VEMU INSTITUTE OF TECHNOLOGY

P.Kothakota, Near Pakala, Chittoor

LAB Manual

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SUBJECT NAME: Computer Network LAB

BRANCH: CSE

YEAR AND SEMESTER: III – I

COURSE: B.TECH

REGULATION: R20

EXP NO: 1	**To Study different types of Network cables**
Date:	

AIM: To Study different types of Network cables (Copper and Fiber) and prepare cables (Straight and Cross) to connect Two or more systems. Use crimping tool to connect jacks. Use LAN tester to connect the cables.

Hardware Requirement

RJ-45 connector,

Climping Tool,

Twisted pair Cable

Software Requirement

Command Prompt and Packet Tracer.

Apparatus (Components): RJ-45 connector, Climping Tool, Twisted pair Cable

Procedure: To do these practical following steps should be done:

1. Start by stripping off about 2 inches of the plastic jacket off the end of the cable. Be very careful at this point, as to not nick or cut into the wires, which are inside. Doing so could alter the characteristics of your cable, or even worse render is useless. Check the wires, **one more time** for nicks or cuts. If there are any, just whack the whole end off, and start over.

2. Spread the wires apart, but be sure to hold onto the base of the jacket with your other hand. You do not want the wires to become untwisted down inside the jacket. Category 5 cable must only have 1/2 of an inch of 'untwisted' wire at the end; otherwise it will be 'out of spec'. At this point, you obviously have ALOT more than 1/2 of an inch of un-twisted wire.

3. You have 2 end jacks, which must be installed on your cable. If you are using a pre-made cable, with one of the ends whacked off, you only have one end to install - the crossed over end. Below are two diagrams, which show how you need to arrange the cables for each type of cable end. Decide at this point which end you are making and examine the associated picture below.

Diagram shows you how to prepare Cross wired connection

RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	

2

Diagram shows you how to prepare straight through wired connection

| RJ45 Pin # (END 1)	Wire Color	Diagram End #1	RJ45 Pin # (END 2)	Wire Color	Diagram End #2
1	White/Orange		1	White/Green	
2	Orange		2	Green	
3	White/Green		3	White/Orange	
4	Blue		4	White/Brown	
5	White/Blue		5	Brown	
6	Green		6	Orange	
7	White/Brown		7	Blue	
8	Brown		8	White/Blue	

Output:

Cable Crimping steps:

1. Remove the outmost vinyl shield for 12mm at one end of the cable (we call this side A-side).

2. Arrange the metal wires in parallel

3. Insert the metal wires into RJ45 connector on keeping the metal wire arrangement.

4. Set the RJ45 connector (with the cable) on the pliers, and squeeze it tightly.
5. Make the other side of the cable (we call this side B-side) in the same way.

6. After you made it, you don't need to take care of the direction of the cable.
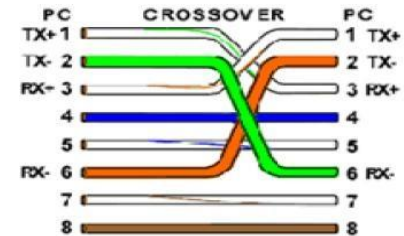
IO connector crimping: Run the full length of Ethernet cable in place, from endpoint to endpoint, making sure to leave excess.

At one end, cut the wire to length leaving enough length to work, but not too

much excess.Strip off about 2 inches of the Ethernet cable sheath.

Align each of the colored wires according to the

layout of the jack.Use the punch down tool to insert

each wire into the jack.

Repeat the above steps for the second RJ45 jack.
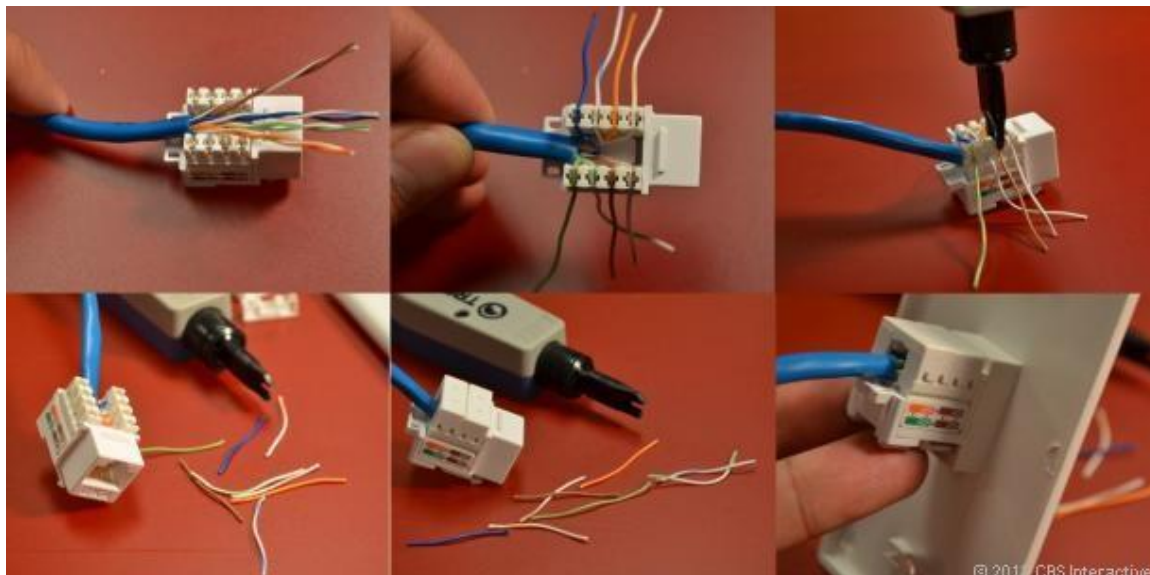
Testing the crimped cable using a cable tester:

Step 1 : Skin off the cable jacket 3.0 cm long cable stripper up to cable
Step 2: Untwist each pair and straighten each wire 190 0 1.5 cm long.
Step 3 : Cut all the wires

4

Step 4 : Insert the wires into the RJ45 connector right white orange left brown the pins facing up
Step 5 : Place the connector into a crimping tool, and squeeze hard so that the handle reaches its fullswing.
Step 6: Use a cable tester to test for proper continuity

Result:

Cable Crimping, Standard Cabling and Cross Cabling, IO connector crimping and testing the crimpedcable using a cable tester are done successfully

5

EXP-1(b): To Install and configure Network Devices: HUB, Switch and Routers. Consider both manageable and non-manageable switches. Do the logical configuration of the system. Set the bandwidth of different ports.

To Study of following Network Devices in Detail
- Repeater
- Hub
- Switch
- Bridge
- Router
- Gate Way

Apparatus (Software): No software or hardware needed.

Procedure: Following should be done to understand this practical.

1. **Repeater:** Functioning at Physical Layer. A **repeater** is an electronic device that receives a signal and retransmits it at a higher level and/or higher power, or onto the other side of an obstruction, so that the signal can cover longer distances. Repeater have two ports ,so cannot be use to connect for more than two devices

2. **Hub:** An Ethernet hub, active hub, network hub, repeater hub, hub or concentrator is a device for connecting multiple twisted pair or fiber optic Ethernet devices together and making them act as a single network segment. Hubs work at the physical layer (layer 1) of the OSI model. The device is a form of multiport repeater. Repeater hubs also participate in collision detection, forwarding a jam signal to all ports if it detects a collision.

3. **Switch:** A **network switch** or **switching hub** is a computer networking device that connects network segments. The term commonly refers to a network bridge that processes and routes data at the data link layer (layer 2) of the OSI model. Switches that additionally process data at the network layer (layer 3 and above) are often referred to as Layer 3 switches or multilayer switches.

4. **Bridge:** A **network bridge** connects multiple network segments at the data link layer (Layer 2) of the OSI model. In Ethernet networks, the term *bridge* formally means a device that behaves according to the IEEE 802.1 D standard. A bridge and switch are very much alike; a switch being a bridge with numerous ports. *Switch* or *Layer 2 switch* is often used interchangeably with *bridge*. Bridges can analyze incoming data packets to determine if the bridge is able to send the given packet to another segment of the network.

5. **Router:** A **router** is an electronic device that interconnects two or more computer networks, and selectively interchanges packets of data between them. Each data packet contains address information that a router can use to determine if the source and destination are on the same network, or if the data packet must be transferred from one network to another. Where multiple routers are used in a large collection of interconnected networks, the routers exchange information about target system addresses, so that each router can build up a table showing the preferred paths between any two systems on the interconnected networks.

6. **Gate Way:** In a communications network, a network node equipped for interfacing with another network that uses different protocols.

• A gateway may contain devices such as protocol translators, impedance matching devices, rate converters, fault isolators, or signal translators as necessary to provide system interoperability. It also requires the establishment of mutually acceptable administrative procedures between both networks.

• A protocol translation/mapping gateway interconnects networks with different network protocol technologies by performing the required protocol conversions.

EXP 1(c):

Install and Configure Wired and Wireless NIC and transfer files between systems in Wired LANand Wireless LAN. Consider both adhoc and infrastructure mode of operation.

Objectives

Part 1: Identify and Work with PC NICs
Part 2: Identify and Use the System Tray Network Icons

Background / Scenario

This lab requires you to determine the availability and status of the network interface cards (NICs) on the PC that you use. Windows provides a number of ways to view and work with your NICs.

In this lab, you will access the NIC information of your PC and change the status of these cards.

Required Resources

1 PC (Windows 7 or 8 with two NICs, wired and wireless, and a wireless connection)

Part 1: Identify and Work with PC NICs

In Part 1, you will identify the NIC types in the PC that you are using. You will explore different ways to extract information about these NICs and how to activate and deactivate them.

Procedure:

(a) Install the network card:

Disconnect all cables connected to the computer and open the case. Locate an available PCI slot (white slots) and insert the network card and secure the card with the screw that came with it. Once the adapterhas been installed and secured close the computer case, connect all the cables and turn it on.

After installing the adapter driver it should be working find, now let's configure the card for use on anetwork.

Click on the Start button and select Settings then

Control Panel.Double click on the System icon
Click on the Hardware tab.

7

Click on Device Manager.

You will see a list of devices installed in your compquter.

If necessary, click on the + sign next to Network Adapters to expand the list.

Ensure that there is no yellow exclamation mark (!) next to the Network Adapter. This indicates a possibleproblem with the card or configuration.

Double click on your network driver (e.g. NE2000

Compatible).In the Device Status box you should

see the message:

This Device is working correctly.
If you do not see this message or if there is no Network Adapter displayed, then your Ethernet card willprobably need configuring.

Result:
Installation and configuration of Wired and Wireless (remotely) NIC and transfer files between systems inLAN and Wireless LAN between two systems in a LAN have been done successfully.

EXP NO: 2	Networking commands
Date:	

AIM: Work with the commands Ping, Tracert, Ipconfig, pathping, telnet, ftp, getmac, ARP, Hostname, Nbtstat, netdiag, and Nslookup

Procedure:

Configure Internet connection and use IPCONFIG, PING:

1. Open Command Prompt, and then type ipconfig. From the display of the ipconfig command, ensure that the network adapter for the TCP/IP configuration you are testing is not in a Media disconnected state.
2. At the command prompt, ping the loopback address by typing ping 127.0.0.1.
3. Ping the IP address of the computer.
4. Ping the IP address of the default gateway. If the ping command fails, verify that the default gateway IPaddress is correct and that the gateway (router) is operational.
5. Ping the IP address of a remote host (a host that is on a different subnet).
   If the ping command fails, verify that the remote host IP address is correct, that the remote host is operational, and that all of the gateways (routers) between this computer and the remote host are operational.
6. Ping the IP address of the DNS server.
   If the ping command fails, verify that the DNS server IP address is correct that the DNS server is operational, and that all of the gateways (routers) between this computer and the DNS server are operational.

Tracer to debug the network issues.

Tracer network:
Open Command Prompt, and type the following:

tracert host_name
  Or
tracert ip_address
where host_name or ip_address is the host name or IP address, respectively, of the remote computer.

If you do not want the tracert command to resolve and display the names of all routers in the path, use the -d parameter. This expedites the display of the path. For example, to trace a path from this computer towww.microsoft.com without displaying the router names, type the following at a command prompt:
Net stat utilities to debug the network issues:

Displays active TCP connections, ports on which the computer is listening, Ethernet statistics, the IP routing table, IPv4 statistics (for the IP, ICMP, TCP, and UDP protocols), and IPv6 statistics (for the IPv6, ICMPv6, TCP over IPv6, and UDP over IPv6 protocols).

9

Used without parameters, netstat displays active TCP connections

Syntax
netstat [-a] [-e] [-n] [-o] [-p *Protocol*] [-r] [-s] [*Interval*]

Parameters
-a
Displays all active TCP connections and the TCP and UDP ports on which the computer is listening.
-e
Displays Ethernet statistics, such as the number of bytes and packets sent and received
This parametercan be combined with -s.
-n
Displays active TCP connections, however, addresses and port numbers are expressed numerically and noattempt is made to determine names.
-o
Displays active TCP connections and includes the process ID (PID) for each connection. You can find the application based on the PID on the Processes tab in Windows Task Manager. This parameter can be combined with -a, -n, and -p.
-p *Protocol*

Shows connections for the protocol specified by *Protocol*. In this case, the *Protocol* can be tcp, udp, tcpv6, or udpv6. If this parameter is used with -s to display statistics by protocol, *Protocol* can be tcp, udp,icmp, ip, tcpv6, udpv6, icmpv6, or ipv6.
-s
Displays statistics by protocol. By default, statistics are shown for the TCP, UDP, ICMP, and IP protocols. If the IPv6 protocol for Windows XP is installed, statistics are shown for the TCP over IPv6, UDP over IPv6, ICMPv6, and IPv6 protocols. The -p parameter can be used to specify a set of protocols.
-r
Displays the contents of the IP routing table. This is equivalent to the route print command.
Interval

Redisplays the selected information every *Interval* seconds. Press CTRL+C to stop the redisplay. If thisparameter is omitted, netstat prints the selected information only once.
/? Displays help at the command prompt

Result:

Thus the Configure Internet connection and use IPCONFIG, PING / Tracer and Net stat utilities toestablish interconnection between systems have been done successful

EXP NO: 3	**Find all the IP addresses on your network**
Date:	

AIM: To Find all the IP addresses on your network. Unicast, Multicast, and Broadcast on your network.

Description:

There are two ways to find IP addresses on network systems. You can find them manually or you can use an IP scanner, which is designed to automatically find the IP addresses within a certain range.

With a scanner such as SolarWinds IPAM, you can run automated scans to identify new devices and more easily manage IP addresses.

The basic steps for manually creating a list of device IP addresses on a network include:

- Get to the command line by opening a terminal window.
- Type the right command for your system. On Linux, type the command "ifconfig" and press Return. On Windows, type the command "ipconfig" and press Return.
- Get more information by typing the command "arp -a."
- You should now see a basic list of the IP addresses for devices connected to your network.
- You can then input this information into an information storage tool, like a spreadsheet, that you'll need to update by hand each time you attempt a new discovery.

This process is unnecessarily time consuming and vulnerable to errors. Not only are you forced to manually run commands to find IP addresses whenever you want to see the devices connected to your network, but you also need to identify which of the IP addresses you find are new or have changed.

Use an IP discovery tool to automatically locate devices and collect critical data

SolarWinds® IP Address Manager (IPAM) is designed to eliminate the need to manually manage IP address data with complicated spreadsheets. By quickly identifying and updating device information, an automated IP address discovery tool can help make IP discovery quicker and a more accurate process.

SolarWinds IPAM is built to scan your network for all IP addresses, checks their statuses, and reports the details. IP Address Manager also uses automated subnet discovery and neighbor discovery techniques to help identify and accurately manage IP subnets, address blocks, and connected hosts to present real-time IP address information.

Output:
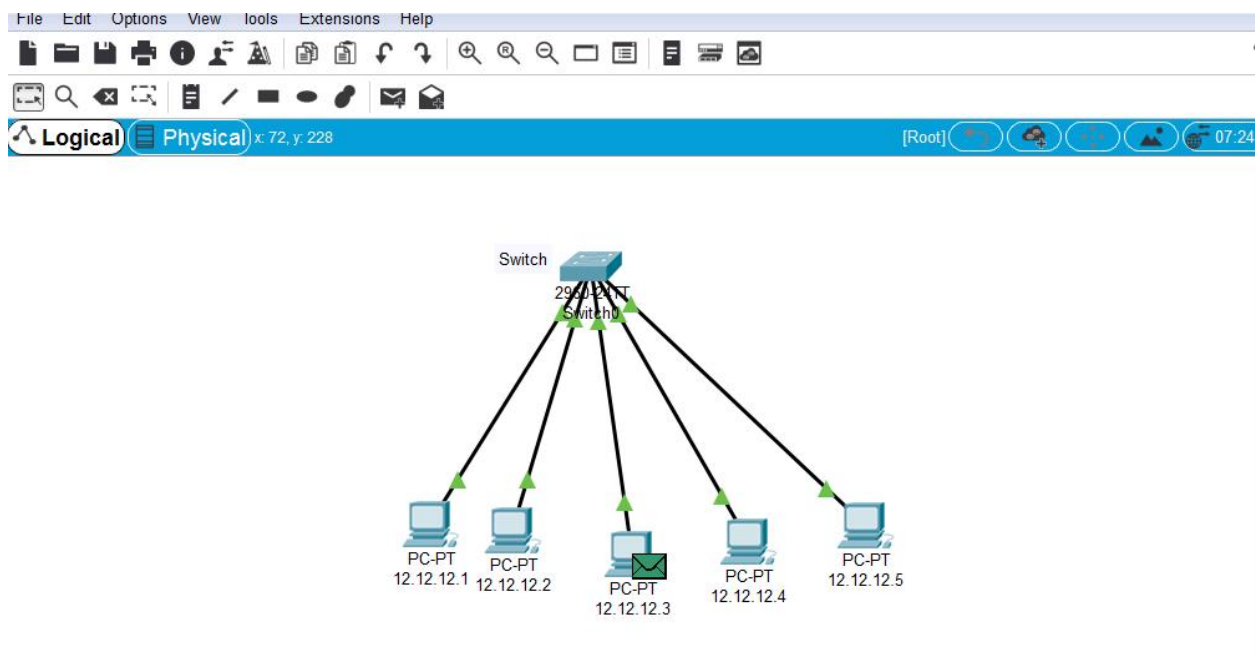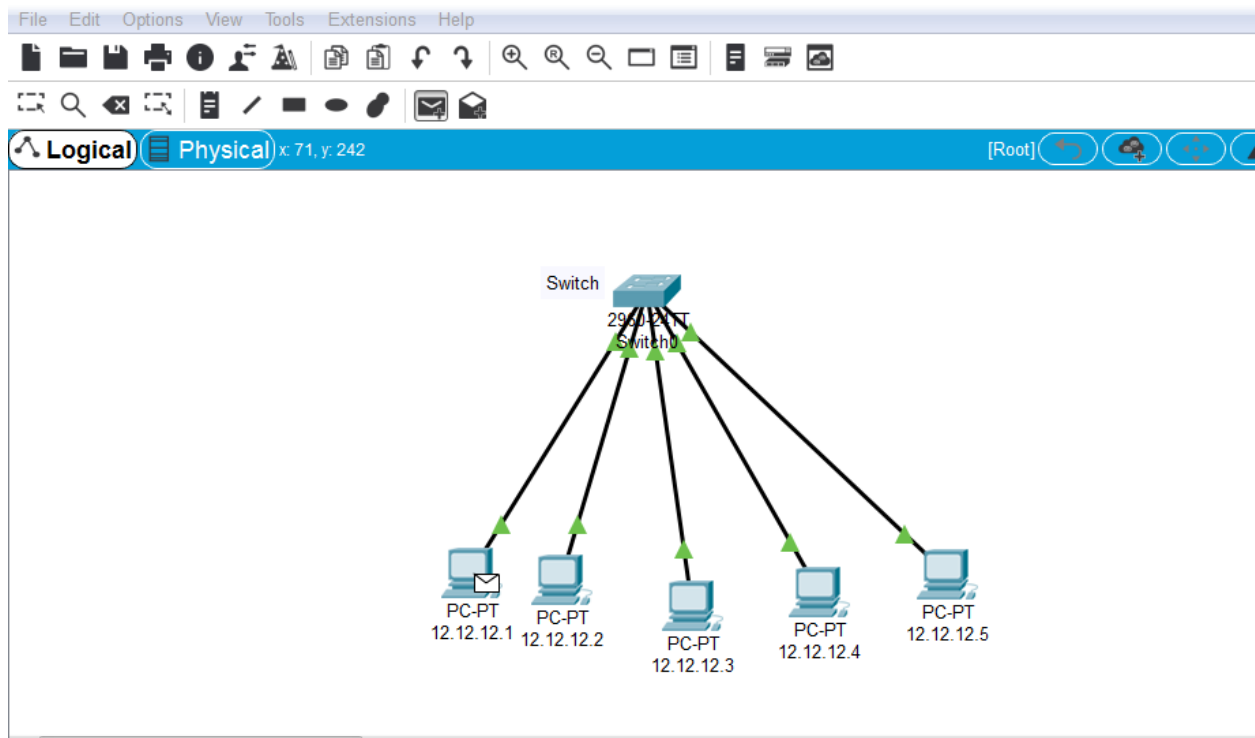
EXP NO: 4	
Date:	**Packet tracer software to build network topology**

AIM: To Use Packet tracer software to build network topology and configure using Distance vector routing protocol.

Description:

It is under dynamic routing algorithm. This algorithm operates by having each route maintains a table giving the least known distance to reach destination and include line in used to get these. These are updated by changing information with neighbour. This is called "Bell mann ford algorithm" and "fod fick" algorithm.

Procedure:

- Open the Cisco Packet Tracer software.
- Add the router and PCs accourding to our design.
- Configure all the routers and PCs.
- Trace the destination in PC's command prompt.
- Verify the output.

Distance vector routing algorithm:

```
#include<stdlib.h>
#define nul 1000
#define nodes 10
int no;
struct node
{
int a[nodes][4];
}router[nodes];
void init(int r)
{
int i;
for(i=1;i<=no;i++)
{
router[r].a[i][1]=i;
router[r].a[i][2]=999;
router[r].a[i][3]=nul;
}
router[r].a[r][2]=0;
router[r].a[r][3]=r;
```

13

```
}
void inp(int r)
{
int i;
printf("\nEnter distance from the node %d to other nodes",r);
printf("\nPls Enter 999 of there is no direct route\n",r);
for(i=1;i<=no;i++)
{
if(i!=r)
{
printf("\nEnter dist to node %d:",i);
scanf("%d",&router[r].a[i][2]);
router[r].a[i][3]=i;
}}}
void display(int r)
{
int i,j;
printf("\n\nThe routing table for node %d is as follows:",r);
for(i=1;i<=no;i++)
{
if(router[r].a[i][2]>=99)
printf("\n\t\t\t%d\tno link\tno hop",router[r].a[i][1]);
else
printf("\n\t\t\t%d\t%d",router[r].a[i][1],router[r].a[i][2],router[r].a[i][3]);
}}
void dv_algo(int r)
{
int i,j,z;
for(i=0;i<=no;i++)
{
if(router[r].a[i][2]!=999&&router[r].a[i][2]!=0)
{
for(j=1;j<=no;j++)
{
z=router[r].a[i][2]+router[i].a[j][2];
if(router[r].a[j][2]>z)
{
router[r].a[j][2]=z;
router[r].a[j][3]=i;
}}}}}
```

14

```
int main()
{
int i,j,x,y;
char choice;
printf("Enter the no.of nodes required(less than 10 pls):");
scanf("%d",&no);
for(i=1;i<=no;i++)
{
init(i);
inp(i);
}
printf("\nTne configuration of the nodes after initialization is as follows:");
for(i=1;i<=no;i++)
display(i);
for(i=1;i<=no;i++)
dv_algo(i);
printf("\nThe configuration of the nodes after computation of paths is as follows:");
for(i=1;i<=no;i++)
display(i);
while(1)
{
printf("\n\nWanna continue (y/n):");
scanf("%c",&choice);
if(choice=='n')
break;
printf("\nEnter the nodes btn which the shortest path is to be found:\n");
scanf("%d%d",&x,&y);
printf("\nThe length of the shortest path is %d",router[x].a[y][2]);
}}
```

15

```
Enter the no.of nodes required(less than 10 pls):3

Enter distance from the node 1 to other nodes
Pls Enter 999 of there is no direct route

Enter dist to node 2:1

Enter dist to node 3:2

Enter distance from the node 2 to other nodes
Pls Enter 999 of there is no direct route

Enter dist to node 1:1

Enter dist to node 3:2

Enter distance from the node 3 to other nodes
Pls Enter 999 of there is no direct route

Enter dist to node 1:1

Enter dist to node 2:999

Tne configuration of the nodes after initialization is as follows:

The routing table for node 1 is as follows:
                      1                0
                      2                1
                      3                2

The routing table for node 2 is as follows:
                      1                1
                      2                0
                      3                2

The routing table for node 3 is as follows:
                      1                1
                      2        no link no hop
                      3                0
The configuration of the nodes after computation of paths is as follows:

The routing table for node 1 is as follows:
                      1                0
                      2                1
                      3                2

The routing table for node 2 is as follows:
                      1                1
                      2                0
                      3                2

The routing table for node 3 is as follows:
                      1                1
                      2                2
                      3                0

Wanna continue (y/n):
Enter the nodes btn which the shortest path is to be found:
1
```

Output:
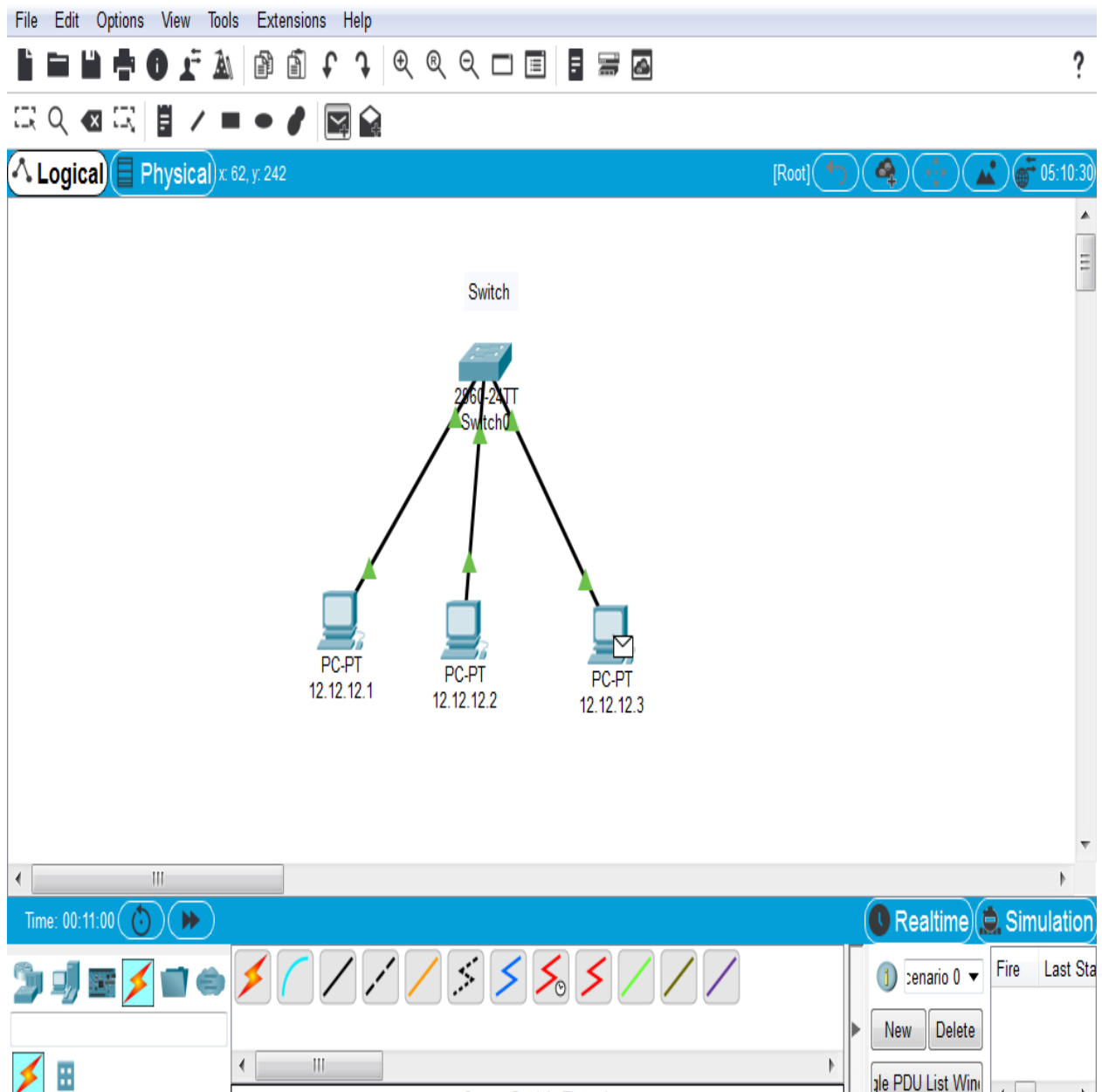
EXP NO: 5	**Link State routing protocol**
Date:	

AIM: Use Packet tracer software to build network topology and configure using Link State routing protocol.

Procedure:
Link state routing works on the following principle.
- Discover the neighbour and keep their network address.
- Measure the delay or cost to each of its neighbour.
- Construct a packet telling all it has just learned.
- Send the packet to all router.
- Compute the shortest path to every router.

Open the Cisco Packet Tracer software.
- Add the router and PCs according to our design.
- Configure all the routers and PCs.
- Trace the destination in PC's command prompt.
- Verify the output.

Link State Routing Protocol Algorithm:

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<string.h>
#include<math.h>
#define IN 99
#define N 6
int dijkstra(int cost[][N],int source,int target);
char
interface[6][6][20]={{"0","0","0","0","0","0"},{"0","0","192.1.1.1","0","200.1.1.1"},{"0","0","0
","0",
"198.1.1.2","0"},{"0","192.1.1.3","0","198.1.1.3","0","200.1.1.2"},{"0","0","200.1.1.3","0","200
.1.1.4
","0"}};
int main()
{
int cost[N][N],i,j,w,ch,co;
char ip[10];
int source,target,x,y;
printf("\t the shortest path algorithm(DIJKSRTRA'S ALGORITHM in c\n\n");
for(i=1;i<N;i++)
```

18

```
for(j=1;j<N;j++)
cost[i][j]=IN;
for(x=1;x<N;x++)
{
for(y=x+1;y<N;y++)
{
printf("enter the weight of the path between node%d and %d:",x,y);
scanf("%d",&w);
cost[x][y]=cost[y][x]=w;
}
printf("\n");
}
for(x=1;x<N;x++)
{
for(y=1;y<N;y++)
{
printf("%s:\t",interface[x][y]);
//scanf("%s",&ip);
//interface[x][y][20]=ip;
}
printf("\n");
}
printf("\nEnter the source:");
scanf("%d",&source);
printf("\nEnter the target");
scanf("%d",&target);
co=dijsktra(cost,source,target);
printf("\nThe Shortest Path:%d",co);
return 0;
}
int dijsktra(int cost[][N],int source,int target)
{
int dist[N],prev[N],selected[N]={0},i,m,min,start,d,j,x,y;
char path[N];int path1[N];
for(i=0;i<N;i++)
{
dist[i]=IN;
prev[i]=-1;
}
start=source;
```

19

```
selected[start]=1;
dist[start]=0;
while(selected[target]==0)
{
min=IN;
m=0;
for(i=1;i<N;i++)
{
d=dist[start]+cost[start][i];
if(d<dist[i]&&selected[i]==0)
{
dist[i]=d;
prev[i]=start;
}
if(min>dist[i]&&selected[i]==0)
{
min=dist[i];
m=i;
}}
start=m;
selected[start]=1;
}
start=target;
j=0;
while(start!=-1)
{
path[j++]=start+64;
path1[j++]=start;
start=prev[start];
}
path[j]='\0';
strrev(path);
printf("%s",path);
printf("\n");
for(j=j-1;j>=0;j--)
{
printf("%d\t",path1[j]);
if(j>0)
{
x=path1[j];
```

20

```
y=path1[j-1];
printf("%s\t%s\n",interface[x][y],interface[y][x]);
}}
return dist[target];
}
```

```
            the shortest path algorithm(DIJKSRIRA'S ALGORITHM in c

enter the weight of the path between node1 and 2:1
enter the weight of the path between node1 and 3:99
enter the weight of the path between node1 and 4:1
enter the weight of the path between node1 and 5:99

enter the weight of the path between node2 and 3:99
enter the weight of the path between node2 and 4:99
enter the weight of the path between node2 and 5:1

enter the weight of the path between node3 and 4:1
enter the weight of the path between node3 and 5:99

enter the weight of the path between node4 and 5:1


0:      192.1.1.1:      0:      200.1.1.1:      :
0:      0:      0:      198.1.1.2:      0:
192.1.1.3:      0:      198.1.1.3:      0:      200.1.1.2:
0:      200.1.1.3:      0:      200.1.1.4:      0:
:       :       :       :       :

Enter the source:3

Enter the target5
```

```
6       weight of the path between node%d and %d:
4
2401
5       0
32765
The Shortest Path:2
```

Output:
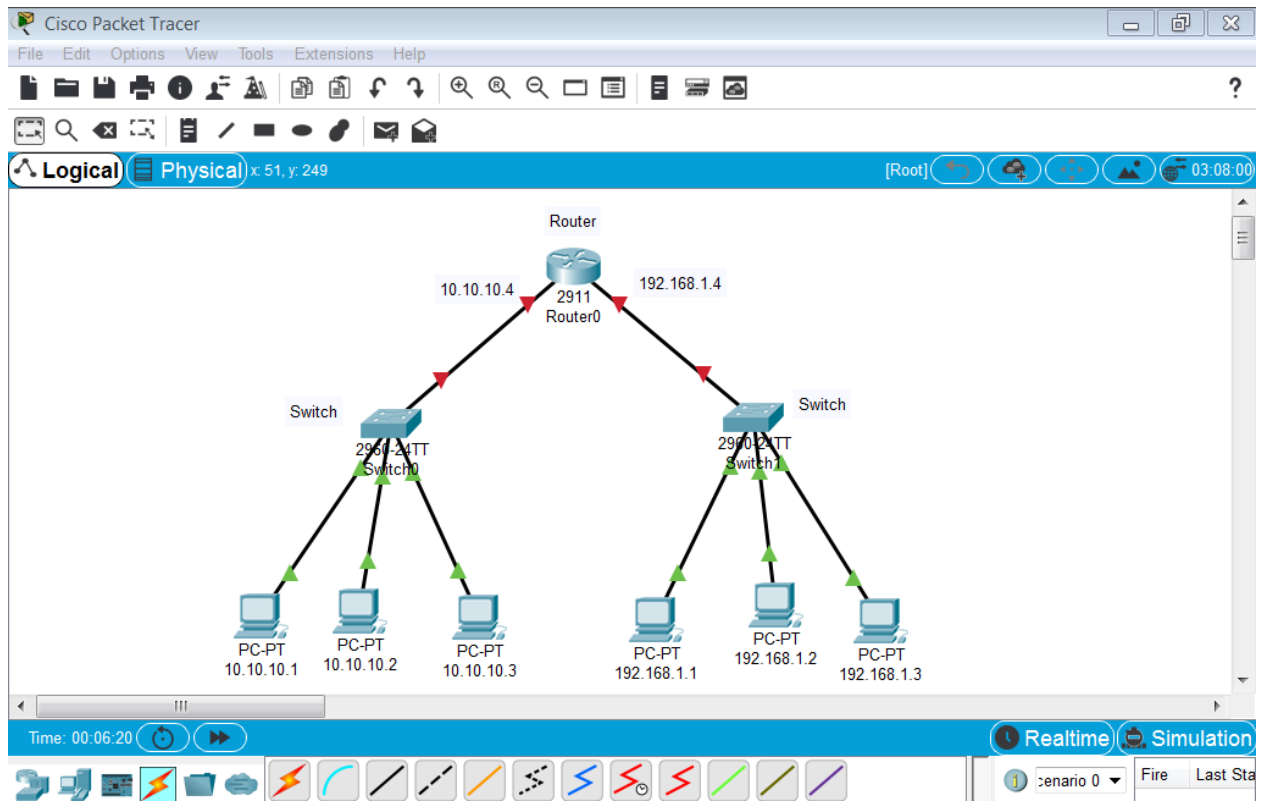
EXP NO: 6	**JAVA RMI Basic Calculator**
Date:	

AIM: To using JAVA RMI write a program to implement Basic Calculator.

RMI called as remote method innovaction, by using rmi to develop calculator program in java

Program:

Calc.java

import java.rmi.Remote;

import java.rmi.RemoteException;

public interface Calc extends Remote

```
{
public long addition(long a, long b)throws RemoteException;



public long subtraction(long a, long b) throws RemoteException;


public long multiplication(long a, long b) throws RemoteException;


public long divition(long a, long b) throws RemoteException;


}
```

Calcimpl.java

23

```java
import java.rmi.RemoteException;

import java.rmi.server.UnicastRemoteObject;

public class Calcimpl extends UnicastRemoteObject implements Calc

    {


    protected Calcimpl()throws RemoteException


        {

        super();

        }


    public long addition(long a,long b)throws RemoteException


        {


        return a+b;


        }


    public long subtraction(long a,long b)throws RemoteException


        {

        return a-b;

        }
```

```java
    public long multiplication(long a,long b)throws RemoteException


        {


        return a*b;

        }


    public long divition(long a, long b)throws java.rmi.RemoteException


        {

        return a/b;

        }

    }
```

Calcserv.java

```java
import java.rmi.Naming;


public class CalcServer

{


    CalcServer()

    {

    try

        {

        Calc c = new Calcimpl();

Naming.rebind("rmi://localhost:1099/CalculatorService", c);
```

25

```
        }

    catch (Exception e)

        {

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

        }

    }

        public static void main(String args[])

        {

        new CalcServer();

        }

}



CalculatorClient

import java.rmi.Naming;

import java.rmi.RemoteException;

import java.net.MalformedURLException;

import java.rmi.NotBoundException;


public class CalculatorClient

    {


public static void main(String[] args)

        {

        try
```
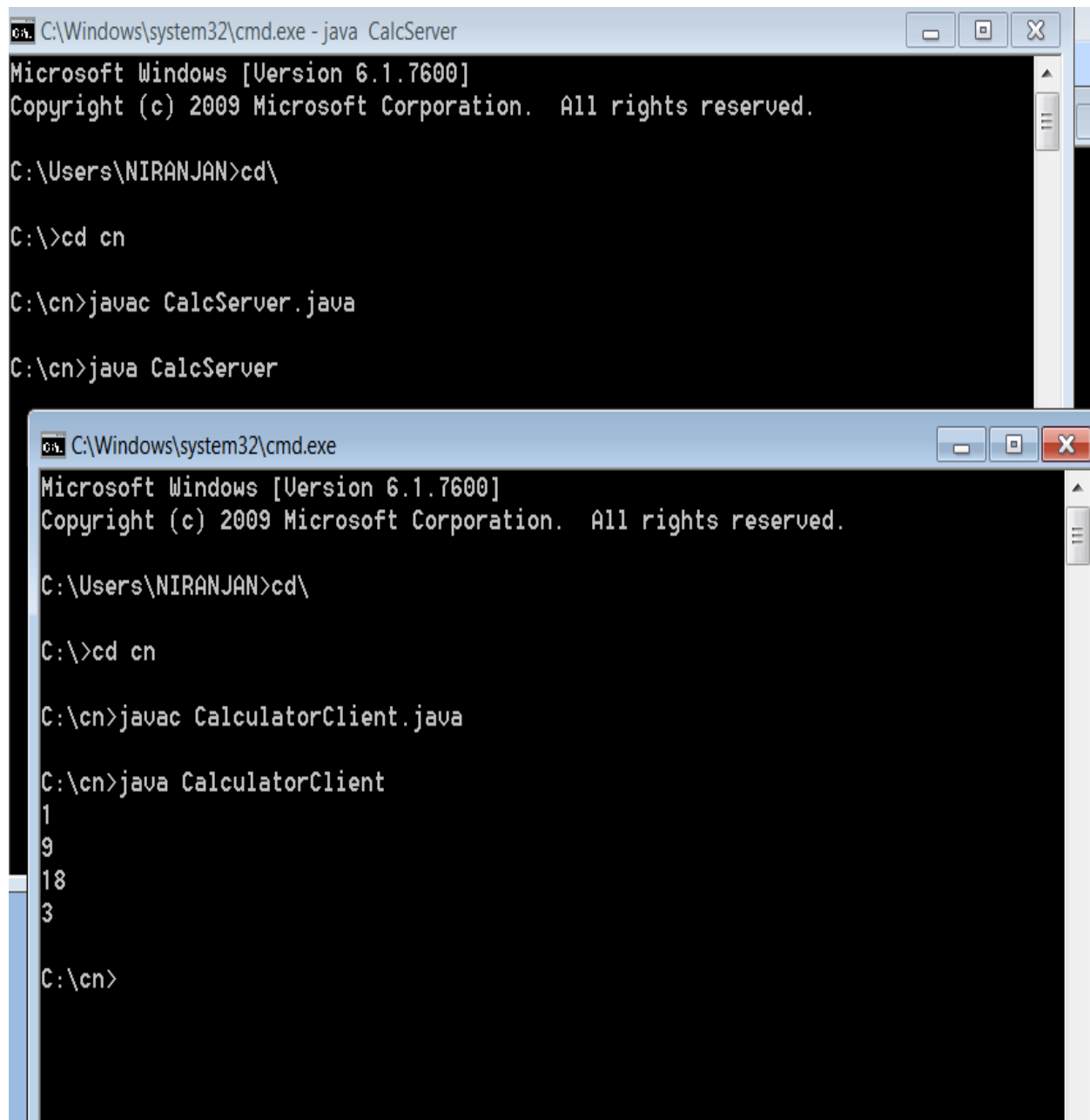
```
        {

        Calc c = (Calc)


        Naming.lookup("rmi://localhost/CalculatorService");


        System.out.println( c.subtraction(4, 3) );

        System.out.println( c.addition(4, 5) );

        System.out.println( c.multiplication(3, 6) );

        System.out.println( c.divition(9, 3) );

        }

     catch (MalformedURLException murle)

        {

     System.out.println();

        System.out.println("MalformedURLException");

        System.out.println(murle);

        }

        catch (RemoteException re)

        {

        System.out.println();

        System.out.println("RemoteException");

        System.out.println(re);

        }

        catch (NotBoundException nbe)

        {

        System.out.println();
```

27

```
System.out.println("NotBoundException");

System.out.println(nbe);

}

catch (java.lang.ArithmeticExceptionae)


{

System.out.println();

System.out.println("java.lang.ArithmeticException");

System.out.println(ae);

}

}

}
```

Output:

EXP NO: 7	**Chatting application using JAVA TCP and UDP sockets**
Date:	

AIM: Implement a Chatting application using JAVA TCP and UDP sockets.

Procedure:
1. In any Client/Server Application, we need to run the server before the client, because the server keeps waiting for the client to be connected.
2. Server keeps listening for the client on an assigned IP & Port
3. For establishing connection client must know the IP & Port of the server.
4. When we start Client Application, It creates a connection to the server.
5. After the Successful connection Client & Server Applications can send & receive messages.

Source Code:

charsever.java

```java
import java.net.*;
import java.io.*;
public class chatserver
{
public static void main(String args[]) throws Exception
{
ServerSocket ss=new ServerSocket(2000);
Socket sk=ss.accept();
BufferedReader cin=new BufferedReader(new
InputStreamReader(sk.getInputStream()));
PrintStream cout=new PrintStream(sk.getOutputStream());
BufferedReader stdin=new BufferedReader(new InputStreamReader(System.in));
String s;
while ( true )
{
s=cin.readLine();
if (s.equalsIgnoreCase("END"))
{
cout.println("BYE");
break;
}
System. out.print("Client : "+s+"\n");
System.out.print("Server : ");
s=stdin.readLine();
```

30

```
cout.println(s);
}
ss.close();
sk.close();
cin.close();
cout.close();
stdin.close();
}
}
```

chatclient.java

```java
import java.net.*;
import java.io.*;
public class chatclient
{
public static void main(String args[]) throws Exception
{
Socket sk=new Socket("127.0.0.1",2000);
BufferedReader sin=new BufferedReader(new
InputStreamReader(sk.getInputStream()));
PrintStream sout=new PrintStream(sk.getOutputStream());
BufferedReader stdin=new BufferedReader(new InputStreamReader(System.in));
String s;
while ( true )
{
System.out.print("Client : ");
s=stdin.readLine();
sout.println(s);
s=sin.readLine();
System.out.print("Server : "+s+"\n");
if ( s.equalsIgnoreCase("BYE") )
break;
}
sk.close();
sin.close();
sout.close();
stdin.close();
}
}
```

output:
Server:

E:\nwlab>javac *.java
E:\nwlab>java chatserver
Client : hi
Server : hi

Client:

E:\nwlab>java chatclient
Client : hi
Server : hi

Client :

EXP NO: 8	**Hello and Echo commands using JAVA**
Date:	

AIM: Hello command is used to know whether the machine at the other end is working or not. Echo command is used to measure the round trip time to the neighbour. Implement Hello and Echo commands using JAVA.

Source Code:

```java
import java.util.Scanner;
public class Echo
{
  public static void main (String[] args)
  {
    String inData;
    Scanner scan = new Scanner( System.in );

    System.out.println("Enter the data:");
    inData = scan.nextLine();

