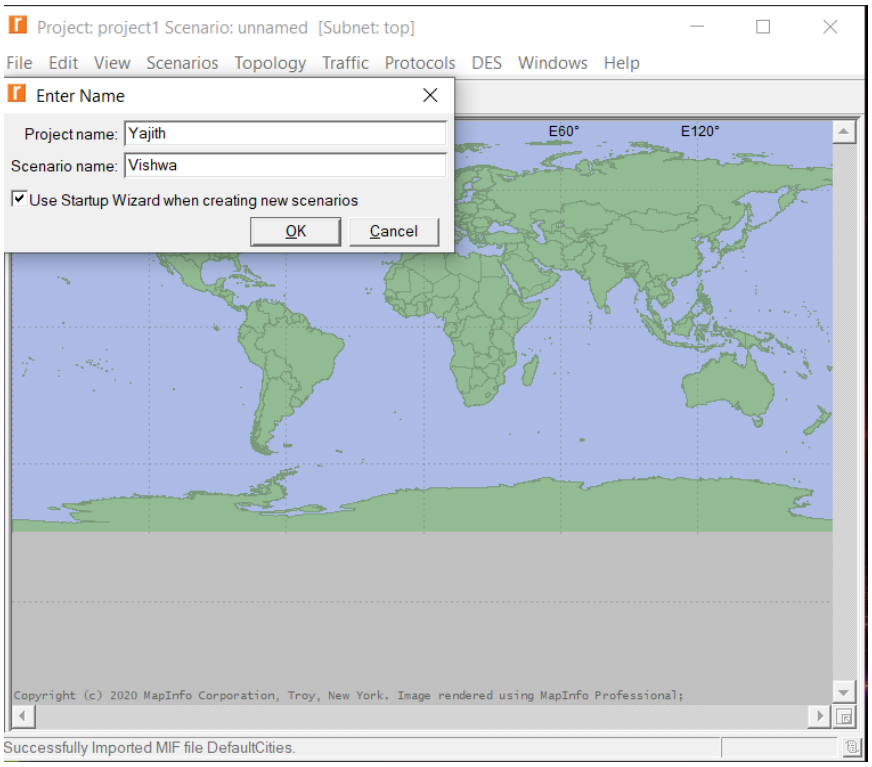
Routing Information Protocol (RIP) is a dynamic routing protocol which uses hop count as a routing metric to find the best path between the source and the destination network. It is a distance vector routing protocol which has AD value 120 and works on the application layer of OSI model. RIP uses port number 520.

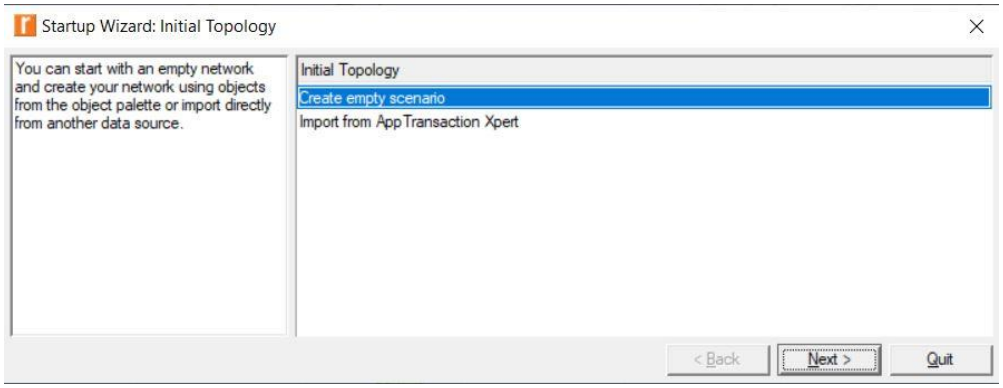
Installation and Procedure:

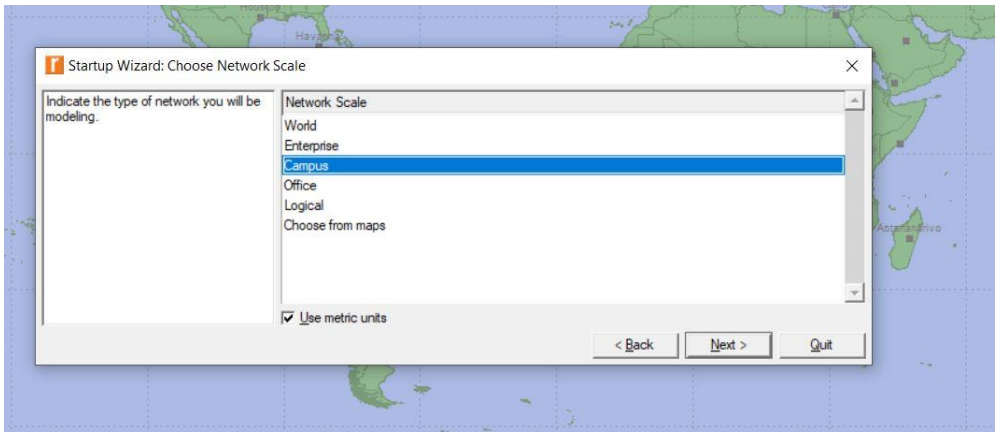
1. Download OPNET from the website https://cms-api.riverbed.com/portal/community_home
2. Run the Installer and proceed next. Click on License Agreement and click next. After Installation give finish.
3. Open OPNET and create a new project

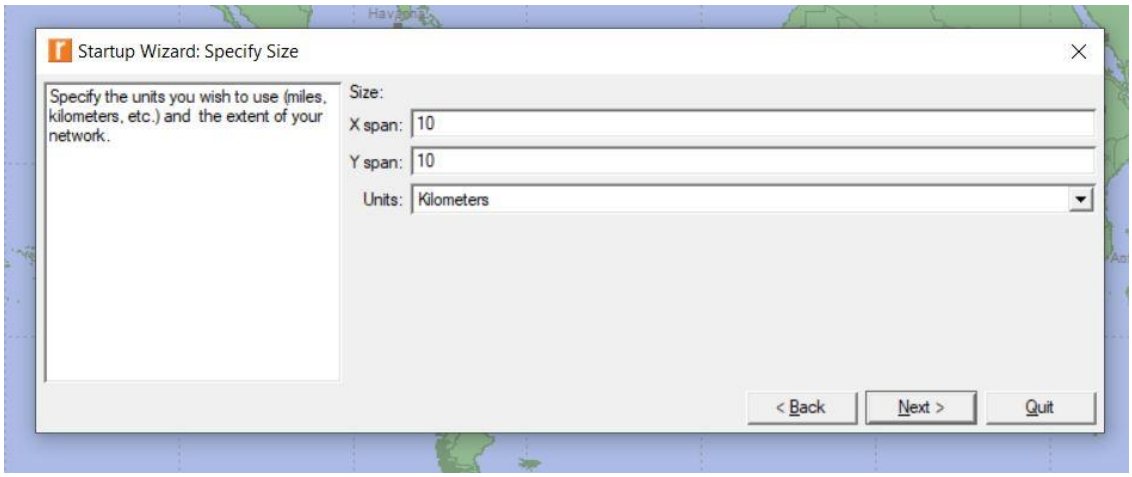


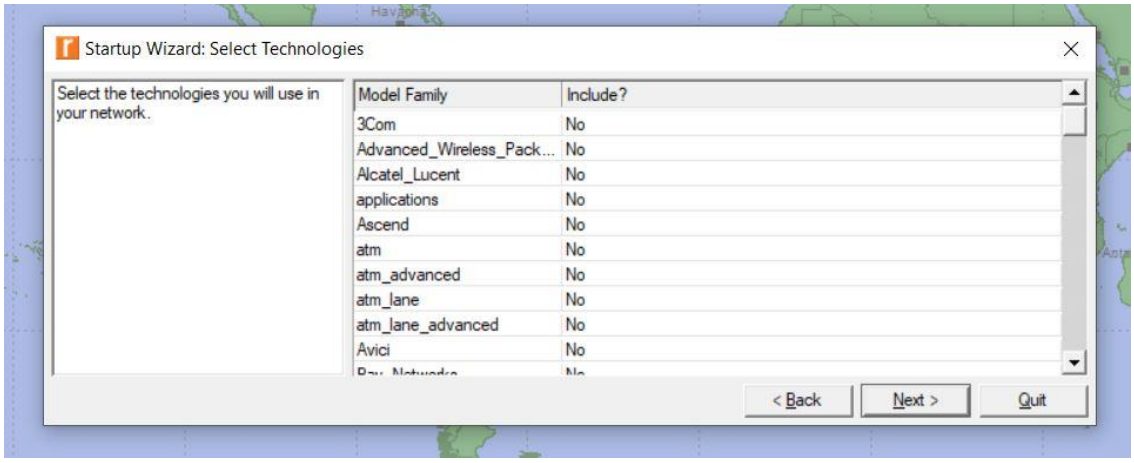
4. Give project name and scenario name
5. In the Startup Wizard give Create empty scenario and click next.
6. In the network scale select Campus and give next
7. Accept the defaults and proceed next. At last give finish. The project window is opened in OPNET.

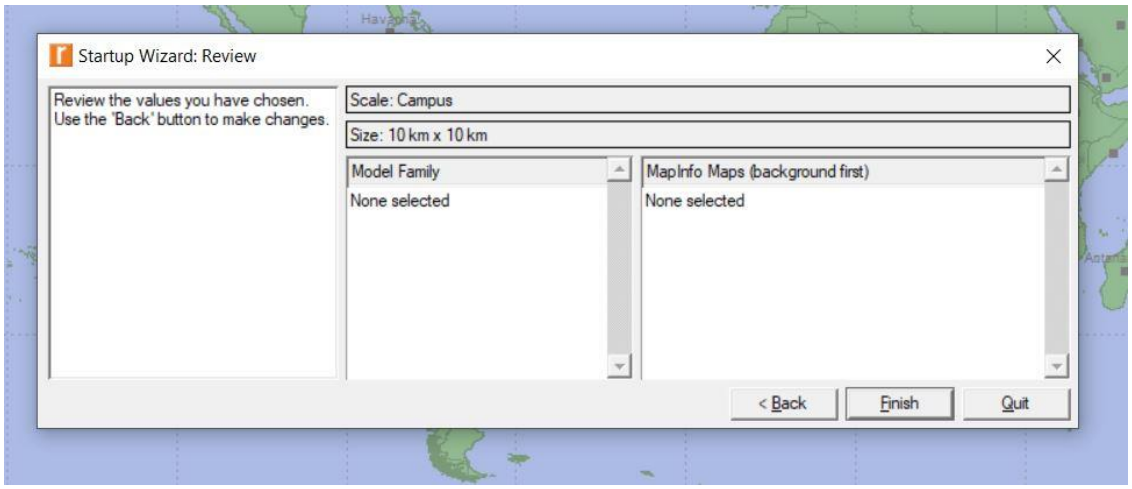




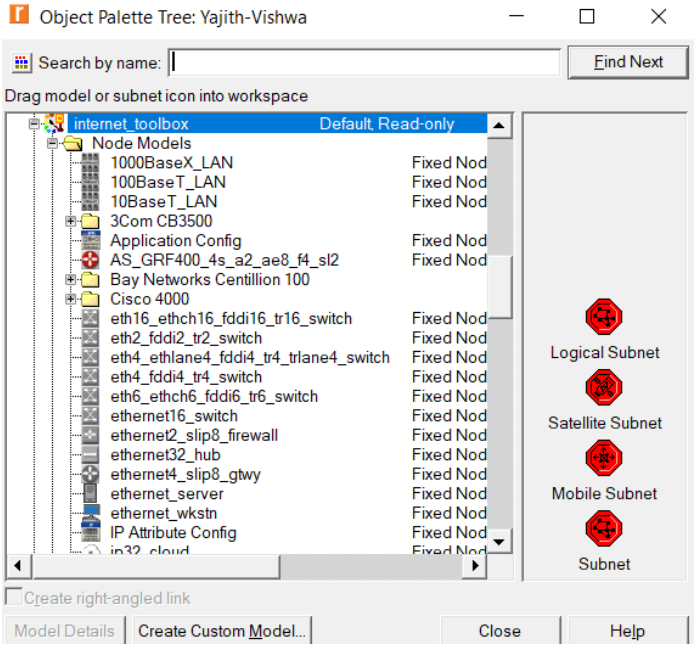

Startup Wizard: Choose Network Scale
Indicate the type of network you will be modeling.
Network Scale
World
Enterprise
Campus
Office
Logical
Choose from maps
Use metric units
< Back
Next >
Quit

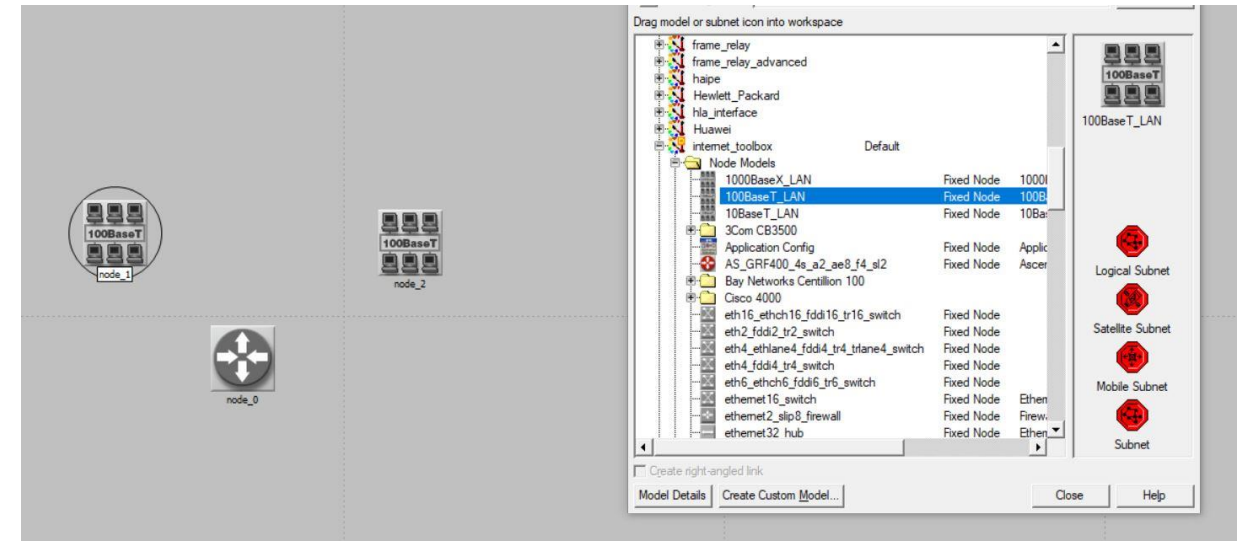

Startup Wizard: Specify Size
Specify the units you wish to use (miles, kilometers, etc.) and the extent of your network.
Size:
X span: 10
Y span: 10
Units: Kilometers
< Back
Next >
Quit

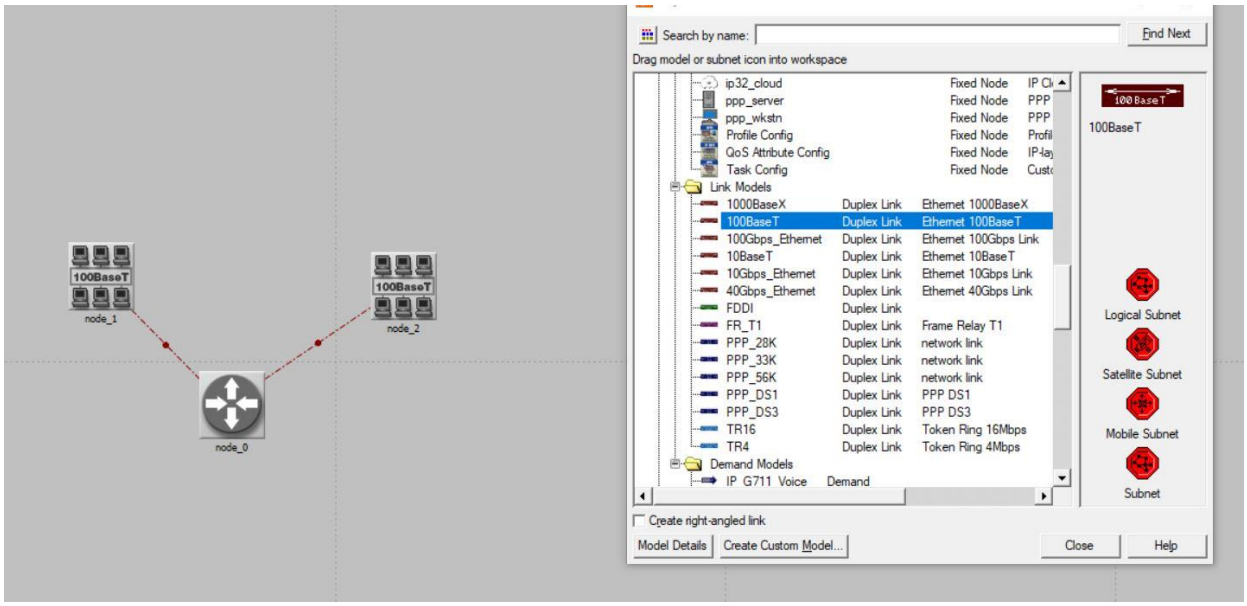

Startup Wizard: Select Technologies
Select the technologies you will use in your network.
Model Family
Include?
3Com No
Advanced_Wireless_Pack... No
Alcatel_Lucent No
applications No
Ascend No
atm No
atm_advanced No
atm_lane No
atm_lane_advanced No
Avici No
< Back
Next >
Quit



8. Open Object Palette and select ethernet4 router
9. Click and drag it to the project
10. Click and drag two 100BaseT_LAN to the project
11. For connecting the router and lan use 100BaseT duplex link

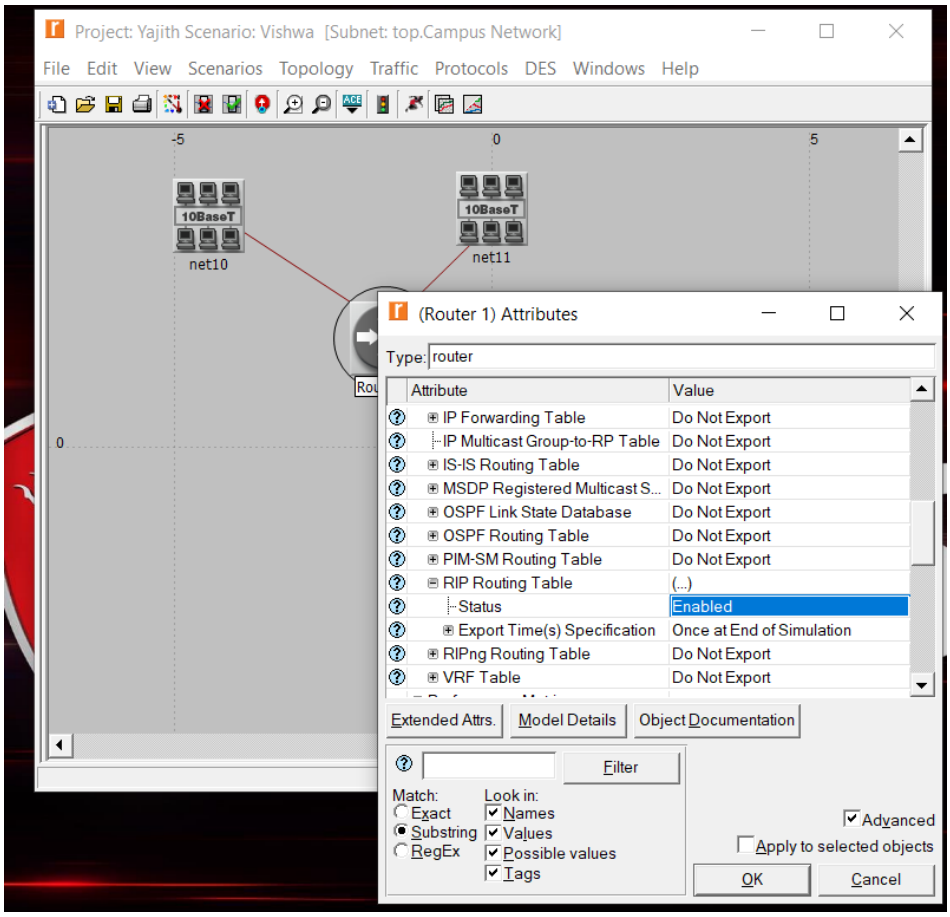






12. Right click the Router and Rename it as “Router”

13. Right click the router and select Edit Attributes



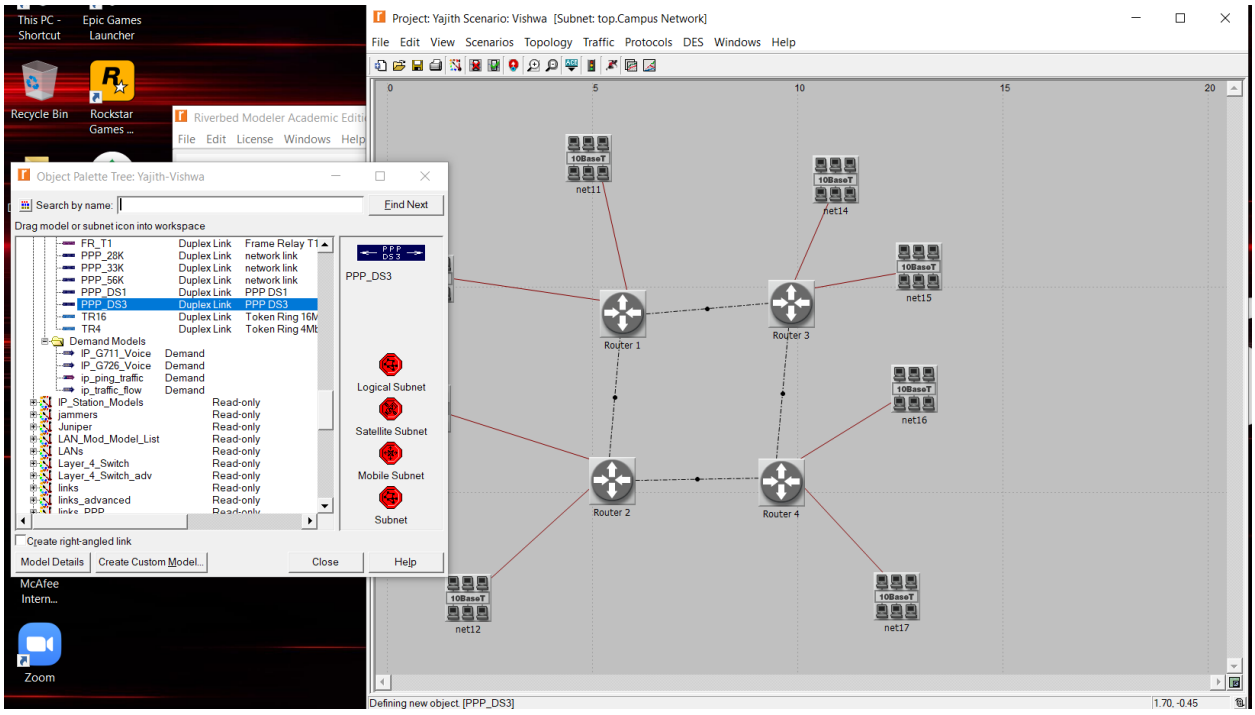
14. In the RIP Routing Table set Status = Enabled

15. In the Loop back interfaces set Number of Rows = 1

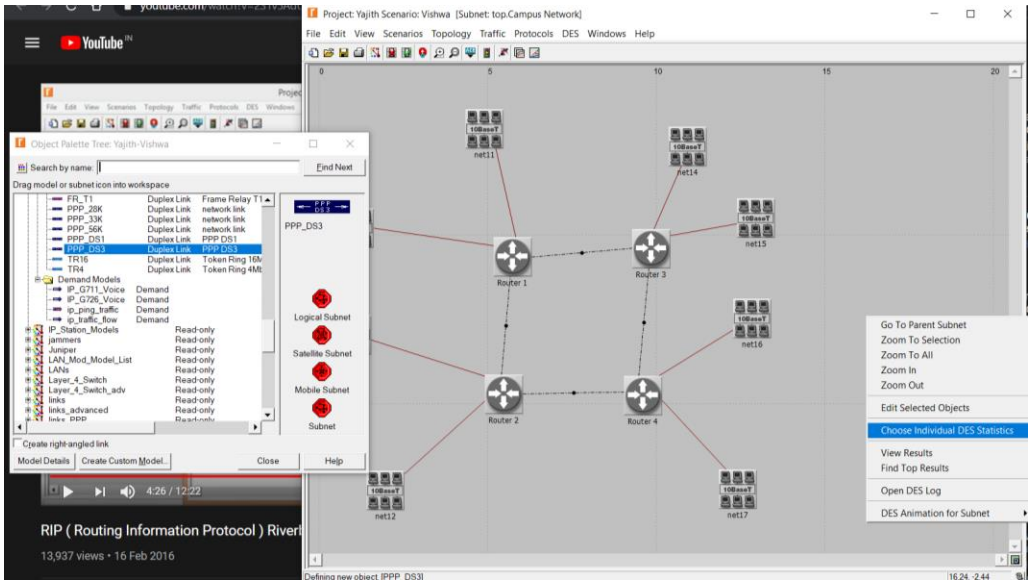
16. Select the router and two nodes and copy it

17. Paste it for 3 times to make a network

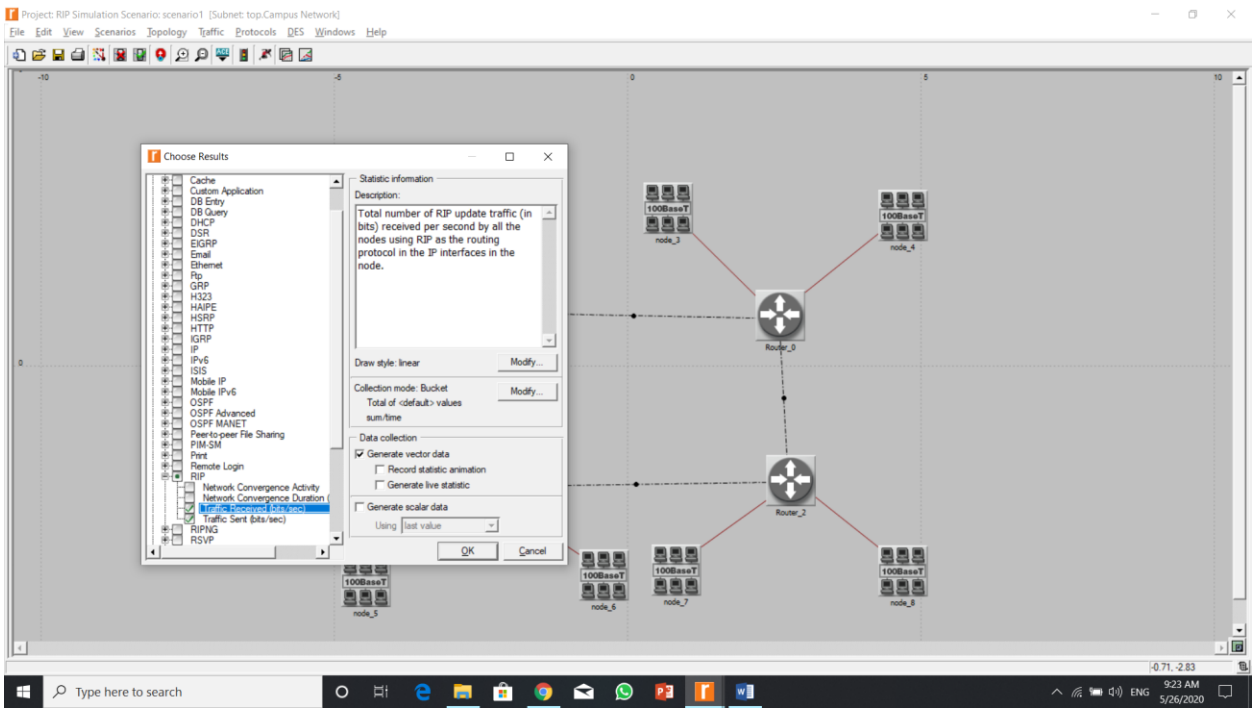
18. Open Object Palette and click on PPP DS3 duplex link to connect all the routers



19. Right click inside the workspace and choose “Choose individual DES Statistics”

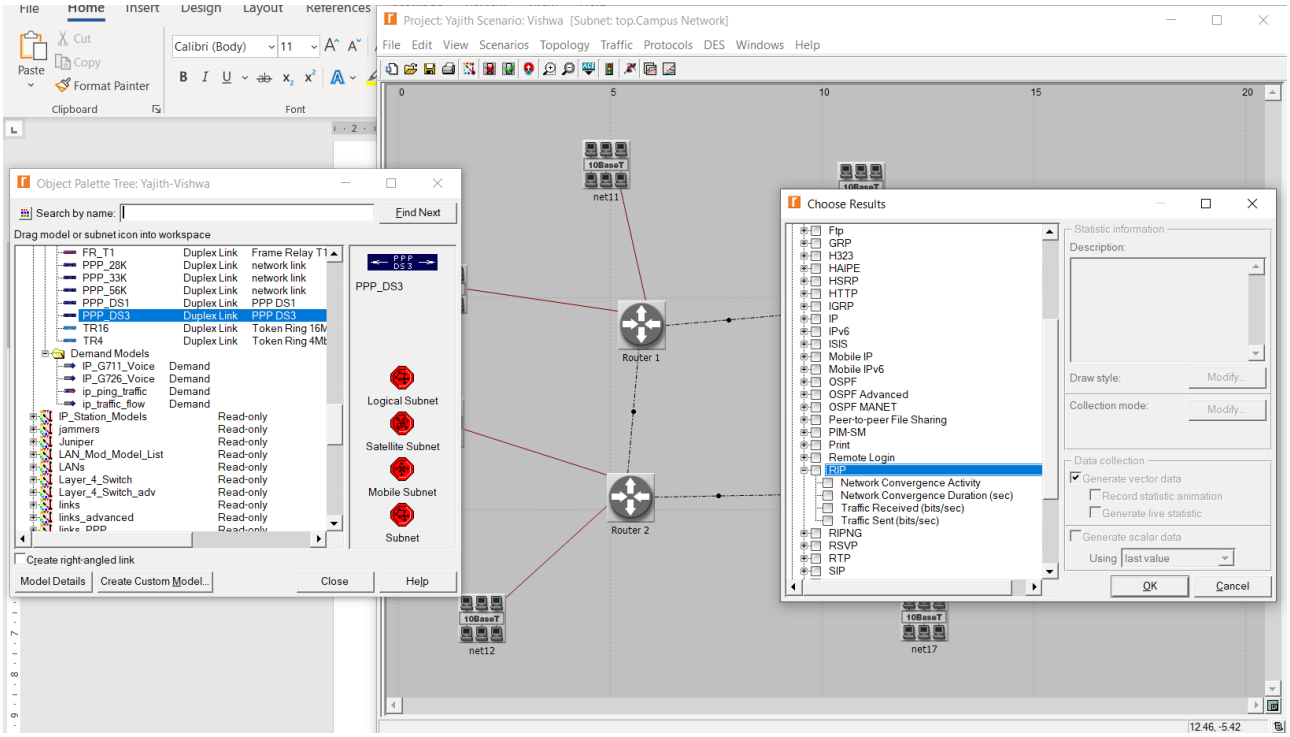


20. In the Choose Results box navigate to RIP and check Traffic Received, Traffic Sent

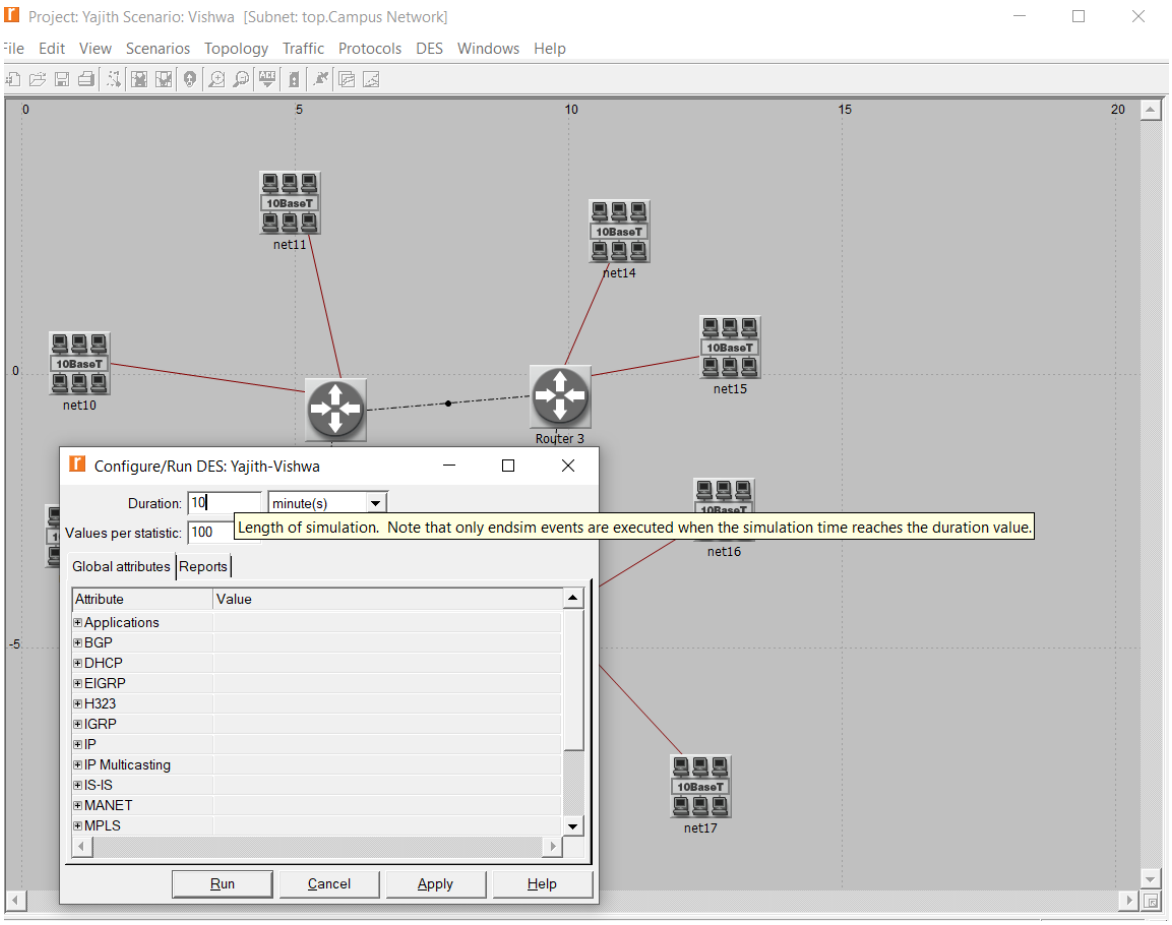


21. Expand the next cell and navigate to Route Table

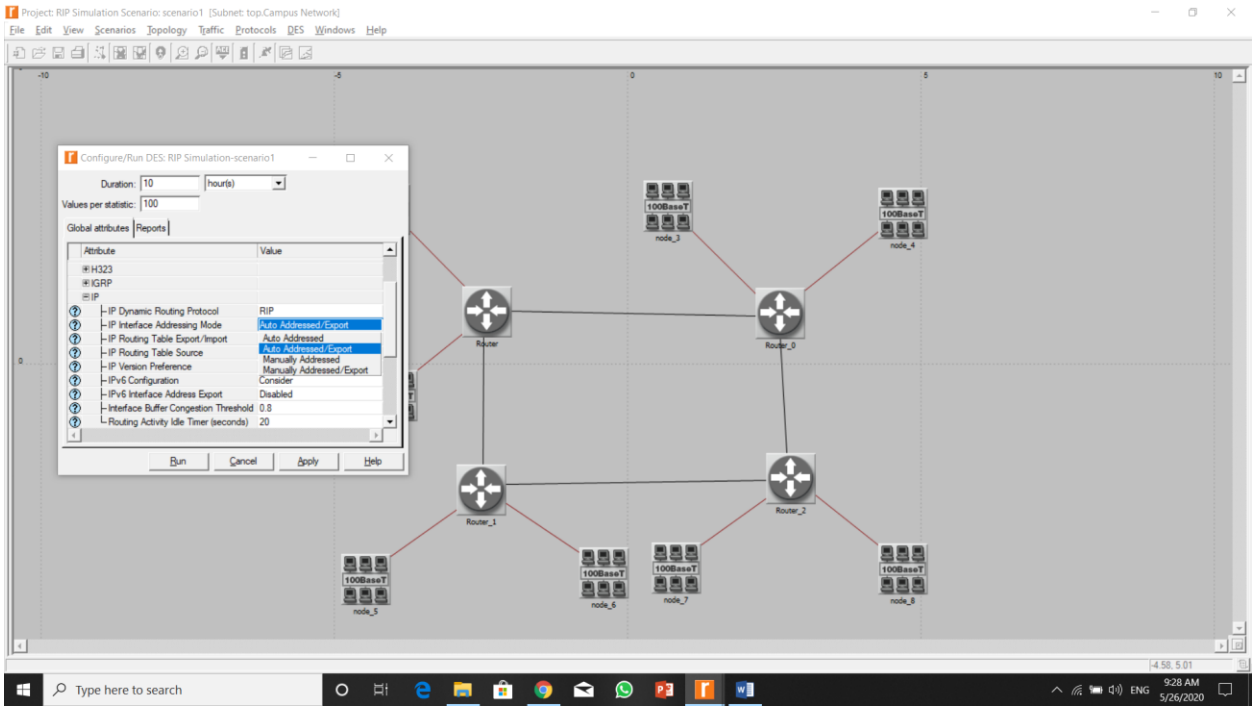
22. Check total number of updates and give OK.



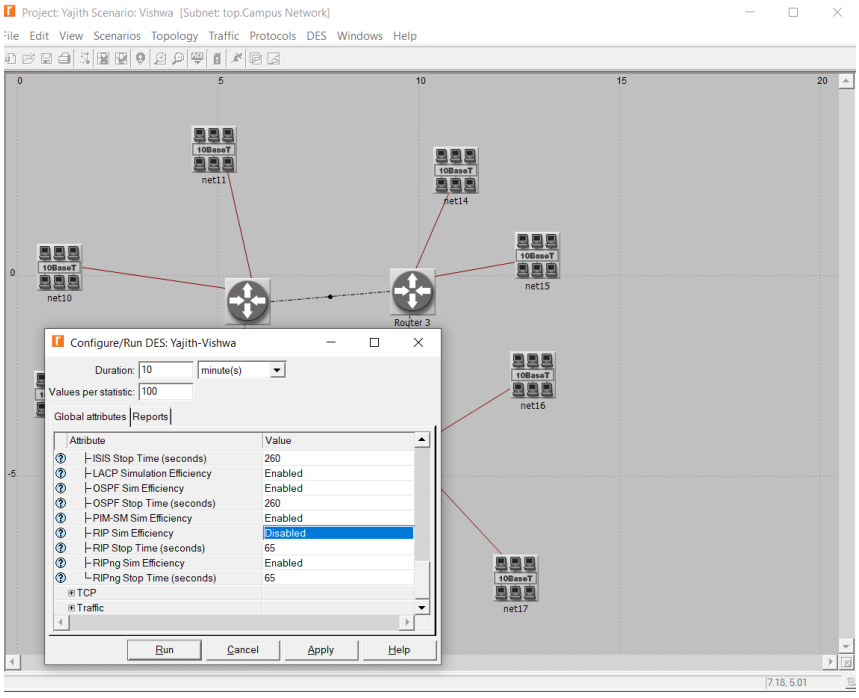
23. Run the simulation. Set duration to some time.



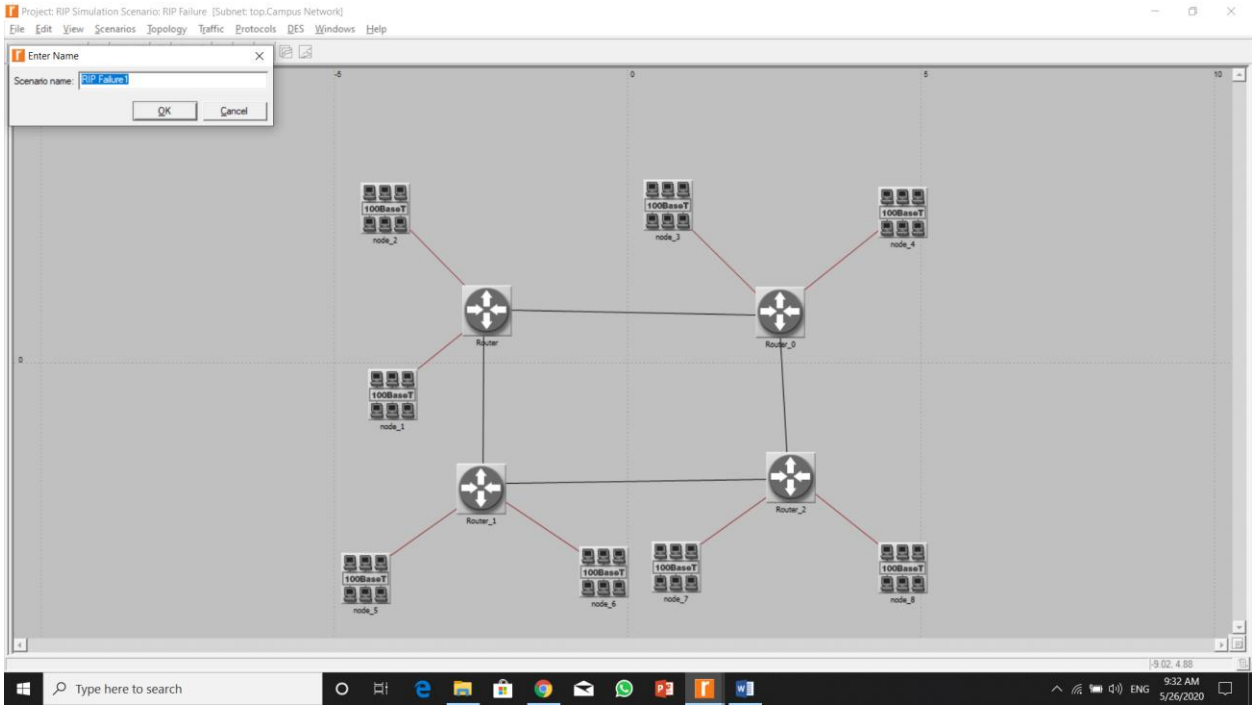
24. Under IP set IP Interface Addressing Mode to Auto Addressed/Export



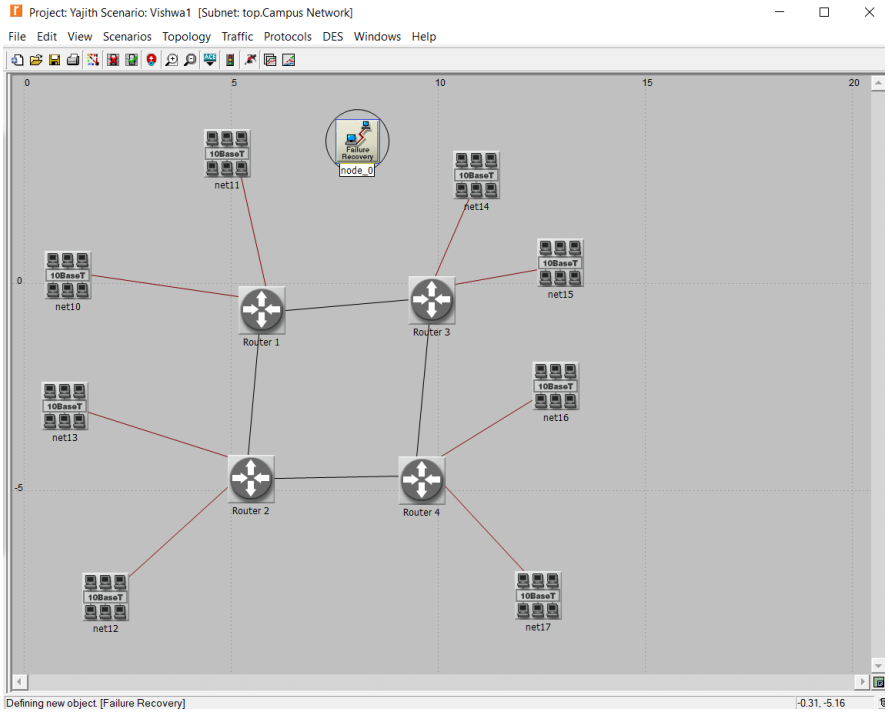
25. Set RIP sim efficiency to Disabled and run it.



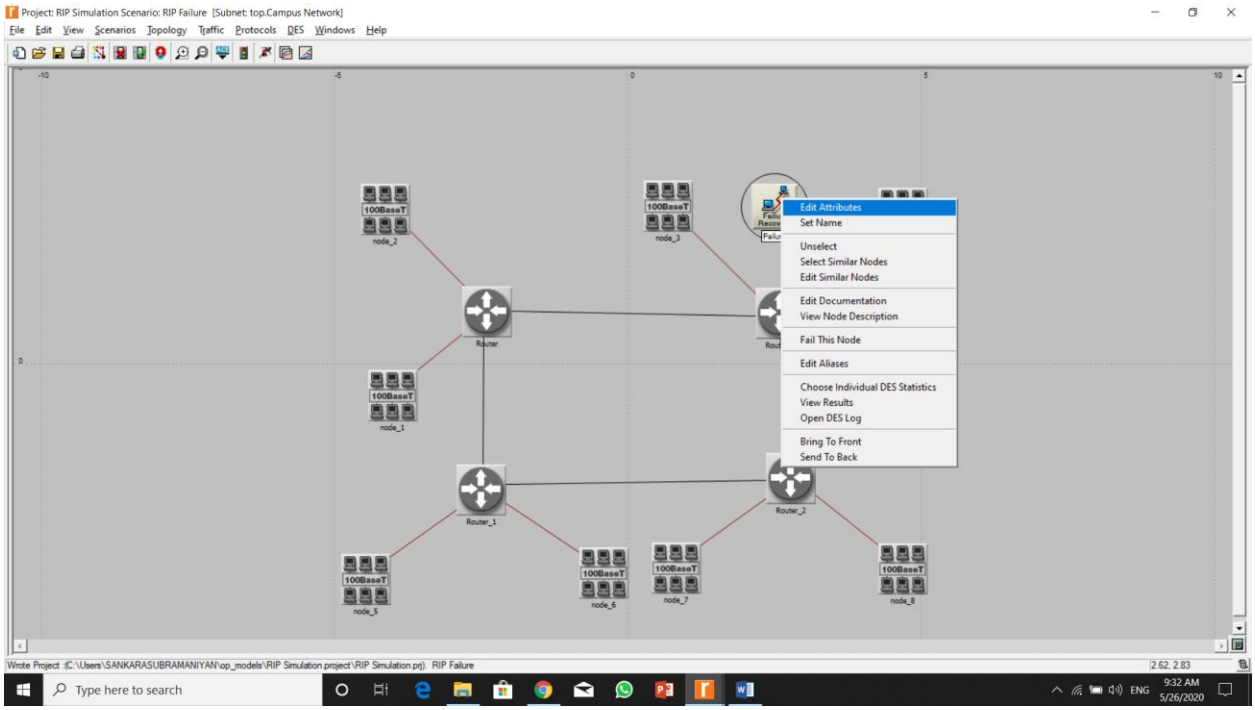
26. Duplicate the scenario and give the name as RIP Failure



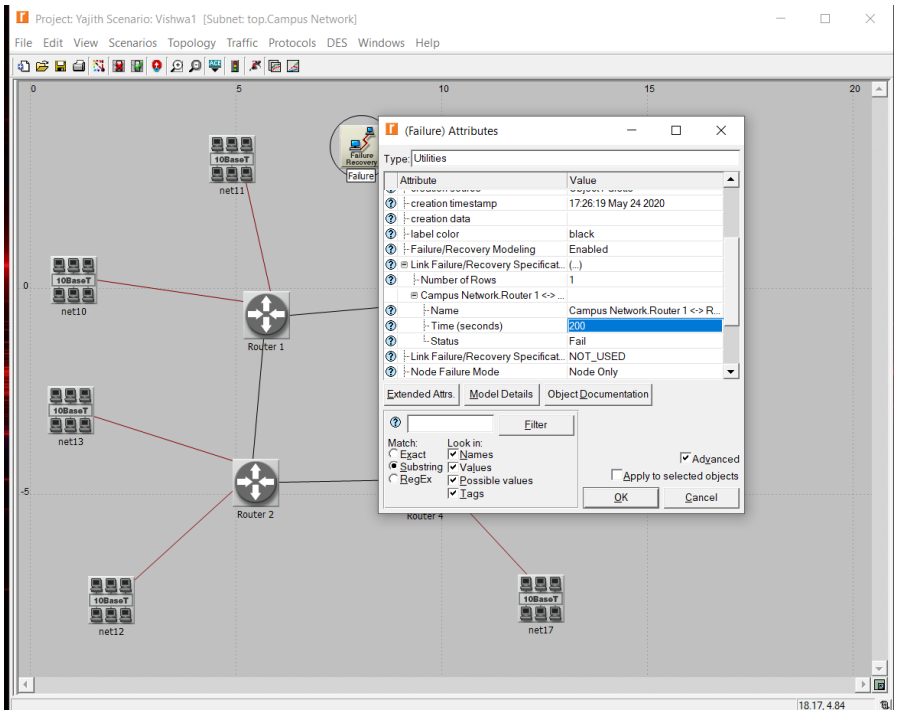
27. Select Failure Recovery from Object Palette and import to our project.



28. Right click it and select Edit Attributes

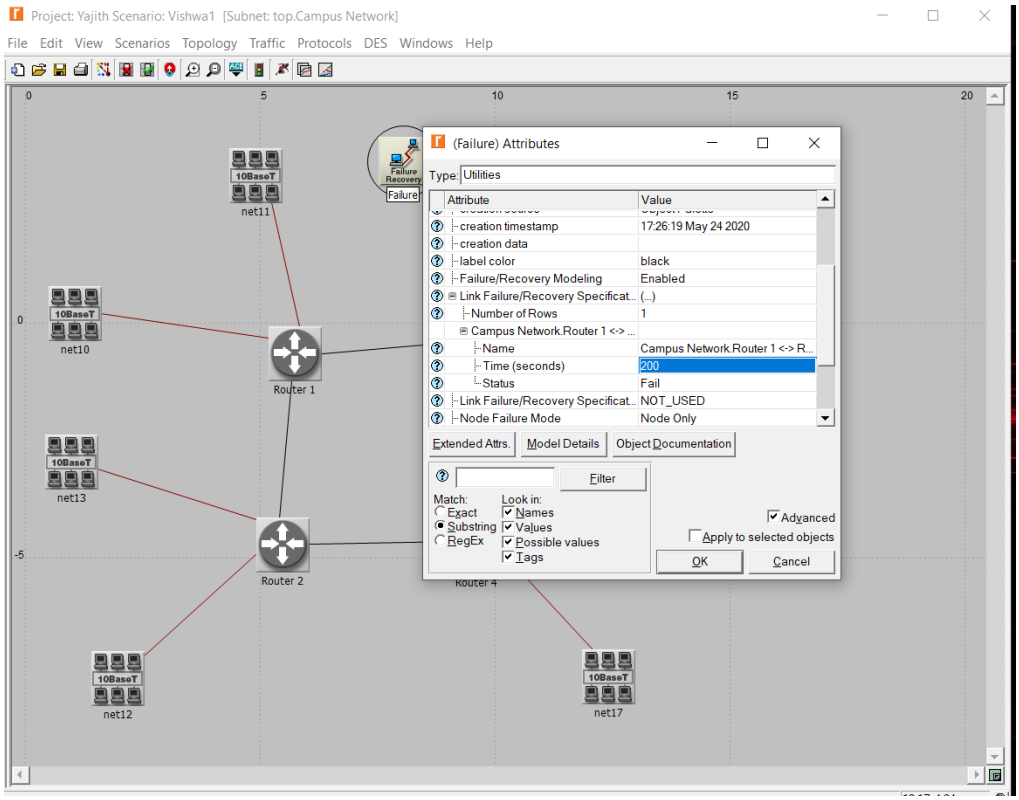


29. Set Link Failure/Recovery Specification -> Number of rows = 1

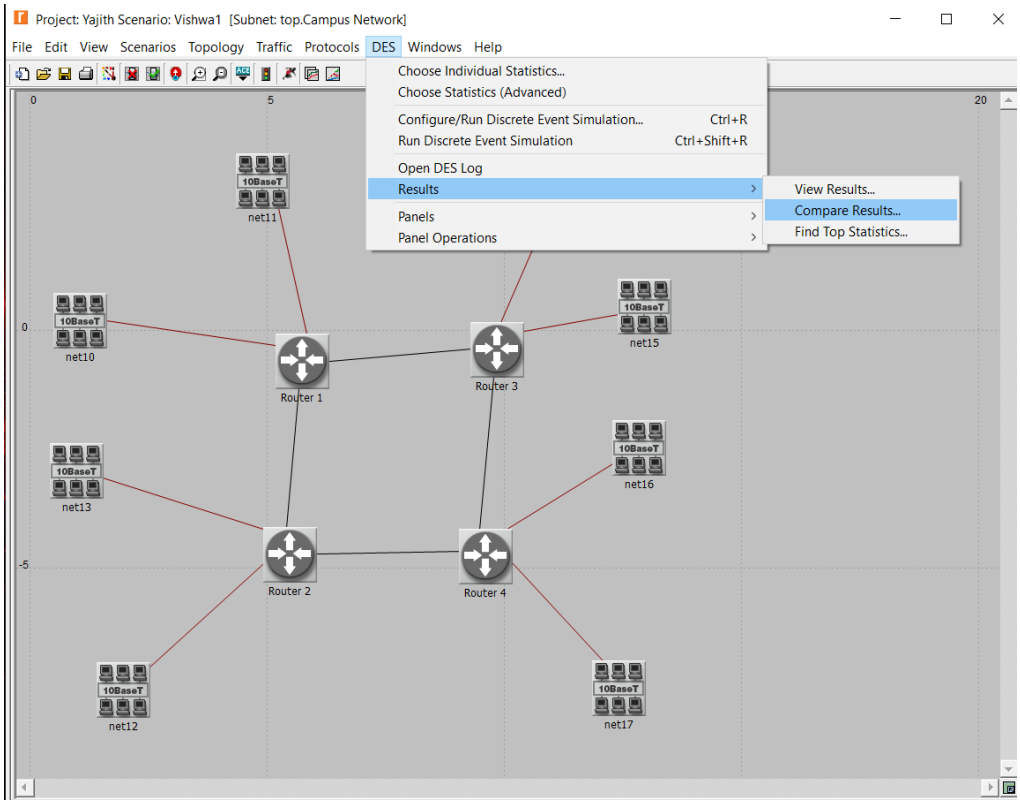


30. Set Campus Network Router -> Time = 200 Name = Router_0 – Router_2

31. Click Ok and Run the simulation

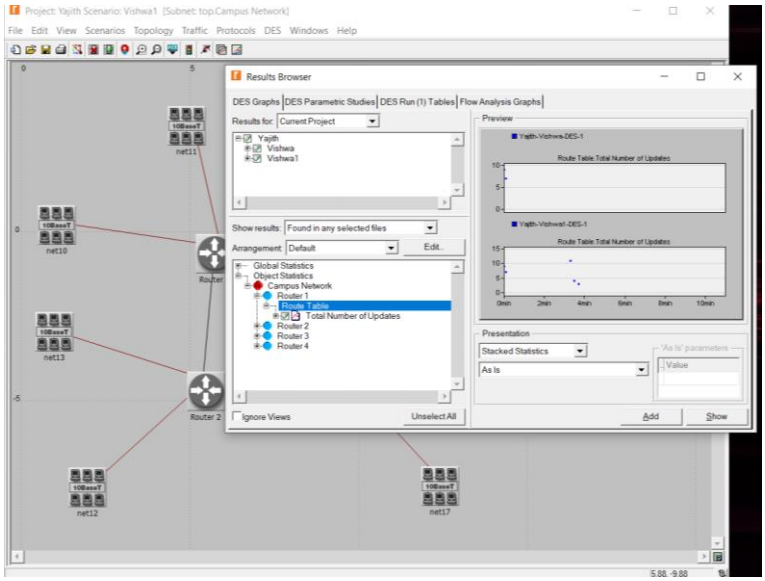


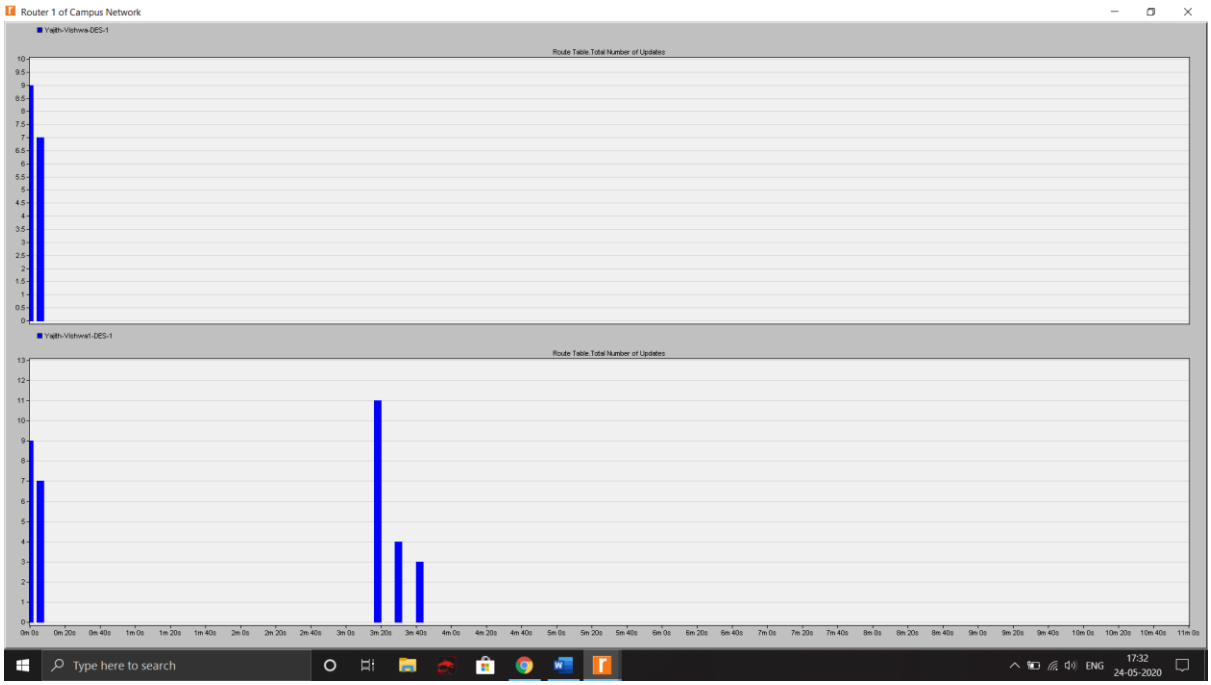
32. In the menu bar select DES -> Results -> Compare Results

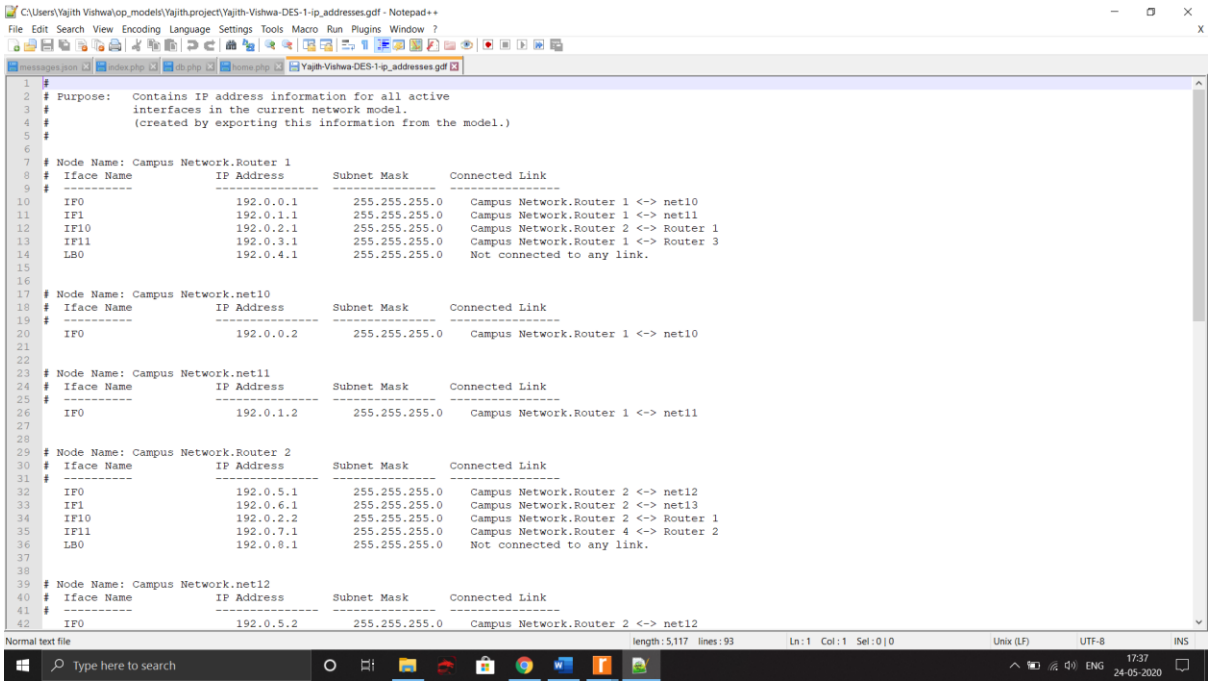


33. Select the scenarios to compare. Select the router to display the failures

34. The blue dot indicates the failure

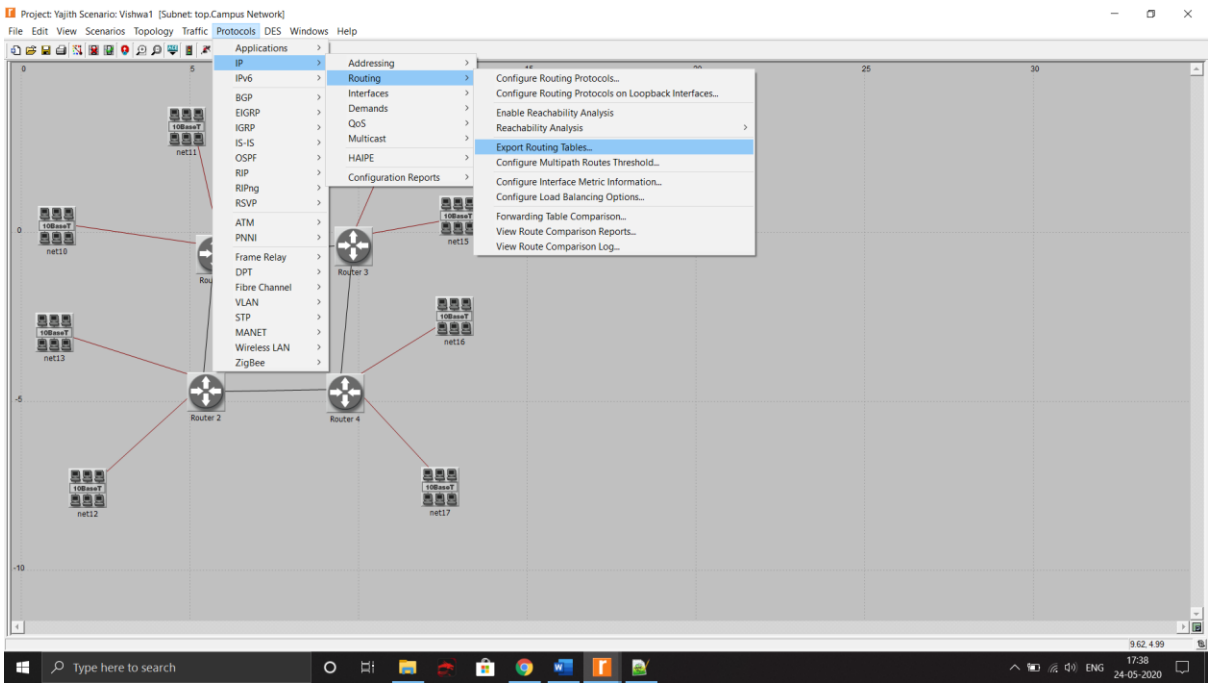

Project: Yajith Scenario: Vishwa1 [Subnet: top.Campus Network]
File Edit View Scenarios Topology Traffic Protocols DES Windows Help
Results Browser
DES Graphs | DES Parametric Studies | DES Run (1) Tables | Flow Analysis Graphs
Results for: Current Project
Yajith
Vishwa
Vishwa1
Show results: Found in any selected files
Arrangement: Default
Edit...
Global Statistics
Object Statistics
Campus Network
Router 1
Route Table
Total Number of Updates
Router 2
Router 3
Router 4
Ignore Views
Unselect All
Preview
Yajith-Vishwa-DES-1
Route Table.Total Number of Updates
Yajith-Vishwa1-DES-1
Route Table.Total Number of Updates
Presentation
Stacked Statistics
As Is
Value
Add
Show
net11
net10
net13
net12
net17
Router
Router 2

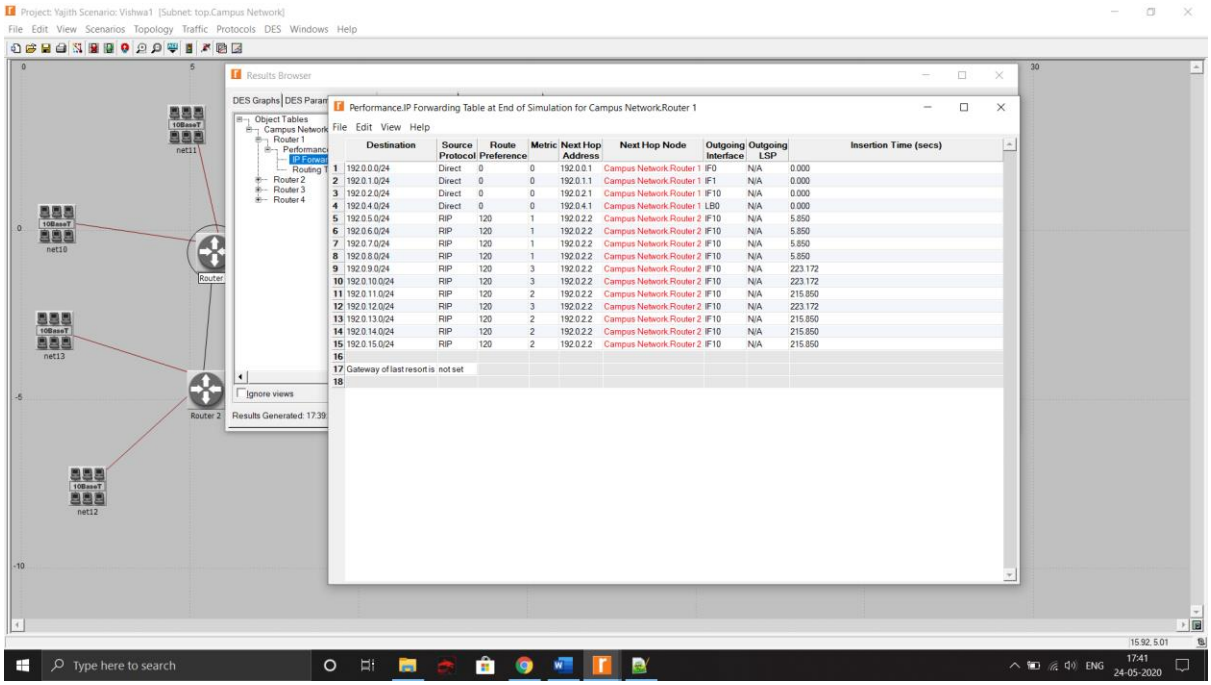

Router 1 of Campus Network
Yajith-Vishwa-DES-1
Route Table.Total Number of Updates
Yajith-Vishwa1-DES-1
Route Table.Total Number of Updates
Type here to search
ENG
17:32
24-05-2020

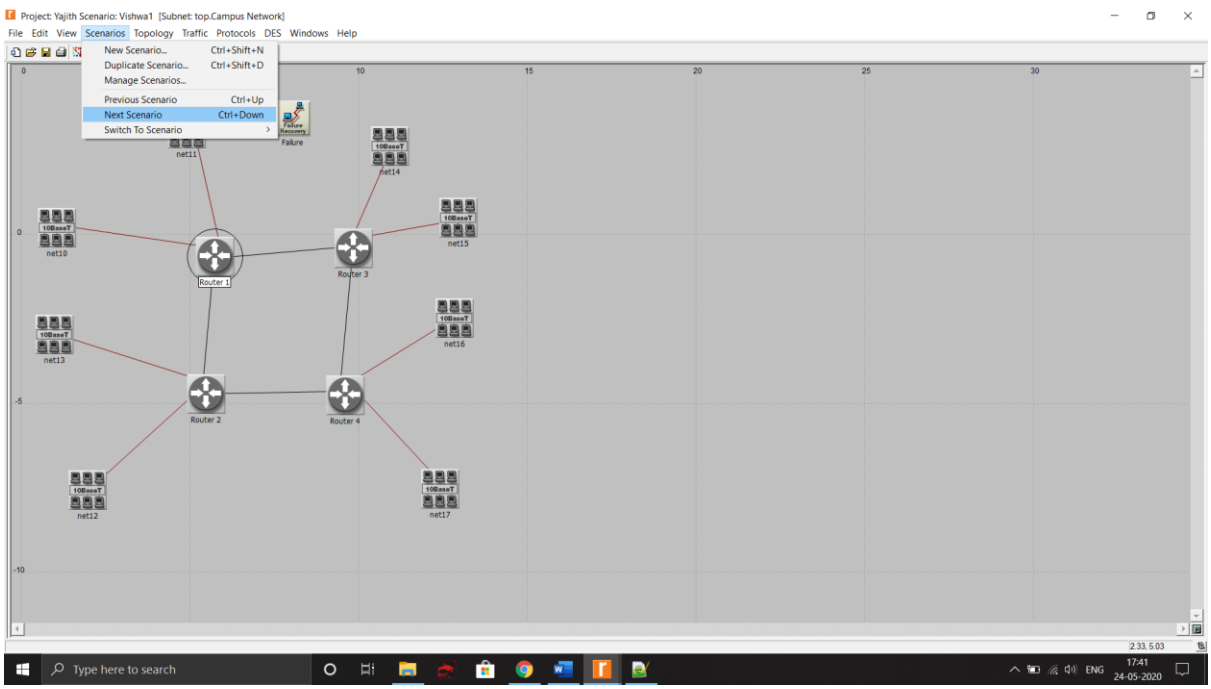


35. Go to Protocols -> IP -> Routing -> Export Routing tables.

36. The RIP table is displayed. Switch to the first scenario



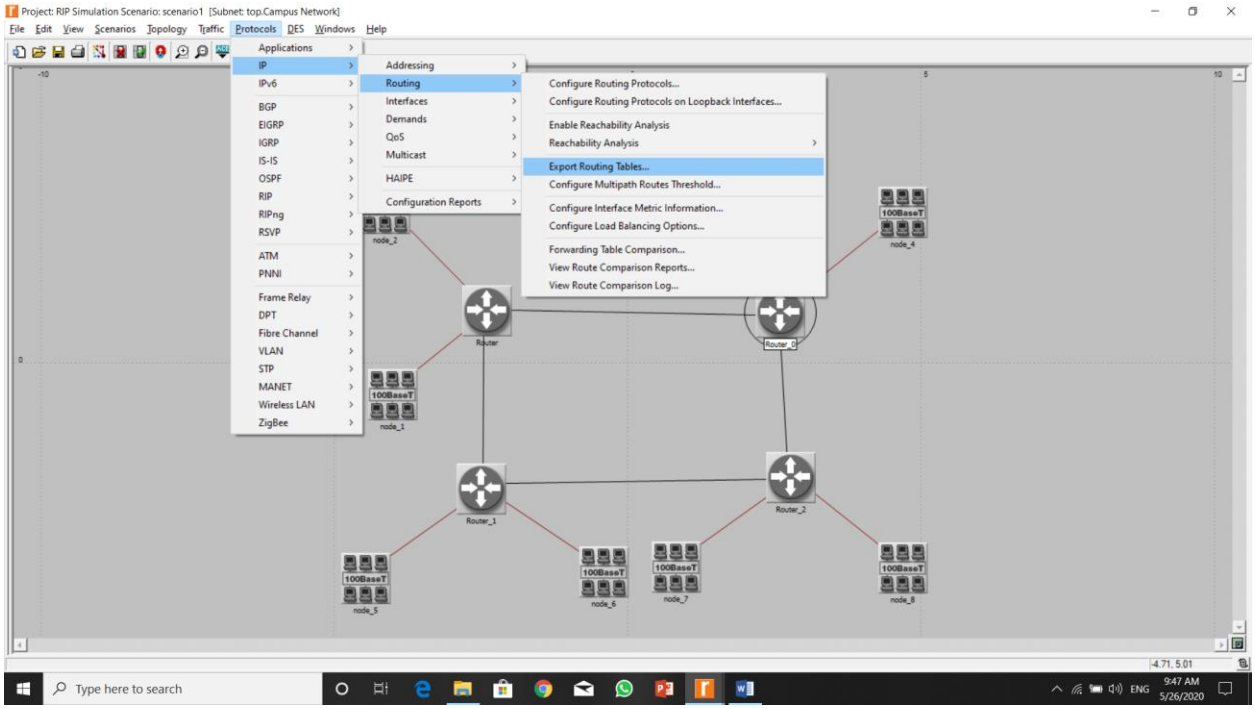

Project: Yajith Scenario: Vishwa1 [Subnet: top.Campus Network]
File Edit View Scenarios Topology Traffic Protocols DES Windows Help
Results Browser
DES Graphs DES Param
Object Tables
Campus Network
Router 1
Performanc
IP Forwa
Routing T
Router 2
Router 3
Router 4
Ignore views
Results Generated: 17:39
Performance.IP Forwarding Table at End of Simulation for Campus Network.Router 1
File Edit View Help
Destination | Source Protocol | Route Preference | Metric | Next Hop Address | Next Hop Node | Outgoing Interface | Outgoing LSP | Insertion Time (secs)
1 192.0.0.0/24 Direct 0 0 192.0.0.1 Campus Network.Router 1 IF0 N/A 0.000
2 192.0.1.0/24 Direct 0 0 192.0.1.1 Campus Network.Router 1 IF1 N/A 0.000
3 192.0.2.0/24 Direct 0 0 192.0.2.1 Campus Network.Router 1 IF10 N/A 0.000
4 192.0.4.0/24 Direct 0 0 192.0.4.1 Campus Network.Router 1 LB0 N/A 0.000
5 192.0.5.0/24 RIP 120 1 192.0.2.2 Campus Network.Router 2 IF10 N/A 5.850
6 192.0.6.0/24 RIP 120 1 192.0.2.2 Campus Network.Router 2 IF10 N/A 5.850
7 192.0.7.0/24 RIP 120 1 192.0.2.2 Campus Network.Router 2 IF10 N/A 5.850
8 192.0.8.0/24 RIP 120 1 192.0.2.2 Campus Network.Router 2 IF10 N/A 5.850
9 192.0.9.0/24 RIP 120 3 192.0.2.2 Campus Network.Router 2 IF10 N/A 223.172
10 192.0.10.0/24 RIP 120 3 192.0.2.2 Campus Network.Router 2 IF10 N/A 223.172
11 192.0.11.0/24 RIP 120 2 192.0.2.2 Campus Network.Router 2 IF10 N/A 215.850
12 192.0.12.0/24 RIP 120 3 192.0.2.2 Campus Network.Router 2 IF10 N/A 223.172
13 192.0.13.0/24 RIP 120 2 192.0.2.2 Campus Network.Router 2 IF10 N/A 215.850
14 192.0.14.0/24 RIP 120 2 192.0.2.2 Campus Network.Router 2 IF10 N/A 215.850
15 192.0.15.0/24 RIP 120 2 192.0.2.2 Campus Network.Router 2 IF10 N/A 215.850
16
17 Gateway of last resort is not set
18
net10
net11
net12
net13
Router
Router 2
15.92, 5.01
Type here to search
ENG
17:41
24-05-2020

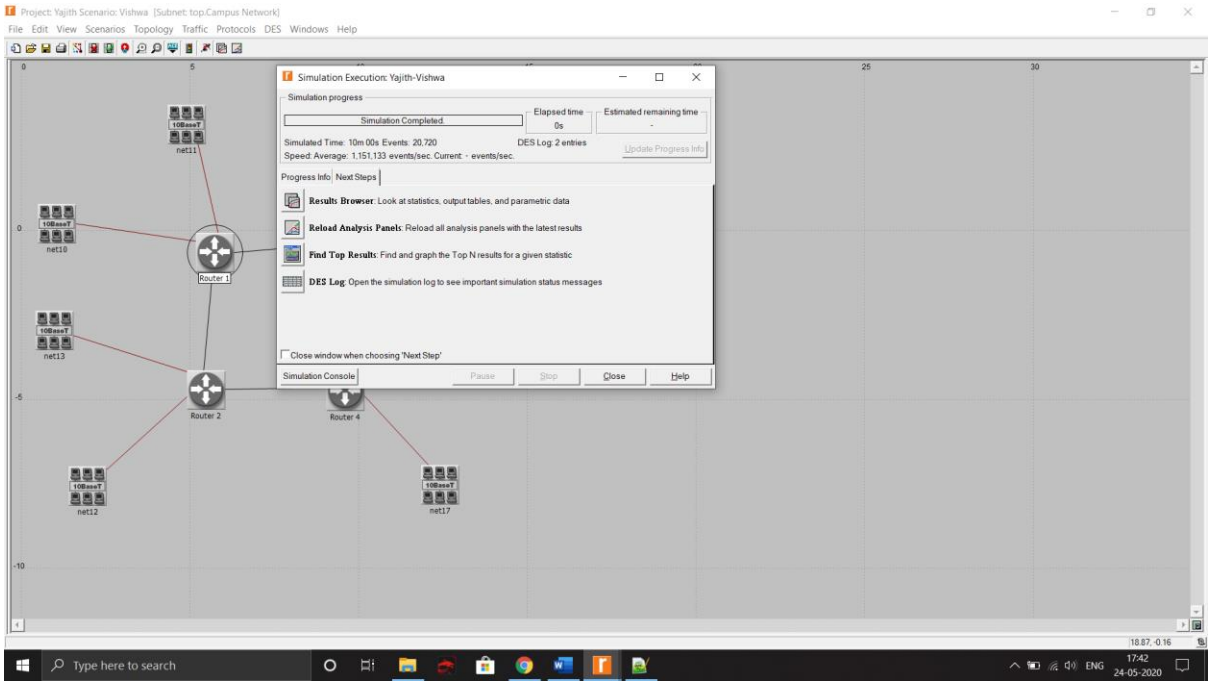

Project: Yajith Scenario: Vishwa1 [Subnet: top.Campus Network]
File Edit View Scenarios Topology Traffic Protocols DES Windows Help
New Scenario... Ctrl+Shift+N
Duplicate Scenario... Ctrl+Shift+D
Manage Scenarios...
Previous Scenario Ctrl+Up
Next Scenario Ctrl+Down
Switch To Scenario
Failure Recovery
Failure
net10
net11
net12
net13
net14
net15
net16
net17
Router 1
Router 2
Router 3
Router 4
2.33, 5.03
Type here to search
ENG
17:41
24-05-2020

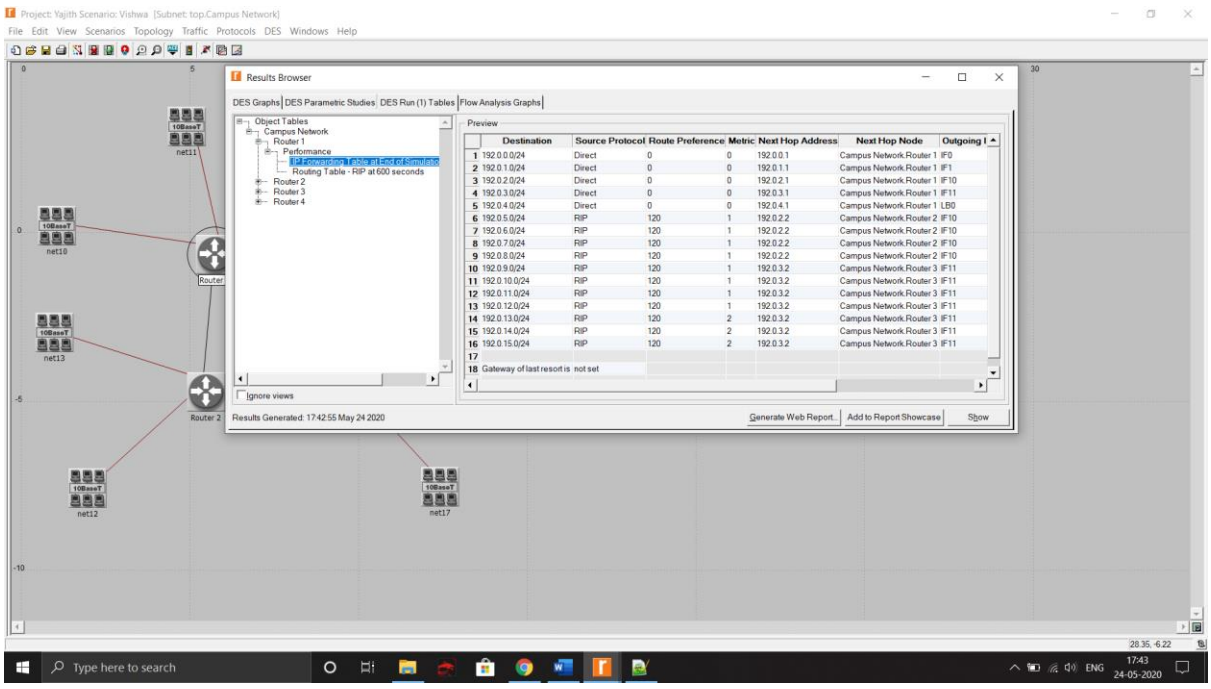
37. Export the routing table here also.

38. Then, run the simulation here.

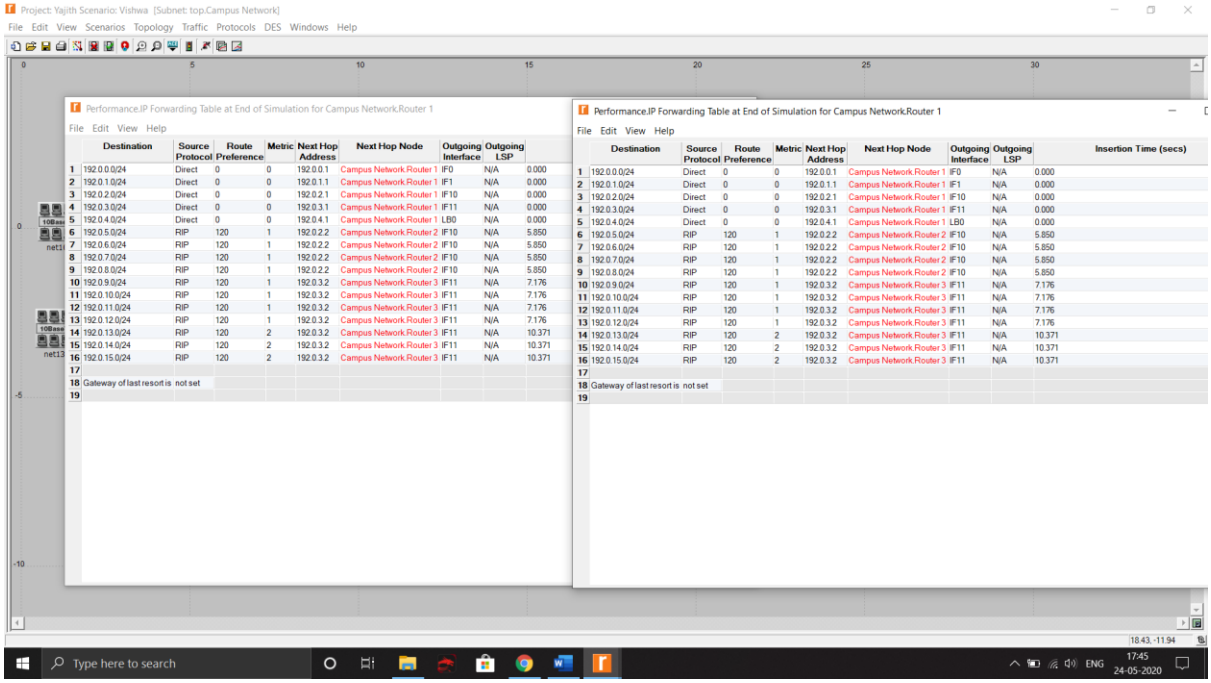
39. Click DES -> Results -> View Results







40. The RIP table is displayed here also (Without failure condition)


Project: Yajith Scenario: Vishwa [Subnet: top.Campus Network]
File Edit View Scenarios Topology Traffic Protocols DES Windows Help
Performance.IP Forwarding Table at End of Simulation for Campus Network.Router 1
File Edit View Help
Destination | Source Protocol | Route Preference | Metric | Next Hop Address | Next Hop Node | Outgoing Interface | Outgoing LSP
1 192.0.0.0/24 Direct 0 0 192.0.0.1 Campus Network.Router 1 IF0 N/A 0.000
2 192.0.1.0/24 Direct 0 0 192.0.1.1 Campus Network.Router 1 IF1 N/A 0.000
3 192.0.2.0/24 Direct 0 0 192.0.2.1 Campus Network.Router 1 IF10 N/A 0.000
4 192.0.3.0/24 Direct 0 0 192.0.3.1 Campus Network.Router 1 IF11 N/A 0.000
5 192.0.4.0/24 Direct 0 0 192.0.4.1 Campus Network.Router 1 LB0 N/A 0.000
6 192.0.5.0/24 RIP 120 1 192.0.2.2 Campus Network.Router 2 IF10 N/A 5.850
7 192.0.6.0/24 RIP 120 1 192.0.2.2 Campus Network.Router 2 IF10 N/A 5.850
8 192.0.7.0/24 RIP 120 1 192.0.2.2 Campus Network.Router 2 IF10 N/A 5.850
9 192.0.8.0/24 RIP 120 1 192.0.2.2 Campus Network.Router 2 IF10 N/A 5.850
10 192.0.9.0/24 RIP 120 1 192.0.3.2 Campus Network.Router 3 IF11 N/A 7.176
11 192.0.10.0/24 RIP 120 1 192.0.3.2 Campus Network.Router 3 IF11 N/A 7.176
12 192.0.11.0/24 RIP 120 1 192.0.3.2 Campus Network.Router 3 IF11 N/A 7.176
13 192.0.12.0/24 RIP 120 1 192.0.3.2 Campus Network.Router 3 IF11 N/A 7.176
14 192.0.13.0/24 RIP 120 2 192.0.3.2 Campus Network.Router 3 IF11 N/A 10.371
15 192.0.14.0/24 RIP 120 2 192.0.3.2 Campus Network.Router 3 IF11 N/A 10.371
16 192.0.15.0/24 RIP 120 2 192.0.3.2 Campus Network.Router 3 IF11 N/A 10.371
17
18 Gateway of last resort is not set
19
Performance.IP Forwarding Table at End of Simulation for Campus Network.Router 1
File Edit View Help
Destination | Source Protocol | Route Preference | Metric | Next Hop Address | Next Hop Node | Outgoing Interface | Outgoing LSP | Insertion Time (secs)
1 192.0.0.0/24 Direct 0 0 192.0.0.1 Campus Network.Router 1 IF0 N/A 0.000
2 192.0.1.0/24 Direct 0 0 192.0.1.1 Campus Network.Router 1 IF1 N/A 0.000
3 192.0.2.0/24 Direct 0 0 192.0.2.1 Campus Network.Router 1 IF10 N/A 0.000
4 192.0.3.0/24 Direct 0 0 192.0.3.1 Campus Network.Router 1 IF11 N/A 0.000
5 192.0.4.0/24 Direct 0 0 192.0.4.1 Campus Network.Router 1 LB0 N/A 0.000
6 192.0.5.0/24 RIP 120 1 192.0.2.2 Campus Network.Router 2 IF10 N/A 5.850
7 192.0.6.0/24 RIP 120 1 192.0.2.2 Campus Network.Router 2 IF10 N/A 5.850
8 192.0.7.0/24 RIP 120 1 192.0.2.2 Campus Network.Router 2 IF10 N/A 5.850
9 192.0.8.0/24 RIP 120 1 192.0.2.2 Campus Network.Router 2 IF10 N/A 5.850
10 192.0.9.0/24 RIP 120 1 192.0.3.2 Campus Network.Router 3 IF11 N/A 7.176
11 192.0.10.0/24 RIP 120 1 192.0.3.2 Campus Network.Router 3 IF11 N/A 7.176
12 192.0.11.0/24 RIP 120 1 192.0.3.2 Campus Network.Router 3 IF11 N/A 7.176
13 192.0.12.0/24 RIP 120 1 192.0.3.2 Campus Network.Router 3 IF11 N/A 7.176
14 192.0.13.0/24 RIP 120 2 192.0.3.2 Campus Network.Router 3 IF11 N/A 10.371
15 192.0.14.0/24 RIP 120 2 192.0.3.2 Campus Network.Router 3 IF11 N/A 10.371
16 192.0.15.0/24 RIP 120 2 192.0.3.2 Campus Network.Router 3 IF11 N/A 10.371
17
18 Gateway of last resort is not set
19
Type here to search
ENG
17:45
24-05-2020

RESULT:

Thus, the simulation of RIP protocol using riverbed is done. The RIP routing table is viewed and the results are verified.

Evaluation:

Parameter	Max Marks	Marks Obtained
Opnet Simulation	10	
Clarity in explanation	10	
Sub Total	20	
Completion of experiment on time	3	
Documentation	7	
Total	30	
Signature of the faculty with Date		