



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 1 of



ST. JOSEPH'S COLLEGE OF ENGINEERING AND TECHNOLOGY, PALAI

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VIth SEMESTER

CS334 Network Programming Lab

Staff-in-Charge:

Assistant Professor in Computer Science and Engineering
St. Joseph's College of Engineering and Technology, Palai

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 2 of

1. Every Machine should be having two numbers for identification, one for station and a permanent serial number. It will never be changed.
2. Preventive maintenance should be done in every machine with the schedule attached below.
3. Antivirus update should be done monthly in internet lab and once in six months at computer center.
4. Every preventive maintenance action taken should be recorded properly and the head of the department monthly should verify it. The record should show the machine number, date of action, person responsible and action taken.
5. All the complaints reported (hardware complaints, UPS problems, etc.) should be entered in the breakdown register kept in the computer center.



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 3 of

PREVENTIVE MAINTENANCE REGISTER FOR SERVERS

- | | |
|-------------------------|----------|
| 1. Disk Defragmentation | 2 Years |
| 2. Scandisk | 1 Month |
| 3. Checking Hardware | 6 Months |
| 4. Anti-virus Updation | 1 Month |

PREVENTIVE MAINTENANCE REGISTER FOR PC's at Computer Center

- | | |
|-------------------------|----------|
| 1. Disk Defragmentation | 2 Years |
| 2. Scandisk | 1 Month |
| 3. Checking Hardware | 6 Months |
| 4. Anti-virus Updation | 1 Month |

PREVENTIVE MAINTENANCE REDISTER FOR PC's at Internet Lab

- | | |
|-------------------------|----------|
| 1. Disk Defragmentation | 2 Years |
| 2. Scandisk | 1 Month |
| 3. Checking Hardware | 6 Months |
| 4. Anti-virus Updation | Daily |

PREVENTIVE MAINTENANCE REGISTER FOR Printers.

- | | |
|-------------------------------------|----------|
| 1. Lubricating Springs | 6 Months |
| 2. Check the ribbon, Cartridge etc. | 6 Months |
| 3. Clean the Printer | 6 Months |

PREVENTIVE MAINTENANCE REGISTER FOR Scanners.

- | | |
|----------------------|----------|
| 1. Clean the Scanner | 6 Months |
|----------------------|----------|

PREVENTIVE MAINTENANCE REGISTER FOR Modems.

- | | |
|---------------------|----------|
| 1. Clean the Modems | 6 Months |
|---------------------|----------|

PREVENTIVE MAINTENANCE REGISTER FOR Campus Automation Software

- | | |
|------------------------------|---------|
| 1. Backup the Database to CD | Monthly |
|------------------------------|---------|

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 4 of

Course No.	Course Name	L-T-P-Credits	Year of Introduction
CS334	Network Programming Lab	0-0-3-1	2015
Pre-requisite <ol style="list-style-type: none">1. CS307 Data Communication2. CS306 Computer Networks			
Course Objectives <ol style="list-style-type: none">1. To introduce Network related commands and configuration files in Linux Operating System.2. To introduce tools for Network Traffic Analysis and Network Monitoring.3. To practice Network Programming using Linux System Calls.4. To design and deploy Computer Networks.			
List of Exercises/ Experiments (12 Exercises/ Experiments are to be completed . Exercises/ Experiments marked with * are mandatory) <ol style="list-style-type: none">1. Getting started with Basics of Network configurations files and Networking Commands in Linux.2. To familiarize and understand the use and functioning of System Calls used for Operating system and network programming in Linux.3. <u>Familiarization and implementation of programs related to Process and thread.</u>4. <u>Implement the First Readers-Writers Problem.</u>5. <u>Implement the Second Readers-Writers problem.</u>6. <u>Implement programs for Inter Process Communication using PIPE, Message Queue and Shared Memory.</u>7. Implement Client-Server communication using Socket Programming and TCP as transport layer protocol.*8. Implement Client-Server communication using Socket Programming and UDP as transport layer protocol.*9. Implement a multi user chat server using TCP as transport layer protocol.*10. Implement Concurrent Time Server application using UDP to execute the program at remoteserver. Client sends a time request to the server, server sends its system time back to the client. Client displays the result.*11. Implement and simulate algorithm for Distance vector routing protocol.12. Implement and simulate algorithm for Link state routing protocol.			



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 5 of

13. Implement Simple Mail Transfer Protocol.*
14. Develop concurrent file server which will provide the file requested by client if it exists. If not server sends appropriate message to the client. Server should also send its process ID (PID) to clients for display along with file or the message.*
15. Using Wireshark observe data transferred in client server communication using UDP and identify the UDP datagram.
16. Using Wireshark observe Three Way Handshaking Connection Establishment, Data Transfer and Three Way Handshaking Connection Termination in client server communication using TCP.
17. Develop a packet capturing and filtering application using raw sockets.
18. Design and configure a network with multiple subnets with wired and wireless LANs using required network devices. Configure the following services in the network- TELNET, SSH, FTP server, Web server, File server, DHCP server and DNS server.*
19. Install network simulator NS-2 in any of the Linux operating system and simulate wired and wireless scenarios.

Expected Outcome

Student is able to

1. *Use network related commands and configuration files in Linux Operating System.*
2. *Develop operating system and network application programs.*
3. *Analyze network traffic using network monitoring tools.*



LIST OF EXPERIMENTS

EXPNO	TITLE
1.	Familiarization of network configuration files and networking commands in LINUX
2.	Familiarization of system calls in UNIX operating system
3.	Familiarization of process and thread with help of a C program.
4.	Implement Socket programming using TCP/IP protocol.
5.	Implement Socket programming using UDP protocol.
6.	Implement multichat using TCP.
7.	Implement Concurrent Time Server application using UDP to execute the program at remote server.
8.	Implement SMTP protocol.
9.	Implement concurrent file servers which provide file on request by client.
10.	Design and configure a network with multiple subnets with wired and wireless LAN's. Configure TELNET, SSH, FTP, DHCP and DNS server.
11.	Installation and understanding NS2.
12.	Use wireshark to observe a three way handshake connection establishment.



Exp. No: 01

Network Configuration Files and Networking Commands

AIM :

Write different LINUX scripts to familiarization network configuration files and networking commands.

LINUX scripts:

Networking commands:

1. ifconfig:

ifconfig (interface configurator) command is use to initialize an interface, assign IP Address to interface and enable or disable interface on demand. With this command you can view IP Address and Hardware / MAC address assign to interface and also MTU (Maximum transmission unit) size.

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:28:FD:4C
inet addr:192.168.50.2  Bcast:192.168.50.255  Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fe28:fd4c/64 Scope:Link
UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:6093 errors:0 dropped:0 overruns:0 frame:0
TX packets:4824 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:6125302 (5.8 MiB)  TX bytes:536966 (524.3 KiB)
Interrupt:18 Base address:0x2000
lo        Link encap:Local Loopback
inet addr:127.0.0.1  Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING  MTU:16436  Metric:1
RX packets:8 errors:0 dropped:0 overruns:0 frame:0
TX packets:8 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:480 (480.0 b)  TX bytes:480 (480.0 b)
```

2. PING Command

PING (Packet INternet Groper) command is the best way to test connectivity between two nodes. Whether it is Local Area Network (LAN) or Wide Area Network (WAN). Ping use ICMP (Internet Control Message Protocol) to communicate to other devices. You can ping host name of ip address using below command.



```
# ping 4.2.2.2
PING 4.2.2.2 (4.2.2.2) 56(84) bytes of data.
64 bytes from 4.2.2.2: icmp_seq=1 ttl=44 time=203 ms
64 bytes from 4.2.2.2: icmp_seq=2 ttl=44 time=201 ms
64 bytes from 4.2.2.2: icmp_seq=3 ttl=44 time=201 ms
OR
# ping www.tecmint.com
PING tecmint.com (50.116.66.136) 56(84) bytes of data.
64 bytes from 50.116.66.136: icmp_seq=1 ttl=47 time=284 ms
64 bytes from 50.116.66.136: icmp_seq=2 ttl=47 time=287 ms
64 bytes from 50.116.66.136: icmp_seq=3 ttl=47 time=285 ms
```

3. TRACEROUTE Command

traceroute is a network troubleshooting utility which shows number of hops taken to reach destination also determine packets traveling path. Below we are tracing route to global DNS server IP Address and able to reach destination also shows path of that packet is traveling.

```
# traceroute 4.2.2.2
traceroute to 4.2.2.2 (4.2.2.2), 30 hops max, 60 byte packets
 1  192.168.50.1 (192.168.50.1)  0.217 ms  0.624 ms  0.133 ms
 2  227.18.106.27.mysipl.com (27.106.18.227)  2.343 ms  1.910 ms  1.799 ms
 3  221-231-119-111.mysipl.com (111.119.231.221)  4.334 ms  4.001 ms  5.619 ms
 4  10.0.0.5 (10.0.0.5)  5.386 ms  6.490 ms  6.224 ms
 5  gi0-0-0.dgw1.bom2.pacific.net.in (203.123.129.25)  7.798 ms  7.614 ms  7.378 ms
 6  115.113.165.49.static-mumbai.vsnl.net.in (115.113.165.49)  10.852 ms  5.389 ms  4.322 ms
 7  ix-0-100.tcore1.MLV-Mumbai.as6453.net (180.87.38.5)  5.836 ms  5.590 ms  5.503 ms
 8  if-9-5.tcore1.WYN-Marseille.as6453.net (80.231.217.17)  216.909 ms  198.864 ms  201.737 ms
 9  if-2-2.tcore2.WYN-Marseille.as6453.net (80.231.217.2)  203.305 ms  203.141 ms  202.888 ms
10  if-5-2.tcore1.WV6-Madrid.as6453.net (80.231.200.6)  200.552 ms  202.463 ms  202.222 ms
```

4. NETSTAT Command

Netstat (Network Statistic) command display connection info, routing table information etc. To displays routing table information use option as -r.

```
# netstat -r
Kernel IP routing table
Destination        Gateway            Genmask           Flags     MSS Window  irtt Iface
192.168.50.0        *                  255.255.255.0     U           0 0          0 eth0
link-local          *                  255.255.0.0       U           0 0          0 eth0
default            192.168.50.1      0.0.0.0           UG          0 0          0 eth0
```

5. NSLOOKUP Command

nslookup command also use to find out DNS related query. The following examples shows A Record (IP Address) of tecmint.com.



```
# nslookup www.tecmint.com
Server:      4.2.2.2
Address:     4.2.2.2#53
Non-authoritative answer:
www.tecmint.com canonical name = tecmint.com.
Name:   tecmint.com
Address: 50.116.66.136
```

6. ROUTE Command

route command also shows and manipulate ip routing table. To see default routing table in Linux, type the following command.

```
# route
Kernel IP routing table
Destination    Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.50.0   *               255.255.255.0   U        0      0        0 eth0
link-local     *               255.255.0.0     U        1002   0        0 eth0
default        192.168.50.1   0.0.0.0         UG        0      0        0 eth0
```

7. HOST Command

host command to find name to IP or IP to name in IPv4 or IPv6 and also query DNS records.

```
# host www.google.com
www.google.com has address 173.194.38.180
www.google.com has address 173.194.38.176
www.google.com has address 173.194.38.177
www.google.com has address 173.194.38.178
www.google.com has address 173.194.38.179
www.google.com has IPv6 address 2404:6800:4003:802::1014
```

8. ARP Command

ARP (Address Resolution Protocol) is useful to view / add the contents of the kernel's ARP tables. To see default table use the command as.

```
# arp -e
Address                  HWtype  HWaddress          Flags Mask            Iface
192.168.50.1             ether    00:50:56:c0:00:08   C                     eth0
```

9. HOSTNAME Command

hostname is to identify in a network. Execute hostname command to see the hostname of your box. You can set hostname permanently in /etc/sysconfig/network. Need to reboot box once set a proper hostname.

```
# hostname
tecmint.com
```



Network Configuration Files

Before delving into the interface configuration files, let us first itemize the primary configuration files used in network configuration. Understanding the role these files play in setting up the network stack can be helpful when customizing a Red Hat Enterprise Linux system.

The primary network configuration files are as follows:

/etc/hosts

The main purpose of this file is to resolve hostnames that cannot be resolved any other way. It can also be used to resolve hostnames on small networks with no DNS server. Regardless of the type of network the computer is on, this file should contain a line specifying the IP address of the loopback device (**127.0.0.1**) as **localhost.localdomain**. For more information, refer to the **hosts** man page.

/etc/resolv.conf

This file specifies the IP addresses of DNS servers and the search domain. Unless configured to do otherwise, the network initialization scripts populate this file. For more information about this file, refer to the **resolv.conf** man page.

/etc/sysconfig/network

This file specifies routing and host information for all network interfaces. For more information about this file and the directives it accepts,

/etc/sysconfig/network-scripts/ifcfg-*<interface-name>*

For each network interface, there is a corresponding interface configuration script. Each of these files provide information specific to a particular network interface.

RESULT:

The commands are executed and the understood the configuration commands of network in LINUX



Exp No: 02

Familiarization of System Call in UNIX Operating System

AIM :

Program to familiarize system call- fork using C programming language.

ALGORITHM :

Step1: Start.

Step 2:Show message "Hello world".

Step 3:Invoke fork() system call and return value of operation to id.

Step 4:if id>0 then got step 5, else go to step6

Step 5: Print "This is parent section" and its process id, goto step

Step 6:if id==0 the goto step 7, else goto step8

Step 7: Print "child create" and its Id" also display parent process id, goto step

Step8: Print " fork failed"

Step 9:Stop

PROGRAM:

```
#include <stdio.h>
#include <unistd.h>
```

```
intmain()
{
    intid;
    printf("Hello, World!\n");

    id=fork();
    if(id>0)
    {
        /*parent process*/
        printf("This is parent section [Process id: %d].\n",getpid());
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 12 of

```
}  
elseif(id==0)  
{  
    /*child process*/  
    printf("fork created [Process id: %d].\n",getpid());  
    printf("fork parent process id: %d.\n",getppid());  
}  
else  
{  
    /*fork creation faile*/  
    printf("fork creation failed!!!\n");  
}  
  
return0;  
}
```

OUTPUT:

```
Hello, World!  
This is parent section [Process id: 1252].  
fork created [Process id: 1253].  
fork parent process id: 1252.
```

RESULT:

The program is executed and the output is verified



Exp No: 03

Familiarization of Process and Thread in UNIX Operating System

AIM :

Program to familiarize process and thread using C programming language.

ALGORITHM :

Step 1: Start

Step2: Declare a variable MAX and initialize it with value 10.

Step3: Declare method odd to print odd numbers till MAX

Step4: Declare a method even to print even number till MAX.

Step 5: Declare two thread variables t1 and t2

Step 6: Create two thread odd and even

Step 7: Invoke two thread odd and even

Step 8: Join both threads

Step 9: Stop

PROGRAM:

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

int MAX = 10;
int count = 0;

void *even(void *arg)
{
    printf("This is even thread()\n");
    while(count < MAX)
    if(count % 2 == 0)
    printf("%d ", count++);
    pthread_exit(0);
}

void *odd(void *arg)
```



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 14 of

```
{  
printf("This is odd thread()\n");  
while(count < MAX)  
if(count % 2 == 1)  
printf("%d ", count++);  
pthread_exit(0);  
}  
  
int main()  
{  
pthread_t t1;  
pthread_t t2;  
  
pthread_create(&t1;, 0, &even;, NULL);  
pthread_create(&t2;, 0, &odd;, NULL);  
  
pthread_join(t1, 0);  
pthread_join(t2, 0);  
  
return 0;  
}
```

OUTPUT:

1 0 2 3 4 5 6 7 9 8

RESULT:

The Program is executed and the output is verified



Exp No:04

Implantation of TCP/IP Protocol.

AIM :

Write a program to implement TCP/IP protocol using java programming language .

ALGORITHM :

SERVER:

Step 1: Start

Step 2: Server initializes the ServerSocket class and waiting for connection request from client .

Step3: On getting request accept connection with Socket class object.

Step4 :Read message from client and print.

Step5: Read input from keyboard and send back to client

Step6: Repeat step 4 to 5 until close connection

Step 7: Stop

CLIENT:

Step 1: Start

Step 2: Client initializes the Socket class and send connection request to server .

Step3: On accepting connection request got to setp 4, else go to step 7.

Step4 :Read input from keyboard and send to server

Step5: Read message from server and print.

Step6: Repeat step 4 to 5 until close connection

Step 7: Stop



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 16 of

PROGRAM:

```
//tcpserver.java

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

class tcpserver {
    public static void main(String args[]) throws Exception
    {

        String clientsentence, serversentence;
        ServerSocket welcomesocket = new ServerSocket(6789);

        while(true)
        {
            Socket connectionsocket = welcomesocket.accept();

            System.out.println("server side ");

            BufferedReader infromclient = new BufferedReader(new
            InputStreamReader(connectionsocket.getInputStream()));

            clientsentence = infromclient.readLine();
            System.out.println("from client : "+clientsentence);

            BufferedReader infromuser = new BufferedReader(new
            InputStreamReader(System.in));

            System.out.println("Enter a message: ");
            serversentence = infromuser.readLine();

            DataOutputStream outtoclient = new
            DataOutputStream(connectionsocket.getOutputStream());
            ;

            outtoclient.writeBytes(serversentence + '\n');
        }
    }
}

//tcpclient.java

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

class tcpclient
{
    public static void main(String args[]) throws Exception
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 17 of

```
{  
  
    String sentence, replysentence;  
  
    System.out.println("client side");  
    Socket clientsocket=new Socket("192.168.1.141", 6789);  
  
    BufferedReaderinfromuser= new BufferedReader(new  
InputStreamReader(System.in));  
  
    System.out.println("Enter a message: ");  
    sentence=infromuser.readLine();  
  
    DataOutputStreamouttoserver=new  
DataOutputStream(clientsocket.getOutputStream());  
    outtoserver.writeBytes(sentence + '\n');  
  
    BufferedReaderinfromserver= new BufferedReader(new  
InputStreamReader(clientsocket.getInputStream()));  
  
    replysentence=infromserver.readLine();  
    System.out.println("from server: " + replysentence);  
  
    clientsocket.close();  
}  
}
```

OUTPUT

```
[sjcet@localhost ~]$ javac tcpserver.java  
[sjcet@localhost ~]$ java tcpserver
```

```
server side  
from client : welcome
```

```
Enter a message:  
hello
```

```
server side  
from client : how are you
```

```
Enter a message:  
i am fine
```

```
[sjcet@localhost ~]$  
[sjcet@localhost ~]$ javac tcpclient.java
```



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 18 of

```
[sjcet@localhost ~]$ javatcpclient
```

clientside

Enter a message:

welcome

from server: hello

```
[sjcet@localhost ~]$ javac tcpclient.java
```

```
[sjcet@localhost ~]$ javatcpclient
```

client side

Enter a message:

how are you

from server: i am fine

```
[sjcet@localhost ~]$
```

RESULT

The inter process communication using TCP has been successfully executed and tested.



Exp No:05

Implementation of UDP protocol

AIM :

Write a Java program to set up inter process communication over a connectionless service.
(Using User Datagram Protocol- UDP sockets)

ALGORITHM :

SERVER:

Step 1: Start

Step 2: Server initializes the DatagramSocket class and waiting for message from client

Step3: :Read message from client and print.

Step4: Read input from keyboard and send back to client

Step5: Repeat step 3 to 4 until close connection

Step 6: Stop

CLIENT:

Step 1: Start

Step 2: Client initializes the DatagramSocket class

Step3: Read input from keyboard and send server

Step5: Read message from server and print.

Step6: Repeat step 4 to 5 until close connection

Step 7: Stop

PROGRAM:

```
//udpserver.java  
  
import java.io.*;
```



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 20 of

```
import java.net.*;

classudpserver {
    public static void main(String args[]) throws Exception
    {

        DatagramSocketserversocket=new DatagramSocket(9876);
        byte[] receivedata=new byte[1024];
        byte[] senddata=new byte[1024];

        while(true)
        {
            System.out.println("server side ");

            DatagramPacketreceivepacket=new
            DatagramPacket(receivedata,receivedata.length);
            serversocket.receive(receivepacket);
            String sentence=new String(receivepacket.getData());

            System.out.println("from client: "+sentence);

            InetAddressipaddress=receivepacket.getAddress();
            int port=receivepacket.getPort();

            BufferedReaderinfromuser=new BufferedReader(new
            InputStreamReader(System.in));
            System.out.println("Enter a message: ");
            String serversentence=infromuser.readLine();

            senddata=serversentence.getBytes();
            DatagramPacketsendpacket=new
            DatagramPacket(senddata, senddata.length, ipaddress,
            port);
            serversocket.send(sendpacket);
        }
    }
}

//udpclient.java

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

classudpclient {
    public static void main(String args[]) throws Exception
    {

        DatagramSocketclientsocket=new DatagramSocket();
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 21 of

```
        InetAddress ipAddress=InetAddress.getByName("192.168.1.141");
    };
    clientsocket.close();
    byte[] senddata=new byte[1024];
    byte[] receivedata=new byte[1024];

    System.out.println("client side ");

    BufferedReader infromuser=new BufferedReader(new
    InputStreamReader(System.in));
    System.out.println("Enter a message: ");
    String sentence=infromuser.readLine();

    senddata=sentence.getBytes();
    DatagramPacket sendpacket=new DatagramPacket(senddata,
    senddata.length, ipAddress, 9876);
    clientsocket.send(sendpacket);

    DatagramPacket receivepacket=new
    DatagramPacket(receivedata, receivedata.length);
    clientsocket.receive(receivepacket);
    String replysentence=new String(receivepacket.getData());

    System.out.println("from server: "+replysentence);

    clientsocket.close();
}
}
```

OUTPUT

At the server

```
[sjcet@localhost ~]$ javac udpserver.java
```

```
[sjcet@localhost ~]$ java udpserver
```

server side

from client: welcome

Enter a message:hello

server side

from client: how are you

Enter a message:i am fine

server side

```
[sjcet@localhost ~]$
```



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 22 of

At the client

```
[sjcet@localhost ~]$ javac udpclient.java
```

```
[sjcet@localhost ~]$ javaudpclient
```

client side

Enter a message:welcome

from server: hello

```
[sjcet@localhost ~]$ javac udpclient.java
```

```
[sjcet@localhost ~]$ javaudpclient
```

client side

Enter a message: how are you

from server: i am fine

RESULT

The inter process communication has been successfully executed and tested.

Prepared By:

Approved By:



Exp No:06

Implementation of Multichat

AIM :

Write a program to implementation of Multichat using TCP protocol.

ALGORITHM :

SERVER:

Step 1: Start

Step 2: Server initializes the ServerSocket class and waiting for connection request from client .

Step3: On getting request from two clients accept connection with two Socket class object.

Step4 :Read message from client1 and print and send to client 2 and also read from client2 to cleint1.

Step6: Repeat step 4 until close connection

Step 7: Stop

CLIENT1:

Step 1: Start

Step 2: Client initializes the Socket class and send connection request to server .

Step3: On accepting connection request got to setp 4, else go to step 7.

Step4 :Read input from keyboard and send to server

Step5: Read message from server and print.

Step6: Repeat step 4 to 5 until close connection

Step 7: Stop

CLIENT2:

Step 1: Start

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 24 of

Step 2: Client initializes the Socket class and send connection request to server .

Step3: On accepting connection request got to setp 4, else go to step 7.

Step4 :Read input from keyboard and send to server

Step5: Read message from server and print.

Step6: Repeat step 4 to 5 until close connection

Step 7: Stop

PROGRAM:

//Server program

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

public class server1 implements Runnable
{

String str;
ServerSocket ser;
Socket s,s1;
BufferedReader br;
BufferedReader br1;
Thread t1=null;
Thread t2=null;
PrintWriter pw,pw1;

public server1()
{

try
{
ser=new ServerSocket(5000);
System.out.println("Server running");
s=ser.accept();
System.out.println("Client1 connected");
s1=ser.accept();
System.out.println("Client2 connected");
br=new BufferedReader(new InputStreamReader(s.getInputStream()));
br1=new BufferedReader(new InputStreamReader(s1.getInputStream()));
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**

Department of Computer Science and Engineering

WORK INSTRUCTION

PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 25 of

```
pw=new PrintWriter(s.getOutputStream(),true);  
pw1=new PrintWriter(s1.getOutputStream(),true);  
}
```

```
catch(Exception e)  
{  
  
}
```

```
public void run()  
{try{  
do  
{  
  
if(Thread.currentThread()==t1)  
{
```

```
str=br.readLine();  
pw1.println(str);
```

```
str=br1.readLine();  
pw.println(str);  
}
```

```
}while(true);}catch(Exception e1){}  
}
```

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

```
server1serv=new server1();  
serv.t1=new Thread(serv);  
serv.t2=new Thread(serv);  
serv.t1.start();  
serv.t2.start();
```

```
}
```

```
//Client1 program
```

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

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 26 of

```
import java.net.*;

public class client1 implements Runnable
{

String str=null;

Socket s;
BufferedReader br;
BufferedReader br1;
Thread t1=null;
Thread t2=null;
PrintWriter pw;

public client1()
{

try
{
System.out.println("Server running");
s=new Socket("localhost",5000);
System.out.println("Client1 connected");
br=new BufferedReader(new InputStreamReader(System.in));
br1=new BufferedReader(new InputStreamReader(s.getInputStream()));
pw=new PrintWriter(s.getOutputStream(),true);
}

catch(Exception e)
{
}

}

public void run()
{
try
{
do
{

if(Thread.currentThread()==t1)
{
str=br.readLine();pw.println(str);
}

else
```



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 27 of

```
{  
str=br1.readLine();System.out.println("Client2:"+str);  
  
}  
}  
while(true);  
}  
catch(Exception e1){}  
}  
  
}  
  
public static void main(String args[])  
{  
try  
{  
client1 cli=new client1();  
cli.t1=new Thread(cli);  
cli.t2=new Thread(cli);  
cli.t1.start();  
cli.t2.start();  
}  
catch(Exception e)  
{  
}  
  
}  
  
}  
}
```

//Client2 program

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

```
public class client11 implements Runnable  
{
```

```
String str=null;  
Socket s;  
BufferedReaderbr;  
BufferedReader br1;  
Thread t1=null;  
Thread t2=null;  
PrintWriterpw;
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**

Department of Computer Science and Engineering

WORK INSTRUCTION

PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 28 of

```
public client11()
{
try
{
System.out.println("Server running");
s=new Socket("localhost",5000);
System.out.println("Client1 connected");
br=new BufferedReader(new InputStreamReader(System.in));
br1=new BufferedReader(new InputStreamReader(s.getInputStream()));
pw=new PrintWriter(s.getOutputStream(),true);
}
catch(Exception e)
{}
}
public void run()
{
try
{
do
{

if(Thread.currentThread()==t1)
{
str=br.readLine();pw.println(str);

}

else
{
str=br1.readLine();System.out.println("Client1:"+str);

}
}
while(true);
}
catch(Exception e1){}
}

public static void main(String args[])
{
try
{
client11 cli=new client11();
cli.t1=new Thread(cli);
cli.t2=new Thread(cli);
cli.t1.start();
cli.t2.start();
}
}
```

Prepared By:

Approved By:

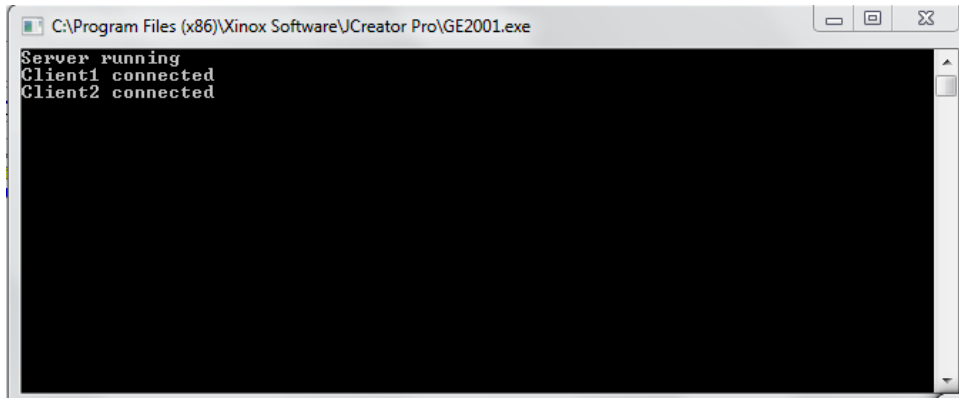


**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

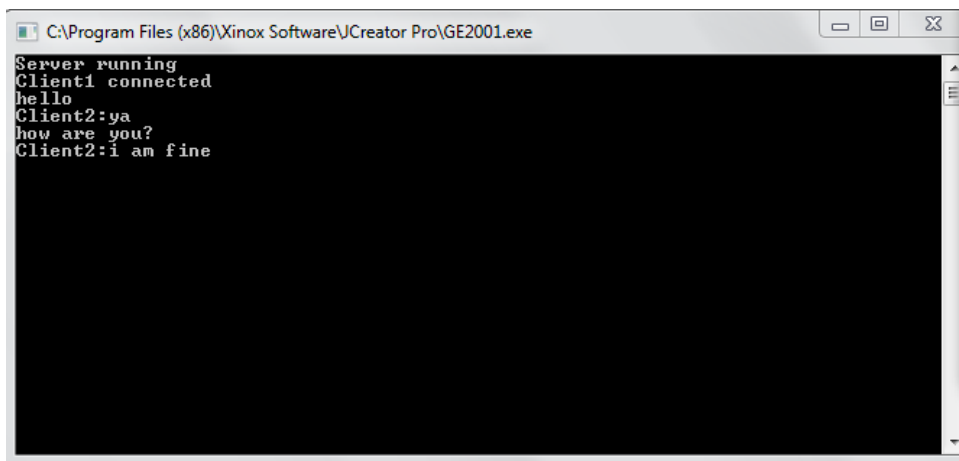
Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 29 of

```
catch(Exception e){}  
}  
}
```

OUTPUT



```
C:\Program Files (x86)\Xinox Software\JCreator Pro\GE2001.exe  
Server running  
Client1 connected  
Client2 connected
```



```
C:\Program Files (x86)\Xinox Software\JCreator Pro\GE2001.exe  
Server running  
Client1 connected  
hello  
Client2:ya  
how are you?  
Client2:i am fine
```



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 30 of

```
C:\Program Files (x86)\Xinox Software\JCreator Pro\GE2001.exe
Server running
Client1 connected
Client1:hello
ya
Client1:how are you?
i am fine
```

RESULT:

The program is executed and the output is verified.

Prepared By:

Approved By:



Exp No:07

Concurrent Time Server application using UDP

AIM :

Write a program to implement Concurrent Time Server application using UDP to execute the program at remote server..

ALGORITHM :

SERVER:

Step 1: Start

Step 2: Import necessary packages

Step3: Create object or server class and bind it to rmi registry.

Step4 : Implement the function date1 to find system time and return the time.

Step 7: Stop

CLIENT:

Step 1: Start

Step 2: Client initializes the necessary variables.

Step3: Perform lookup for server entry while doing the object creation.

Step4 :Send the request for time to interface

Step5: Read replay message and display the result

Step6: Stop

INTERFACE:

Step 1: Start

Step 2: Define the remote method date1 with necessary parameters

Step 3: Stop

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 32 of

PROGRAM

/* the Remote Interfacetim.java */

```
import java.rmi.*;  
  
public interface tim extends Remote  
{  
    String date1() throws RemoteException;  
}
```

/*Server.java*/

```
import java.rmi.*;  
import java.rmi.server.*;  
import java.util.*;  
  
public class tmserver extends UnicastRemoteObject implements tim  
{  
    public tmserver() throws RemoteException  
    {}  
  
    public static void main(String args[]) throws Exception  
    {  
        System.out.println("hiii this is server");  
  
        tmserver s = new tmserver();  
        Naming.rebind("tmserver", s);  
    }  
  
    public String date1()  
    {
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 33 of

```
return(new Date().toString());  
  
}  
  
}/*Client.java*/  
  
import java.rmi.*;  
  
import java.net.*;  
  
import java.rmi.*;  
  
public class Clientm  
{  
  
public static void main(String args[])  
{  
  
try  
{  
  
String URL="rmi://localhost/tmserver";  
  
tim m=(tim)Naming.lookup(URL);  
  
  
  
System.out.println("The date and time is"+m.date1());  
  
}  
  
catch(Exception e)  
{  
  
System.out.println(e);  
  
}  
  
}}}
```

COMPLIATION STEPS

Step1:

compile all java files Clientm.java , tmserver.java, tim.java

Step 2:

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 34 of

generate stubs and skeleton

To generate stubs and skeleton ,use the tool called RMI compiler which is invoked from
command line

rmictmserver

Step 3:

Install files on the client and server machine.

Copy Clienttm.class,tmserver_Stub.class,multiply.class to a directory on the client machine.

tmserver.class,tmserver_Skel.class ,tmserver_Stub.class,tim.class to a directory on theserver machine.

Step4:

Start the RMI registry on the server machine

rmiregistry

Step5:

Start the Server

javatmserver

Step6:

Start the client

Java Clentm

OUTPUT

Server starts

The date and time is:04/02/2018 09:30:33

RESULT

The RMI program has been executed and tested successfully.

Prepared By:

Approved By:



Exp No:08

Implement SMTP protocol

AIM :

Write a program to implement SMTP protocol.

ALGORITHM :

SERVER:

Step 1: Start

Step 2: Server initializes the ServerSocket class and waiting for connection request from client .

Step3: On getting request accept connection with Socket class object.

Step4 :Read message from client and do necessary action after checking the code.

Step5: Do the action and send corresponding status code back to client

Step6: Repeat step 4 to 5 until close connection

Step 7: Stop

CLIENT:

Step 1: Start

Step 2: Client initializes the Socket class and send connection request to server .

Step3: On accepting connection request got to step 4, else go to step 7.

Step4 :Get the code from user for action and send to server

Step5: Read message from server and do necessary reply action .

Step6: Repeat step 4 to 5 until close connection

Step 7: Stop



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 36 of

PROGRAM

SMTP SERVER

```
import java.io.*;

import java.net.*;

class SMTPS {

public static void main(String args[]) throws Exception

{

String clientsentence,serversentence;

BufferedReaderinfromuser= new BufferedReader(new InputStreamReader(System.in));

ServerSocketwelcomesocket=new ServerSocket(6789);

System.out.println("server side ");

Socket connectionsocket=welcomesocket.accept();

System.out.println("Connection request accepted ");

DataOutputStreamouttoclient=new DataOutputStream(connectionsocket.getOutputStream());

BufferedReaderinfromclient=new BufferedReader(new

InputStreamReader(connectionsocket.getInputStream()));

System.out.println("SEND 220 ");

outtoclient.writeBytes("220"+"\n");

//Receiving EHLO
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 37 of

```
clientsentence=infromclient.readLine();

System.out.println("from client : "+clientsentence);


System.out.println("PRESS ENTER TO RESPOND");

infromuser.readLine();


System.out.println("SEND 250 to client ");

outtoclient.writeBytes("250 mail server at your service"+"\\n");

//recevg MAIL from


clientsentence=infromclient.readLine();

System.out.println("from client : "+clientsentence);


//reply 250

System.out.println("PRESS ENTER TO RESPOND");

infromuser.readLine();

System.out.println("SEND 250 to client ");

outtoclient.writeBytes("250 OK"+"\\n");
```



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 38 of

//receiving RCPT

```
clientsentence=infromclient.readLine();  
  
System.out.println("from client : "+clientsentence);
```

//reply

```
System.out.println("PRESS ENTER TO RESPOND");  
  
infromuser.readLine();  
  
System.out.println("SEND 250 to client ");  
  
outtoclient.writeBytes("250 OK"+'\n');
```

//receiving DATA

```
clientsentence=infromclient.readLine();  
  
System.out.println("from client : "+clientsentence);
```

//reply

```
System.out.println("PRESS ENTER TO RESPOND");  
  
infromuser.readLine();  
  
System.out.println("SEND 354 to client ");  
  
outtoclient.writeBytes("354 OK"+'\n');
```

//receivin mail body

```
clientsentence=infromclient.readLine();  
  
System.out.println("Mail body : "+clientsentence);
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 39 of

```
//receiving dot
clientsentence=infromclient.readLine();

System.out.println("from client DOT received : "+clientsentence);
```

```
//reply
System.out.println("PRESS ENTER TO RESPOND");

infromuser.readLine();

System.out.println("SEND 250 to client ");

outtoclient.writeBytes("250 OK"+"\n");
```

```
//receiving QUIT
clientsentence=infromclient.readLine();

System.out.println("from client : "+clientsentence);

}

}
```

SMTP CLIENT

```
import java.io.*;

import java.net.*;

class SMTPC {

public static void main(String args[]) throws Exception

{

String sentence, replysentence;

BufferedReader infromuser= new BufferedReader(new InputStreamReader(System.in));
```



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 40 of

```
System.out.println("client side");

Socket clientsocket=new Socket("localhost", 6789);

BufferedReaderinfromserver= new BufferedReader(new InputStreamReader(clientsocket.getInputStream()));

replysentence=infromserver.readLine();

System.out.println("Received "+replysentence);


System.out.println("sending EHLO to mailserver");

DataOutputStreamouttoserver=new DataOutputStream(clientsocket.getOutputStream());

outtoserver.writeBytes("EHLO" + "\n");


System.out.println("Waiting response from server");


replysentence=infromserver.readLine();

System.out.println("Received : "+replysentence);


//send MAIL FROM

System.out.println("Enter the your email address");

sentence=infromuser.readLine();

System.out.println("MAIL FROM:<"+sentence+">");

outtoserver.writeBytes("MAIL FROM:<"+sentence+">" + "\n");


//receiving reply
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 41 of

```
System.out.println("Waiting response from server");

replysentence=infromserver.readLine();

System.out.println("Received: "+replysentence);


//send RCPT to

System.out.println("Enter the destination email address");

sentence=infromuser.readLine();

System.out.println("RCPT TO:<" + sentence + ">");

outtoserver.writeBytes("RCPT TO:<" + sentence + ">" + "\n");


//receiving reply

System.out.println("Waiting response from server");

replysentence=infromserver.readLine();

System.out.println("Received : "+replysentence);


//send DATA to

System.out.println("Sending DATA command");

outtoserver.writeBytes("DATA" + "\n");

//receiving reply 354

System.out.println("Waiting response from server");

replysentence=infromserver.readLine();

System.out.println("Received: "+replysentence);
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 42 of

//Mail body

//send RCPT to

```
System.out.println("Enter your mail data");
```

```
sentence=infromuser.readLine();
```

```
System.out.println("Sending to Server");
```

```
outtoserver.writeBytes(sentence + '\n');
```

```
System.out.println("Sending DOT to Server");
```

```
outtoserver.writeBytes("DOT ." + '\n');
```

//receiving reply

```
System.out.println("Waiting response from server");
```

```
replysentence=infromserver.readLine();
```

```
System.out.println("Received: "+replysentence);
```

```
System.out.println("Sending QUIT to Server");
```

```
outtoserver.writeBytes("QUIT" + '\n');
```

```
}
```

```
}
```

OUTPUT

Server

server side

Connection request accepted

SEND 220

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 43 of

from clientEHLO

PRESS ENTER TO RESPOND

SEND 250 to client

from client: abc@gmail.com

PRESS ENTER TO RESPOND

SEND 250 to client

from client:qwe@gmail.com

PRESS ENTER TO RESPOND

SEND 250 to client

from client :

PRESS ENTER TO RESPOND

SEND 354 to client

Mail body :asdsdasdsadasdsadxcbfghb xcv

from client DOT received :

PRESS ENTER TO RESPOND

SEND 250 to client

from client :

Client

client side

Received220

sending EHLO to mailserver

Waiting response from server

Received :250

Enter the your email address: abc@gmail.com

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 44 of

MAIL FROM:

Waiting response from server

Received: 250

Enter the destination email address :qwe@gmail.com

RCPT TO:

Waiting response from server

Received :250

Sending DATA command

Waiting response from server

Received: 354

Enter your mail data :asdsdasdsadasdsadxcbfghb xcv

Sending to Server

Sending DOT to Server

Waiting response from server

Received: 250

Sending QUIT to Server

RESULT

The program has been executed and tested successfully.

Prepared By:

Approved By:



Exp No:09

Concurrent File Server

AIM :

Write a program to implement concurrent file server to provide file on request by client.

ALGORITHM :

SERVER:

Step 1: Start

Step 2: Server initializes the ServerSocket class and waiting for connection request from client .

Step3: On getting request accept connection with Socket class object.

Step4 :Readfilename from client and check the file.

Step5: Get the content of file and send back to client

Step6: Close connection

Step 7: Stop

CLIENT:

Step 1: Start

Step 2: Client initializes the Socket class and send connection request to server .

Step3: On accepting connection request goto step 4, else goto step 7.

Step4 :Get the file name and send to server

Step5: Read file content from server and display it.

Step6: Close connection

Step 7: Stop



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 46 of

PROGRAM

FileServer

```
import java.io.*;

import java.net.*;

class server

{

    public static int getPID()

    {

        String processName = java.lang.management.ManagementFactory.getRuntimeMXBean().getName();

        return Integer.parseInt(processName.split("@")[0]);

    }

    public static void main(String args[]) throws Exception

    {

        ServerSocket s = new ServerSocket(7000);

        System.out.println("Server ready for connection");

        Socket p = s.accept();

        BufferedReader br1 = new BufferedReader(new InputStreamReader(p.getInputStream()));

        String fname = br1.readLine();

        System.out.println("File" + fname + " received from client");

        DataOutputStream out = new DataOutputStream(p.getOutputStream());

        String str;

        try {

            BufferedReader br2 = new BufferedReader(new FileReader(fname));
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 47 of

```
int n=fserver.getPID();
str=Integer.toString(n);
out.writeBytes(str+'\n');
while((str=br2.readLine()) != null)
{
out.writeBytes(str+'\n');
}
}
catch(Exception e)
{
int n=fserver.getPID();
str=Integer.toString(n);
out.writeBytes(str+'\n');
out.writeBytes("file not found"+"\n");
}
s.close();
p.close();
br1.close();
//br2.close();
}
}
```

FileClient

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 48 of

```
import java.io.*;

import java.net.*;

class client

{

public static void main(String args[]) throws Exception

{

Socket o=new Socket("localhost",7000);

System.out.println("Enter the file name");

BufferedReader br1=new BufferedReader(new InputStreamReader(System.in));

String fname=br1.readLine();

DataOutputStream out=new DataOutputStream(o.getOutputStream());

out.writeBytes(fname+"\n");

BufferedReader br2=new BufferedReader(new InputStreamReader(o.getInputStream()));

String str;

str=br2.readLine();

System.out.println(str);

while((str=br2.readLine()) != null)

{

System.out.println(str);

}

br1.close();

br2.close();

}

}
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 49 of

OUTPUT

Server

Server started
Accepted connection
File **hello.txt** received from client

Client

Enter file name: hello.txt

Hello how are you. I am Fine. Hope you too.

RESULT

The program has been executed and tested successfully.

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 50 of

Prepared By:

Approved By:



Exp No:10

Designing and Configure of Network

AIM :

To configure a network in wired LAN and configure TELNET, SSH, DHCP and DNS and FTP server.

Telnet

Telnet is a protocol used on the Internet or local area networks to provide a bidirectional interactive text-oriented communication facility using a virtual terminal connection. User data is interspersed in-band with Telnet control information in an 8-bit byte oriented data connection over the Transmission Control Protocol (TCP).

Telnet was developed in 1969 beginning with RFC 15, extended in RFC 854, and standardized as Internet Engineering Task Force (IETF) Internet Standard STD 8, one of the first Internet standards. The name stands for "teletype network".

Historically, Telnet provided access to a command-line interface (usually, of an operating system) on a remote host, including most network equipment and operating systems with a configuration utility (including systems based on Windows NT).[clarification needed] However, because of serious security concerns when using Telnet over an open network such as the Internet, its use for this purpose has waned significantly in favor of SSH.

The term telnet is also used to refer to the software that implements the client part of the protocol. Telnet client applications are available for virtually all computer platforms. Telnet is also used as a verb. To telnet means to establish a connection using the Telnet protocol, either with command line client or with a programmatic interface. For example, a common directive might be: "To change your password, telnet into the server, log in and run the passwd command." Most often, a user will be telnetting to a Unix-like server system or a network device (such as a router) and obtaining a login prompt to a command line text interface or a character-based full-screen manager.

```
student@debian:~$ su -  
Password:  
root@debian:~# telnet  
telnet> open 127.0.0.1
```



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 52 of

```
root@debian:~# telnet 127.0.0.1
```

```
Trying 127.0.0.1...
```

```
Connected to 127.0.0.1.
```

```
Escape character is '^['.
```

```
Debian GNU/Linux 9
```

```
debian login: root
```

```
Password:
```

```
Linux debian 4.9.0-4-amd64 #1 SMP Debian 4.9.51-1 (2017-09-28) x86_64
```

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

```
exam1@CGLAB:~$ su -
```

```
Password:
```

```
root@CGLAB:~# ipconfig
```

```
-su: ipconfig: command not found
```

```
root@CGLAB:~# ifconfig
```

```
eth1    Link encap:Ethernet HWaddr 00:21:97:19:99:47
```

```
inet addr:172.16.51.238 Bcast:172.16.63.255 Mask:255.255.240.0
```

```
inet6 addr: fe80::221:97ff:fe19:9947/64 Scope:Link
```

```
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
```

```
RX packets:3607 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:3791 errors:0 dropped:0 overruns:0 carrier:0
```

```
collisions:0 txqueuelen:1000
```

```
RX bytes:3246819 (3.0 MiB) TX bytes:305643 (298.4 KiB)
```

```
Interrupt:20 Base address:0xee00
```

```
lo      Link encap:Local Loopback
```

```
inet addr:127.0.0.1 Mask:255.0.0.0
```

```
inet6 addr: ::1/128 Scope:Host
```

```
UP LOOPBACK RUNNING MTU:16436 Metric:1
```

```
RX packets:960 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:960 errors:0 dropped:0 overruns:0 carrier:0
```

```
collisions:0 txqueuelen:0
```

```
RX bytes:61108 (59.6 KiB) TX bytes:61108 (59.6 KiB)
```



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 53 of

SSH

Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. The best known example application is for remote login to computer systems by users.

SSH provides a secure channel over an unsecured network in a client-server architecture, connecting an SSH client application with an SSH server. Common applications include remote command-line login and remote command execution, but any network service can be secured with SSH. The protocol specification distinguishes between two major versions, referred to as SSH-1 and SSH-2.

The most visible application of the protocol is for access to shell accounts on Unix-like operating systems, but it sees some limited use on Windows as well. In 2015, Microsoft announced that they would include native support for SSH in a future release.

SSH was designed as a replacement for Telnet and for unsecured remote shell protocols such as the Berkeley rlogin, rsh, and rexec protocols. Those protocols send information, notably passwords, in plaintext, rendering them susceptible to interception and disclosure using packet analysis. The encryption used by SSH is intended to provide confidentiality and integrity of data over an unsecured network, such as the Internet, although files leaked by Edward Snowden indicate that the National Security Agency can sometimes decrypt SSH, allowing them to read the contents of SSH sessions.

On 6 July 2017, the government transparency organization WikiLeaks confirmed that the US Central Intelligence Agency had developed tools that can be installed on computers running Microsoft Windows or GNU/Linux operating systems to intercept SSH connections started by SSH clients on the compromised systems.

The SSH protocol (also referred to as Secure Shell) is a method for secure remote login from one computer to another. It provides several alternative options for strong authentication, and it protects the communications security and integrity with strong encryption. It is a secure alternative to the non-protected login protocols (such as telnet, rlogin) and insecure file transfer methods (such as FTP).

```
root@CGLAB:~# ssh exam1@localhost
```

```
The authenticity of host 'localhost (::1)' can't be established.
```

```
ECDSA key fingerprint is 8d:e3:6c:e1:9e:4b:95:7f:94:a2:76:12:10:84:ae:b1.
```

```
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
```

```
exam1@localhost's password:
```

```
Linux CGLAB 3.2.0-4-686-pae #1 SMP Debian 3.2.65-1 i686
```

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 54 of

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

```
exam1@CGLAB:~$ ls
```

```
ajay.c ajay.c~ Desktop Documents Downloads Music Pictures Public Templates Videos
```

```
exam1@CGLAB:~$ cat /etc/host
```

```
host.conf hostname hosts hosts.allow hosts.deny
```

```
exam1@CGLAB:~$ cat /etc/hosts
```

```
127.0.0.1 localhost
```

```
127.0.1.1 CGLAB
```

```
172.16.48.5 CGLAB CGLAB.NIS-CSE-NETWORK
```

The following lines are desirable for IPv6 capable hosts

```
::1 localhost ip6-localhost ip6-loopback
```

```
ff02::1 ip6-allnodes
```

```
ff02::2 ip6-allrouters
```

```
exam1@CGLAB:~$ ping CGLAB
```

```
PING CGLAB (127.0.1.1) 56(84) bytes of data.
```

```
64 bytes from CGLAB (127.0.1.1): icmp_req=1 ttl=64 time=0.043 ms
```

```
64 bytes from CGLAB (127.0.1.1): icmp_req=2 ttl=64 time=0.037 ms
```

```
64 bytes from CGLAB (127.0.1.1): icmp_req=3 ttl=64 time=0.038 ms
```

```
64 bytes from CGLAB (127.0.1.1): icmp_req=4 ttl=64 time=0.035 ms
```

```
64 bytes from CGLAB (127.0.1.1): icmp_req=5 ttl=64 time=0.036 ms
```

```
64 bytes from CGLAB (127.0.1.1): icmp_req=6 ttl=64 time=0.033 ms
```

```
^C
```

```
--- CGLAB ping statistics ---
```

```
6 packets transmitted, 6 received, 0% packet loss, time 4998ms
```

```
rtt min/avg/max/mdev = 0.033/0.037/0.043/0.003 ms
```

FTP

The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.

FTP is built on a client-server model architecture and uses separate control and data connections between the client and the server. FTP users may authenticate themselves with a clear-text sign-in protocol, normally in the form of a username and password, but can connect anonymously if the server is configured to allow it. For secure transmission that protects the username and password, and encrypts the content, FTP is often secured with SSL/TLS (FTPS). SSH File Transfer Protocol (SFTP) is sometimes also used instead; it is technologically different.

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

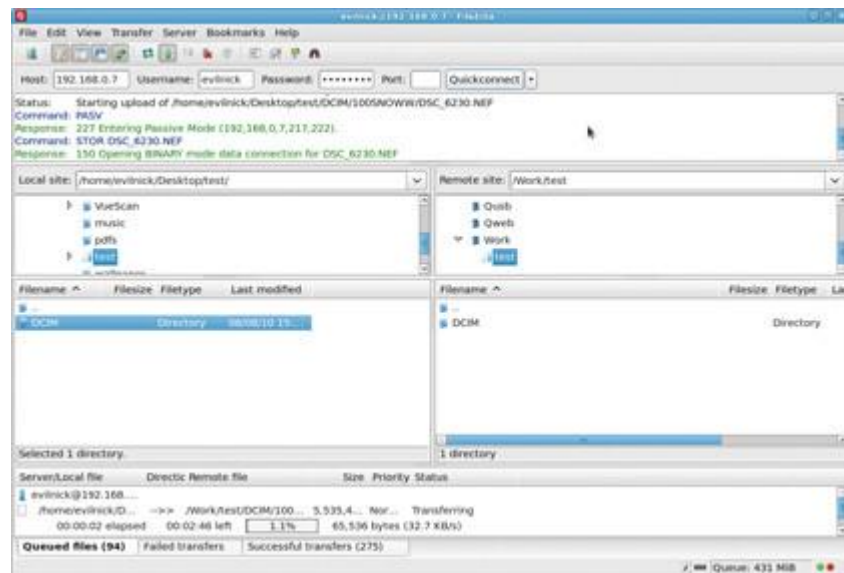
Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 55 of

The first FTP client applications were command-line programs developed before operating systems had graphical user interfaces, and are still shipped with most Windows, Unix, and Linux operating systems. Many FTP clients and automation utilities have since been developed for desktops, servers, mobile devices, and hardware, and FTP has been incorporated into productivity applications, such as web page editors.

FTP is a great way to transfer files to the cloud, and you have some pretty good choices of FTP clients on Linux. However, our favorite is the powerful, cross-platform FileZilla.

FileZilla

Platform: Windows, Mac, Linux



Features

- A highly configurable interface that supports drag-and-drop for easy file transfer
- Supports FTP, FTP over SSL/TLS (FTPS) and SFTP protocols
- Supports resume and transfer of large files over 4GB
- Tabs for opening multiple connections
- Simple bookmarking system for oft-used servers
- Configurable transfer speed limits
- Advanced search feature with filename filtering
- Directory comparison and sync

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 56 of

- A network configuration wizard
- Remote file editing
- Many more advanced features

FileZilla has just about any configuration option you can imagine. If you have to use FTP a lot, FileZilla will let you transfer your files in any way you see fit, as well as search through large servers to find just the file you're looking for. If you know what you're doing, you can even tweak a lot of the more advanced options to optimize the speed of your transfers. At the same time, it's pretty simple to use, at least for how powerful it is. It can be a bit intimidating at first to new users, but with a few clicks of the mouse, one can pare down the interface to something more manageable. If you need more than just the occasional file upload or download, FileZilla is a winner.

Like we said, FileZilla can be a bit intimidating for beginners, especially if you have to root around the preferences. If you only need basic FTP functions, you'd probably be happier with something simpler like gftp, or one of the FTP-enabled desktop file browsers. Also, FileZilla's interface, while configurable, isn't exactly the prettiest (okay, it's ugly as sin), and seems unnecessarily cluttered. It's not a huge issue, but again, if you don't need its advanced features, there's no reason to trudge through its interface when you have other choices.

If FileZilla isn't for you, the next client I'd actually recommend aren't standalone FTP clients at all—but desktop file managers, like Nautilus for GNOME and Konqueror for KDE. Both have some pretty solid FTP features built-in, and provide desktop integration that make dealing with FTP super easy. You deal with them almost exactly like you would with any other file on your computer; you can view them with your default programs, copy them anywhere, and so on. If you aren't a super heavy FTP user, this is a pretty awesome way to get those few files transferred.

If you want a dedicated FTP client but FileZilla is just a bit too much, you might like gftp. It hasn't really had an update since 2008, but it works, and it's very, very simple to use. I still think the desktop integration of Nautilus and Konqueror makes them better for most users, but if you aren't using either of them as your file browser (say, on a low-powered LXDE system), gftp will get the job done.

FireFTP is actually not a separate program, but instead, a Firefox extension. It isn't quite as powerful as the others, but if you just want simple file uploading and downloading through the FTP protocol, it's a really convenient way to do it. Plus, it doesn't require installing another program onto your machine, which is pretty nice.



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 57 of

DHCP

DHCP (Dynamic Host Configuration Protocol) is a network management protocol used to dynamically assign an Internet Protocol (IP) address to any device, or node, on a network so they can communicate using IP. DHCP automates and centrally manages these configurations rather than requiring network administrators to manually assign IP addresses to all network devices. DHCP can be implemented on small local networks as well as large enterprise networks.

DHCP will assign new IP addresses in each location when devices are moved from place to place, which means network administrators do not have to manually initially configure each device with a valid IP address or reconfigure the device with a new IP address if it moves to a new location on the network. Versions of DHCP are available for use in Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6).

DHCP runs at the application layer of the Transmission Control Protocol/IP (TCP/IP) protocol stack to dynamically assign IP addresses to DHCP clients and to allocate TCP/IP configuration information to DHCP clients. This includes subnet mask information, default gateway IP addresses and domain name system (DNS) addresses.

DHCP is a client-server protocol in which servers manage a pool of unique IP addresses, as well as information about client configuration parameters, and assign addresses out of those address pools. DHCP-enabled clients send a request to the DHCP server whenever they connect to a network.

Clients configured with DHCP broadcast a request to the DHCP server and request network configuration information for the local network to which they're attached. A client typically broadcasts a query for this information immediately after booting up. The DHCP server responds to the client request by providing IP configuration information previously specified by a network administrator. This includes a specific IP address as well as for the time period, also called a lease, for which the allocation is valid. When refreshing an assignment, a DHCP client requests the same parameters, but the DHCP server may assign a new IP address based on policies set by administrators.

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 58 of

```
dns-update-style interim;  
ignore client-updates;  
  
subnet 192.168.0.0 netmask 255.255.255.0 {  
# --- default gateway  
    option routers 192.168.0.1;  
    option subnet-mask 255.255.255.0;  
  
    option nis-domain "domain.org";  
    option domain-name "domain.org";  
    option domain-name-servers 192.168.1.1;  
  
    option time-offset -18000; # Eastern Standard Time  
#    option ntp-servers 192.168.1.1;  
#    option netbios-name-servers 192.168.1.1;  
# --- Selects point-to-point node (default is hybrid). Don't change this unless  
# -- you understand Netbios very well  
#    option netbios-node-type 2;  
  
    range dynamic-bootp 192.168.0.128 192.168.0.254;  
    default-lease-time 21600;  
    max-lease-time 43200;  
  
"dhcpd.conf" 31L, 852C                                4.9      Top
```

DNS

The domain name system (DNS) is the way that internet domain names are located and translated into internet protocol (IP) addresses. The domain name system maps the name people use to locate a website to the IP address that a computer uses to locate a website. For example, if someone types TechTarget.com into a web browser, a server behind the scenes will map that name to the IP address 206.19.49.149.

Web browsing and most other internet activity rely on DNS to quickly provide the information necessary to connect users to remote hosts. DNS mapping is distributed throughout the internet in a hierarchy of authority. Access providers and enterprises, as well as governments, universities and other organizations, typically have their own assigned ranges of IP addresses and an assigned domain name; they also typically run DNS servers to manage the mapping of those names to those addresses. Most URLs are built around the domain name of the web server that takes client requests. For example, the URL for this page is <http://searchnetworking.techtarget.com/definition/domain-name-system>.

DNS servers answer questions from both inside and outside their own domains. When a server receives a request from outside the domain for information about a name or address inside the domain, it provides the authoritative answer. When a server receives a request from inside its own domain for information about a name or address outside that domain, it passes the request out to another server -- usually one managed by its internet service provider. If that server does not know the answer or the authoritative source for the answer, it will reach out to the DNS servers for the top-level domain -- e.g., for all of .com or



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 59 of

.edu. Then, it will pass the request down to the authoritative server for the specific domain -- e.g., techtarget.com or stkate.edu; the answer flows back along the same path.

To promote efficiency, servers can cache the answers they receive for a set amount of time. This allows them to respond more quickly the next time a request for the same lookup comes in. For example, if everyone in an office needs to access the same training video on a particular website on the same day, the local DNS server will ordinarily only have to resolve the name once, and then it can serve all the other requests out of its cache. The length of time the record is held -- the time to live -- is configurable; longer values decrease the load on servers, shorter values ensure the most accurate responses.

RESULT

The program to design and configure an network is successfully implemented .



Exp No:11

Familiarisation of NS2

AIM :

Install and understanding of NS2 in Linux platform . Also write a program to implement UDP protocol in NS2.

NS2

ns (from network simulator) is a name for a series of discrete event network simulators, specifically ns-1, and ns-2. All of them are discrete-event computer network simulators, primarily used in research and teaching.

Ns-2 began as a revision of ns-1. In 1995 ns development was supported by DARPA through the VINT project at LBL, Xerox PARC, UCB, and USC/ISI. In 2000, ns-2 development was support through DARPA with SAMAN and through NSF with CONSER, both at USC/ISI, in collaboration with other researchers including ACIRI.

Ns-2 incorporates substantial contributions from third parties, including wireless code from the UCB Daedalus and CMU Monarch projects and Sun Microsystems. For documentation on recent changes, see the version 2 change

Installation of NS2 in Debian

// These commands used for installation of NS2

```
sudo apt-get install ns2
```

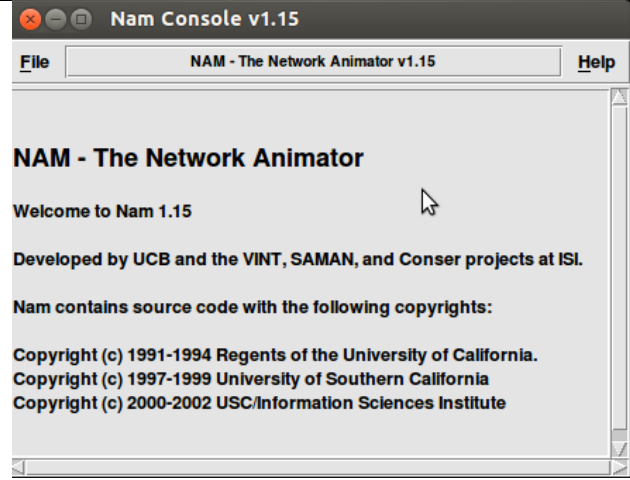
```
sudo apt-get install nam
```

```
mallik@mallik-HP-ProBook-4540s:~$ sudo apt-get install ns2
Reading package lists... Done
Building dependency tree
Reading state information... Done
ns2 is already the newest version (2.35+dfsg-2ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
mallik@mallik-HP-ProBook-4540s:~$
```



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 61 of



Implement UDP protocol in NS2

Algorithm

- Step1 : Start
- Step2 :Creates two nodes – n0, n1.
- Step3 :Add a duplex link between n0 and n1 with the speed of 1Mb/s and 5 ms delay
- Step4 :Create a UDP connection between no and n1 and name it udp
- Step5 :Create a CBR application and attach it to udp
- Step6 :Schedule the time in which the CBR agents should start
- Step7 :Schedule the time for which the simulation should run
- Step8 :Call NAM to animate this topology
- Step9 :Stop

Program

```
set ns [new Simulator]
set nf [open out.nam w]
$ns namtrace-all $nf
proc finish {} {
    global ns nf
    $ns flush-trace
    close $nf
    exec nam out.nam &
    exit 0
}
set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
set udp [new Agent/UDP]
$ns attach-agent $n0 $udp
set null [new Agent/Null]
```

Prepared By:

Approved By:



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**

Department of Computer Science and Engineering

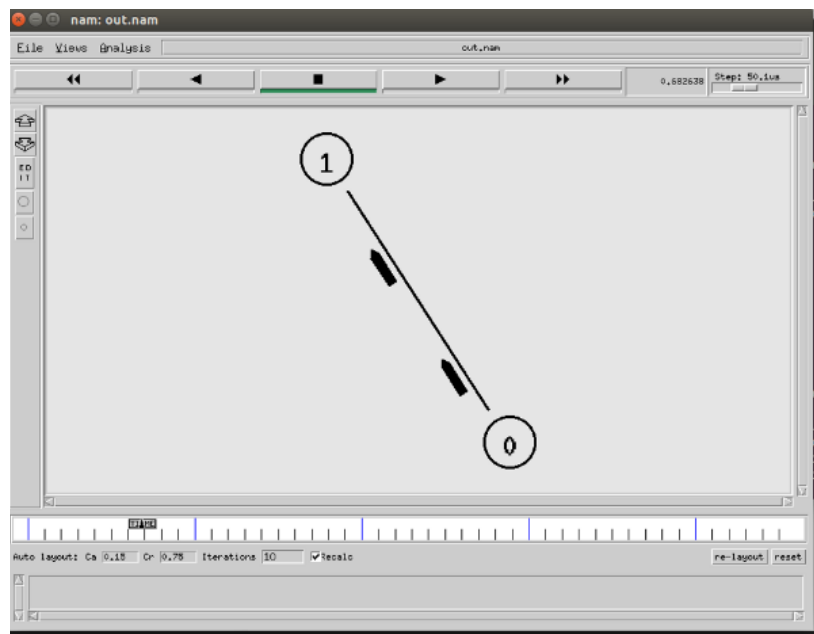
WORK INSTRUCTION

PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 62 of

```
$ns attach-agent $n1 $null
$ns connect $udp $null
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 500
$cbr0 set interval_ 0.005
$cbr0 attach-agent $udp
$ns at 0.5 "$cbr0 start"
$ns at 4.5 "$cbr0 stop"
$ns at 5.0 "finish"
$ns run
```

Output



Result

The program executed successfully and output obtained.

Prepared By:

Approved By:



Exp No:12

Familiarisation of Wireshark

AIM :

Install and familiarize wireshark in Linux platform .

Wireshark

Wireshark is a free and open source packet analyzer. It is used for network troubleshooting, analysis, software and communications protocol development, and education. Originally named Ethereal, the project was renamed Wireshark in May 2006 due to trademark issues.

Wireshark is cross-platform, using the Qt widget toolkit in current releases to implement its user interface, and using pcap to capture packets; it runs on Linux, macOS, BSD, Solaris, some other Unix-like operating systems, and Microsoft Windows. There is also a terminal-based (non-GUI) version called TShark. Wireshark, and the other programs distributed with it such as TShark, are free software, released under the terms of the GNU General Public License.

Wireshark is an open source tool for profiling network traffic and analyzing packets. Such a tool is often referred to as a network analyzer, network protocol analyzer or sniffer.

Wireshark, formerly known as Ethereal, can be used to examine the details of traffic at a variety of levels ranging from connection-level information to the bits that make up a single packet. Packet capture can provide a network administrator with information about individual packets such as transmit time, source, destination, protocol type and header data. This information can be useful for evaluating security events and troubleshooting network security device issues.

Wireshark will typically display information in three panels. The top panel lists frames individually with key data on a single line. Any single frame selected in the top pane is further explained in the tool's middle panel. In this section of the display, Wireshark shows packet details, illustrating how various aspects of the frame can be understood as belonging to the data link layer, network layer, transport layer or application layer. Finally, Wireshark's bottom pane displays the raw frame, with a hexadecimal rendition on the left and the corresponding ASCII values on the right.



**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 64 of

Steps to install Wireshark

Step1: Update the repository:

```
sudo apt-get update
```

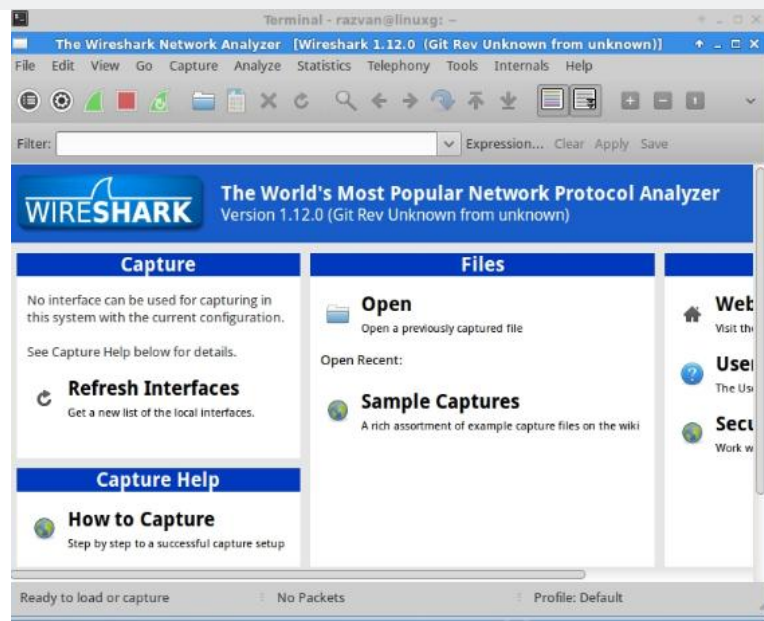
Step 2: Install wireshark

```
sudo apt-get install wireshark
```

```
root@linuxhelp1:~# apt-get install wireshark -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  geoip-database-extra javascript-common libc-ares2 libjs-openlayers libnghttp2-14 libnl-route-3-200 libqstools-pi
  libqt5multimedia5-plugins libqt5multimedia5widgets5 libsmi2ltdb1 libwireshark-data libwireshark8 libwireshark-qt
  libwsutil7 wireshark-common wireshark-qt
Suggested packages:
  apache2 | lighttpd | httpd snmp-mibs-downloader wireshark-doc
The following NEW packages will be installed:
  geoip-database-extra javascript-common libc-ares2 libjs-openlayers libnghttp2-14 libnl-route-3-200 libqstools-pi
  libqt5multimedia5-plugins libqt5multimedia5widgets5 libsmi2ltdb1 libwireshark-data libwireshark8 libwireshark-qt
  libwsutil7 wireshark-common wireshark-qt
0 upgraded, 18 newly installed, 0 to remove and 416 not upgraded.
Need to get 31.1 MB of archives.
After this operation, 136 MB of additional disk space will be used.
Get:1 http://in.archive.ubuntu.com/ubuntu xenial/universe amd64 geoip-database-extra all 20160408-1 [12.1 MB]
Get:2 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu xenial/main amd64 libwireshark-data all 2.2.1+ga6fbd27-1-xenial1 [931 kB]
Get:3 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu xenial/main amd64 libwsutil7 amd64 2.2.1+ga6fbd27-1-xenial1 [91.0 kB]
Get:4 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu xenial/main amd64 libwireshark-qt amd64 2.2.1+ga6fbd27-1-xenial1 [194 kB]
Get:5 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu xenial/main amd64 libwireshark8 amd64 2.2.1+ga6fbd27-1-xenial1 [47.2 kB]
Get:6 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu xenial/main amd64 libwireshark8 amd64 2.2.1+ga6fbd27-1-xenial1 [12.0 MB]
Get:7 http://in.archive.ubuntu.com/ubuntu xenial/main amd64 javascript-common all 11 [6,066 B]
Get:8 http://in.archive.ubuntu.com/ubuntu xenial/universe amd64 libnghttp2-14 amd64 1.7.1-1 [71.1 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu xenial/main amd64 libnl-route-3-200 amd64 3.2.27-1 [124 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu xenial/main amd64 libqt5multimedia5widgets5 amd64 5.5.1-4ubuntu2 [35.9 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu xenial/main amd64 libqstools-pi amd64 5.5.1-4ubuntu2 [65.4 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu xenial/main amd64 libqt5multimedia5-plugins amd64 5.5.1-4ubuntu2 [175 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu xenial/main amd64 libsmi2ltdb1 amd64 0.4.8+dfsg2-11 [101 kB]
Get:14 http://in.archive.ubuntu.com/ubuntu xenial/universe amd64 libjs-openlayers all 2.13.1+ds2-2 [677 kB]
Get:15 http://in.archive.ubuntu.com/ubuntu xenial/main amd64 libc-ares2 amd64 1.10.0-3 [33.9 kB]
Get:16 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu xenial/main amd64 wireshark-common amd64 2.2.1+ga6fbd27-1-xenial1 [221 kB]
Get:17 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu xenial/main amd64 wireshark-qt amd64 2.2.1+ga6fbd27-1-xenial1 [3,347 kB]
Get:18 http://ppa.launchpad.net/wireshark-dev/stable/ubuntu xenial/main amd64 wireshark amd64 2.2.1+ga6fbd27-1-xenial1 [37.2 kB]
Fetched 31.1 MB in 1min 46s (292 kB/s)
Preconfiguring packages ...
Selecting previously unselected package geoip-database-extra.
Reading database ... 95%
```

Step 3: Run wireshark:

```
sudo wireshark
```

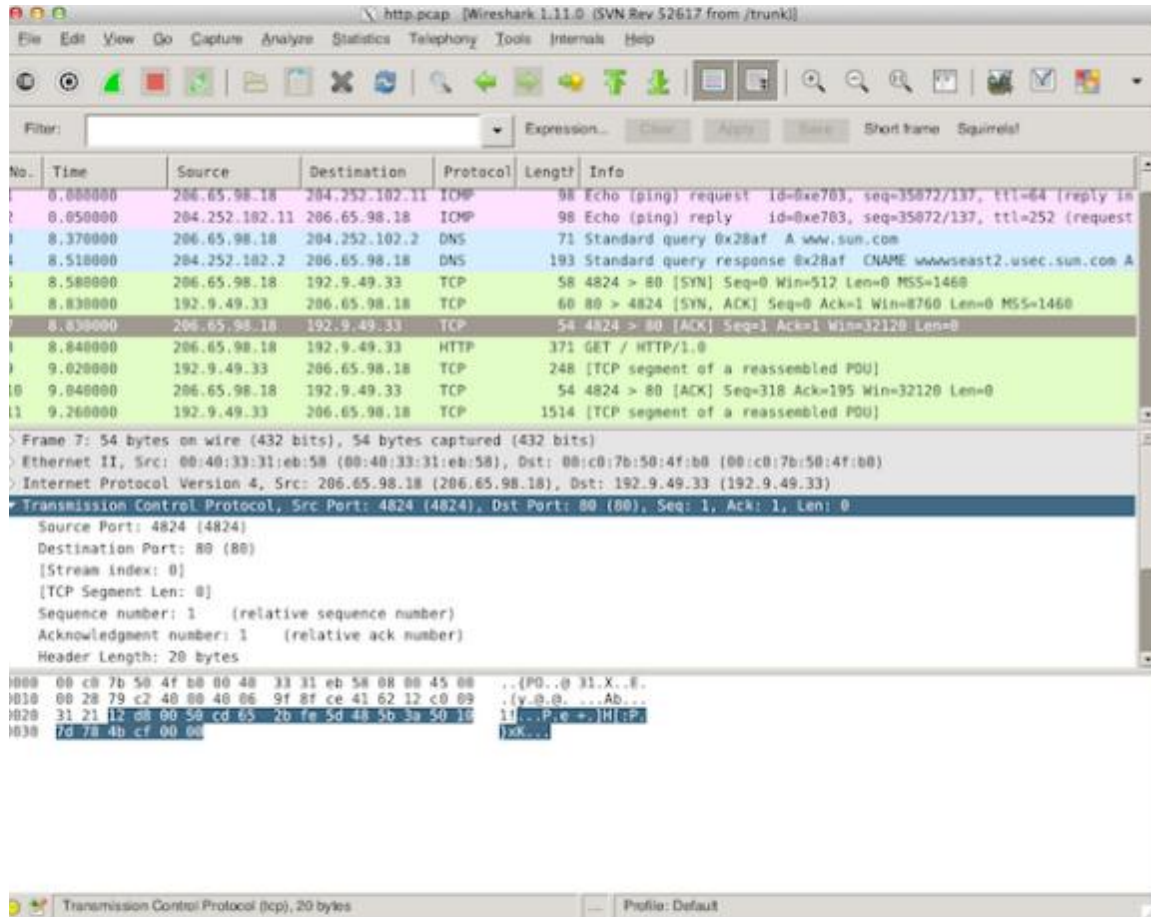




**ST. JOSEPH'S COLLEGE OF ENGINEERING AND
TECHNOLOGY**
Department of Computer Science and Engineering
WORK INSTRUCTION
PREVENTIVE MAINTENANCE FOR COMPUTER LABORATORIES

Doc. No. :
Issue Status :00
Revision Status:00
Date :
Page 65 of

Output



Result

Installation of wireshark is done and packet tracing using wireshark is done successfully.