    System.out.println("You entered:" + inData );
  }
}
```

Output:
C:\> javac Echo.java
C:\> java Echo
Enter the data:
This is what the user typed.
You entered: This is what the user typed.

33

EXP NO: 9	**Inspect HTTP Traffic**
Date:	

AIM: Using Wireshark perform the following operations:
- Inspect HTTP Traffic
- .Inspect HTTP Traffic from a Given IP Address,
- Inspect HTTP Traffic to a Given IP Address,
- Reject Packets to Given IP Address,
- Monitor Apache and MySQL Network Traffic.

Procedure:

Wireshark is a software protocol analyzer, or "packet sniffer" application, used for network troubleshooting, analysis, software and protocol development, and education. As data streams travel back and forth over the network, the sniffer "captures" each protocol data unit (PDU) and can decode and analyze its content according to the appropriate RFC or other specifications.

Wireshark is a useful tool for anyone working with networks and can be used with most labs in the CCNA courses for data analysis and troubleshooting. This lab provides instructions for downloading and installing Wireshark.

Required Resources

- PC (Windows 7, 8, or 10 with internet access)

Download and Install Wireshark

Wireshark has become the industry standard packet-sniffer program used by network engineers. This open source software is available for many different operating systems, including Windows, Mac, and Linux. In this lab, you will download and install the Wireshark software program on your PC.

Step 1: Download Wireshark.

a. Wireshark can be downloaded from www.wireshark.org.
b. Click the icon above **Download**

34
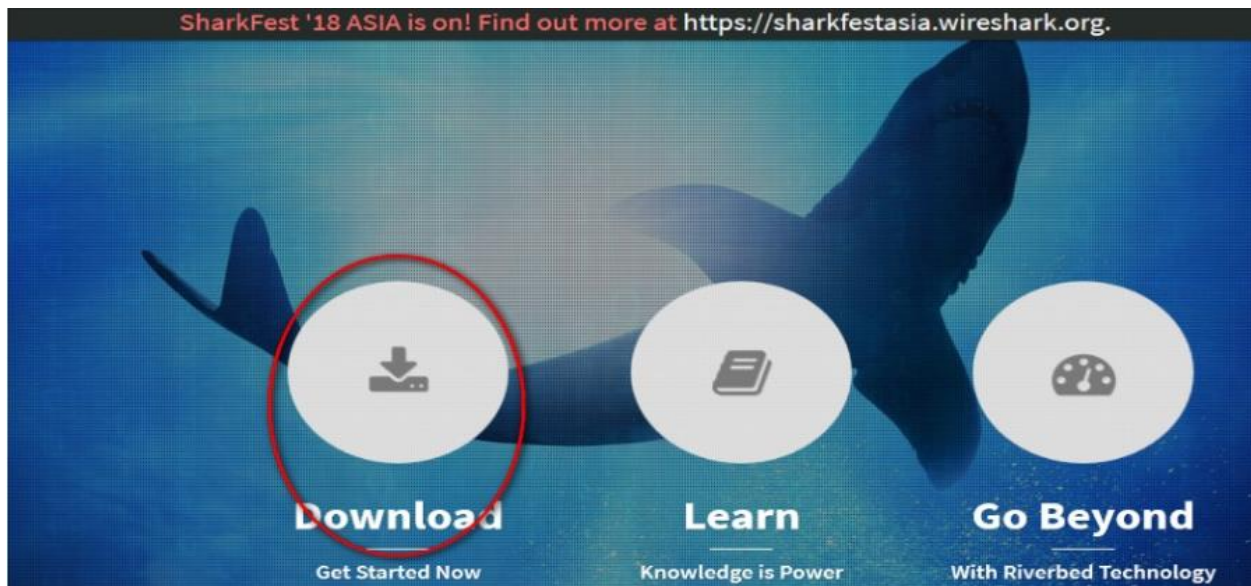
C. Choose the software version you need based on your PC architecture and operating system. For instance, if you have a 64-bit PC


Step 2: Install Wireshark.

a. The downloaded file is named **Wireshark-win64-x.x.x.exe**, where **x** represents the version number. Double-click the file to start the installation process.

b. Respond to any security messages that may display on your screen. If you already have a copy of Wireshark on your PC, you will be prompted to uninstall the old version before

35

installing the new version. It is recommended that you remove the old version of Wireshark prior to installing another version. Click Yes to uninstall the previous version of Wireshark.
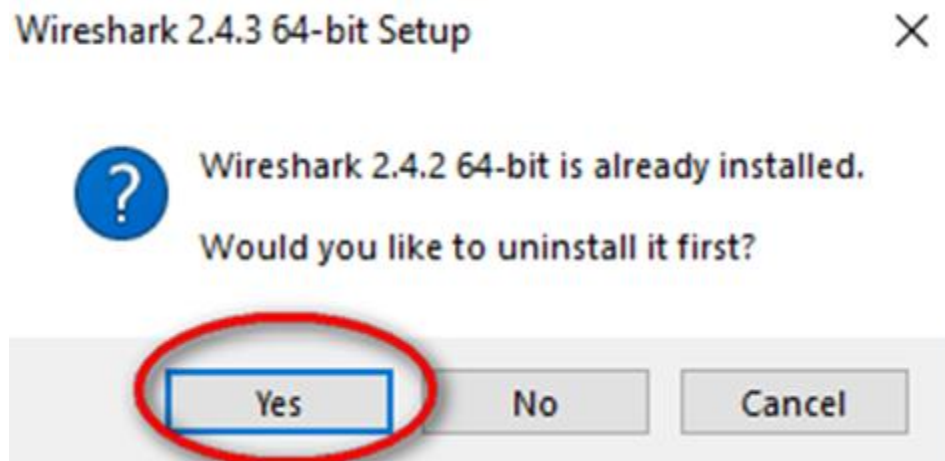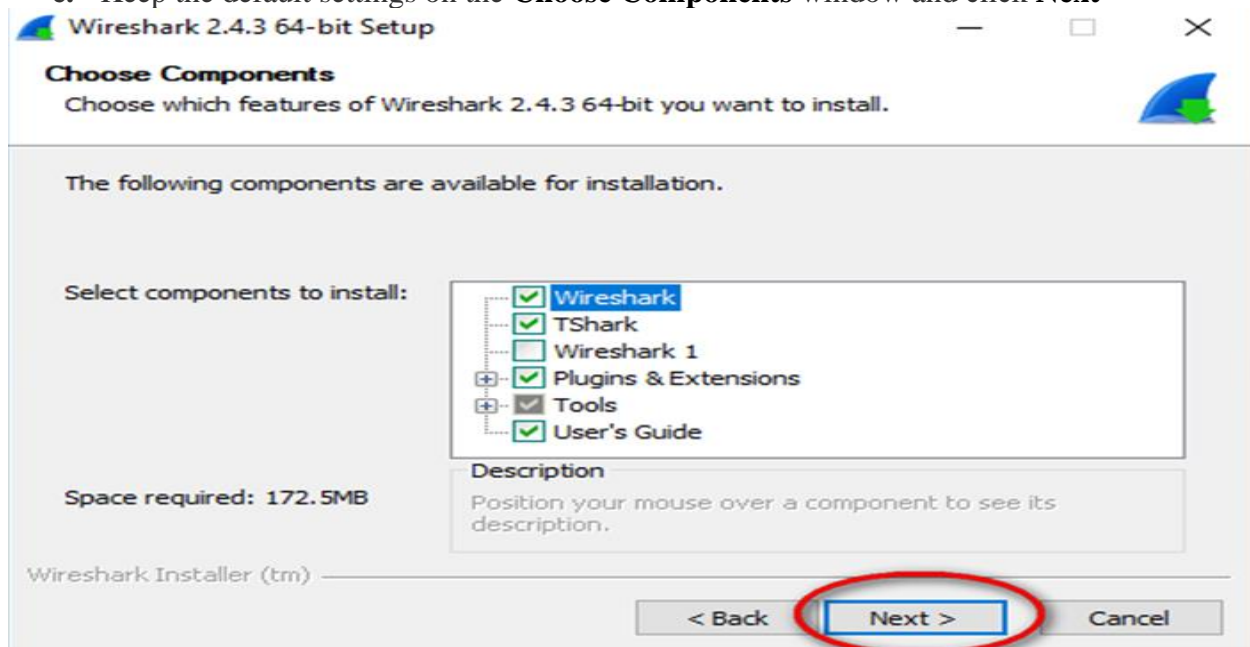

c. If this is the first time that you have installed Wireshark, or after you have completed the uninstall process, you will navigate to the **Wireshark Setup** wizard. Click **Next**.
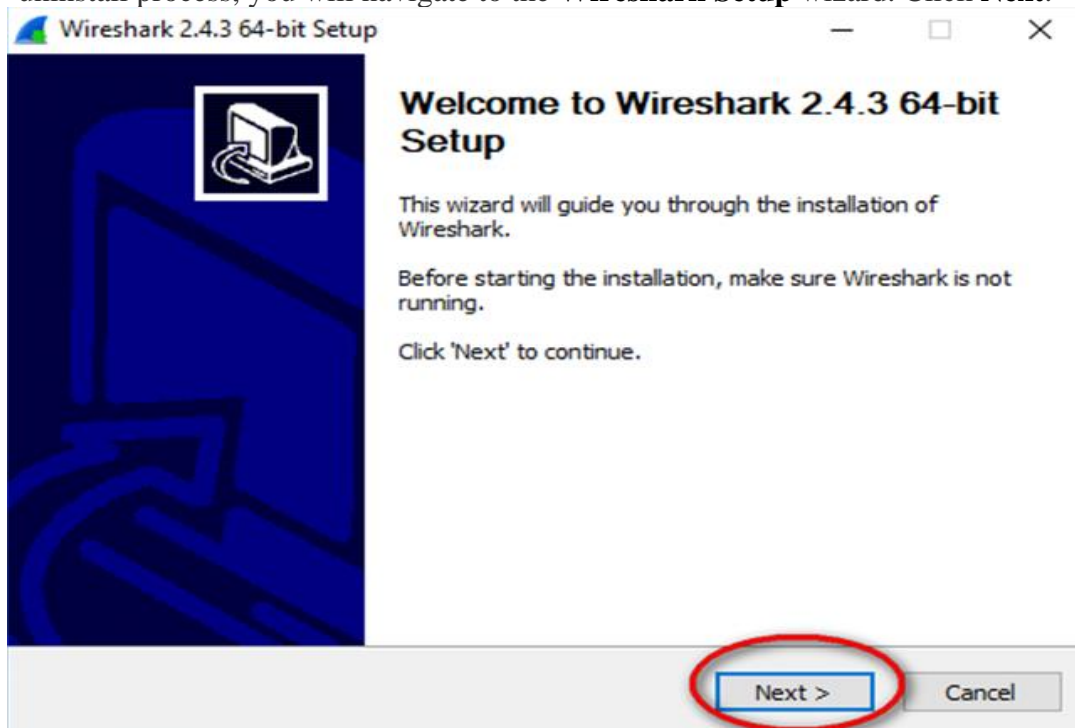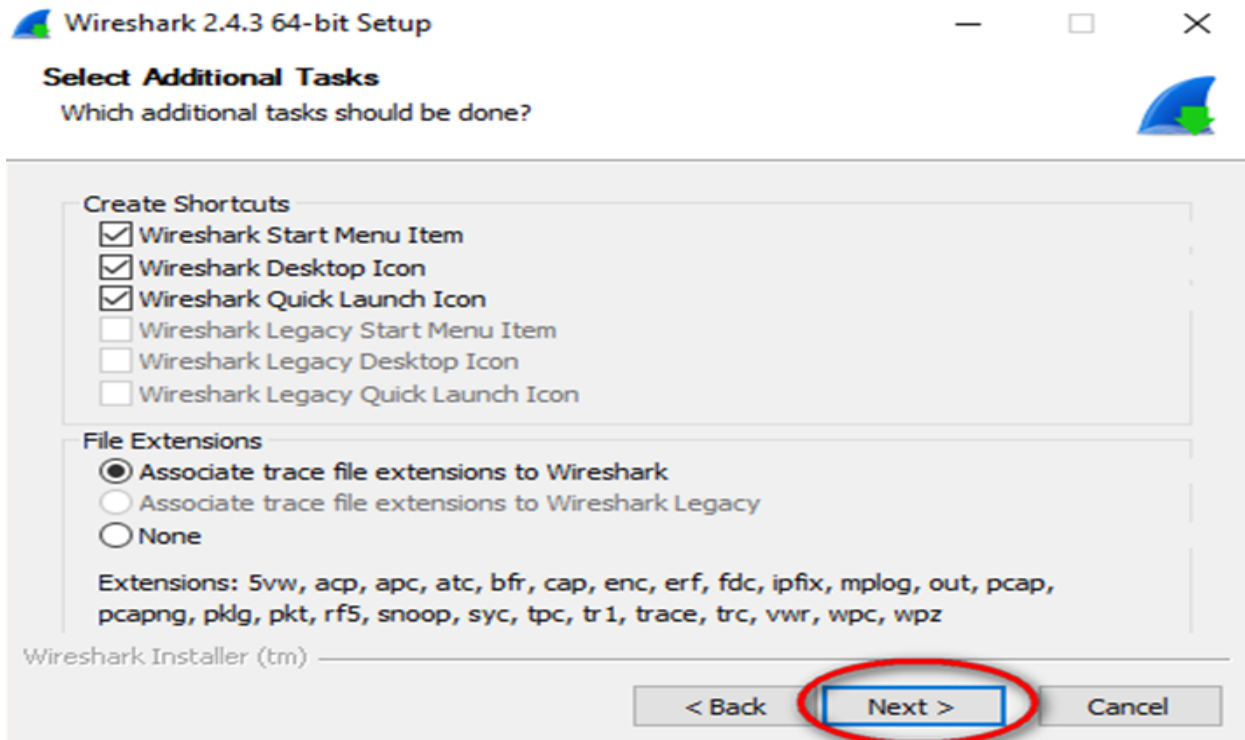

d. Continue advancing through the installation process. Click **I Agree** when the License Agreement window displays
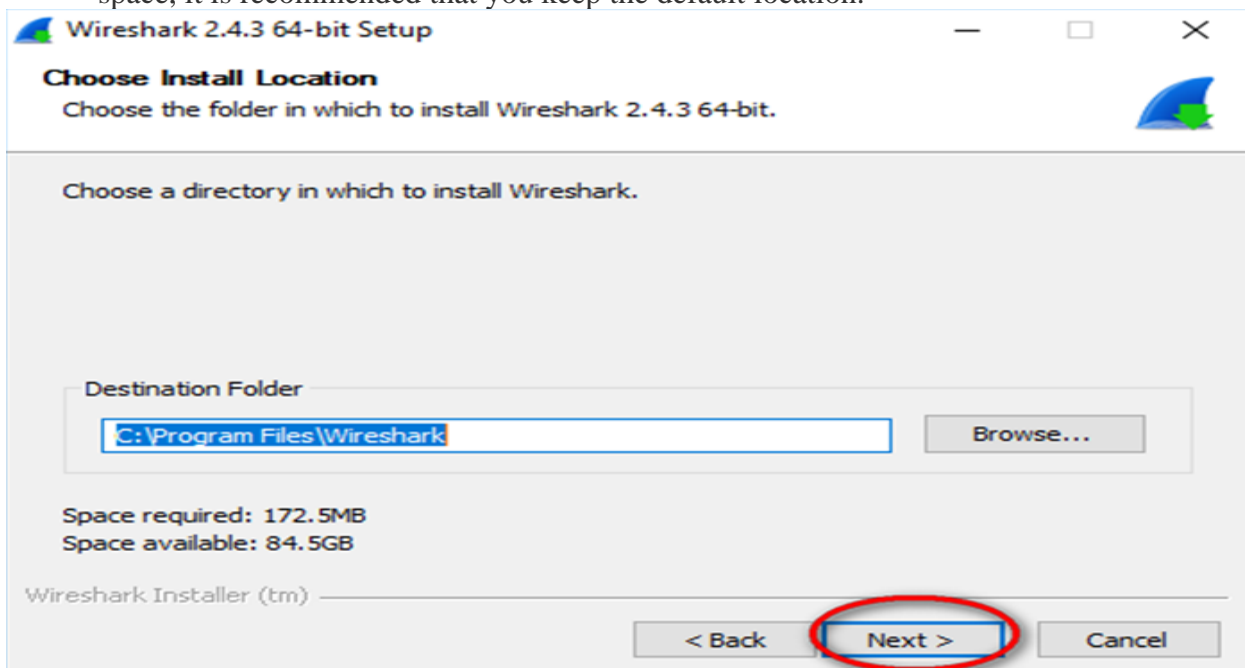
36
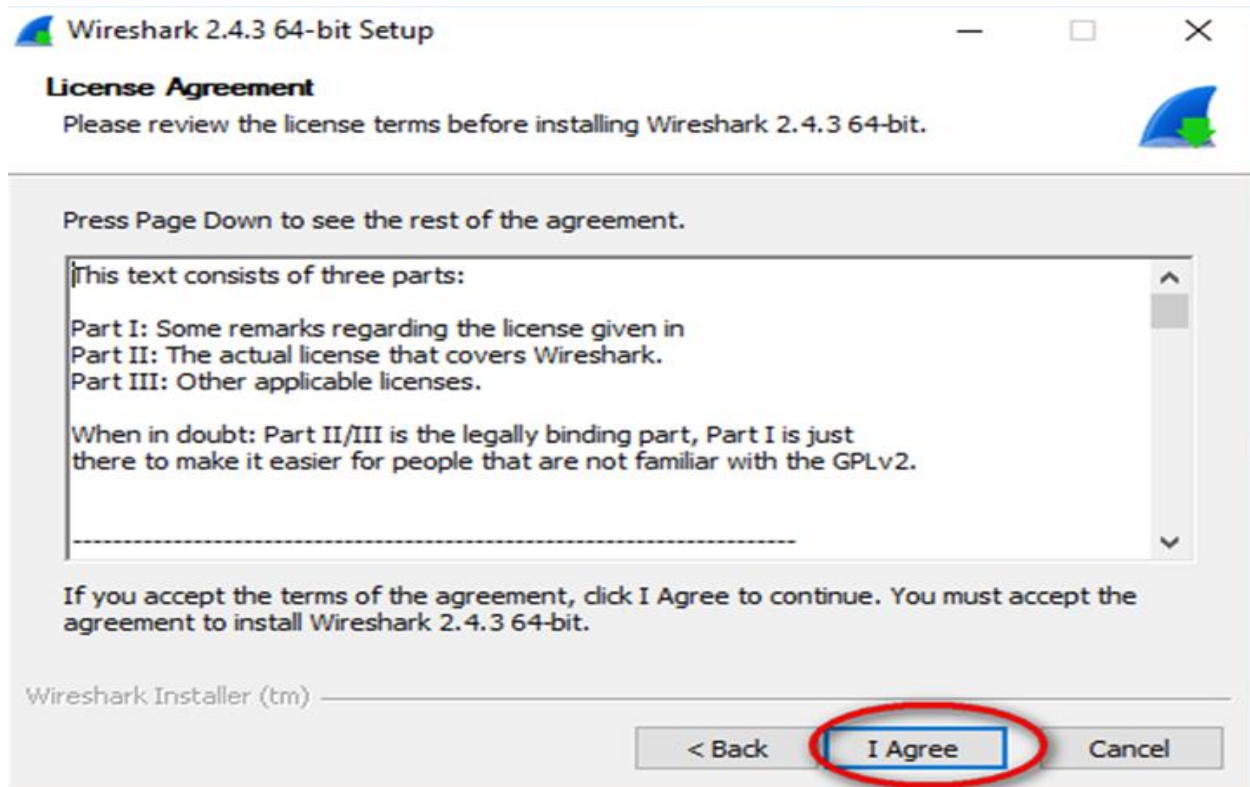
e. Keep the default settings on the **Choose Components** window and click **Next**

f. Choose your desired shortcut options and click **Next**.

g. You can change the installation location of Wireshark, but unless you have limited disk space, it is recommended that you keep the default location.

h. To capture live network data, WinPcap must be installed on your PC. If WinPcap is already installed on your PC, the Install check box will be unchecked. If your installed version of WinPcap is older than the version that comes with Wireshark, it is recommended that you allow the newer version to be installed by clicking the Install

38

WinPcap x.x.x (version number) check box.

**i.** Finish the WinPcap Setup wizard if installing WinPcap
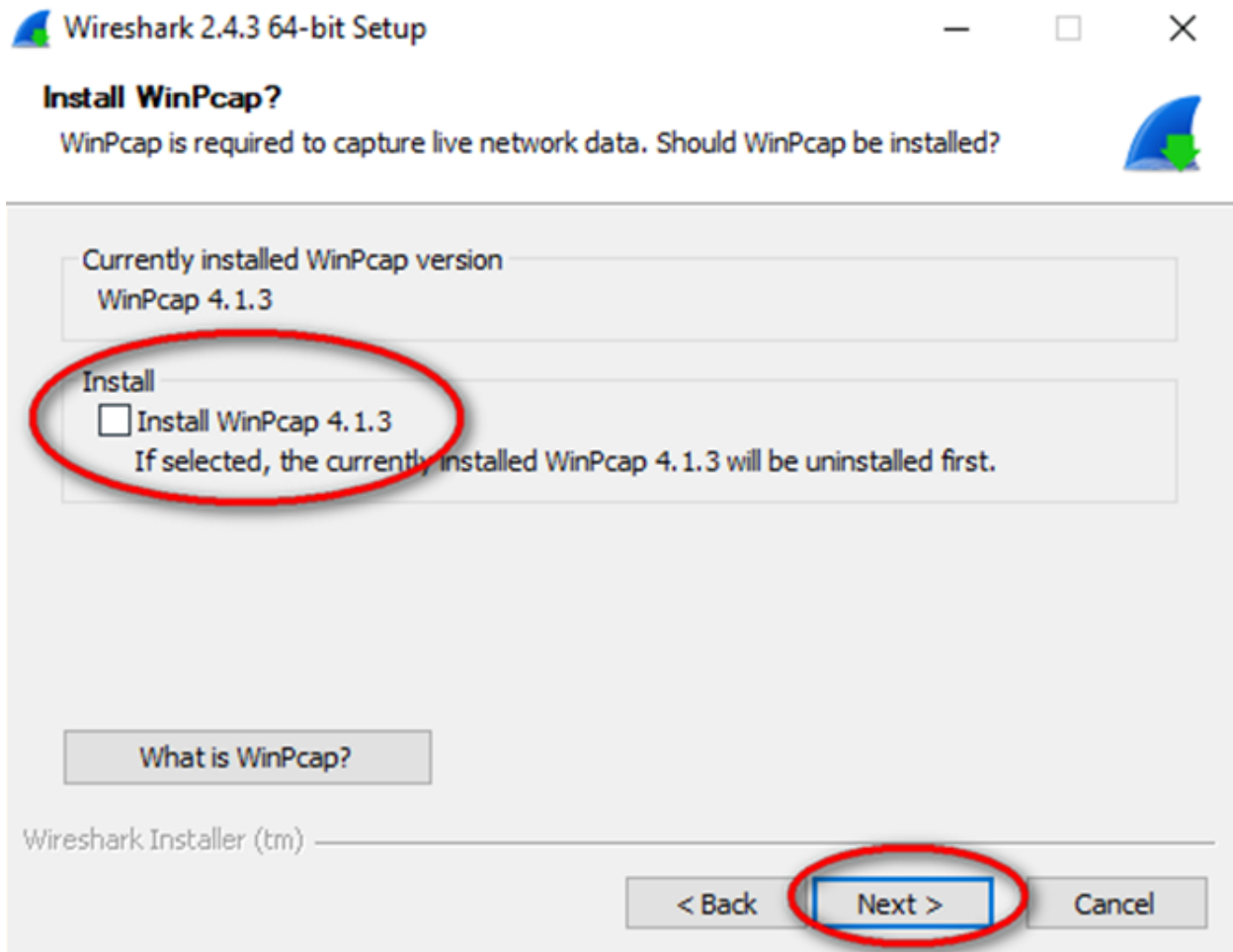
j. In addition, USBPcap can be installed on your PC. If USBPcap is already installed on your PC, the Install check box will be unchecked. If your installed version of USBPcap is older than the version that comes with Wireshark, it is recommended that you allow the newer version to be installed by clicking the **Install USBPcap x.x.x** (version number) check box.

Note: Because USBcap is still experimental, it is recommended that you **DO NOT** install USBcap unless you need to capture USB traffic.
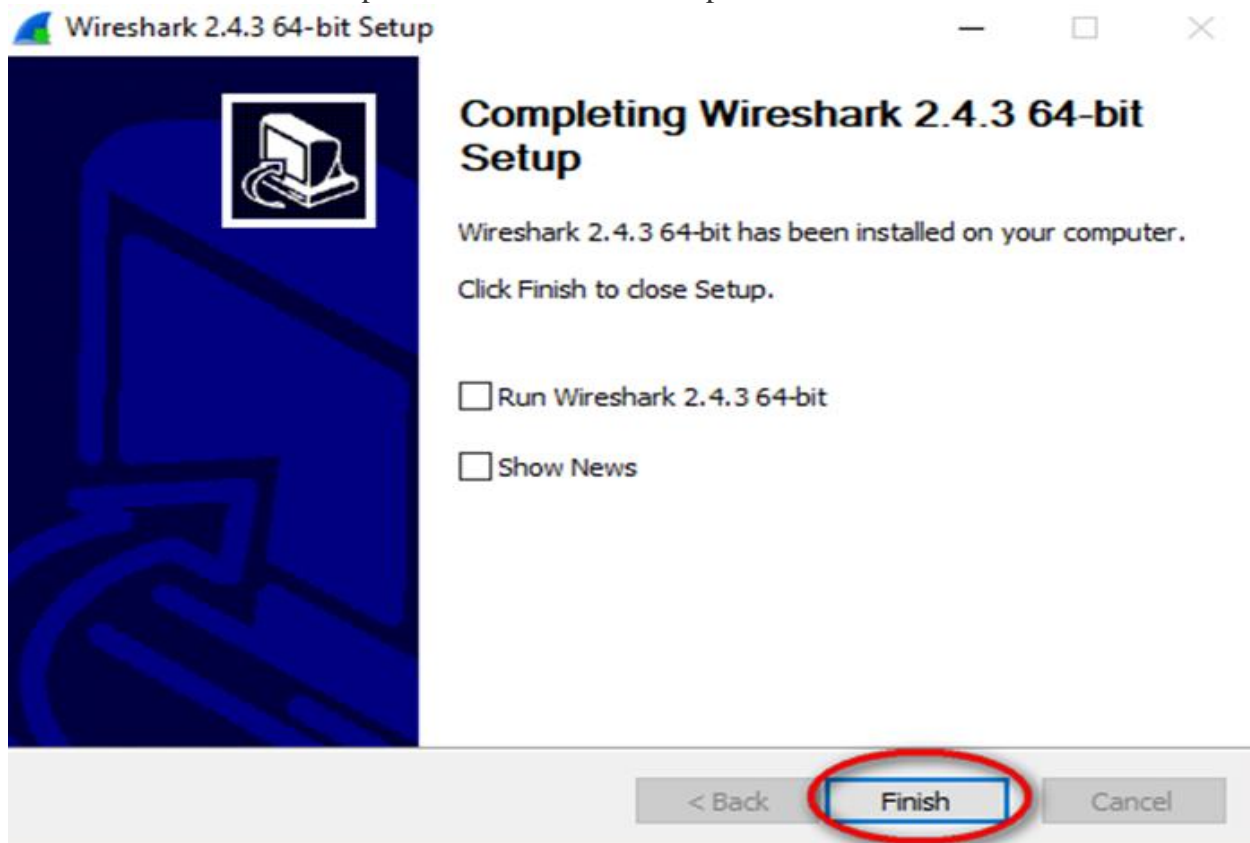
k. Finish the **USBPcap Setup** wizard if installing USBPcap

39
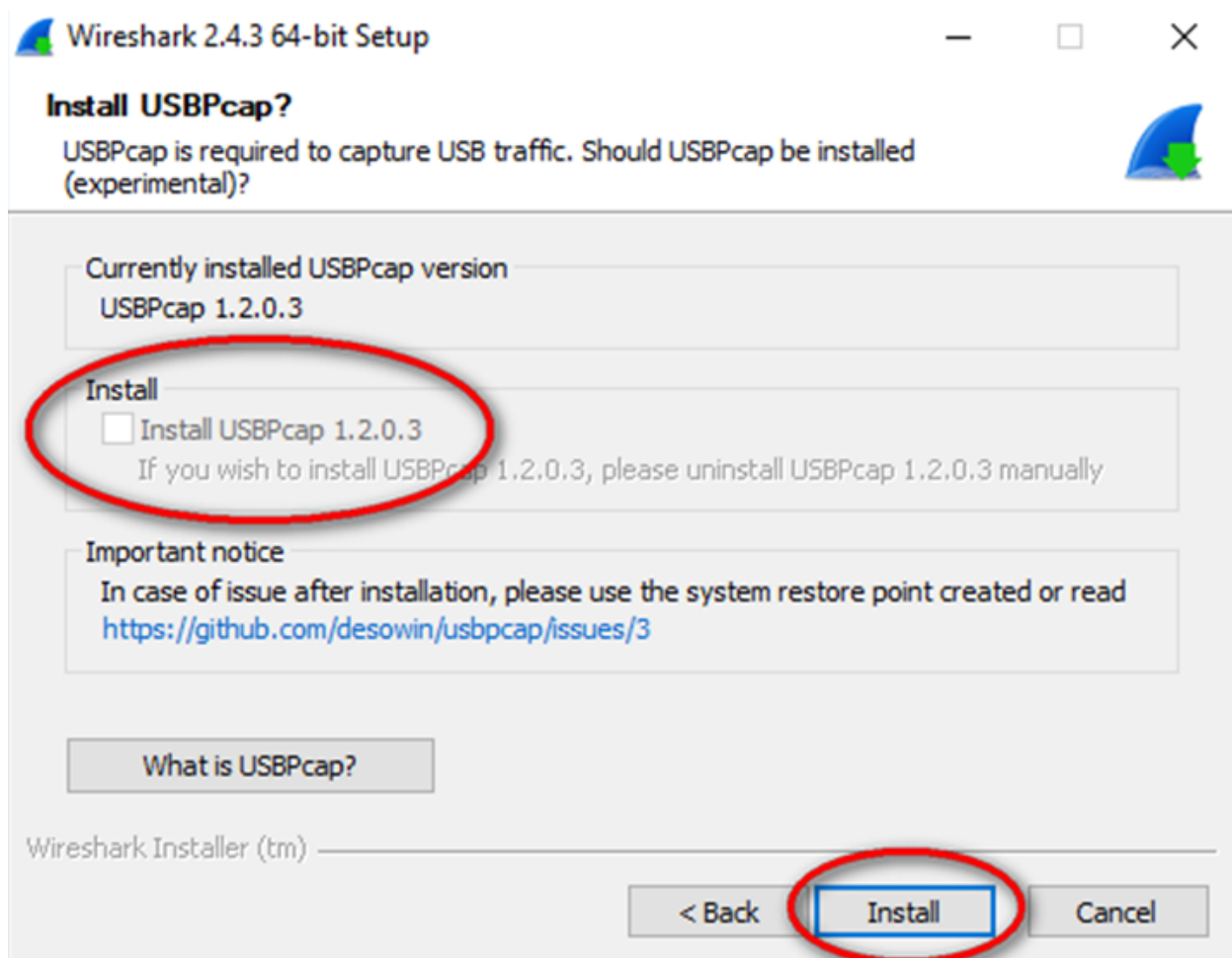
l. Wireshark starts installing its files, and a separate window displays with the status of the installation. Click **Next** when the installation is complete.

40
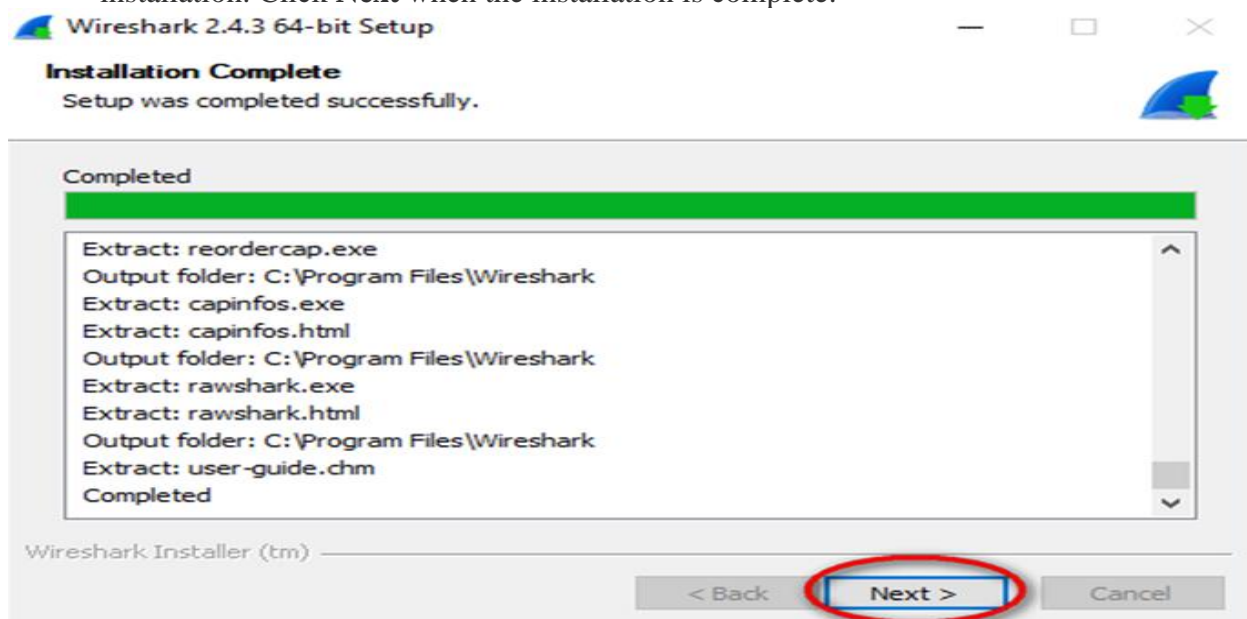
m. Click **Finish** to complete the Wireshark install process.

Result:

EXP NO: 10	**Network Simulater-2**
Date:	

AIM: Install Network Simulator 2/3. Create a wired network using dumbbell topology. Attach agents, generate both FTP and CBR traffic, and transmit the traffic. Vary the data rates and evaluate the performance using metric throughput, delay, jitter and packet loss.

Description:

Network Simulator (Version 2), widely known as NS2, is simply an event driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. Due to its flexibility and modular nature, NS2 has gained constant popularity in the networking research community since its birth in 1989. Ever since, several revolutions and revisions have marked the growing maturity of the tool, thanks to substantial contributions from the players in the field. Among these are the University of California and Cornell University who developed the REAL network simulator,1 the foundation which NS is based on. Since 1995 the Defense Advanced Research Projects Agency (DARPA) supported development of NS through the Virtual Inter Network Testbed (VINT) project . Currently the National Science Foundation (NSF) has joined the ride in development. Last but not the least, the group of Researchers and developers in the community are constantly working to keep NS2 strong and versatile.

NS uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols requires a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important. On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios.

In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important. NS meets both of these needs with two languages, C++ and OTcl.

Tcl scripting

- o Tcl is a general purpose scripting language. [Interpreter]
- o Tcl runs on most of the platforms such as Unix, Windows, and Mac.
- o The strength of Tcl is its simplicity.
- o It is not necessary to declare a data type for variable prior to the usage.

42

Basics of TCL

Syntax: command arg1 arg2 arg3

Hello World!

puts stdout{Hello, World!} Hello, World!

Variables Command Substitution

set a 5 set len [string length foobar]

set b $a set len [expr [string length foobar] + 9]

Wired TCL Script Components

Create the event scheduler

Open new files & turn on the tracing

Create the nodes

Setup the links

Configure the traffic type (e.g., TCP, UDP, etc)

Set the time of traffic generation (e.g., CBR, FTP)

Terminate the simulation

NS Simulator Preliminaries.

1. Initialization and termination aspects of the ns simulator.

2. Definition of network nodes, links, queues and topology.

3. Definition of agents and of applications.

4. The nam visualization tool.

5. Tracing and random variables.

Initialization and Termination of TCL Script in NS-2

An ns simulation starts with the command

set ns [new Simulator]

Which is thus the first line in the tcl script? This line declares a new variable as using the set command, you can call this variable as you wish, In general people declares it as ns because it is an instance of the Simulator class, so an object the code[new Simulator] is indeed the installation of the class Simulator using the reserved word new.

Output:

43

EXP NO: 11	**Network Simulater-2 FTP & CBR Traffic**
Date:	

AIM: Create a static wireless network. Attach agents, generate both FTP and CBR traffic, and transmit the traffic. Vary the data rates and evaluate the performance using metric throughput, delay, jitter and packet loss.

DESCRIPTION

NS uses two languages because simulator has two different kinds of things it needs to do. On one hand, detailed simulations of protocols requires a systems programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For these tasks run-time speed is important and turn-around time (run simulation, find bug, fix bug, recompile, re-run) is less important. On the other hand, a large part of network research involves slightly varying parameters or configurations, or quickly exploring a number of scenarios.

In these cases, iteration time (change the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of this part of the task is less important. NS meets both of these needs with two languages, C++ and OTcl.

Tcl scripting

- o Tcl is a general purpose scripting language. [Interpreter]
- o Tcl runs on most of the platforms such as Unix, Windows, and Mac.
- o The strength of Tcl is its simplicity.
- o It is not necessary to declare a data type for variable prior to the usage.

In order to have output files with data on the simulation (trace files) or files used for visualization (nam files), we need to create the files using ―open command:

#Open the Trace file

> **set tracefile1 [open out.tr w]**
> **$ns trace-all $tracefile1**

#Open the NAM trace file

> **set namfile [open out.nam w]**
> **$ns namtrace-all $namfile**

The above creates a dta trace file called out.tr and a nam visualization trace file called out. nam.Within the tcl script, these files are not called explicitly by their names, but instead by pointers that are declared above and called ―tracefile1 and ―namfile respectively. Remark that

44

they begins with a # symbol. The second line open the file —out.tr to be used for writing, declared with the letter —w. The third line uses a simulator method called trace-all that have as parameter the name of the file where the traces will go.

Define a "finish" procedure

Proc finish { } {
global ns tracefile1 namfile
$ns flush-trace
Close $tracefile1
Close $namfile
Exec nam out.nam &
Exit 0
}

Definition of a network of links and nodes
The way to define a node is

set n0 [$ns node]

Once we define several nodes, we can define the links that connect them. An example of a definition of a link is:

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

Which means that $n0 and $n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10Mb per sec for each direction.
To define a directional link instead of a bi-directional one, we should replace —duplex-link by —simplex-link.
In ns, an output queue of a node is implemented as a part of each link whose input is that node. We should also define the buffer capacity of the queue related to each link. An example would be:

#set Queue Size of link (n0-n2) to 20

FTP over TCP
TCP is a dynamic reliable congestion control protocol. It uses Acknowledgements created by the destination to know whether packets are well received.
There are number variants of the TCP protocol, such as Tahoe, Reno, NewReno, Vegas. The type of agent appears in the first line:
#Setup a UDP connection

set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n5 $null
$ns connect $udp $null

45

$udp set fid_2

#setup a CBR over UDP connection
The below shows the definition of a CBR application using a UDP agent
The command $ns attach-agent $n4 $sink defines the destination node. The command $ns
connect $tcp $sink finally makes the TCP connection between the source and destination nodes.

set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set packetsize_ 100
$cbr set rate_ 0.01Mb
$cbr set random_ false

TCP has many parameters with initial fixed defaults values that can be changed if mentioned
explicitly. For example, the default TCP packet size has a size of 1000bytes.This can be
changed to another value, say 552bytes, using the command $tcp set packetSize_ 552.
When we have several flows, we may wish to distinguish them so that we can identify them
with different colors in the visualization part. This is done by the command $tcp set fid_ 1 that
assigns to the TCP connection a flow identification of ―1.We shall later give the flow
identification of ―2‖ to the UDP connection.

Output:

EXP NO: 12	**Network Simulater-2 FTP & CBR Traffic**
Date:	

AIM: Create a mobile wireless network. Attach agents, generate both FTP and CBR traffic, and transmit the traffic. Vary the data rates and evaluate the performance using metric throughput, delay, jitter and packet loss.

Description:

A wireless sensor network (WSN) consists of a large number of small sensor nodes that are deployed in the area in which a factor is to be monitored. In wireless sensor network, energy model is one of the optional attributes of a node. The energy model denotes the level of energy in a mobile node. The components required for designing energy model includes initialEnergy, txPower, rxPower, and idlePower. The "initialEnergy" represents the level of energy the node has at the initial stage of simulation. "txPower" and "rxPower" denotes the energy consumed for transmitting and receiving the packets. If the node is a sensor, the energy model should include a special component called "sensePower". It denotes the energy consumed during the sensing operation. Apart from these components, it is important to specify the communication range (RXThresh_) and sensing range of a node (CSThresh_). The sample 18.tcl designs a WSN in which sensor nodes are configured with different communication and sensing range. Base Station is configured with highest communication range. Data Transmission is established between nodes using UDP agent and CBR traffic.

Algorithm:
1. Create a simulator object
2. Define setting options for wireless channel
3. Create trace file and name file
4. Setup topography object and nodes
5. Provide initial location of mobile nodes
6. Setup a UDP connection between nodes
7. Printing the window size

Program:
```
# Define setting options
set val(chan) Channel/WirelessChannel ;# channel type
set val(prop) Propagation/TwoRayGround ;# radio-propagation model
set val(netif) Phy/WirelessPhy ;# network interface type
set val(mac) Mac/802_11 ;# MAC type
```

47

```
set val(ifq) Queue/DropTail/PriQueue ;# interface queue type
set val(ll) LL ;# link layer type
set val(ant) Antenna/OmniAntenna ;# antenna model
set val(ifqlen) 50 ;# max packet in ifq
set val(nn) 10 ;# number of mobilenodes
set val(rp) DSDV ;# routing protocol
set val(x) 500 ;# X dimension of topography
set val(y) 400 ;# Y dimension of topography
set val(stop) 150 ;# time of simulation end
set ns [new Simulator]
#Creating trace file and nam file
set tracefd [open dsdv.tr w]
set windowVsTime2 [open win.tr w]
set namtrace [open dsdv.nam w]
$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)
# set up topography object
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)
# configure the nodes
$ns node-config -adhocRouting $val(rp) \
-llType $val(ll) \
-macType $val(mac) \
-ifqType $val(ifq) \
-ifqLen $val(ifqlen) \
-antType $val(ant) \
-propType $val(prop) \
-phyType $val(netif) \
-channelType $val(chan) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace ON
for {set i 0} {$i < $val(nn) } { incr i } {
set node_($i) [$ns node]
}
# Provide initial location of mobilenodes
$node_(0) set X_ 5.0
```

48

```
$node_(0) set Y_ 800.0
$node_(0) set Z_ 0.0
$node_(1) set X_ 8.0
$node_(1) set Y_ 650.0
$node_(1) set Z_ 0.0
$node_(2) set X_ 60.0
$node_(2) set Y_ 450.0
$node_(2) set Z_ 0.0
$node_(3) set X_ 10.0
$node_(3) set Y_ 480.0
$node_(3) set Z_ 0.0
#another four
$node_(4) set X_ 350.0
$node_(4) set Y_ 500.0
$node_(4) set Z_ 0.0
$node_(5) set X_ 150.0
$node_(5) set Y_ 850.0
$node_(5) set Z_ 0.0
$node_(6) set X_ 200.0
$node_(6) set Y_ 500.0
$node_(6) set Z_ 0.0
#$node_(7) set X_ 320.0
$node_(7) set X_ 320.0
$node_(7) set Y_ 650.0
$node_(7) set Z_ 0.0
#another four
$node_(8) set X_ 250.0
$node_(8) set Y_ 700.0
$node_(8) set Z_ 0.0
$node_(9) set X_ 400.0
$node_(9) set Y_ 800.0
$node_(9) set Z_ 0.0
####################################
#copy of the data
####################################
#####Attack Node
$node_(5) color Green
$ns at 10.0 "$node_(5) color Green"
$ns at 10.0 "$node_(5) label EndUser"
# Set a udp connection between node_(5) and node_(8)
```

49

```
set udp [new Agent/UDP]
set sink [new Agent/LossMonitor]
#$ns attach-agent $node_(5) $udp
#$ns attach-agent $node_(8) $sink
$ns attach-agent $node_(8) $udp
$ns attach-agent $node_(5) $sink
$ns connect $udp $sink
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$node_(8) color Green
$ns at 10.0 "$node_(8) color yellow"
$ns at 10.0 "$node_(5) label node5 "
$ns at 10.0 "$node_(8) label node8 "
$ns at 10.0 "$cbr start"
$ns at 10.45 "$cbr stop"
# Set a udp connection between node_(8) and node_(6)
set udp1 [new Agent/UDP]
set sink1 [new Agent/LossMonitor]
$ns attach-agent $node_(8) $udp1
$ns attach-agent $node_(6) $sink1
$ns connect $udp1 $sink1
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$node_(6) color yellow
$ns at 10.0 "$node_(6) color yellow"
$ns at 10.0 "$node_(6) label node6 "
$ns at 10.0 "$cbr1 start"
$ns at 10.45 "$cbr1 stop"
# Set a udp connection between node_(8) and node_(6)
set udp2 [new Agent/UDP]
set sink2 [new Agent/LossMonitor]
$ns attach-agent $node_(5) $udp2
$ns attach-agent $node_(8) $sink2
$ns connect $udp2 $sink2
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$node_(6) color yellow
$ns at 12.0 "$cbr2 start"
$ns at 13.45 "$cbr2 stop"
# Set a udp connection between node_(8) and node_(6)
```

50

```
set udp3 [new Agent/UDP]
set sink3 [new Agent/LossMonitor]
$ns attach-agent $node_(8) $udp3
$ns attach-agent $node_(6) $sink3
$ns connect $udp3 $sink3
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp1
$node_(6) color yellow
$ns at 14.0 "$cbr3 start"
$ns at 16.45 "$cbr3 stop"
# Printing the window size
proc plotWindow {tcpSource file} {
global nss
set time 0.01
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
#$ns at 10.1 "plotWindow $tcp $windowVsTime2"
# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
# 30 defines the node size for nam
$ns initial_node_pos $node_($i) 30
}
# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
#$ns at $val(stop) "$node_($i) reset";
#$ns at 200.25 "$node_($i) reset";
$ns at 21.25 "$node_($i) reset";
}
$ns at 0.00 "$ns trace-annotate \"Wireless Mac Ptotocol \""
$ns at 10.0 "$ns trace-annotate \" Data send in node5 to node8 \""
$ns at 10.0 "$ns trace-annotate \" Data send in node6 to node8 \""
$ns at 10.45 "$ns trace-annotate \" Data Collision \""
$ns at 12.00 "$ns trace-annotate \"Data send in node5 to node8\""
$ns at 14.00 "$ns trace-annotate \"Data send in node6 to node8\""
$ns at 21.25 "$ns trace-annotate \"End simulation\""
$ns at 21.25 "$ns nam-end-wireless 21.01"
$ns at 21.25 "stop"
$ns at 22.01 "puts \"end simulation\" ; $ns halt"
```

51

```
proc stop { } {
global ns tracefd namtrace
$ns flush-trace
close $tracefd
close $namtrace
exec nam dsdv.nam &
exit 0
}
$ns run
```

Expected Output:
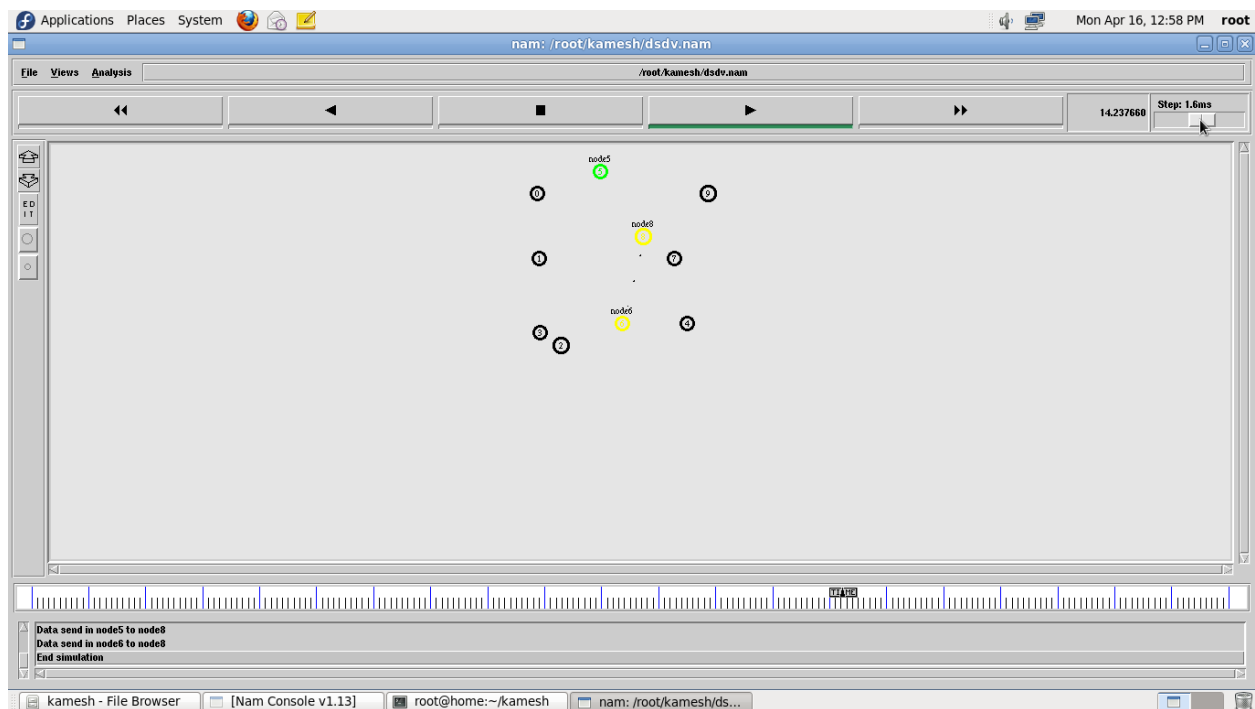
ADDITIONAL EXPERIMENTS

EXP NO: 13 Date:	SIMULATE A MOBILE ADHOC NETWORK

AIM: To simulate a Mobile Adhoc network (MANET) using NS2.

SOFTWARE REQUIREMENTS:

Network Simulator -2

THEORY:

A mobile ad hoc network or MANET does not depend on a fixed infrastructure for its networking operation. MANET is an autonomous and short-lived association of group of mobile nodes that communicate with each other over wireless links. A node can directly communicate to the nodes that lie within its communication range. If a node wants to communicate with a node that is not directly within its communication range, it uses intermediate nodes as routers.

ALGORITHM:

1. Create a simulator object
2. Set the values for the parameter
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create the nodes that forms a network numbered from 0 to 3
5. Schedule events and run the program.

PROGRAM:
```
set val(chan) Channel/WirelessChannel
set val(prop) Propagation/TwoRayGround
set val(netif) Phy/WirelessPhy
set val(mac) Mac/802_11
set val(ifq) Queue/DropTail/PriQueue
set val(ll) LL
set val(ant) Antenna/OmniAntenna
set val(ifqlen) 50
set val(nn) 3
set val(rp) DSDV
set ns [new Simulator]
set tf [open output.tr w]
$ns trace-all $tf
set tf1 [open output.nam w]
```

53

```
$ns namtrace-all-wireless $tf1 100 100
set topo [new Topography]
$topo load_flatgrid 100 100
create-god $val(nn)
$ns node-config -adhocRouting $val(rp) \
-llType $val(ll) \
-macType $val(mac) \
-ifqType $val(ifq) \
-ifqLen $val(ifqlen) \
-antType $val(ant) \
-propType $val(prop) \
-phyType $val(netif) \
-channelType $val(chan) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace OFF \
-macTrace OFF \
-movementTrace OFF
set node0 [$ns node]
set node1 [$ns node]
set node2 [$ns node]
$ns initial_node_pos $node0 10
$ns initial_node_pos $node1 10
$ns initial_node_pos $node2 10
$node0 set X_ 25.0
$node0 set Y_ 50.0
$node0 set Z_ 0.0
$node1 set X_ 50.0
$node1 set Y_ 50.0
$node1 set Z_ 0.0
$node2 set X_ 65.0
$node2 set Y_ 50.0
$node2 set Z_ 0.0
set tcp1 [new Agent/TCP]
$ns attach-agent $node0 $tcp1
set ftp [new Application/FTP]
$ftp attach-agent $tcp1
set sink1 [new Agent/TCPSink]
$ns attach-agent $node2 $sink1
$ns connect $tcp1 $sink1
```

54

$ns at 10.0 "$node1 setdest 50.0 90.0 0.0"
$ns at 50.0 "$node1 setdest 50.0 10.0 0.0"
$ns at 0.5 "$ftp start"
$ns at 1000 "$ftp stop"
$ns at 1000 "finish"
proc finish { } {
global ns tf tf1
$ns flush-trace
close $tf
exec nam output.nam &
exit 0
}
$ns run

EXPECTED OUTPUT:
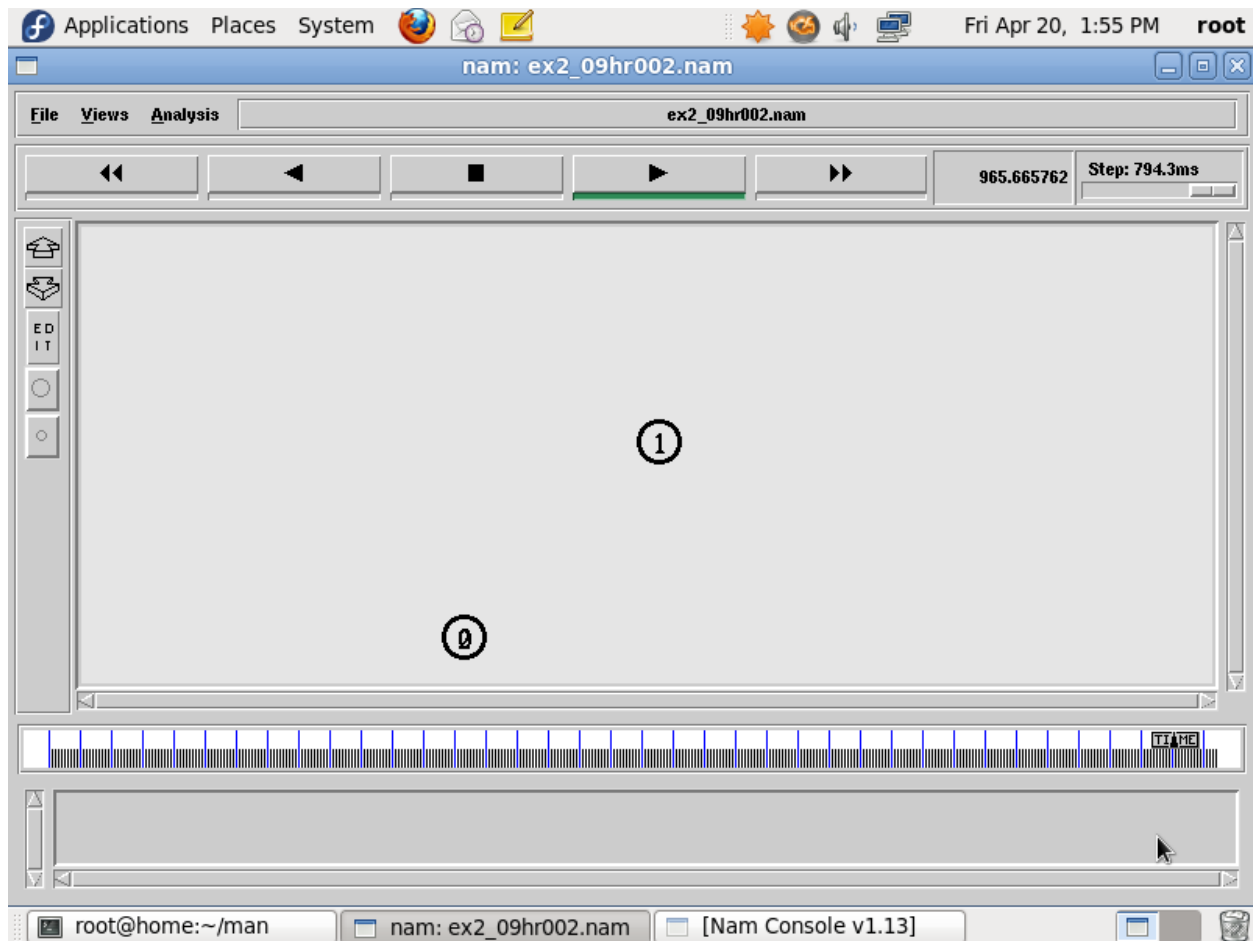
EXP NO: 14	SIMULATE A MOBILE ADHOC NETWORK
Date:	

AIM: To implement a Transport Control Protocol in sensor network through the simulator.

SOFTWARE REQUIREMENTS:

Network Simulator -2

THEORY:

The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP). Therefore, the entire suite is commonly referred to as TCP/IP. TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating by an IP network. Major Internet applications such as the World Wide Web, email, remote administration, and file transfer rely on TCP. Applications that do not require reliable data stream service may use the User Datagram Protocol (UDP), which provides a connectionless datagram service that emphasizes reduced latency over reliability.

ALGORITHM:

1. Create a simulator object
2. Define different colors for different data flows
3. Open a nam trace file and define finish procedure then close the trace file, and execute nam on trace file.
4. Create six nodes that forms a network numbered from 0 to 5
5. Create duplex links between the nodes and add Orientation to the nodes for setting a LAN topology
6. Setup TCP Connection between n(0) and n(4)
7. Apply FTP Traffic over TCP
8. Setup UDP Connection between n(1) and n(5)
9. Apply CBR Traffic over UDP.
10. Schedule events and run the program.

PROGRAM:
set ns [new Simulator]
#Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red
#Open the Trace files
set file1 [open out.tr w]
set winfile [open WinFile w]

56

```
$ns trace-all $file1
#Open the NAM trace file
set file2 [open out.nam w]
$ns namtrace-all $file2
#Define a 'finish' procedure
proc finish {} {
global ns file1 file2
$ns flush-trace
close $file1
close $file2
exec nam out.nam &
exit 0
}
#Create six nodes
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]
$n1 color red
$n1 shape box
#Create links between the nodes
$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
set lan [$ns newLan "$n3 $n4 $n5" 0.5Mb 40ms LL Queue/DropTail MAC/Csma/Cd Channel]
#Setup a TCP connection
set tcp [new Agent/TCP/Newreno]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink/DelAck]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_ 1
$tcp set window_ 8000
$tcp set packetSize_ 552
#Setup a FTP over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
```

57

$ftp set type_ FTP

$ns at 1.0 "$ftp start"
$ns at 124.0 "$ftp stop"

# next procedure gets two arguments: the name of the
# tcp source node, will be called here "tcp",
# and the name of output file.
proc plotWindow {tcpSource file} {
global ns
set time 0.1
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "$now $cwnd"
$ns at [expr $now+$time] "plotWindow $tcpSource $file" }
$ns at 0.1 "plotWindow $tcp $winfile"
$ns at 5 "$ns trace-annotate \"packet drop\""
PPP
$ns at 125.0 "finish"
$ns run

Expected Output: