OBJECTIVES

✓ To understand the basic concepts of Networking, Networking devices.
✓ To Learn socket programming.
✓ To have hands on experience on various networking commands.
✓ To implement protocols like DNS, SNMP and HTTP.
✓ To gain knowledge about the network simulation tools.

COURSE CONTENTS

1. Study of system administration and network administration
2. Study of socket programming and client server model using TCP and UDP
3. Implementation of
   a. Sliding window protocol
   b. Stop and wait protocol
4. Write a code simulating PING and TRACEROUTE commands
5. Applications using TCP Sockets like
   a. File transfer
   b. Remote command execution
   c. Chat
   d. Concurrent server
6. Create a socket for HTTP for webpage upload and download
7. Implementation of Subnetting
8. Applications
   a. DNS
   b. SNMP
9. Study of Network Simulator(NS)
10. Study of Wire Shark Tool for SDN and Hypervisor for Network Virtualization.
11. Perform a case study about the different routing algorithms to select the network path with its optimum and economical during data transfer.
   a. Link State routing
   b. Flooding
   c. Distance vector

TOTAL: 45
System administrator:

System administrator, or sysadmin, is a person who is responsible for the upkeep, configuration, and reliable operation of computer systems; especially multi-user computers, such as servers. The system administrator seeks to ensure that the uptime, performance, resources, and security of the computers he or she manages meet the needs of the users, without exceeding the budget.

Skills:

- Entails a knowledge of operating systems and applications
- Problem solving Technique.
- To understand the behaviour of software in order to deploy it and for troubleshooting problem.

Responsibilities of the System Administrator:

- User account management ,
- Hardware management
- Perform file system backups, restores
- Install and configure new software and services
- Keep systems and services operating
- Monitor system and network ,Troubleshoot problems
- Maintain documentation
- Audit security ,Help users,
- Performance tuning.

Network administration:

A network administrator, sometimes called a systems administrator, is responsible for keeping an organization’s computer network up to date and running smoothly. Any company or organization that uses multiple computers or software platforms needs a network admin to coordinate the different systems. Network admins will especially be in high demand as companies and organizations invest in newer, faster technology and mobile networks. Growth is also expected in the healthcare industry as the use of information technology increases.

Skills of network administrator:

Network admins are responsible for the back end networks, software and hardware as well as for the teams and end users they support. That means a network admin should have a combination of problem solving and people skills.
**Analysing and critical thinking**

Network admins need to explore and solve problems logically and consistently. “The ability to take the concepts you’ve learned in school and understand how they work and affect other concepts is the bread and butter of being a network administrator,” says Brad Meyer, systems administrator at Technology Advice. Even if you don’t yet know the solution, he believes thinking critically will help you get there.

**Time management**

Network admins juggle several projects, people and problems simultaneously. This means it’s essential to be organized in the present and looking ahead to prepare for what’s coming next. It’s like spinning plates with a little practice a network admin can keep everything balanced.

**Interpersonal skills**

Network admins work with a range of people, from network engineers to help desk employees to end users, explains Eric Jeffery, founder of IT solutions firm Gungon Consulting. He says bridging the gap between diverse groups of people requires patience and understanding.

**Lifelong learning**

The technology field is constantly changing, which means network admins must be willing to learn and evolve with it. Good network admins are able to adapt to new techniques and technologies throughout their careers.

**Responsibilities of the Network Administrator**

As a network administrator, the tasks generally fall into the following areas:

- Designing and planning the network
- Setting up the network
- Maintaining the network
- Expanding the network

Each task area corresponds to a phase in the continuing life cycle of a network. You might be responsible for all the phases, or you might ultimately specialize in a particular area, for example, network maintenance.

**Designing the Network:**

The first phase in the life cycle of a network involves creating its design, a task not usually performed by new network administrators. Designing a network involves making decisions about the type of network that best suits the needs of your organization. In larger sites this task is performed by a senior network architect: an experienced network administrator familiar with both network software and hardware.
**Setting Up the Network:**

After the new network is designed, the second phase of network administration begins, which involves setting up and configuring the network. This consists of installing the hardware that makes up the physical part of the network, and configuring the files or databases, hosts, routers, and network configuration servers. The tasks involved in this phase are a major responsibility for network administrators. You should expect to perform these tasks unless your organization is very large, with an adequate network structure already in place.

**Maintaining the Network:**

The third phase of network administration consists of on-going tasks that typically constitute the bulk of your responsibilities. They might include:

- Adding new host machines to the network
- Administering network security
- Administering network services, such as NFS services, name services, and electronic mail
- Troubleshooting network problems

"Configuring Network Clients" explains how to set up new hosts on an existing network. "General Troubleshooting Tips" contains hints for solving network problems.

**Expanding the Network:**

The longer a network is in place and functioning properly, the more your organization might want to expand its features and services. Initially, you can increase network population by adding new hosts and expanding network services by providing additional shared software. But eventually, a single network will expand to the point where it can no longer operate efficiently. That is when it must enter the fourth phase of the network administration cycle: expansion.

Several options are available for expanding your network:

1. Setting up a new network and connecting it to the existing network using a machine functioning as a router, thus creating an internetwork.
2. Configuring machines in users' homes or in remote office sites and enabling these machines to connect over telephone lines to your network.
3. Connecting your network to the Internet, thus enabling users on your network to retrieve information from other systems throughout the world.
4. Configuring UUCP communications, enabling users to exchange files and electronic mail with remote machines.
Objective:
To implement socket programming date and time display from client to server using TCP and UDP Sockets.

Learning Outcomes:
After the completion of this experiment, student will be able to

- Write, execute and debug programs which use TCP Socket API.
- understand the use of client/server architecture in application development
- Develop a client-server application by using TCP and UDP
- Understand how the connection is established using a socket between a client and a server and also understands time and date retrieval from the server.

Problem Statement:

- A server program to establish the socket connection with the client.
- A client program which on establishing a connection retrieves the time and date of the system and displays it.

Concept:

Socket Programming

- Sockets provide the communication mechanism between two computers. A client program creates a socket on its end of the communication and attempts to connect that socket to a server.
- When the connection is made, the server creates a socket object on its end of the communication. The client and server can now communicate by writing to and reading from the socket.

Client–server model

- The client–server model of computing is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.
- Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system.
- A server host runs one or more server programs which share their resources with clients. A client does not share any of its resources, but requests a server's content or service function.
- Clients therefore initiate communication sessions with servers which await incoming requests. Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.
Algorithm:

Server:
Step 1: Create a server socket and bind it to port.
Step 2: Listen for new connection and when a connection arrives, accept it.
Step 3: Send server’s date and time to the client.
Step 4: Read client’s IP address sent by the client.
Step 5: Display the client details.
Step 6: Repeat steps 2-5 until the server is terminated.
Step 7: Close the server socket.

Client
Step 1: Create a client socket and connect it to the server’s port number.
Step 2: Retrieve its own IP address using built-in function.
Step 3: Send its address to the server.
Step 4: Display the date & time sent by the server.
Step 5: Close the client socket

System and Software tools required:
- Package Required : Java Compiler
- Operating System : UBUNTU
- Minimum Requirement : Pentium III or Pentium IV with 2GB RAM 40 GB hard disk

TCP Program:
dateserver.java

//import java packages
import java.net.*; import java.io.*; import java.util.*;

/*… Register service on port 8020…*/
ss=new ServerSocket(8020);

/*… Wait and accept a connection…*/
s=ss.accept();

/*… Get a communication stream associated with the socket…*/
ps=new PrintStream(s.getOutputStream());

/* …To get system time…*/
Date d=new Date();
ps.println(d);

/* …To get system time…*/
dis=new DataInputStream(s.getInputStream());
inet=dis.readLine();System.out.println("THE CLIENT SYSTEM ADDRESS IS :"+inet);
/* …This method is used to request for closing or terminating an object…*/
ps.close(); }}

dateclient.java

/* …Socket class is having a constructor through this Client program can request to server to get connection…*/
Socket soc;
DataInputStream dis;
String sdate;
PrintStreams;

/*…getLocalHost() method: Returns the name of the local computer…*/
InetAddressia=InetAddress.getLocalHost();

/*… Open your connection to a server, at port 8020…*/
soc=new Socket(ia,8020);

/*… Get an input file handle from the socket and read the input…*/

/*…getInputStream() - This method take the permission to write the data from client program to server program and server program to client program…*/
dis=new DataInputStream(soc.getInputStream());
sdate=dis.readLine();
System.out.println("THE date in the server is:"+sdate);

/* …getOutputStream() - This method is used to take the permission to read data from client system by the server or from the server system by the client…*/
ps=new PrintStream(soc.getOutputStream());
ps.println(ia);}

Output:
UDP Program:

**Server.java**
/*…import java packages...*/
import java.net.*; import java.io.*; import java.util.*;
/*..receiving the packet from client...*/
DatagramPacket rp=new DatagramPacket(rd,rd.length); ss.receive(rp);
InetAddress ip= rp.getAddress(); int port=rp.getPort();
/*… getting system time...*/
Date d=new Date();
/*… converting it to String...*/
String time= d + "";
/*… converting that String to byte...*/
sd=time.getBytes();
/*…sending the data to the client...*/
DatagramPacket sp=new DatagramPacket(sd,sd.length,ip,port);
ss.send(sp);

**Clientnew.java**
/*…send the data to the server(data,length,ip address and port number)...*/
DatagramPacket sp=new DatagramPacket(sd,sd.length,ip,1234);
DatagramPacket rp=new DatagramPacket(rd,rd.length);
/*…To Send the data...*/
cs.send(sp); cs.receive(rp);
String time=new String(rp.getData()); System.out.println(time);
/*…This method is used to request for closing or terminating an object...*/
cs.close(); } }
Test cases:
1. Check for the successful connection establishment
2. Retrieve the time and date of the system
3. Change the port number in both server and client programs. Recompile
4. Run client and server on separate computers and try to retrieve the date and time using TCP and UDP.

Viva Questions:
1. Define Socket.
2. What is Socket address?
3. What is port number?
4. Give the differences between connection oriented and connectionless service
5. What is UDP?
6. List the features of UDP.
7. Give the differences between TCP and UDP
8. Why UDP is called unreliable protocol?
9. How UDP socket is created?
10. Discuss the disadvantages of UDP.
11. List the steps involved in client server communication using UDP socket
12. What is datagram socket?

Result:
Thus the program for implementing to display date and time from client to server using TCP Sockets was executed successfully and output verified using various samples.
EX. NO.3a IMPLEMENTATION OF SLIDING WINDOW PROTOCOL

Objective:

To write and simulate the sliding window protocol using ns2 toolkit.

Learning Outcomes:

After the completion of this experiment, student will be able to

- Interpret the concepts of windowing techniques.
- Understand the sliding window implementation in TCP.
- Understand how the reliable in order delivery is obtained using this protocol
- Understand how the window size is fixed and the packets are transmitted in a network.

Problem Statement:

- A program to simulate the sliding window concept and to evaluate the reliability of TCP.

Algorithm:

Step1: Create the nodes in the network
Step2: Give the link between the nodes created.
Step3: Create a TCP agent.
Step4: Give the corresponding id for the agents which have been created.
Step5: Attach the created agents to the respective nodes
Step6: Specify the application program which is running on the node
Step7: Set the window size for the TCP nodes
Step8: Give the start and termination time for the application o the run in the network.
Step9: Repeat the above steps for the number of TCP agents necessary for your topology.
Step10: Execute the file using NAM (network animation monitor) and view the output.

System and Software tools required:

Package Required : Java Compiler
Operating System : UBUNTU
Minimum Requirement : Pentium III or Pentium IV with 2GB RAM 40 GB hard disk

Execution of the Program:

$ nsfilename.tcl //running the program
**Sample Code:**

**Nodes creation:**
- `set ns [new Simulator]`
- `set tracefile1 [open out.tr w]`
- `$ns trace-all $tracefile1`
- `setnamfile [open out.nam w]`
- `$ns namtrace-all $namfile`
- `set n0 [$ns node]`
- `set n1 [$ns node]`

**Colouring the node:**
- `$n0 color red`
- `$n1 color red`

**Giving the link between nodes**
- `$ns duplex-link $n0 $n3 0.5Mb 250ms DropTail`
- `$ns duplex-link $n1 $n3 0.5Mb 250ms DropTail`

**Creating TCP agents and application**
- `set tcp1 [new Agent/TCP]`
- `$tcp1 set fid_ 1`
- `$ns attach-agent $n0 $tcp1`
- `set sink1 [new Agent/TCPSink]`
- `$ns attach-agent $n7 $sink1`
- `set ftp [new Application/FTP]`
- `$ftp attach-agent $tcp1`
- `$ns connect $tcp1 $sink1`
- `$tcp1 set window_ 5`
- `$ns at 0.3 "$ftp start"`
- `$ns at 5.0 "$ftp stop"`

**Executing the code**
- `proc finish {} {
  global ns tracefile1 namfile
  $ns flush-trace
  close $tracefile1
  close $namfile
  execnam -a out.nam&
  exit 0
} $ns at 8.2 "finish"$ns run`
Sample Output:

Test cases:
1. Create 7 nodes
2. Give correct orientation of nodes
3. Make a node as a source node and another as a destination node
4. Create 4 TCP agents
5. Make the window size as 5.
6. Check how the packets travel when the window size is fixed.

Analysis of result:
From the experiment, the student will be able to
1. Analyse the purpose of windowing techniques.
2. Realize how sliding window helps in reliability.
3. Understand the sliding window implementation in TCP.
4. Analyse the working of sliding window protocol using simulation.

Viva Questions
1. What is sliding window?
2. What is send sliding window?
3. What is receive sliding window?
4. What is timeout?
5. What is sequence number?
6. How sliding window protocol addresses a lost packet?
7. How a frame is resent in sliding window protocol?
8. What is the criterion for the window to slide the slots?
9. What are the different types of protocols used in windowing techniques?
10. Define RTT

Result:
Thus the Simulation for sliding window protocol was written and implemented successfully.
Objective:
To write a java program to perform Stop and Wait protocol.

Learning outcome:
After the completion of this experiment, student will be able to
- Interpret the concepts of reliable transmission.
- Understand how the reliable in order delivery is obtained using this protocol.
- Understand how to send information between two connected devices.

Problem Statement:
- A program to evaluate the reliability of stop and wait protocol.

Concept: Stop and wait:
"stop-n-wait" (sometimes known as "positive acknowledgement with retransmission") is the fundamental technique to provide reliable transfer under unreliable packet delivery system.

Client: Sends packet (frame), waits for an ACK from the server in a given time (timeout), if ACK not received, the packet is sent again, all packets have a number in case one is lost to know which one.

Server: waits for packets, sends ACK when packet received within the timeout, ACK sent must have the same number as packet received.

Algorithm:
Step 1: Start the program.
Step 2: Create the socket by specifying the address and establishes the connection
Step 3: Send and receive information.
Step 4: The sender sends one frame, stops until it receives confirmation from the receiver and then sends the next frame.
Step 5: If your frames reach the server it will send ACK signal to client otherwise it will send NACK signal to client.
Step 6: Stop the program

System and Software tools required:
- Package Required: Java Compiler
- Operating System: UBUNTU
- Minimum Requirement: Pentium III or Pentium IV with 2GB RAM 40 GB hard disk
Sample Code:

Sender.java

//import java packages
import java.io.*;
import java.net.*;

/*...System.in is an InputStream, we create an InputStreamReader which reads bytes from System.in...*/
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Waiting for Connection....");

/*... Register service on port 2004...*/
sender = new Socket("localhost",2004);
sequence=0;

/*... Get a communication stream associated with the socket...*/
out=new ObjectOutputStream(sender.getOutputStream());
out.flush();
in=new ObjectInputStream(sender.getInputStream());
str=(String)in.readObject();
System.out.println("reciver> "+str);
System.out.println("Enter the data to send....");

/*...readline() method read a line of text...*/
packet=br.readLine();
n=packet.length();
do{try{ if(i<n){
    msg=String.valueOf(sequence);
    msg=msg.concat(packet.substring(i,i+1));}
  else if(i==n){ msg="end";

  }else{
    out.writeObject(msg):break;
  }
}

/*...method writeObject is used to write an object to the stream...*/
out.writeObject(msg);
sequnce=(sequence==0)?1:0;

/*... method flushes this output stream and forces any buffered output bytes to be written out...*/
out.flush();
Reciever.java
ServerSocket reciever;
Socket connection = null;
ObjectOutputStream out;
ObjectInputStream in;
public void run() {

/* … It reads bytes and decodes them into characters using a specified charset … */
try { BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

/* … Open your connection to a server, at port 2004 … */
reciever = new ServerSocket(2004, 10);
System.out.println("waiting for connection...");
/* … Wait and accept a connection … */
connection = reciever.accept();
sequence = 0;
System.out.println("Connection established :");

/* … getOutputStream() - This method is used to take the permission to read data from client system by the server or from the server system by the client … */
out = new ObjectOutputStream(connection.getOutputStream());
out.flush();

/* … getInputStream() - This method take the permission to write the data from client program to server program and server program to client program … */
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); sequence = (sequence == 0) ? 1 : 0;
System.out.println("\n\nreceiver">"+packet); }
else { System.out.println("\n\nreceiver">"+packet + " duplicate data"); }
if (i < 3) {
/* … The method writeObject is used to write an object to the stream … */
out.writeObject(String.valueOf(sequence)); i++; }
else out.writeObject(String.valueOf((sequence + 1) % 2));

}
Sample Output:
Test cases:

1. Check for the successful connection establishment.
2. Send one data frame to the receiver.
3. Get ACK from the receiver.
4. Check when the data loss can occur.

Viva Questions

1. What happens when a sender doesn’t receive a packet after a certain amount of time?
2. Which packet number is assigned to a new packet if “packet 0” didn’t receive an ACK?
3. Can you identify issue(s) with the Stop-and-Wait Protocol?
4. What is time out?
5. Difference between sliding window and stop and wait?
6. In which layer stop and wait protocol is used?
7. What are the rules for sender in stop and wait?
8. What are the rules for receiver in stop and wait?

Result:

Thus the program for implementing stop and wait protocol was executed successfully.
**EX. NO.4 WRITE A CODE SIMULATING PING AND TRACEROUTE COMMANDS**

**Objective:**
To write the java program for simulating ping and trace route commands.

**Learning outcome:**
After the completion of this experiment, student will be able to
- Understand that Ping is a networking utility program or a tool to test if a particular host is reachable.
- Understand the use of ping command and discover the routes that packets actually take when traveling to their destination by using tracert command.

**Problem Statement:**
To write the java program for simulating ping and trace route commands.

**Concept: PING Command**

The ping command is a very common method for troubleshooting the accessibility of devices.

It uses a series of Internet Control Message Protocol (ICMP) Echo messages to determine:
- Whether a remote host is active or inactive.
- The round-trip delay in communicating with the host.
- Packet loss.

The ping command first sends an echo request packet to an address, and then waits for a reply.

The ping is successful only if:
- the echo request gets to the destination, and
- The destination is able to get an echo reply back to the source within a predetermined time called a timeout. The default value of this timeout is two seconds on Cisco routers.

**TRACEROUTE Command**

1. The trace route command is used to discover the routes that packets actually take when traveling to their destination. The device (for example, a router or a PC) sends out a sequence of User Datagram Protocol (UDP) data grams to an invalid port address at the remote host.

2. Three data grams are sent, each with a Time-To-Live (TTL) field value set to one. The TTL value of 1 causes the datagram to "timeout" as soon as it hits the first router in the path; this router then responds with an ICMP Time Exceeded Message (TEM) indicating that the datagram has expired.

3. Another three UDP messages are now sent, each with the TTL value set to 2, which causes the second router to return ICMP TEMs. This process continues until the packets actually reach the other destination.
4. Since these data grams are trying to access an invalid port at the destination host, ICMP Port
Unreachable Messages are returned, indicating an unreachable port; this event signals the Trace
route program that it is finished.

**Algorithm:**

Server
**Step1:** Start the program.
**Step2:** Import necessary packages.
**Step3:** Initialize the ping server with both sockets as null value.
**Step4:** Start the server socket.
**Step5:** At the client give the IP address of the server(by using **ifconfig** command in command prompt).
**Step6:** The client program is then started by starting socket.
**Step7:** At the receiver end, the client is pinged and traced.
**Step8:** Stop the program.

Client
**Step1:** Start the program.
**Step2:** Import necessary packages.
**Step3:** Initialize the ping client with both sockets as null value.
**Step4:** Start the socket.
**Step5:** Get the IP address of the server.
**Step6:** Ping the server.
**Step7:** At the receiver end, the server is pinged and traced.
**Step8:** Stop the program.

**System and Software tools required:**

<table>
<thead>
<tr>
<th>Package Required</th>
<th>Java Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>UBUNTU</td>
</tr>
<tr>
<td>Minimum Requirement</td>
<td>Pentium III or Pentium IV with 2GB RAM 40 GB hard disk</td>
</tr>
</tbody>
</table>

**Sample code:**

**pingclient.java**

/*…localhostport name and 5555-port number…*/
Socket s=new Socket("127.0.0.1",5555);

/*… Get an input file handle from the socket and read the input…*/
DataInputStream dis=new DataInputStream(s.getInputStream());
PrintStream out=new PrintStream(s.getOutputStream()); while(c<4){

returns the current time in milliseconds...*/
t1=System.currentTimeMillis();
str="Welcome to network programming world":
out.println(str);

/*...readline() method read a line of text...*/
System.out.println(dis.readLine());
t2=System.currentTimeMillis();
System.out.println(";TTL="+(t2-t1)+"ms"); c++;

pingserver.java

/*...ServerSocket object is used to establish the communication with clients...*/
ServerSocket ss=new ServerSocket(5555);

/*...accept(): Used to accept the client request...*/
Socket s=ss.accept();
int c=0; while(c<4) {

/*... Get an input file handle from the socket and read the input...*/
DataInputStream dis=new DataInputStream(s.getInputStream());
PrintStream out=new PrintStream(s.getOutputStream());

/*...readline() method read a line of text...*/
String str=dis.readLine();
out.println("Reply from"+ InetAddress.getLocalHost()+";Length"+str.length()); c++;

Sample Output:
Testcase:
1. Verify that a computer can communicate over network with another computer by using ping cmd.
2. Trace the route from source to destination.
3. Check the ip level connectivity by specifying the domain name.

Viva questions:
1. Does ping can measure round trip time?
2. What is ping sweep to trace the path from source to destination?
4. Difference between traceroute and ping.
5. What utility is used to find the number of routers between a source and destination?
6. Which protocol does ping use?
7. What is the ‘ping’ command useful for?
8. How do you trace route an ip address?

Result:
Thus the program for implementing ping and trace route command was executed successfully.
Objective:
To write a java program for file transfer using TCP Sockets.

Learning outcome:
After the completion of this experiment, student will be able to
- Develop a file transfer application by using TCP Socket
- Understand the reliability of TCP protocol.
- Understand how to transfer of computer files between a client and server on a computer network.
- Develop a client-server application by using TCP

Problem Statement:
To transfer computer files between a client and server on a computer network using TCP Sockets.

Concept:
Our aim is to send a file from the client machine to the server machine using TCP Communication.

Algorithm
Server
Step1: Import java packages and create class file server.
Step2: Create a new server socket and bind it to the port.
Step3: Accept the client connection
Step4: Get the file name and stored into the BufferedReader.
Step5: Create a new object class file and realine.
Step6: If file is exists then FileReader read the content until EOF is reached.
Step7: Stop the program.

Client
Step1: Import java packages and create class file server.
Step2: Create a new server socket and bind it to the port.
Step3: Now connection is established.
Step4: The object of a BufferedReader class is used for storing data content which has been retrieved from socket object.
Step5: The content of file is displayed in the client window and the connection is closed.
Step6: Stop the program.
**System and Software tools required:**

- Package Required: Java Compiler
- Operating System: UBUNTU
- Minimum Requirement: Pentium III or Pentium IV with 2GB RAM 40 GB hard disk

**Sample Code:**

**Se.java**

```java
/** … Register service on port 15123… */
ServerSocket serverSocket = new ServerSocket(15123);

/** … Wait and accept a connection… */
Socket socket = serverSocket.accept();
System.out.println("Accepted connection : "+socket);

/** … enter the source file name which is to be transferred to client… */
File transferFile = new File("sunithanandhinifiletransfer.doc");
byte[] bytearray = new byte[(int)transferFile.length()];

/** … FileInputStream is meant for reading streams of raw bytes such as image data… */
FileInputStream fin = new FileInputStream(transferFile);
BufferedInputStream bin = new BufferedInputStream(fin);
bin.read(bytearray,0,bytearray.length);

/** … Get a communication stream associated with the socket… */
OutputStream os = socket.getOutputStream();
System.out.println("Sending Files...");
os.write(bytearray,0,bytearray.length);
os.flush();socket.close();

**Cl.java**

/** … Open your connection to a server, at port 15123… */
Socket socket = new Socket("127.0.0.1",15123);
byte[] bytearray = new byte[filesize];
InputStream is = socket.getInputStream();

/** … enter the destination file name which is to be transferred … */
FileOutputStream fos = new FileOutputStream("kalpanasonikafilafiletransfer.doc");
BufferedOutputStream bos = new BufferedOutputStream(fos);
bytesRead = is.read(bytearray,0,bytearray.length);
currentTot = bytesRead;
```
do { bytesRead = is.read(bytearray, currentTot, (bytearray.length-currentTot));
if(bytesRead>= 0) currentTot += bytesRead;}
while(bytesRead> -1);
bos.write(bytearray, 0 , currentTot);
bos.flush(); bos.close(); socket.close();   } 

Sample Output:
Testcase:

1. Send and receive files from or to server/client
2. Change the port number in both server and client programs. Recompile
3. Run Client and Server on separate computers and try to transfer the files.

Viva questions:

1. What are the types of protocol?
2. Define socket.
3. What information is needed to create a TCP Socket?
4. What are the two important TCP Socket classes?
5. What is the difference between the File and Random Access File classes?
6. What are some advantages and disadvantages of Java Sockets?
7. How to build FileInputStream object with byte array as a parameter
8. How to read file in byte array with FileInputStream

RESULT

Thus the java program file transfer application using TCP Sockets was executed.
Objective:
To create a program for executing the remote command using TCP Socket.

Learning outcomes:
After the completion of this experiment, student will be able to
- Develop a program to execute remote the application by using TCP Socket.
- Understand the reliability of TCP protocol.
- Understand how to execute remote commands in computer files between a client and server on a computer network.

Problem Statement:
To create a program for executing a command in the system from a remote point.

Algorithm:
Step1: Start the program.
Step2: Create a server socket at the server side.
Step3: Create a socket at the client side and the connection is set to accept by the server socket using the accept () method.
Step4: In the client side the remote command to be executed is given as input.
Step5: The command is obtained using the readLine () method of Buffer Reader.
Step6: Get the runtime object of the runtime class using getRuntime ().
Step7: Execute the command using the exec () method of runtime.

System and Software tools required:
- Package Required : Java Compiler
- Operating System : UBUNTU
- Minimum Requirement : Pentium III or Pentium IV with 2GB RAM 40 GB hard disk

Sample Code:
RemoteServer.java

/*...getInputStream()-This method take the permission to write the data from client program to server program and server program to client program...*/
BufferedReader Buf = new BufferedReader(new InputStreamReader(System.in));
System.out.print(“ Enter the Port Address : ” );
Port = Integer.parseInt(Buf.readLine());
Socket class is having a constructor through this Client program can request to server to get connection…*/
ServerSocket ss=new ServerSocket(Port);
System.out.println(" Server is Ready To Receive a Command. ");
System.out.println(" Waiting ..... ");

/*… Wait and accept a connection…*/
Socket s=ss.accept();
if(s.isConnected()==true)
System.out.println(" Client Socket is Connected Succefully. ");
InputStream in=s.getInputStream();

/*…getOutputStream()-This method is used to take the permission to read data from client
text system by the server or from the server system by the client…*/
OutputStream ou=s.getOutputStream();
BufferedReader buf=new BufferedReader(new InputStreamReader(in));
String cmd=buf.readLine();
PrintWriter pr=new PrintWriter(ou);
pr.println(cmd);

/*... Runtime.getRuntime() method returns the runtime object associated with the current Java application ...*/
Runtime H=Runtime.getRuntime();
Process P=H.exec(cmd);

RemoteClient.java
BufferedReader Buf=new BufferedReader(new InputStreamReader(System.in));
System.out.print(" Enter the Port Address : ");

/*...Integer.parseInt() java method is used primarily in parsing a String method argument into
an Integer object. The Integer object is a wrapper class for the int primitive data type ...*/
Port=Integer.parseInt(Buf.readLine());
Socket s=new Socket("localhost",Port);
if(s.isConnected()==true)
System.out.println(" Server Socket is Connected Succefully. ");
InputStream in=s.getInputStream();
OutputStream ou=s.getOutputStream();
BufferedReader buf=new BufferedReader(new InputStreamReader(System.in));
BufferedReader buf1=new BufferedReader(new InputStreamReader(in));
PrintWriter pr=new PrintWriter(ou);
System.out.print(" Enter the Command to be Executed : ");

/*... readline() method read a line of text ...*/

pr.println(buf.readLine());
pr.flush();
String str=buf1.readLine();

Sample Output:

Testcase:

1. Check some other commands to execute remotely.
2. Try to run some other commands remotely.
**Viva questions:**

1. Define remote object.
2. Give the features of TCP.
3. What is the use of accept function?
4. Which function is used to establish connection from client to server?
5. Which line of code gets the remote command for execution process?
6. Mention some other commands to be executed remotely.

**Result:**

Thus the java program to execute remote command using TCP Sockets was done successfully.
Objective:

To implement a CHAT application, where the Client establishes a connection with the Server. The Client and Server can send as well as receive messages at the same time. Both the Client and Server exchange messages.

Learning Outcomes:

After the completion of this experiment, student will be able to

- Develop a chat application by using TCP Socket
- Understand the reliability of TCP protocol.
- Understand how the reliable in order delivery is obtained using this protocol
- Develop a client-server application by using TCP

Problem Statement:

- A server program to establish the socket connection with the client for performing chat.
- A client program which on establishing a connection with the server for performing chat.

Concept:

1. It uses TCP socket communication. We have a server as well as a client.
2. Both can be run in the same machine or different machines. If both are running in the machine, the address to be given at the client side is local host address.
3. If both are running in different machines, then in the client side we need to specify the ip address of machine in which server application is running.

Algorithm:

Server

Step1: Start the program and create server and client sockets.
Step2: Use input streams to get the message from user.
Step3: Use output streams to send message to the client.
Step4: Wait for client to display this message and write a new one to be displayed by the server.
Step5: Display message given at client using input streams read from socket.
Step6: Stop the program.

Client

Step1: Start the program and create a client socket that connects to the required host and port.
Step2: Use input streams read message given by server and print it.
Step3: Use input streams; get the message from user to be given to the server.
Step4: Use output streams to write message to the server.
Step5: Stop the program.
**System and Software tools required:**

<table>
<thead>
<tr>
<th>Package Required</th>
<th>Java Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>UBUNTU</td>
</tr>
<tr>
<td>Minimum Requirement</td>
<td>Pentium III or Pentium IV with 2GB RAM 40 GB hard disk</td>
</tr>
</tbody>
</table>

**Sample Program:**

**GossipServer.java**

ServerSocketersock = new ServerSocket(3000);
System.out.println("Server ready for chatting");
Socket sock = sersock.accept();

/*…reading from keyboard (keyRead object)…*/
BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));

/*…sending to client (pwrite object)…*/
OutputStreamostream = sock.getOutputStream();
PrintWriterpwrite = new PrintWriter(ostream, true);

/*… receiving from server (receiveRead object)…*/
InputStreamistream = sock.getInputStream();
BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));
String receiveMessage, sendMessage;
while(true)
{if((receiveMessage = receiveRead.readLine()) != null)
{System.out.println(receiveMessage); }

**GossipClient.java**

Socket sock = new Socket("127.0.0.1", 3000);

/*…reading from keyboard (keyRead object)…*/
BufferedReader keyRead = new BufferedReader(new InputStreamReader(System.in));

/*…sending to client (pwrite object)…*/
OutputStreamostream = sock.getOutputStream();
PrintWriterpwrite = new PrintWriter(ostream, true);

/*… receiving from server (receiveRead object)…*/
InputStreamistream = sock.getInputStream();
BufferedReader receiveRead = new BufferedReader(new InputStreamReader(istream));

System.out.println("Start the chitchat, type and press Enter key");
String receiveMessage, sendMessage;
while(true)
{
    sendMessage = keyRead.readLine(); /*... keyboard reading ...*/
pwrite.println(sendMessage); /*... sending to server...*/
pwrite.flush(); /*... flush the data...*/
    if((receiveMessage = receiveRead.readLine()) != null) /*...receive from server...*/
    {System.out.println(receiveMessage);} /*... displaying at DOS prompt...*/
}

Sample Output:

Test cases:
1. Check for the successful connection establishment
2. Send and receive messages from or to server/client
3. Change the port number in both server and client programs. Recompile
4. Run TCPClient and TCPServer on separate computers and try to chat.
Viva Questions:

1. How the reliability is achieved using TCP?
2. What is sequence number?
3. What is inorder delivery?
4. What is congestion?
5. Define fast retransmit
6. Define TCP 3way handshaking
7. Define ACK number
8. Difference between TCP and UDP
9. Justify why TCP is called as connection oriented protocol.
10. What are the two versions of IP Addressing?
11. List the values that AF_INET can take.
12. What are the steps involved in establishing a connection with the server using TCP Socket

Result:

Thus the java program to chat using TCP Sockets was executed successfully.
Objective:
To write a program for concurrent communication of client to the server using TCP Socket.

Learning Outcomes:
After the completion of this experiment, student will be able to
- Understand the reliability of TCP protocol.
- Understand how a server can handle multiple clients at the same time in parallel
- Develop a client-server application by using TCP

Problem Statement:
To have a reliable concurrent communication between client and the server.

Concept:
Concurrent Server: The server can be iterative, 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.

Algorithm:
Server
Step 1: Create a socket and bind to the address. Leave socket unconnected.
Step 2: Leave socket in passive mode, making it ready for use by a server.
Step 3: Repeatedly call accept to receive the next request from a client to handle the response with the through socket.

Client
Step 1: Begin with a connection passed from the server (i.e., a socket for the connection).
Step 2: Use input streams; get the message from user to be given to the server.
Step 3: Use input streams read message given by server and print it.
Step 4: Use output streams to write message to the server.
Step 5: Close the connection and exit, i.e., slave terminates after handling all requests from one client.

System and Software tools required:
- Package Required: Java Compiler
- Operating System: UBUNTU
- Minimum Requirement: Pentium III or Pentium IV with 2GB RAM 40 GB hard disk
Sample Program:

ConServer.java

/*… Register service on port 8020...*/
ServerSocket ss=new ServerSocket(8500);
System.out.println("Waiting for client...");
while(true)

/*... ServerSocket in order to listen for and accept connections from clients...*/
{Socket s=ss.accept();

/*...getInputStream()-This method take the permission to write the data from client program to
server program and server program to client program...*/
BufferedReader br=new BufferedReader(new InputStreamReader(s.getInputStream()));
cli_name=br.readLine();
System.out.println("\nCLIENT NAME: "+cli_name);
no=Integer.parseInt(br.readLine());
sql=no*no;
PrintWriter pw=new PrintWriter(s.getOutputStream(),true);
pw.println(sql);
System.out.println("OUTPUT - The square of "+no+" is "+sql);
}

ConClient1.java

{Socket s=new Socket("localhost",8500);
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

/*...Integer.parseInt() java method is used primarily in parsing a String method argument into
an Integer object. The Integer object is a wrapper class for the int primitive data type...*/
int num=Integer.parseInt(br.readLine());

/*...getOutputStream()-This method is used to take the permission to read data from client
system by the server or from the server system by the client...*/
PrintWriter pw=new PrintWriter(s.getOutputStream(),true);
pw.println("Client 1");
pw.println(num);
BufferedReader br1=new BufferedReader(new InputStreamReader(s.getInputStream()));
intsqu=Integer.parseInt(br1.readLine());
System.out.println("Square of "+num+" is "+squ+
");
ConClient2.java

Socket s=new Socket("localhost",8500);
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("CLIENT 2:\nEnter the number to find square: ");
int num=Integer.parseInt(br.readLine());
PrintWriter pw=new PrintWriter(s.getOutputStream(),true);
pw.println("Client 2");
pw.println(num);
BufferedReader br1=new BufferedReader(new InputStreamReader(s.getInputStream()));
int squ=Integer.parseInt(br1.readLine());
System.out.println("Square of "+num+" is "+squ+"\n");
s.close();

Sample Output:
Testcase:

1. Run two or more client in parallel.
2. Return the result to multiple clients from server.
3. Perform different operations and try to execute the program by using threads.

Viva questions:

1. Define server and what are the types of server?
2. What are the three types of socket function?
3. What are concurrent servers?
4. Define Iterative server
5. Compare Iterative server and concurrent server
6. Explain socket address structure
7. List some character stream support classes
8. What do you mean by socket programming?

Result:

Thus the java program to concurrently communicate using TCP Sockets was executed successfully.
Objective:
To write a java program for creating socket for HTTP web page upload and download.

Learning Outcomes:
After the completion of this experiment, student will be able to
- Implement a program that can be used to understand the implementation of HTTP.
- Send HTTP request to a server and obtain the HTTP response message.
- Display the contents of the resolved file using HTTP server.

Problem Statement:
- A client program to get the file name from the user.
- A HTTP server program that resolves the given file and displays the contents of the file.

Concept: HTTP
1. The Hypertext Transfer Protocol (HTTP) is a protocol used mainly to access data on the WWW.
2. HTTP functions as a combination of FTP and SMTP.
3. SMTP messages are stored and forwarded, but HTTP messages are delivered immediately.
4. The commands from the client to the server are embedded in a request message. The contents of the requested file or other information are embedded in a response message.

Algorithm:
Step1: Set a server port as 80.
Step2: Using HTTP services create a Socket for server by specifying the server port
Step3: Use HTTP socket for connecting the client to the URL.
Step4: Use BufferedReader to output stream to place the response from the server by the client.
Step5: Close the Connection as soon the request is been serviced. Use Malformed URL exception

If any errors in grabbing the server

System and Software tools required:
- Package Required : Java Compiler
- Operating System : UBUNTU
- Minimum Requirement : Pentium III or Pentium IV with 2GB RAM 40 GB hard disk
Sample Program:

Client.java

/* …create file object…*/
import java.io.File;
import java.io.IOException;

/*…used to perform read and write operation…*/
importjavax.imageio.ImageIO;

public class Client{
public static void main(String args[]) throws Exception{
Socket soc;
BufferedImage img = null;
soc=new Socket("localhost",4000);
System.out.println("Client is running.");
try {
System.out.println("Reading image from disk.");

/*…read image file…*/
img = ImageIO.read(new File("kalpanasonika.jpg"));
ByteArrayOutputStream baos = new ByteArrayOutputStream();

/*…write image file…*/
ImageIO.write(img, "jpg", baos);
baos.flush();

/*…we use toByteArray() method of ByteArrayOutputStream class…*/
byte[] bytes = baos.toByteArray();
baos.close();
System.out.println("Sending image to server.");
OutputStream out = soc.getOutputStream();
DataOutputStream dos = new DataOutputStream(out);
dos.writeInt(bytes.length);
dos.write(bytes, 0, bytes.length);
System.out.println("Image sent to server.");
Server.java

//...Create Server Socket...//
ServerSocket server=null;
Socket socket;

//...Register Service port to 4000...//
server=new ServerSocket(4000);
System.out.println("Server Waiting for image");
socket=server.accept(); System.out.println("Client connected.");
InputStream in =socket.getInputStream();
DataInputStream dis = new DataInputStream(in);
intlen = dis.readInt();
System.out.println("Image Size: " + len/1024 + "KB");
byte[] data = new byte[len];
dis.readFully(data);

//...method is used to request for closing or terminating an object...//
dis.close();
in.close();
InputStreamian = new ByteArrayInputStream(data);
BufferedImage bImage = ImageIO.read(ian);

//...create a frame window entitled “server”...//
JFrame f = new JFrame("Server");
ImageIcon icon = new ImageIcon(bImage);

Sample Output:
Testcase:

1. Give the file name as kalpanasonika.jpg from the client.
2. Read the image from the client and send it to the server.
3. Try to display the image and its size in the server.

Viva questions:

1. What is HTTP and WWW?
2. To which OSI layer does IP belong?
3. What HTTP response headers do?
4. What is HTTP session state?
5. What is Secure HTTP?
6. What is the current version of HTML?
7. What is HTTP session and what is session id?
8. What is the main usage of session id?
9. Define cookie and list some real time examples where the cookies are used.

Result:

Thus the program for creating sockets for HTTP web page upload and download was implemented.
EX. NO.7 IMPLEMENTATION OF SUBNETTING

Objective:

To find Subnet Mask and Network ID for given IP Address.

Learning Outcomes:

After the completion of this experiment, student will be able to

- Understand how to calculate subnets in binary and decimal.
- Understand how to do subnet mask calculations and break down IP address classes to route traffic within network.

Problem Statement:

Write a program to implement subnetting and find the subnet masks.

Concept:

If an organization was granted a large block in class A or B, it could divide the addresses into several contiguous groups and assign each group to smaller networks (called subnets) or, in rare cases, share part of the addresses with neighbours.

Algorithm:

Step 1: Get the input from the user by using scanner method.

Step 2: Read the input by using nextLine() and store it.

Step 3: Split the string based on string by using split(“\"”)

Step 4: Convert it into binary.

Step 5: Calculating the network mask by using math and logarithmic

Step 6: Get the first address by ANDding the last n bits with 0.

Step 7: Get the last address by ANDding the last n bits with 1.

System and Software tools required:

- Package Required : Java Compiler
- Operating System : UBUNTU
- Minimum Requirement : Pentium III or Pentium IV with 2GB RAM 40 GB hard disk
Sample Coding:

//...Calculation of mask.../
int bits = (int)Math.ceil(Math.log(n)/Math.log(2));
/*eg if address = 120, log 120/log 2 gives log to the base 2 => 6.9068, ceil gives us upper integer */
System.out.println("Number of bits required for address = "+bits);
int mask = 32-bits;
System.out.println("The subnet mask is = "+mask);

//...Calculation of first address and last address.../
int fbip[] = new int[32];
for(int i=0; i<32;i++) fbip[i] = (int)bip.charAt(i)-48;
//convert cahracter 0,1 to integer 0,1
for(int i=31;i>31-bits;i--)/Get first address by ANDing last n bits with 0
fbip[i] &= 0;
String fip[] = {"","","",""};
for(int i=0;i<32;i++)
  fip[i/8] = new String(fip[i/8]+fbip[i]);
System.out.print("First address is = ");
for(int i=0;i<4;i++){
  System.out.print(Integer.parseInt(fip[i],2));
  if(i!=3) System.out.print( "); } 

Sample Output:

Testcase:

- The first address in the block can be found by setting the rightmost 32-n bits to 0s
- The last address in the block can be found by setting the rightmost 32-n bits to 1s.
- Try to find out the number of address in the block.
Viva questions:

1. What are the advantages of subnetting?

2. Your router has the following IP address on Ethernet0: 172.16.2.1/23. Which of the following can be valid host IDs on the LAN interface attached to the router?
   1) 172.16.1.100  2) 172.16.1.198  3) 172.16.2.255  4) 172.16.3.0
   A. 1 only
   B. 2 and 3 only
   C. 3 and 4 only
   D. None of the above

3. A network administrator is connecting hosts A and B directly through their Ethernet interfaces, as shown in the illustration. Ping attempts between the hosts are unsuccessful. What can be done to provide connectivity between the hosts?

4. What is the maximum number of IP addresses that can be assigned to hosts on a local subnet that uses the 255.255.255.224 subnet mask?

5. If an Ethernet port on a router were assigned an IP address of 172.16.112.1/25, what would be the valid subnet address of this host?

6. What are the network address, broadcast address, and the subnet mask for a host with the IP Address below?
   IP Address: 101.39.85.201/23
   Network Address: 101.39.84.0
   Subnet Mask: 255.255.254.0
   Broadcast Address: 101.39.85.255

7. On a VLSM network, which mask should you use on point-to-point WAN links in order to reduce the waste of IP addresses?

8. To test the IP stack on your local host, which IP address would you ping?

Result:

Thus the program for subnetting was implemented.
Objective:
To implement a program that resolves the IP address from a given domain name.

Learning Outcomes:
After the completion of this experiment, student will be able to
- Understand how the domain name is transformed to an IP address by the DNS server.
- Check for the given domain name from the list of domain names using logical address.
- Understand the implementation of Domain Name System.
- Implement a server program to resolve a given domain name from the server.

Problem Statement:
- A client program to give the domain name to be resolved into an IP address.
- A server program that resolves the given domain from the list of domains which the DNS server has stored.

Concept:
1. The DNS client program sends a request to a DNS server to map the e-mail address to the corresponding IP address.
2. When the Internet was small, mapping was done by using a host file. The host file had only two columns: name and address.
3. The host that needs mapping can contact the closest computer holding the needed information. This method is used by the Domain Name System (DNS).

Algorithm:
Server
Step1: Start the program.
Step2: Create the socket for the server.
Step3: Bind the socket to the port.
Step4: Listen for the incoming client connection.
Step5: Receive the IP address from the client to be resolved.
Step6: Get the domain name for the client.
Step7: Check the existence of the domain in the server.
Step8: If domain matches then send the corresponding address to the client.
Step9: Stop the program execution
Client

Step 1: Start the Program.
Step 2: Create the socket for the client.
Step 3: Connect the socket to the Server.
Step 4: Send the host name to the server to be resolved.
Step 5: If the server corresponds then print the address and terminate the process

System and Software tools required:
- Package Required: Java Compiler
- Operating System: UBUNTU
- Minimum Requirement: Pentium III or Pentium IV with 2GB RAM 40 GB hard disk

Sample Program:
Clientdns12.java

/*... datagram socket is the sending or receiving point for a packet delivery service.*/
DatagramSocket client=new DatagramSocket();

/*...InetAddress class provides methods to get the IP of any host name...*/
InetAddress addr=InetAddress.getByName("127.0.0.1");
byte[] sendbyte=new byte[1024];
byte[] receivebyte=new byte[1024];
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Enter the DOMAIN NAME or IP adress:");
String str=in.readLine();
sendbyte=str.getBytes();
/*…send the data to the server(data,length,ip address and port number)...*/
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("IP address or DOMAIN NAME: "+s.trim());

Serverdns12.java

DatagramSocket server=new DatagramSocket(1309);
while(true)
{byte[] sendbyte=new byte[1024];
byte[] receivebyte=new byte[1024];
/*..receiving the packet from client...*/
DatagramPacket receiver=new DatagramPacket(receivebyte,receivebyte.length);
server.receive(receiver);
String str=new String(receiver.getData());
String s=str.trim();
//System.out.println(s);
InetAddress addr=receiver.getAddress();
int port=receiver.getPort();

/*/... specify the IP address to map with its domain name...*/
String ip[]=>Contact: 165.165.80.80,"165.165.79.1";
/*/domain name...*/
String name[]=new String("www.skct.edu","www.sonika.com");
for(int i=0;i<ip.length;i++)
if(s.equals(ip[i]))
{sendbyte=name[i].getBytes();
 DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,port);
 server.send(sender);
 break;}
else if(s.equals(name[i])){
 sendbyte=ip[i].getBytes();
 DatagramPacket sender=new DatagramPacket(sendbyte,sendbyte.length,addr,port);
 server.send(sender);

Sample Output:
Test cases:
1. Give the domain name as google.com
2. Give the domain name as computernetworks.in

Analysis of Result:
From this experiment, student will be able to
1. Deduce the need for the DNS server.
2. Relate the purpose of the DNS in real time when they type URL’s in the browser.
3. Analyze how the DNS server resolves the IP address from the domain name given by the end user.

Viva Questions:
1. What is DNS?
2. What is the port number of DNS?
3. What is the main purpose of DNS server?
4. What is forward lookup?
5. What is reverse lookup?
6. What are the different types of DNS Server?
7. What are the different types of Resource Records in bind?
8. What are the different types of Resource Records in bind?
9. What is the main use of port number in DNS?
10. Mention some real time applications of DNS

Result:
Thus the DNS application program was executed.
**Objective:**

To write a java program for SNMP application.

**Learning Outcomes:**

After the completion of this experiment, student will be able to

- Understand fundamental operations for monitoring and maintaining an internet.
- Develop a simple client which will display the hardware information.
- Understand the implementation of SNMP.

**Problem Statement:**

To develop a simple program which will display the hardware information.

**Concept:**

1. Simple Network Management Protocol (SNMP) is a framework for managing devices in an internet using TCP/IP.
2. It provides a set of fundamental operations for monitoring and maintaining an internet.
3. SNMP uses the concept of manager and agent
   - A manager is a host that runs the SNMP client program.
   - A managed station called an agent, is a router that runs the SNMP server program

**Algorithm:**

**Step 1:** Using start method the system is used to receive SNMP request.

**Step 2:** The assigned Port no 162 are used to send and reciver trap.

**Step 3:** Set the address using the format ("udp:127.0.0.1/161")

**Step 4:** Create a variable binding and add the object identifier in OID format.

**Step 5:** Create the Protocol data unit object

**Step 6:** Listen enable listening for SNMP packet by using listen ()

**Step 8:** Get method asynchronous GET request PDU is send to the given target.

**Step 7:** Send a new get request for single OID and return response event for the request only, if no timeout has occurred.

**Step 8:** Send a new get request for multiple OIDS and return response event for the request only, if no timeout has occurred.

**Step 9:** Create target method which contains information about where the data should be fetched and how to return.

**Step 10:** If response is not NULL then no time out has occurred and the response was successfully delivered.
**System and Software tools required:**

<table>
<thead>
<tr>
<th>Package Required</th>
<th>Operating System</th>
<th>Minimum Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Compiler</td>
<td>UBUNTU</td>
<td>Pentium III or Pentium IV with 2GB RAM 40 GB hard disk</td>
</tr>
</tbody>
</table>

**Sample Code:**

```java
public SNMPManager(String add)
{
    address = add;
    public static void main(String[] args) throws IOException {

        /*...Port 161 is used for Read and Other operations, Port 162 is used for the trap generation ...*/
        SNMPManager client = new SNMPManager("udp:127.0.0.1/161");
        client.start();

        /*...OID - .1.3.6.1.2.1.1.1.0 =>SysDec, OID - .1.3.6.1.2.1.1.5.0 =>SysName...*/
        String sysDescr = client.getAsString(new OID(".1.3.6.1.2.1.1.1.0"));
        System.out.println(sysDescr);
    }

    /*...the listen() method listens for answers...*/
    private void start() throws IOException {
        TransportMapping transport = new DefaultUdpTransportMapping();
        snmp = new Snmp(transport);
        transport.listen();

        /*...Method which takes a single OID and returns the response from the agent as a String...*/
        public String getAsString(OID oid) throws IOException {
            ResponseEvent event = get(new OID[]{oid});
            return event.getResponse().get(0).getVariable().toString();
        }

        /*...This method is capable of handling multiple OIDs paramoids Eturn throws IOException ...*/
        public ResponseEvent get(OID oids[]) throws IOException {
            PDU pdu = new PDU();
            for (OID oid : oids) {
                pdu.add(new VariableBinding(oid));
            }
            pdu.setType(PDU.GET);
            ResponseEvent event = snmp.send(pdu, getTarget(), null); if(event != null) {
            return event;
        }
        throw new RuntimeException("GET timed out");
    }

    /*... This method returns a Target, which contains information about where the data should be
```
fetched and how to return ...*/
private Target getTarget() {
    Address targetAddress = GenericAddress.parse(address);
    CommunityTarget target =
        new CommunityTarget();
    target.setCommunity(new OctetString("public");
    target.setAddress(targetAddress);
    target.setRetries(2);
    target.setTimeout(1500);
    target.setVersion(SnmpConstants.version2c);
    return target;
}

**Sample Output:**

Hardware: x86 Family 6 Model 23 Stepping 10 AT/AT COMPATIBLE – Software: Windows

2000 Version 5.1 (Build 2600 Multiprocessor Free)

**Test cases:**
- Verify SNMP Service are installed.
- Developed a simple client which will display the hardware information.

**Viva Questions:**
1. State the advantages of SNMP.
2. What are the requests messages support SNMP and explain it?
3. Which application layer protocol is used by network management frameworks to manage and monitor network devices?
4. What are the Main difference between SNMPv3 and SNMPv2 ?
5. In Network Management System, what is the term responsible for controlling access to network based on predefined policy ?
6. Structure of Management Information (SMI), is guideline of SNMP why?
7. Adjusting network components and features, can be a daily occurrence in a large network is called
   A. Reconciliation
   B. Reservation
   C. Restoration
   D. Reconfiguration

**RESULT**

Thus the SNMP application was implemented.
EX. NO.9 STUDY OF NETWORK SIMULATOR (NS)

Aim:

To Study of Network simulator and Simulation of Congestion Control Algorithms using NS.

NET WORK SIMULATOR (NS2)

Ns overview

- Ns programming: A Quick start
- Case study I: A simple Wireless network
- Case study II: Create a new agent in Ns
- Ns Status
- Periodical release (ns-2.26, Feb 2003)
- Platform support
- FreeBSD, Linux, Solaris, Windows and Mac

Ns Functionalities

Routing, Transportation, Traffic sources, queuing disciplines, QoS

Wireless

Ad hoc routing, mobile IP, sensor-MAC Tracing, visualization and various utilities. NS (Network Simulators) Most of the commercial simulators are GUI driven, while some network simulators are CLI driven. The network model / configuration describe the state of the network (nodes, routers, switches and links) and the events (data transmissions, packet error etc.). The important outputs of simulations are the trace files. Trace files log every packet, every event that occurred in the simulation and are used for analysis. Network simulators can also provide other tools to facilitate visual analysis of trends and potential trouble spots.

Most network simulators use discrete event simulation, in which a list of pending "events" is stored, and those events are processed in order, with some events triggering future events such as the event of the arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node. Simulation of networks is a very complex task. For example, if congestion is high, then estimation of the average occupancy is challenging because of high variance. To estimate the likelihood of a buffer overflow in a network, the time required for an accurate answer can be extremely large. Specialized techniques such as "control variants" and "importance sampling" have been developed to speed simulation.
Examples of network simulators

There are many both free/open-source and proprietary network simulators. Examples of notable network simulation software are, ordered after how often they are mentioned in research papers:

- ns (open source)
- OPNET (proprietary software)
- NetSim (proprietary software)

Uses of network simulators

Network simulators serve a variety of needs. Compared to the cost and time involved in setting up an entire test bed containing multiple networked computers, routers and data links, network simulators are relatively fast and inexpensive. They allow engineers, researchers to test scenarios that might be particularly difficult or expensive to emulate using real hardware - for instance, simulating a scenario with several nodes or experimenting with a new protocol in the network. Network simulators are particularly useful in allowing researchers to test new networking protocols or changes to existing protocols in a controlled and reproducible environment. A typical network simulator encompasses a wide range of networking technologies and can help the users to build complex networks from basic building blocks such as a variety of nodes and links. With the help of simulators, one can design hierarchical networks using various types of nodes like computers, hubs, bridges, routers, switches, links, mobile units etc. Various types of Wide Area Network (WAN) technologies like TCP, ATM, IP etc. and Local Area Network (LAN) technologies like Ethernet, token rings etc., can all be simulated with a typical simulator and the user can test, analyse various standard results apart from devising some novel protocol or strategy for routing etc. Network simulators are also widely used to simulate battlefield networks in Network-centric warfare. There are a wide variety of network simulators, ranging from the very simple to the very complex. Minimally, a network simulator must enable a user to represent a network topology, specifying the nodes on the network, the links between those nodes and the traffic between the nodes. More complicated systems may allow the user to specify everything about the protocols used to handle traffic in a network. Graphical applications allow users to easily visualize the workings of their simulated environment. Text-based applications may provide a less intuitive interface, but may permit more advanced forms of customization.

Packet loss occurs when one or more packets of data travelling across a computer network fail to reach their destination. Packet loss is distinguished as one of the three main error types encountered in digital communications; the other two being bit error and spurious packets caused due to noise. Packets can be lost in a network because they may be dropped when a queue in the network node overflows. The amount of packet loss during the steady state is another important property of a congestion control scheme. The larger the value of packet loss, the more difficult it is for transport layer protocols to
maintain high bandwidths, the sensitivity to loss of individual packets, as well as to frequency and patterns of loss among longer packet sequences is strongly dependent on the application itself.

**Throughput**

This is the main performance measure characteristic, and most widely used. In communication networks, such as Ethernet or packet radio, throughput or network throughput is the average rate of successful message delivery over a communication channel. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot. This measure how soon the receiver is able to get a certain amount of data send by the sender. It is determined as the ratio of the total data received to the end to end delay. Throughput is an important factor which directly impacts the network performance.

**Delay**

Delay is the time elapsed while a packet travels from one point e.g., source premise or network ingress to destination premise or network degrees. The larger the value of delay, the more difficult it is for transport layer protocols to maintain high bandwidths. We will calculate end to end delay.

**Queue Length**

A queuing system in networks can be described as packets arriving for service, waiting for service if it is not immediate, and if having waited for service, leaving the system after being served. Thus queue length is very important characteristic to determine that how well the active queue management of the congestion control algorithm has been working.
Wireshark:

Wireshark is a network packet analyzer. A network packet analyzer will try to capture network packets and tries to display that packet data as detailed as possible. We could think of a network packet analyzer as a measuring device used to examine what’s going on inside a network cable, just like a voltmeter is used by an electrician to examine what’s going on inside an electric cable.

In the past, such tools were either very expensive, proprietary, or both. However, with the advent of Wireshark, all that has changed. Wireshark is perhaps one of the best open source packet analyzers available today. Wireshark is an open source software project, and is released under the GNU General Public License (GPL). We can freely use Wireshark on any number of computers you like, without worrying about license keys or fees or such. In addition, all source code is freely available under the GPL.

Here are some things Wireshark does not provide:

- Wireshark isn’t an intrusion detection system. It will not warn you when someone does strange things on your network that he/she isn’t allowed to do. However, if strange things happen, Wireshark might help you figure out what is really going on.
- Wireshark will not manipulate things on the network, it will only "measure" things from it. Wireshark doesn’t send packets on the network or do other active things (except for name resolutions, but even that can be disabled).

Example people use Wireshark for:

- Network administrators use it to troubleshoot network problems
- Network security engineers use it to examine security problems
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals
- Beside these examples Wireshark can be helpful in many other situations too.

Features Wireshark provides:

- Available for UNIX and Windows.
- Capture live packet data from a network interface.
- Open files containing packet data captured with tcpdump/WinDump, Wireshark, and a number of other packet capture programs.
- Import packets from text files containing hex dumps of packet data.
- Display packets with very detailed protocol information.
Wireshark can capture traffic from many different network media types - and despite its name - including wireless LAN as well. Which media types are supported, depends on many things like the operating system you are using.

**How to Use Wireshark to Capture, Filter and Inspect Packets**

Wireshark, a network analysis tool formerly known as **Ethereal**, captures packets in real time and display them in human-readable format. Wireshark includes filters, color coding, and other features that let you dig deep into network traffic and inspect individual packets. It captures the packets, filtering them, and inspecting them. You can use Wireshark to inspect a suspicious program’s network traffic, analyze the traffic flow on your network, or troubleshoot network problems.

**Getting Wireshark**

We can download Wireshark for Windows or macOS from its official website. If you’re using Linux or another UNIX-like system, you’ll probably find Wireshark in its package repositories. For example, if you’re using Ubuntu, you’ll find Wireshark in the Ubuntu Software Center.
Capturing Packets

After downloading and installing Wireshark, you can launch it and double-click the name of a network interface under Capture to start capturing packets on that interface. For example, if you want to capture traffic on your wireless network, click your wireless interface. You can configure advanced features by clicking Capture > Options, but this isn’t necessary for now.

As soon as you click the interface’s name, you’ll see the packets start to appear in real time. Wireshark captures each packet sent to or from your system.

If you have promiscuous mode enabled it’s enabled by default you’ll also see all the other packets on the network instead of only packets addressed to your network adapter. To check if promiscuous mode is enabled, click Capture > Options and verify the “Enable promiscuous mode on all interfaces” checkbox is activated at the bottom of this window.

Click the red “Stop” button near the top left corner of the window when you want to stop capturing traffic.
Color Coding
You’ll probably see packets highlighted in a variety of different colors. Wireshark uses colors to help you identify the types of traffic at a glance. By default, light purple is TCP traffic, light blue is UDP traffic, and black identifies packets with errors—for example, they could have been delivered out of order. To view exactly what the color codes mean, click View >Coloring Rules. You can also customize and modify the coloring rules from here, if you like.

Sample Captures
If there’s nothing interesting on your own network to inspect, Wireshark’s wiki has you covered. The wiki contains a page of sample capture files that you can load and inspect. Click File > Open in Wireshark and browse for your downloaded file to open one. You can also save your own captures in Wireshark and open them later. Click File > Save to save your captured packets.
Filtering Packets

If you’re trying to inspect something specific, such as the traffic a program sends when phoning home, it helps to close down all other applications using the network so you can narrow down the traffic. Still, you’ll likely have a large amount of packets to sift through. That’s where Wireshark’s filters come in.

The most basic way to apply a filter is by typing it into the filter box at the top of the window and clicking Apply (or pressing Enter). For example, type “dns” and you’ll see only DNS packets. When you start typing, Wireshark will help you autocomplete your filter.

![Wireshark filter example](image)

You can also click Analyze > Display Filters to choose a filter from among the default filters included in Wireshark. From here, you can add your own custom filters and save them to easily access them in the future. For more information on Wireshark’s display filtering language, read the Building display filter expressions page in the official Wireshark documentation.

![Wireshark display filters](image)

Another interesting thing you can do is right-click a packet and select Follow > TCP Stream. You’ll see the full TCP conversation between the client and the server. You can also click other protocols in the Follow menu to see the full conversations for other protocols, if applicable.
Close the window and you’ll find a filter has been applied automatically. Wireshark is showing you the packets that make up the conversation.

**Inspecting Packets**

Click a packet to select it and you can dig down to view its details.

You can also create filters from here — just right-click one of the details and use the Apply as Filter submenu to create a filter based on it.
Wireshark is an extremely powerful tool, and this tutorial is just scratching the surface of what you can do with it. Professionals use it to debug network protocol implementations, examine security problems and inspect network protocol internals.

**Software-defined networking (SDN):**

Software-defined networking (SDN) is an approach to computer networking that allows network administrators to programmatically initialize, control, change, and manage network behaviour dynamically via open interfaces and abstraction of lower-level functionality. SDN is meant to address the fact that the static architecture of traditional networks doesn't support the dynamic, scalable computing and storage needs of more modern computing environments such as data centers. This is done by decoupling or disassociating the system that makes decisions about where traffic is sent (the SDN controller, or control plane) from the underlying systems that forward traffic to the selected destination (the data plane).

SDN was commonly associated with the Open Flow protocol (for remote communication with network plane elements for the purpose of determining the path of network packets across network switches) since the latter's emergence in 2011. Since 2012 however, many companies have moved away from Open Flow, and have embraced different techniques. These include Cisco Systems' Open Network Environment and Nicira's network virtualization platform.
Network Hypervisor:

A network hypervisor is a program that provides an abstraction layer for network hardware. Network hypervisors allow network engineers to create virtual networks that are completely decoupled and independent from the underlying physical network. The hypervisor enables segments of a virtual network to be managed independently and to be provisioned dynamically. In the world of hyperscale computing, physical components that make up the network (such as switches, routers and firewalls) can be virtualized to accommodate the traffic demands created by server virtualization. Once networking components have been virtualized, network engineers need to be able to programmatically control the provisioning and management of these resources. While some capabilities are provided by the hypervisors in each virtual server, more robust programming is required to treat all servers and services in the network as a single pool of resources. That's where the network hypervisor comes in.

Not all vendors are using the term "network hypervisor" to describe the program that provides an abstraction layer for network hardware. Some vendors are using the label software-defined networking controller instead.
EX. NO.11 PERFORM A CASE STUDY ABOUT THE DIFFERENT ROUTING ALGORITHMS TO SELECT THE NETWORK PATH WITH ITS OPTIMUM AND ECONOMICAL DURING DATA TRANSFER.

a) LINK STATE ROUTING  b) FLOODING  c) DISTANCE VECTOR

a) LINK STATE ROUTING

Aim:

To study the link state routing flooding and distance vector routing.

Link State routing

Routing is the process of selecting best paths in a network. In the past, the term routing was also used to mean forwarding network traffic among networks. However this latter function is much better described as simply forwarding. Routing is performed for many kinds of networks, including the telephone network (circuit switching), electronic data networks (such as the Internet), and transportation networks. This article is concerned primarily with routing in electronic data networks using packet switching technology. In packet switching networks, routing directs packet forwarding (the transit of logically addressed network packets from their source toward their ultimate destination) through intermediate nodes. Intermediate nodes are typically network hardware devices such as routers, bridges, gateways, firewalls, or switches. General-purpose computers can also forward packets and perform routing, though they are not specialized hardware and may suffer from limited performance. The routing process usually directs forwarding on the basis of routing tables which maintain a record of the routes to various network destinations. Thus, constructing routing tables, which are held in the router's memory, is very important for efficient routing. Most routing algorithms use only one network path at a time. Multipath routing techniques enable the use of multiple alternative paths. In case of overlapping/equal routes, the following elements are considered in order to decide which routes get installed into the routing table (sorted by priority):

1. Prefix-Length: where longer subnet masks are preferred (independent of whether it is within a routing protocol or over different routing protocol)
2. Metric: where a lower metric/cost is preferred (only valid within one and the same routing protocol)
3. Administrative distance: where a lower distance is preferred (only valid between different routing protocols)

Routing, in a more narrow sense of the term, is often contrasted with bridging in its assumption that network addresses are structured and that similar addresses imply proximity within the network. Structured addresses allow a single routing table entry to represent the route to a group of devices. In large networks, structured addressing (routing, in the narrow sense) outperforms unstructured addressing (bridging). Routing has become the dominant form of addressing on the Internet. Bridging is still widely used within localized environments.
b) FLOODING

Flooding is a simple routing algorithm in which every incoming packet is sent through every outgoing link except the one it arrived on. Flooding is used in bridging and in systems such as Usenet and peer-to-peer file sharing and as part of some routing protocols, including OSPF, DVMRP, and those used in ad-hoc wireless networks. There are generally two types of flooding available, Uncontrolled Flooding and Controlled Flooding. Uncontrolled Flooding is the fatal law of flooding. All nodes have neighbours and route packets indefinitely. More than two neighbours create a broadcast storm. Controlled Flooding has its own two algorithms to make it reliable, SNCF (Sequence Number Controlled Flooding) and RPF (Reverse Path Flooding). In SNCF, the node attaches its own address and sequence number to the packet, since every node has a memory of addresses and sequence numbers. If it receives a packet in memory, it drops it immediately while in RPF, the node will only send the packet forward. If it is received from the next node, it sends it back to the sender.

Algorithm

There are several variants of flooding algorithm. Most work roughly as follows:

1. Each node acts as both a transmitter and a receiver.
2. Each node tries to forward every message to every one of its neighbours except the source node. This results in every message eventually being delivered to all reachable parts of the network. Algorithms may need to be more complex than this, since, in some case, precautions have to be taken to avoid wasted duplicate deliveries and infinite loops, and to allow messages to eventually expire from the system. A variant of flooding called selective flooding partially addresses these issues by only sending packets to routers in the same direction. In selective flooding the routers don't send every incoming packet on every line but only on those lines which are going approximately in the right direction.

Advantages

Since flooding naturally utilizes every path through the network, it will also use the shortest path. This algorithm is very simple to implement.

Disadvantages

Flooding can be costly in terms of wasted bandwidth. While a message may only have one destination it has to be sent to every host. In the case of a ping flood or a denial of service attack, it can be harmful to the reliability of a computer network. Messages can become duplicated in the network further increasing the load on the networks bandwidth as well as requiring an increase in processing complexity to disregard duplicate messages. Duplicate packets may circulate forever, unless certain precautions are taken: Use a hop count or a time to live count and include it with each packet. This value should take into account the number of nodes that a packet may have to pass through on the way to its destination. Have each node keep track of every packet seen and only forward each packet once. Enforce a network topology without loops.
c) DISTANCE VECTOR ROUTING PROTOCOL USING NS2

In computer communication theory relating to packet-switched networks, a **distance vector routing protocol** is one of the two major classes of routing protocols, the other major class being the link-state protocol. Distance-vector routing protocols use the Bellman–Ford algorithm, Ford–Fulkerson algorithm, or DUAL FSM (in the case of Cisco Systems protocols) to calculate paths.

A distance-vector routing protocol requires that a router informs its neighbours of topology changes periodically. Compared to link-state protocols, which require a router to inform all the nodes in a network of topology changes, distance-vector routing protocols have less computational complexity and message overhead. The term distance vector refers to the fact that the protocol manipulates vectors (arrays) of distances to other nodes in the network. The vector distance algorithm was the original ARPANET routing algorithm and was also used in the internet under the name of RIP (Routing Information Protocol). Examples of distance-vector routing protocols include RIPv1 and RIPv2 and IGRP.

**Method**

Routers using distance-vector protocol do not have knowledge of the entire path to a destination. Instead they use two methods:

1. **Direction** in which router or exit interface a packet should be forwarded.
2. **Distance** from its destination

Distance-vector protocols are based on calculating the direction and distance to any link in a network. "Direction" usually means the next hop address and the exit interface. "Distance" is a measure of the cost to reach a certain node. The least cost route between any two nodes is the route with minimum distance. Each node maintains a vector (table) of minimum distance to every node. The cost of reaching a destination is calculated using various route metrics. RIP uses the hop count of the destination whereas IGRP takes into account other information such as node delay and available bandwidth. Updates are performed periodically in a distance-vector protocol where all or part of a router's routing table is sent to all its neighbors that are configured to use the same distance-vector routing protocol. RIP supports cross-platform distance vector routing whereas IGRP is a Cisco Systems proprietary distance vector routing protocol. Once a router has this information it is able to amend its own routing table to reflect the changes and then inform its neighbors of the changes. This process has been described as routing by rumor because routers are relying on the information they receive from other routers and cannot determine if the information is actually valid and true. There are a number of features which can be used to help with instability and inaccurate routing information.

EGP and BGP are not pure distance-vector routing protocols because a distance-vector protocol calculates routes based only on link costs whereas in BGP, for example, the local route preference value takes priority over the link cost.
Count-to-infinity problem

The Bellman–Ford algorithm does not prevent routing loops from happening and suffers from the count to infinity problem. The core of the count-to-infinity problem is that if A tells B that it has a path somewhere, there is no way for B to know if the path has B as a part of it. To see the problem clearly, imagine a subnet connected like A–B–C–D–E–F, and let the metric between the routers be "number of jumps". Now suppose that A is taken offline. In the vector-update-process B notices that the route to A, which was distance 1, is down – B does not receive the vector update from A. The problem is, B also gets an update from C, and C is still not aware of the fact that A is down – so it tells B that A is only two jumps from C (C to B to A), which is false. This slowly propagates through the network until it reaches infinity (in which case the algorithm corrects itself, due to the relaxation property of Bellman–Ford).

Sample program:

```plaintext
set ns [new Simulator]
set nr [open thro.tr w]
$ns trace-all $nr
setnf [open thro.nam w]
$ns namtrace-all $nf
proc finish {} {global ns nr nf
$ns flush-trace
close $nf
close $nr
execnamthro.nam&
exit 0}
for { set i 0 } { $i < 12 } { incr i 1 }
{ set n($i) [$ns node]} for { set i 0 } { $i < 8 } { incr i } {{
$ns duplex-link $n($i) $n([expr $i+1]) 1Mb 10ms DropTail }
$ns duplex-link $n(0) $n(8) 1Mb 10ms DropTail
$ns duplex-link $n(1) $n(10) 1Mb 10ms DropTail
$ns duplex-link $n(0) $n(9) 1Mb 10ms DropTail
$ns duplex-link $n(9) $n(11) 1Mb 10ms DropTail
$ns duplex-link $n(10) $n(11) 1Mb 10ms DropTail
```

66
$ns duplex-link $n(11) $n(5) 1Mb 10ms DropTail
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp0
set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp0 $null0
set udp1 [new Agent/UDP]
$ns attach-agent $n(1) $udp1
$cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set interval_ 0.005
$cbr1 attach-agent $udp1
set null0 [new Agent/Null]
$ns attach-agent $n(5) $null0
$ns connect $udp1 $null0
$ns rtproto DV
$ns rtmodel-at 10.0 down $n(11) $n(5)
$ns rtmodel-at 15.0 down $n(7) $n(6)
$ns rtmodel-at 30.0 up $n(11) $n(5)
$ns rtmodel-at 20.0 up $n(7) $n(6)
$udp0 set fid_ 1
$udp1 set fid_ 2
$ns color 1 Red
$ns color 2 Green
$ns at 1.0 "$cbr0 start"
$ns at 2.0 "$cbr1 start"
$ns at 45 "finish"
$ns run
Sample Output:

Link made down: