MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY

DEPARTMENT OF INFORMATION TECHNOLOGY

IT5511 – COMPUTER NETWORKS LABORATORY

LABORATORY RECORD NOTE BOOK

REGISTER NUMBER : 2020506054

NAME : MONICA S

SEMESTER : 5

DEPARTMENT OF INFORMATION TECHNOLOGY

ANNA UNIVERSITY, MIT CAMPUS

CHROMEPET, CHENNAI – 600 044

BONAFIDE CERTIFICATE

Certified that the bonafide record of the practical work done by ~~Mr~~/Ms.Monica S, Register Number: 2020506054 of **Fifth** Semester **B.Tech Information Technology** during the academic period from **August 2022 to December 2022** in the IT5511 – Computer Networks Laboratory.

Date: Course Instructor: Arulalan .V

Examiner:

TABLE OF CONTENTS

Ex. No: 1	STUDY OF BASIC NETWORK COMMANDS
Date: 29/08/2022	

Aim:

To get introduced on basic commands related to Networking on Windows.

Commands and output:

1) Ipconfig:

Displays all current TCP/IP network configuration values and refreshes Dynamic Host Configuration Protocol (DHCP) and Domain Name System (DNS) settings. Used without parameters, **ipconfig** displays the IP address, subnet mask, and default gateway for all adapters.

```
C:\Users\Tester>ipconfig /all

Windows IP Configuration

   Host Name . . . . . . . . . . . . : Tester-PC
   Primary Dns Suffix  . . . . . . . :
   Node Type . . . . . . . . . . . . : Hybrid
   IP Routing Enabled. . . . . . . . : No
   WINS Proxy Enabled. . . . . . . . : No

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Intel(R) PRO/1000 MT Desktop Adapter
   Physical Address. . . . . . . . . : 08-00-27-B2-FB-C6
   DHCP Enabled. . . . . . . . . . . : Yes
   Autoconfiguration Enabled . . . . : Yes
   Link-local IPv6 Address . . . . . : fe80::894:2250:e148:e97%11(Preferred)
   IPv4 Address. . . . . . . . . . . : 10.0.2.15(Preferred)
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Lease Obtained. . . . . . . . . . : Tuesday, October 04, 2022 7:28:57 AM
   Lease Expires . . . . . . . . . . : Wednesday, October 05, 2022 7:29:00 AM
   Default Gateway . . . . . . . . . : 10.0.2.2
   DHCP Server . . . . . . . . . . . : 10.0.2.2
   DHCPv6 IAID . . . . . . . . . . . : 235405351
   DHCPv6 Client DUID. . . . . . . . : 00-01-00-01-2A-79-D0-C7-08-00-27-B2-FB-C6

   DNS Servers . . . . . . . . . . . : 192.168.18.1
   NetBIOS over Tcpip. . . . . . . . : Enabled

Tunnel adapter isatap.{CEF60CB0-4EB7-4F07-9770-008A7EB6FA69}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
   Description . . . . . . . . . . . : Microsoft ISATAP Adapter
   Physical Address. . . . . . . . . : 00-00-00-00-00-00-00-E0
   DHCP Enabled. . . . . . . . . . . : No
   Autoconfiguration Enabled . . . . : Yes
```

2) Ping command:

The **Ping** command is used to check the destination IP address to be reached and record the results. The **ping** command displays whether the destination responded and how long it took to receive a reply. If there is an error in the delivery to the destination, the **ping** command displays an error message.

```
C:\Users\Tester>ping google.com

Pinging google.com [142.250.77.174] with 32 bytes of data:
Reply from 142.250.77.174: bytes=32 time=9ms TTL=127
Reply from 142.250.77.174: bytes=32 time=9ms TTL=127
Reply from 142.250.77.174: bytes=32 time=9ms TTL=127
Reply from 142.250.77.174: bytes=32 time=9ms TTL=127

Ping statistics for 142.250.77.174:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 9ms, Maximum = 9ms, Average = 9ms
```

3) Traceroute command:

Traceroute command (tracert) is a utility designed for displaying the time it takes for a packet of information to travel between a local computer and a destination IP address or domain. After running a traceroute command, the results displayed are a list of the 'hops' that data packets take along their path to the designated IP address or domain.

```
C:\Users\Tester>tracert google.com

Tracing route to google.com [142.250.77.174]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  10.0.2.2
  2     3 ms     1 ms     1 ms  192.168.18.1
  3     6 ms     3 ms     3 ms  100.100.0.1
  4    12 ms     7 ms     5 ms  103.82.209.9
  5     9 ms    10 ms     9 ms  103.82.208.218
  6   122 ms     9 ms     8 ms  103.82.210.50
  7    12 ms    11 ms    10 ms  74.125.242.129
  8     9 ms     9 ms     9 ms  209.85.247.229
  9    12 ms     9 ms     9 ms  maa05s17-in-f14.1e100.net [142.250.77.174]

Trace complete.
```

4) Pathping command:

The **pathping** command is a route tracing tool that combines features of the **ping** and **tracert** commands with additional information that neither of those tools provides. The **pathping** command sends packets to each router on the way to a final destination over a period of time, and then computes results based on the packets returned from each hop. Since the command shows the degree of packet loss at any given router or link, it is easy to determine which routers or links might be causing network problems.

```
C:\Users\Tester>pathping google.com

Tracing route to google.com [142.250.77.174]
over a maximum of 30 hops:
  0  Tester-PC [10.0.2.15]
  1  10.0.2.2
  2  192.168.18.1
  3  100.100.0.1
  4  103.82.209.9
  5  103.82.208.218
  6  103.82.210.50
  7  74.125.242.129
  8  209.85.247.229
  9  maa05s17-in-f14.1e100.net [142.250.77.174]

Computing statistics for 225 seconds...
            Source to Here   This Node/Link
Hop  RTT    Lost/Sent = Pct  Lost/Sent = Pct  Address
  0                                           Tester-PC [10.0.2.15]
                                   0/ 100 =  0%   |
  1    0ms     0/ 100 =  0%      0/ 100 =  0%  10.0.2.2
                                   0/ 100 =  0%   |
  2    4ms     0/ 100 =  0%      0/ 100 =  0%  192.168.18.1
                                   0/ 100 =  0%   |
  3  ---     100/ 100 =100%    100/ 100 =100%  100.100.0.1
                                   0/ 100 =  0%   |
  4    8ms     0/ 100 =  0%      0/ 100 =  0%  103.82.209.9
                                   0/ 100 =  0%   |
  5   15ms     0/ 100 =  0%      0/ 100 =  0%  103.82.208.218
                                   0/ 100 =  0%   |
  6   13ms     0/ 100 =  0%      0/ 100 =  0%  103.82.210.50
                                   0/ 100 =  0%   |
  7   13ms     0/ 100 =  0%      0/ 100 =  0%  74.125.242.129
                                   0/ 100 =  0%   |
  8  ---     100/ 100 =100%    100/ 100 =100%  209.85.247.229
                                   0/ 100 =  0%   |
  9   13ms     0/ 100 =  0%      0/ 100 =  0%  maa05s17-in-f14.1e100.net [142.250
.77.174]

Trace complete.
```

5) Arp command:

The address resolution protocol (arp) is a protocol used by the Internet Protocol (IP) , specifically IPv4, to map IP network addresses to the hardware addresses used by a data link protocol. The protocol operates below the network layer as a part of the interface between the OSI network and OSI link layer. It is used when IPv4 is used over Ethernet.

The term address resolution refers to the process of finding an address of a computer in a network. The address is "resolved" using a protocol in which a piece of information is sent by a client process executing on the local computer to a server process executing on a remote computer. The information received by the server allows the server to uniquely identify the network system for which the address was required and therefore to provide the required address. The address resolution procedure is completed when the client receives a response from the server containing the required address.

```
C:\Users\Tester>arp /a

Interface: 10.0.2.15 --- 0xb
  Internet Address      Physical Address      Type
  10.0.2.2              52-54-00-12-35-02     dynamic
  10.0.2.255            ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static
  255.255.255.255       ff-ff-ff-ff-ff-ff     static
```

6) Hostname command:

Display the hostname of the machine the command is being run on.

```
C:\Users\Tester>hostname
Tester-PC
```

7) Netstat command:

The netstat command is used to display the TCP/IP network protocol statistics and information.

```
C:\Users\Tester>netstat /a

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            Tester-PC:0            LISTENING
  TCP    0.0.0.0:445            Tester-PC:0            LISTENING
  TCP    0.0.0.0:554            Tester-PC:0            LISTENING
  TCP    0.0.0.0:2869           Tester-PC:0            LISTENING
  TCP    0.0.0.0:5357           Tester-PC:0            LISTENING
  TCP    0.0.0.0:10243          Tester-PC:0            LISTENING
  TCP    0.0.0.0:49152          Tester-PC:0            LISTENING
  TCP    0.0.0.0:49153          Tester-PC:0            LISTENING
  TCP    0.0.0.0:49154          Tester-PC:0            LISTENING
  TCP    0.0.0.0:49155          Tester-PC:0            LISTENING
  TCP    0.0.0.0:49157          Tester-PC:0            LISTENING
  TCP    10.0.2.15:139          Tester-PC:0            LISTENING
  TCP    [::]:135               Tester-PC:0            LISTENING
  TCP    [::]:445               Tester-PC:0            LISTENING
  TCP    [::]:554               Tester-PC:0            LISTENING
  TCP    [::]:2869              Tester-PC:0            LISTENING
  TCP    [::]:3587              Tester-PC:0            LISTENING
  TCP    [::]:5357              Tester-PC:0            LISTENING
  TCP    [::]:10243             Tester-PC:0            LISTENING
  TCP    [::]:49152             Tester-PC:0            LISTENING
  TCP    [::]:49153             Tester-PC:0            LISTENING
  TCP    [::]:49154             Tester-PC:0            LISTENING
  TCP    [::]:49155             Tester-PC:0            LISTENING
  TCP    [::]:49157             Tester-PC:0            LISTENING
  UDP    0.0.0.0:3702           *:*
  UDP    0.0.0.0:5004           *:*
  UDP    0.0.0.0:5005           *:*
  UDP    0.0.0.0:5355           *:*
  UDP    0.0.0.0:49152          *:*
  UDP    0.0.0.0:54986          *:*
  UDP    0.0.0.0:63239          *:*
  UDP    10.0.2.15:137          *:*
  UDP    10.0.2.15:138          *:*
  UDP    10.0.2.15:1900         *:*
  UDP    10.0.2.15:63237        *:*
  UDP    127.0.0.1:1900         *:*
  UDP    127.0.0.1:63238        *:*
  UDP    [::]:3540              *:*
  UDP    [::]:3702              *:*
```

8) Route command:

Command to manually configure the routes in the routing table.

```
C:\Users\Tester>route -p PRINT
===========================================================================
Interface List
 11...08 00 27 b2 fb c6 ......Intel(R) PRO/1000 MT Desktop Adapter
  1...........................Software Loopback Interface 1
 12...00 00 00 00 00 00 00 e0 Microsoft ISATAP Adapter
===========================================================================

IPv4 Route Table
===========================================================================
Active Routes:
Network Destination        Netmask          Gateway       Interface  Metric
          0.0.0.0          0.0.0.0         10.0.2.2        10.0.2.15     10
         10.0.2.0    255.255.255.0         On-link         10.0.2.15    266
        10.0.2.15  255.255.255.255         On-link         10.0.2.15    266
      10.0.2.255  255.255.255.255         On-link         10.0.2.15    266
         127.0.0.0        255.0.0.0         On-link         127.0.0.1    306
         127.0.0.1  255.255.255.255         On-link         127.0.0.1    306
  127.255.255.255  255.255.255.255         On-link         127.0.0.1    306
         224.0.0.0        240.0.0.0         On-link         127.0.0.1    306
         224.0.0.0        240.0.0.0         On-link         10.0.2.15    266
  255.255.255.255  255.255.255.255         On-link         127.0.0.1    306
  255.255.255.255  255.255.255.255         On-link         10.0.2.15    266
===========================================================================
Persistent Routes:
  None

IPv6 Route Table
===========================================================================
Active Routes:
 If Metric Network Destination      Gateway
  1    306 ::1/128                  On-link
 11    266 fe80::/64                On-link
 11    266 fe80::894:2250:e148:e97/128
                                    On-link
  1    306 ff00::/8                 On-link
 11    266 ff00::/8                 On-link
===========================================================================
Persistent Routes:
  None
```

9) Nslookup command:

MS-DOS utility that enables a user to look up an IP address of a domain or host on a network.

```
C:\Users\Tester>nslookup google.com
Server:  UnKnown
Address:  192.168.18.1

Non-authoritative answer:
Name:    google.com
Addresses:  2404:6800:4007:817::200e
          142.250.77.142
```

10) Nbstat command:

Displays NetBIOS over TCP/IP (NetBT) protocol statistics, NetBIOS name tables for both the local computer and remote computers, and the NetBIOS name cache. Nbtstat allows a refresh of the NetBIOS name cache and the names registered with Windows Internet Name Service (WINS).

```
C:\Users\Tester>nbtstat /n

Local Area Connection:
Node IpAddress: [10.0.2.15] Scope Id: []

                NetBIOS Local Name Table

       Name               Type         Status
    ---------------------------------------------
    TESTER-PC      <20>  UNIQUE      Registered
    TESTER-PC      <00>  UNIQUE      Registered
    WORKGROUP      <00>  GROUP       Registered
    WORKGROUP      <1E>  GROUP       Registered
    WORKGROUP      <1D>  UNIQUE      Registered
    ..__MSBROWSE__.<01>  GROUP       Registered
```

11) Netsh:

Netsh command is powerful utility to view and configure almost all of network adapters. When passed without arguments it lands inside a interactive shell.

```
C:\Users\Tester>netsh trace show interface

Ethernet adapter Local Area Connection:
    Description:     Intel(R) PRO/1000 MT Desktop Adapter
    Interface GUID:  {CEF60CB0-4EB7-4F07-9770-008A7EB6FA69}
    Interface Index: 11
    Interface Luid:  0x6000006000000

Tunnel adapter isatap.{CEF60CB0-4EB7-4F07-9770-008A7EB6FA69}:
    Description:     Microsoft ISATAP Adapter
    Interface GUID:  {CF155101-2E72-41BF-A449-8A5A2D4B7161}
    Interface Index: 12
    Interface Luid:  0x83000004000000
```

12) Getmac command:

The getmac command provides an easy way to find the MAC address of your device. Prints more than one MAC address device has multiple network adapters.

```
C:\Users\Tester>getmac

Physical Address    Transport Name
=================== ==========================================================
08-00-27-B2-FB-C6   \Device\Tcpip_{CEF60CB0-4EB7-4F07-9770-008A7EB6FA69}
```

13) Net command:

The net command allows user to manage many different aspects of a network and its settings such as network shares, users and print jobs etc..

```
C:\Users\Tester>net config workstation
Computer name                        \\TESTER-PC
Full Computer name                   Tester-PC
User name                            Tester

Workstation active on
        NetBT_Tcpip_{CEF60CB0-4EB7-4F07-9770-008A7EB6FA69} (080027B2FBC6)

Software version                     Windows 7 Professional

Workstation domain                   WORKGROUP
Logon domain                         Tester-PC

COM Open Timeout (sec)               0
COM Send Count (byte)                16
COM Send Timeout (msec)              250
The command completed successfully.
```

14) Systeminfo command:

Systeminfo command collects all the important details of a system including processor details, model of the system and network interfaces and puts them in a readable format.

```
C:\Users\Tester>systeminfo

Host Name:                 TESTER-PC
OS Name:                   Microsoft Windows 7 Professional
OS Version:                6.1.7601 Service Pack 1 Build 7601
OS Manufacturer:           Microsoft Corporation
OS Configuration:          Standalone Workstation
OS Build Type:             Multiprocessor Free
Registered Owner:          Tester
Registered Organization:
Product ID:                00371-868-0000007-85350
Original Install Date:     8/1/2022, 10:50:39 PM
System Boot Time:          10/4/2022, 7:28:56 AM
System Manufacturer:       innotek GmbH
System Model:              VirtualBox
System Type:               x64-based PC
Processor(s):              1 Processor(s) Installed.
                           [01]: Intel64 Family 6 Model 140 Stepping 1 GenuineIn
tel ~3110 Mhz
BIOS Version:              innotek GmbH VirtualBox, 12/1/2006
Windows Directory:         C:\Windows
System Directory:          C:\Windows\system32
Boot Device:               \Device\HarddiskVolume1
System Locale:             en-us;English (United States)
Input Locale:              en-us;English (United States)
Time Zone:                 (UTC+05:30) Chennai, Kolkata, Mumbai, New Delhi
Total Physical Memory:     4,096 MB
Available Physical Memory: 3,314 MB
Virtual Memory: Max Size:  8,189 MB
Virtual Memory: Available: 7,413 MB
Virtual Memory: In Use:    776 MB
Page File Location(s):     C:\pagefile.sys
Domain:                    WORKGROUP
Logon Server:              \\TESTER-PC
Hotfix(s):                 2 Hotfix(s) Installed.
                           [01]: KB2534111
                           [02]: KB976902
Network Card(s):           1 NIC(s) Installed.
                           [01]: Intel(R) PRO/1000 MT Desktop Adapter
                                 Connection Name: Local Area Connection
                                 DHCP Enabled:    Yes
                                 DHCP Server:     10.0.2.2
                                 IP address(es)
                                 [01]: 10.0.2.15
                                 [02]: fe80::894:2250:e148:e97
```

15) Ipconfig /flushdns:

This command is only needed if you're having trouble with your networks DNS configuration.

```
C:\Users\Tester>ipconfig /flushdns
Windows IP Configuration
Successfully flushed the DNS Resolver Cache.
```

Result:

Hence, some basic windows networking commands were tested successfully.

Ex. No: 2	SOCKET PROGRAMMING USING TCP – ECHO AND CHAT
Date: 12/09/2022	

Aim:

To create echo and chat application by implementing sockets using TCP protocol in java programs.

1) Echo server and client:

Algorithm:

Server side:

1. Create the server socket and begin listening.
2. Call the accept() method to get new connections.
3. Create input and output streams for the returned socket.
4. Conduct the conversation based on the agreed protocol.
5. Close the client streams and socket.
6. Go back to step 2 or continue to step 7.
7. Close the server socket.

Client side:

1. Create the client socket connection.
2. Acquire read and write streams to the socket.

Source Code:

Server side:

```
import java.io.*;
import java.net.*;
public class TCPechoserver {
    public static void main(String[] args) throws IOException {
        ServerSocket ss = new ServerSocket(8080);
        Socket s = ss.accept();
        System.out.println("[INFO] : Client connected");
        InputStreamReader socIn = new InputStreamReader(s.getInputStream());
        BufferedReader in = new BufferedReader(socIn);
        String x;
        while (true) {
            x = in.readLine();
            System.out.println("[CLIENT] : " + x);
            if (x.equalsIgnoreCase("terminate")){
                System.out.println("[INFO] : Client terminated connection");
                break;
            }
        }
        ss.close();
    }
```

Client side:

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

```
import java.net.*;
public class TCPechoclient {
    public static void main(String[] args) throws IOException {
        Socket s = new Socket("localhost", 8080);
        System.out.println("[INFO] : Server connection established");
        PrintWriter pr = new PrintWriter(s.getOutputStream(), true);
        InputStreamReader sysIn = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(sysIn);
        String y;
        while (true) {
            System.out.print("[SYSTEM] > Enter message:");
            y = br.readLine();
            if (y.equalsIgnoreCase("terminate")) {
                System.out.println("[INFO] : Connection terminated");
                pr.println("terminate");
                break;
            } else
            pr.println(y);
        }
        s.close();
    }
}
```

Output:

Server:

```
❯  java TCPechoserver
[INFO] : Client connected
[CLIENT] : Hello
[CLIENT] : This is a test run
[CLIENT] : Working fine
[CLIENT] : terminate
[INFO] : Client terminated connection
```

Client:

```
❯  java TCPechoclient
[INFO] : Server connection established
[SYSTEM] > Enter message:Hello
[SYSTEM] > Enter message:This is a test run
[SYSTEM] > Enter message:Working fine
[SYSTEM] > Enter message:terminate
[INFO] : Connection terminated
```

2) Chat server and client:

Algorithm:

Server side:

1. Start
2. Declare the variables for the socket.
3. Specify the family,protocol,ip address and port number.
4. Create a socket using socket() function.
5. Bind IP address and port number.
6. Listen and accept the client's request for connection.
7. Read the client's message.
8. Display the client's message.
9. Close the socket.
10. Stop .

Client side:

1. Start
2. Declare the variables for the socket.
3. Specify the family,protocol,ip address and port number.
4. Create a socket using socket() function.
5. Call the connect() function.
6. Read the input message.
7. Send the input message to the server.
8. Display the server's echo.

Source Code:

Server side:

```
import java.io.*;
import java.net.*;
public class TCPchatserver {
    public static void main(String[] args) throws IOException {
        ServerSocket ss = new ServerSocket(8080);
        Socket s = ss.accept();
        System.out.println("[INFO] : Client connected");
        InputStreamReader socIn = new InputStreamReader(s.getInputStream());
        BufferedReader in = new BufferedReader(socIn);
        InputStreamReader sysIn = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(sysIn);
        PrintWriter pr = new PrintWriter(s.getOutputStream(), true);
        String x, y;
        System.out.println("[INFO] : Enter 'terminate' to end connection");
        while (true) {
            x = in.readLine();
            if (x.equalsIgnoreCase("terminate")) {
                System.out.println("[INFO] : Client terminated connection");
                break;
            }
```

```
            System.out.println("[CLIENT] : " + x);
            System.out.print("[SYSTEM] > Enter message:");
            y = br.readLine();
            if (y.equalsIgnoreCase("terminate")) {
                System.out.print("[INFO] : Connection terminated");
                pr.println("terminate");
                break;
            } else
                pr.println(y);
        }
        ss.close();
    }
}
```

Client side:

```
import java.io.*;
import java.net.*;
public class TCPchatclient {
    public static void main(String[] args) throws IOException {
        Socket s = new Socket("localhost", 8080);
        System.out.println("[INFO] : Server connection established");
        PrintWriter pr = new PrintWriter(s.getOutputStream(), true);
        InputStreamReader socIn = new InputStreamReader(s.getInputStream());
        BufferedReader in = new BufferedReader(socIn);
        InputStreamReader sysIn = new InputStreamReader(System.in);
        BufferedReader br = new BufferedReader(sysIn);
        String x, y;
        System.out.println("[INFO] : Enter 'terminate' to end connection");
        while (true) {
            System.out.print("[SYSTEM] > Enter message:");
            y = br.readLine();
            pr.println(y);
            if (y.equalsIgnoreCase("terminate")) {
                System.out.println("[INFO] : Connection terminated");
                break;
            }
            x = in.readLine();
            if (x.equalsIgnoreCase("terminate")){
                System.out.println("[INFO] : Server terminated connection");
                break;
            }
            System.out.println("[SERVER] : " + x);
        }
        s.close();
    }
}
```

Output:

Server:

```
❯  java TCPchatserver
[INFO] : Client connected
[INFO] : Enter 'terminate' to end connection
[CLIENT] : Hello
[SYSTEM] > Enter message:Hi
[CLIENT] : This is test run for chat
[SYSTEM] > Enter message:It is working
[INFO] : Client terminated connection
```

Client:

```
❯  java TCPchatclient
[INFO] : Server connection established
[INFO] : Enter 'terminate' to end connection
[SYSTEM] > Enter message:Hello
[SERVER] : Hi
[SYSTEM] > Enter message:This is test run for chat
[SERVER] : It is working
[SYSTEM] > Enter message:terminate
[INFO] : Connection terminated
```

Result:

Hence, Echo and Chat application were created and executed successfully with TCP protocol using Java.

Ex. No: 3	SOCKET PROGRAMMING USING UDP – ECHO AND CHAT
Date: 19/09/2022	

Aim:

To create echo and chat application by implementing sockets using UDP protocol in java programs.

1) Echo server and client:

Algorithm:

Server side:

1. Start.
2. Declare the variables for the socket.
3. Specify the family, protocol, IP address and port number.
4. Create a socket using socket() function.
5. Bind the IP address and port number.
6. Listen and acccept the client's request for the connection.
7. Read and display the client's message.
8. Stop.

Client side:

1. Start.
2. Declare the variables for the socket.
3. Specify the family, protocol, IP address and port number.
4. Create a socket using socket() function.
5. Call the connect function.
6. Read the input message.
7. Send the input message to the server.
8. Display the message.
9. Close the socket.
10. Stop.

Source Code:

Server side:

```
import java.io.*;
import java.net.*;
public class UDPechoserver {
    public static void main(String[] args) throws IOException {
        DatagramSocket server = new DatagramSocket(4160);
        byte[] buf = new byte[1024];
        DatagramPacket packet = new DatagramPacket(buf, buf.length);
        System.out.println("[INFO] : Server started");
        while (true) {
            server.receive(packet);
            String str = new String(packet.getData(), 0, packet.getLength());
            if (str.equals("terminate")) {
                System.out.println("[INFO] : Client terminated connection");
```

```
                break;
            }
            System.out.println("[CLIENT] : " + str);
        }
        server.close();
    }
}
```

Client side:

```
import java.io.*;
import java.net.*;
public class UDPechoclient {
    public static void main(String[] args) throws IOException {
        DatagramSocket client = new DatagramSocket();
        InetAddress add = InetAddress.getByName("localhost");
        byte buf[] = new byte[1024];
        BufferedReader dis = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("[INFO] : Client started\n[INFO] : Enter 'terminate' to end connection");
        while (true) {
            System.out.print("[SYSTEM] > Enter message:");
            String str = new String(dis.readLine());
            buf = str.getBytes();
            if (str.equalsIgnoreCase("terminate")) {
                System.out.println("[INFO] : Connection terminated");
                client.send(new DatagramPacket(buf, str.length(), add, 4160));
                break;
            }
            client.send(new DatagramPacket(buf, str.length(), add, 4160));
        }
        client.close();
    }
}
```

Output:

Server side:

```
❯  java UDPechoserver
[INFO] : Server started
[CLIENT] : Hello there
[CLIENT] : This is a test run
[CLIENT] : Using UDP
[INFO] : Client terminated connection
```

Client side:

```
❯  java UDPechoclient
[INFO] : Client started
[INFO] : Enter 'terminate' to end connection
[SYSTEM] > Enter message:Hello there
[SYSTEM] > Enter message:This is a test run
[SYSTEM] > Enter message:Using UDP
[SYSTEM] > Enter message:terminate
[INFO] : Connection terminated
```

2) Chat server and client:

Algorithm:

Server side:

1. Start.
2. Create a server socket.
3. Create a datagram packet with buffer to receive data.
4. Get the data through socket.
5. Convert the byte array to string and print it.
6. Similarly get input from server and send message.
7. Repeat the above steps until the clent sends ‘bye’, if so print terminated…
8. Stop.

Client side:

1. Start.
2. Set IP address and port number.
3. Get input from client.
4. Convert it to byte array.
5. Create a packet using the IP address and port.
6. Create the message.
7. Send the packet through the socket.
8. Create another packet with buffer to receive data from server.
9. Convert the byte array accepted into string array and display it.
10. Repeat the above steps until the user enters ‘bye’, if so break out.
11. Stop

Source code:

Server side:

```
import java.io.*;
import java.net.*;
public class UDPchatserver {
    public static void main(String[] args) throws IOException {
        int c_port = 4150, s_port = 4160;
        DatagramSocket server = new DatagramSocket(s_port);
        byte[] buf = new byte[1024];
        DatagramPacket packet = new DatagramPacket(buf, buf.length);
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        InetAddress ia = InetAddress.getLocalHost();
        System.out.println("[INFO] : Server started\n[INFO] : Enter 'terminate' to end connection");
        while (true) {
            server.receive(packet);
            String str = new String(packet.getData(), 0, packet.getLength());
            if (str.equalsIgnoreCase("terminate")) {
                System.out.println("[INFO] : Client terminated connection");
                break;
            }
```

```
            System.out.println("[CLIENT] : " + str);
            System.out.print("[SYSTEM] > Enter message:");
            buf = br.readLine().getBytes();
            server.send(new DatagramPacket(buf, s.length(), ia, c_port));
        }
        server.close();
    }
}
```

Client side:

```
import java.io.*;
import java.net.*;
public class UDPchatclient {
    public static void main(String[] args) throws IOException {
        int c_port = 4150, s_port = 4160;
        DatagramSocket client = new DatagramSocket(c_port);
        InetAddress add = InetAddress.getByName("localhost");
        byte buf[] = new byte[1024];
        DatagramPacket dp = new DatagramPacket(buf, buf.length);
        BufferedReader dis = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("[INFO] : Client started \n[INFO] : Enter 'terminate' to end connection");
        while (true) {
            System.err.print("[SYSTEM] > Enter message:");
            buf = dis.readLine().getBytes();
            if (str.equalsIgnoreCase("terminate")) {
                System.out.println("[INFO] : Connection Terminated");
                client.send(new DatagramPacket(buf, str.length(), add, s_port));
                break;
            }
            client.send(new DatagramPacket(buf, str.length(), add, s_port));
            client.receive(dp);
            String str2 = new String(dp.getData(), 0, dp.getLength());
            if (str2.equalsIgnoreCase("terminate")){
                System.out.println("[INFO] : Server terminated connection");
                break;
            }
            System.out.println("[SERVER] : " + str2);
        }
        client.close();
    }
}
```

Output:

Server side:

```
❯  java UDPchatserver
[INFO] : Server started
[INFO] : Enter 'terminate' to end connection
[CLIENT] : Hello you
[SYSTEM] > Enter message:Hello back
[CLIENT] : This is a test run
[SYSTEM] > Enter message:Chat client using UDP
[INFO] : Client terminated connection
```

Client side:

```
❯  java UDPchatclient
[INFO] : Client started
[INFO] : Enter 'terminate' to end connection
[SYSTEM] > Enter message:Hello you
[SERVER] : Hello back
[SYSTEM] > Enter message:This is a test run
[SERVER] : Chat client using UDP
[SYSTEM] > Enter message:terminate
[INFO] : Connection Terminated
```

Result:

Hence, Echo and Chat application were created and executed successfully with UDP protocol using Java.

Ex. No: 4	HTTP PROTOCOL – GET AND POST
Date: 24/09/2022	

Aim:

To create programs to implement HTTP protocol using get and post methods in java.

1) HTTP GET server and client:

Algorithm:

Server side:

1. Start
2. Declare the variable for the server socket
3. Create a socket using socket() function
4. Listen and accept the client's request for connection
5. Read the client's message'
6. Display the message
7. Establish get HTTP connection for URL accepted.
8. Set other properties to USER_AGENT
9. Get the response code, if the response code is OK, then perform step 10 else go to step 14
10. Accept the message from the URL present in it by using HTTP connection in a buffer

Client side:

1. Start
2. Create a client socket with post 6789
3. Declare the variable for the socket
4. Accept the URL from the user'
5. Write the URL using the dataoutput stream in the socket to be read by answer
6. Read the message and by server
7. Close the server

Source Code:

Server side:

```
import java.io.*;
import java.net.*;
public class HTTPGetServer {
    private static final String USER_AGENT = "Google Chrome";
    static String sendGET(String GET_URL) throws Exception {
        URL URLObj = new URL(GET_URL);
        HttpURLConnection con = (HttpURLConnection) URLObj.openConnection();
        con.setRequestMethod("GET");
        con.setRequestProperty("User-Agent", USER_AGENT);
        int responseCode = con.getResponseCode();
        System.out.println("[INFO] : Response Code : " + responseCode);
        if (responseCode == HttpURLConnection.HTTP_OK) {
            BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
```

```
            }
            in.close();
            return(response.toString());
        }
        else{
            System.out.println("[ERROR] : GET request failed");
            return (null);
        }
    }
    public static void main(String[] args) throws Exception {
        ServerSocket ss=new ServerSocket(6789);
        while(true){
            Socket conSoc= ss.accept();
            BufferedReader ifc =new BufferedReader(new
InputStreamReader(conSoc.getInputStream()));
            DataOutputStream otc =new DataOutputStream(conSoc.getOutputStream());
            String cs=ifc.readLine()+'\n';
            System.out.println("[INFO] : Requested URL : "+cs);
            String GET_URL = cs;
            otc.writeBytes(sendGET(GET_URL)+'\n');
            System.out.println("[INFO] : GET Request successful");
            break;
        }
        ss.close();
    }
}
```

Client side:

```
import java.io.*;
import java.net.*;
public class HTTPGetClient {
    public static void main(String[] args) throws Exception {
        BufferedReader ifu =new BufferedReader(new InputStreamReader(System.in));
        Socket clientSocket=new Socket("localhost",6789);
        DataOutputStream ots=new DataOutputStream(clientSocket.getOutputStream());
        BufferedReader ifs = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
        System.out.print("[SYSTEM] > Enter URL : ");
        String sentence = ifu.readLine();
        ots.writeBytes(sentence+'\n');
        String ms = ifs.readLine();
        System.out.println("[SERVER] : GET Resoponse:\n"+ms);
        clientSocket.close();
    }
}
```

Output:

Server side:

```
❯  java HTTPGetServer
[INFO] : Requested URL : http://localhost/httpget.php?name=tester

[INFO] : Response Code : 200
[INFO] : GET Request successful
```

Client side:

```
❯  java HTTPGetClient
[SYSTEM] > Enter URL : http://localhost/httpget.php?name=tester
[SERVER] : GET Resoponse:
<title> Test page</title><body> <html>          <h1>Welcome here,tester</h1
>       </html></body>
```

2) HTTP POST server and client:

Algorithm:

Server side:

1. Start
2. Declare the variables for the server socket
3. Create a socket using socket() function
4. Listen and accept the client's request for connection
5. Read the client's message
6. Display the client's message
7. Establish post HTTP connection for the URL accepted
8. Set the other properties such as request method and request property to GET and USER_AGENT respectively.
9. Get the response code, if the response code is OK, do the following else print message(error) and quit.
10. Accept the message from the URL present in it by using the HTTP connection in a buffer
11. Convert the buffered message to string format
12. Print this converted string message
13. Send this message back to the client
14. End

Client side:

1. Start
2. Create a client socket using socket() function'
3. Declare the variables for the client socket
4. Accept the past URL from the user
5. Write the URL using the dataoutputstream in the socket to be read by the server
6. Read the message sent by the server
7. Close the socket
8. End

Source code:

Server side:

```
import java.io.*;
import java.net.*;
public class HTTPPostServer {
    private static final String USER_AGENT = "Google Chrome";
    static String sendPOST(String POST_URL) throws IOException{
        URL obj = new URL(POST_URL);
        HttpURLConnection con = (HttpURLConnection) obj.openConnection();
        con.setRequestMethod("POST");
        con.setRequestProperty("User-Agent", USER_AGENT);
        con.setDoOutput(true);
        int responseCode = con.getResponseCode();
        System.out.println("[INFO] : Response Code : " + responseCode);
        if (responseCode == HttpURLConnection.HTTP_OK) {
            BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();
            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            in.close();
            return(response.toString());
        } else{
            System.out.println("[ERROR] : POST Request failed");
            return(null);
        }
    }
    public static void main(String a[]) throws Exception {
        ServerSocket ss=new ServerSocket(6789);
        while(true) {
            Socket consoc= ss.accept();
            BufferedReader ifc =new BufferedReader(new
InputStreamReader(consoc.getInputStream()));
            DataOutputStream otc =new DataOutputStream(consoc.getOutputStream());
            String ps=ifc.readLine()+'\n';
            System.out.println("[INFO] : Requested URL : "+ps);
            String POST_URL = ps;
            otc.writeBytes(sendPOST(POST_URL)+'\n');
            System.out.println("[INFO] : POST Request successful");
            break;
        }
        ss.close();
    }
}
```

Client side:

```
import java.io.*;
import java.net.*;
public class HTTPPostClient {
    public static void main(String a[]) throws Exception {
```

```
        BufferedReader ifu =new BufferedReader(new InputStreamReader(System.in));
        Socket clientSocket=new Socket("localhost",6789);
        DataOutputStream ots=new DataOutputStream(clientSocket.getOutputStream());
        BufferedReader ifs = new BufferedReader(new
    InputStreamReader(clientSocket.getInputStream()));
        System.out.print("[SYSTEM] > Enter URL : ");
        String sentence =ifu.readLine();
        ots.writeBytes(sentence+'\n');
        String ms=ifs.readLine();
        System.out.println("[SERVER] : POST Response :\n"+ms);
        clientSocket.close();
      }
    }
```

Output:

Server side:

```
❯  java HTTPPostServer
[INFO] : Requested URL : http://localhost/httppost.php

[INFO] : Response Code : 200
[INFO] : POST Request successful
```

Client side:

```
❯  java HTTPPostClient
[SYSTEM] > Enter URL : http://localhost/httppost.php
[SERVER] : POST Response :
<title> Test page</title><body> <html>          <h1>Welcome here,</h1>  </h
tml></body>
```

Result:

Hence, programs implementing HTTP get and post protocol were created and executed successfully.

Ex. No: 5	FILE TRANSFER PROTOCOL
Date: 26/09/2022	

Aim:

To create java program that implements FTP by facilitating file transfer between a server and client.

Algorithm:

Server side:

1. Start the program.
2. Create the server socket.
3. Call the I/O stream.
4. Print the file has been sent.
5. Send the intimation to the client.
6. Stop the program

Client side:

1. Start the program.
2. Create the client packet.
3. After transferring the packet statementis displayed.
4. Stop the program.

Source code:

Server side:

```
import java.net.*;
import java.io.*;
public class FTPServer {
    Socket ss;
    ServerSocket serverSoc;
    DataInputStream din;
    DataOutputStream dout;
    String fileName;
    File fileObject;
    public FTPServer() throws Exception{
        serverSoc = new ServerSocket(6060);
        System.out.println("[INFO] : Server started at "+serverSoc.getInetAddress());
        ss = serverSoc.accept();
        System.out.println("[INFO] : Socket connected to server from "+ss.getInetAddress());
        din = new DataInputStream(ss.getInputStream());
        dout = new DataOutputStream(ss.getOutputStream());
    }
    public void startTransfer() throws Exception{
        int readByte;
        FileInputStream fStream = new FileInputStream(fileName);
        System.out.println("[INFO] : Requested file tranfer to "+ss.getInetAddress()+" starting");
        while((readByte = fStream.read()) != -1) dout.writeUTF(String.valueOf(readByte));
        dout.writeUTF("-1");
        System.out.println("[INFO] : Requested file tranfer to "+ss.getInetAddress()+" completed
successfully");
        fStream.close();
```

```
        }
        public void sendFile() throws Exception{
            fileName = din.readUTF();
            System.out.println("[INFO] : Requested File name: "+fileName);
            fileObject = new File(fileName);
            if(fileObject.exists()){
                dout.writeUTF("1");
                this.startTransfer();
            }
            else{
                System.out.println("[ERROR] : Requested File not found.");
                dout.writeUTF("0");
            }
        }
        public static void main(String[] args) throws Exception{
            FTPServer instance = new FTPServer();
            instance.sendFile();
            instance.ss.close();
            instance.serverSoc.close();
            System.out.println("[INFO] : Socket disconnected and Server closed");
        }
    }
```

Client side:

```
    import java.io.*;
    import java.net.*;
    import java.nio.file.Path;
    import java.nio.file.Paths;
    import java.util.Scanner;;
    public class FTPClient {
        Socket soc;
        DataInputStream din;
        DataOutputStream dout;
        String fileName;
        public FTPClient() throws Exception{
            soc = new Socket("127.0.0.1",6060);
            System.out.println("[INFO] : Socket connected at "+soc.getInetAddress());
            din = new DataInputStream(soc.getInputStream());
            dout = new DataOutputStream(soc.getOutputStream());
        }
        public void getFile() throws Exception{
            String fileIn;
            Scanner in = new Scanner(System.in);
            System.out.print("[SYSTEM] > Enter File name to download: ");
            fileName = in.nextLine();
            dout.writeUTF(fileName);
            Path file = Paths.get(fileName);
            fileName = file.getFileName().toString();
            if(din.readUTF().equals("1")) {
                System.out.println("[INFO] : The Requested file is available at "+soc.getInetAddress());
                File fileObj = new File(fileName);
                if(fileObj.exists()){
```

```
                System.out.println("[INFO] : The Requested file already existing in current folder");
                System.out.print("[SYSTEM] > Do you want to replace the file[Y/N] :");
                String ch = in.nextLine();
                if(ch.equalsIgnoreCase("n")){
                    System.out.print("[SYSTEM] > Do you want to rename the recieved file[Y/N] :");
                    ch = in.nextLine();
                    if(ch.equalsIgnoreCase("n")){
                        System.out.println("[INFO] : File already exists. No further operations done");
                        in.close();
                        return;
                    }
                    System.out.print("[SYSTEM] > Enter new file name :");
                    fileName = in.nextLine();
                }
            }
            System.out.println("[INFO] : File transfer starting from "+soc.getInetAddress());
            FileOutputStream fout = new FileOutputStream(fileName);
            while(!( fileIn = din.readUTF()).equals("-1")) fout.write(Integer.parseInt(fileIn));
            System.out.println("[INFO] : File transfer completed from "+soc.getInetAddress());
            fout.close();
        }
        else{
            System.out.println("[ERROR] : The requested file was not found on the server");
            System.out.println("[ERROR] : Check file name or try again");
        }
        in.close();
    }
    public static void main(String[] args) throws Exception{
        FTPClient instance = new FTPClient();
        instance.getFile();
        instance.soc.close();
        System.out.println("[INFO] : Socket closed");
    }
}
```

Output:

Server side:

```
❯  java FTPServer
[INFO] : Server started at 0.0.0.0/0.0.0.0
[INFO] : Socket connected to server from /127.0.0.1
[INFO] : Requested File name: /home/monish/Document.pdf
[INFO] : Requested file tranfer to /127.0.0.1 starting
[INFO] : Requested file tranfer to /127.0.0.1 completed successfully
[INFO] : Socket disconnected and Server closed
```

Client side:

```
❯  java FTPClient
[INFO] : Socket connected at /127.0.0.1
[SYSTEM] > Enter File name to download: /home/monish/Document.pdf
[INFO] : The Requested file is available at /127.0.0.1
[INFO] : File transfer starting from /127.0.0.1
[INFO] : File transfer completed from /127.0.0.1
[INFO] : Socket closed
```

Result:

Hence, FTP was implemented using java and file transfer from server to client was performed.

Ex. No: 6	DOMAIN NAME SERVER
Date: 10/10/2022	

Aim:

To create java program that implements DNS by facilitating name/IP resolution between a server and client.

Algorithm:

1. Start the program
2. Enter the system name (dn) after the connection with the server is established.
3. Call the subroutine resolver and pass the system name to the resolver.
4. Open the host files in the DNS server.
5. Check the system name with the name stored in the host file until the end of the file.
6. If the name matches, then fetch the corresponding IP address and display the IP address. Go to step8.
7. If the match is not found, then display the message as system is not logged on.
8. Stop the process.

1) Source code:

Server side:

```
import java.net.*;
class DNSServer {
    public static void main(String args[]) throws Exception
        { DatagramSocket server = new DatagramSocket(1309);
        System.out.println("[INFO] : Server started and waiting for requests");
        byte[] sendbyte = new byte[1024];
        byte[] receivebyte = new byte[1024];
        DatagramPacket receiver = new DatagramPacket(receivebyte, receivebyte.length);
        server.receive(receiver);
        String str = new String(receiver.getData());
        String s = str.trim();
        System.out.println("[INFO] : Request recieved to resolve "+s);
        InetAddress addr = receiver.getAddress();
        int port = receiver.getPort();
        String ip[] = { "165.165.80.80", "165.165.79.1" };
        String name[] = { "www.aptitudesource.com", "www.sharifguys.com" };
        DatagramPacket sender = new DatagramPacket("empty".getBytes(), "empty".getBytes().length,
addr, port);
        for (int i = 0; i < ip.length; i++)
            {if (s.equals(ip[i])) {
                sendbyte = name[i].getBytes();
                sender = new DatagramPacket(sendbyte, sendbyte.length, addr, port);
                break;
            } else if (s.equals(name[i]))
                { sendbyte =
                ip[i].getBytes();
```

```
                sender = new DatagramPacket(sendbyte, sendbyte.length, addr, port);
                break;
            }
        }
        server.send(sender);
        server.close();
    }
}
```

Client side:

```
import java.io.*;
import java.net.*;
class DNSClient {
    public static void main(String args[]) throws Exception
        { DatagramSocket client = new DatagramSocket();
        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.print("[SYSTEM] > Enter Domain name or IP address :");
        sendbyte = in.readLine().getBytes();
        DatagramPacket sender = new DatagramPacket(sendbyte, sendbyte.length, addr, 1309);
        client.send(sender);
        DatagramPacket receiver = new DatagramPacket(receivebyte, receivebyte.length);
        client.receive(receiver);
        String s = new String(receiver.getData());
        if(s.isEmpty()) System.out.println("[SERVER] : The requested address or domain doesn't exist");
        else System.out.println("[SERVER] : The input resolves to :" + s.trim());
        client.close();
    }
}
```

Output:

Server side:

```
❯  java DNSServer
[INFO] : Server started and waiting for requests
[INFO] : Request recieved to resolve www.aptitudesource.com
[INFO] : Request recieved to resolve 165.165.80.80
```

Client side:

```
❯  java DNSClient
[SYSTEM] > Enter Domain name or IP address :www.aptitudesource.com
[SERVER] : The input resolves to :165.165.80.80
[SYSTEM] > Enter Domain name or IP address :165.165.80.80
[SERVER] : The input resolves to :www.aptitudesource.com
```

2) Source code:

Server Side:

```
import java.net.*;
class DNSServerExample {
   public static void main(String args[]) throws Exception
      { DatagramSocket server = new DatagramSocket(1309);
      System.out.println("[INFO] : Server started and waiting for requests");
      byte[] sendbyte = new byte[1024];
      byte[] receivebyte = new byte[1024];
      DatagramPacket receiver = new DatagramPacket(receivebyte, receivebyte.length);
      server.receive(receiver);
      String str = new String(receiver.getData());
      String s = str.trim();
      System.out.println("[INFO] : Request recieved to resolve "+s);
      InetAddress addr = receiver.getAddress();
      int port = receiver.getPort();
      InetAddress IP = InetAddress.getByName(s);
      sendbyte = IP.toString().getBytes();
      DatagramPacket sender = new DatagramPacket("empty".getBytes(),
"empty".getBytes().length, addr, port);
      sender = new DatagramPacket(sendbyte, sendbyte.length, addr, port);
      server.send(sender);
      server.close();
   }
}
```

Client Side:

```
import java.io.*;
import java.net.*;
class DNSClientExample {
   public static void main(String args[]) throws Exception
      { DatagramSocket client = new DatagramSocket();
      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.print("[SYSTEM] > Enter Domain name to resolve :");
      sendbyte = in.readLine().getBytes();
      DatagramPacket sender = new DatagramPacket(sendbyte, sendbyte.length, addr, 1309);
      client.send(sender);
      DatagramPacket receiver = new DatagramPacket(receivebyte, receivebyte.length);
      client.receive(receiver);
      String s = new String(receiver.getData());
      if(s.isEmpty()) System.out.println("[SERVER] : The requested domain doesn't exist");
      else System.out.println("[SERVER] : The input resolves to :" + s.trim());
      client.close();
   }
}
```

Output:

Server Side:

```
❯  java DNSServerExample
[INFO] : Server started and waiting for requests
[INFO] : Request recieved to resolve google.com
[INFO] : Request recieved to resolve duckduckgo.com
```

Client Side:

```
❯  java DNSClientExample
[SYSTEM] > Enter Domain name to resolve :google.com
[SERVER] : The input resolves to :google.com/142.250.195.78
[SYSTEM] > Enter Domain name to resolve :duckduckgo.com
[SERVER] : The input resolves to :duckduckgo.com/40.81.94.43
```

Result:

Hence, DNS was implemented using java from server to client was performed.

Ex. No: 7 Date: 07/11/2022	SIMPLE MAIL TRANSFER PROTOCOL

Aim:

To create java program that implements SMTP by facilitating SMTP protocol from a mail server.

Algorithm:

1. Start
2. Set Senders Mail Address and Password.
3. Get System Properties and put host,port,ssl enable to the property object.
4. Ask the Receivers Mail Address.
5. Get the Session object for the processed password authentication purpose.
6. Create a default Mime Message object to receive the message from the user/sender.
7. Ask the User the Subject and the Message to send it to the reciver.
8. Send the Message and Display the Result.
9. STOP

1) Source code:

```
import javax.mail.*;
import javax.mail.internet.*;
import java.util.Properties;
import java.util.Scanner;
import javax.activation.*;
public class SMTPImplementation
   { public static void main(String[]
   args){
      Scanner in = new Scanner(System.in);
      String SenderUser = "username";
      String SenderMail = "username@gmail.com";
      System.out.print("Enter Password for Authentication:");
      String SenderPassword = in.nextLine();
      System.out.print("Enter Destination Address:");
      String ToMail = in.nextLine();
      String ToHost = "smtp.gmail.com";
      Properties SessionProperties = new Properties();
      SessionProperties.put("mail.smtp.auth","true");
      SessionProperties.put("mail.smtp.starttls.enable","true");
      SessionProperties.put("mail.smtp.host",ToHost);
      SessionProperties.put("mail.smtp.port",587);
      Session CurrentSession = Session.getInstance(SessionProperties,
         new javax.mail.Authenticator(){
            protected PasswordAuthentication
               getPasswordAuthentication(){ return new
               PasswordAuthentication(SenderMail, SenderPassword);
            }
         });
```

```
        try{
            Message ThisMessage = new MimeMessage(CurrentSession);
            ThisMessage.setFrom(new InternetAddress(SenderMail));
            ThisMessage.setRecipients(Message.RecipientType.TO, InternetAddress.parse(ToMail));
            System.out.print("Enter Subject for mail:");
            String Subject = in.nextLine();
            System.out.println("Enter Body of the mail:");
            String Body = in.nextLine();
            ThisMessage.setSubject(Subject);
            ThisMessage.setContent(Body,"text/html");
            Transport.send(ThisMessage);
            System.out.println("The Message was sent successfully...");
        }
        catch(Exception
            e){ e.printStackTra
            ce();
        }
        in.close();
    }
}
```

Output:

```
❯ java SMTPImplementation
Enter Password for Authentication:
Enter Destination Address:
Enter Subject for mail:Test Mail
Enter Body of the mail:
This mail was sent using JavaAPI using SMTP
The Message was sent successfully...
```

2) Source code:

```
import javax.mail.*;
import javax.mail.internet.*;
import java.util.*;
import javax.activation.*;
public class SimpleMailTransferProtocol
    {public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        String SenderUser = "ap.monishkumar";
        String SenderMail = "ap.monishkumar@gmail.com";
        System.out.print("Enter Password for Authentication:");
        String SenderPassword = in.nextLine();
        System.out.print("Enter Destination Address:");
        String ToMail = in.nextLine();
        String ToHost = "smtp.gmail.com";
        Properties SessionProperties = new Properties();
        SessionProperties.put("mail.smtp.auth","true");
```

```
        SessionProperties.put("mail.smtp.starttls.enable","true");
        SessionProperties.put("mail.smtp.host",ToHost);
        SessionProperties.put("mail.smtp.port",587);
        Session CurrentSession = Session.getInstance(SessionProperties,
           new javax.mail.Authenticator(){
              protected PasswordAuthentication
                 getPasswordAuthentication(){ return new
                 PasswordAuthentication(SenderMail, SenderPassword);
              }
           });
        try{
           Message ThisMessage = new MimeMessage(CurrentSession);
           ThisMessage.setFrom(new InternetAddress(SenderMail));
           ThisMessage.setRecipients(Message.RecipientType.TO, InternetAddress.parse(ToMail));
           System.out.print("Enter Subject for mail:");
           String Subject = in.nextLine();
           System.out.println("Enter Body of the mail:");
           String Body = in.nextLine();
           System.out.print("Do you want to add attachment?(y/n):");
           String c = in.nextLine();
           if(c.equalsIgnoreCase("y")){ System.out.print("E
              nter FileName to Attach:");String FileName =
              in.nextLine();
              DataSource FileSource = new FileDataSource(FileName);
              MimeBodyPart PartOne = new MimeBodyPart();
              PartOne.setText(Body);
              MimeBodyPart PartTwo = new MimeBodyPart();
              PartTwo.setDataHandler(new DataHandler(FileSource));
              PartTwo.setFileName(FileName);
              Multipart MessageBody = new MimeMultipart();
              MessageBody.addBodyPart(PartOne);
              MessageBody.addBodyPart(PartTwo);
              ThisMessage.setContent(MessageBody);
           }
           else ThisMessage.setContent(Body,"text/html");
           ThisMessage.setSubject(Subject);
           Transport.send(ThisMessage);
           System.out.println("The Message was sent successfully...");
        }
        catch(Exception
           e){ e.printStackTra
           ce();
        }
        in.close();
     }
  }
```

Output:

```
❯  java SimpleMailTransferProtocol
Enter Password for Authentication:[redacted]
Enter Destination Address:ap.monishkumar@gmail.com
Enter Subject for mail:Test Mail With Attachment
Enter Body of the mail:
This Mail was sent by SMTP using java Mail API with attachment
Do you want to add attachment?(y/n):y
Enter FileName to Attach:/home/monish/image.jpg
The Message was sent successfully...
```

Result:

Hence, SMTP was implemented using java from client to mail server.

Ex. No: 8	POST OFFICE PROTOCOL
Date: 07/11/2022	

Aim:

To create java program that implements POP3 from a client to POP3 compatible mail server.

Algorithm:

1. Start
2. Set up properties for Mail Session.
3. Create a Javax.mail Authenticator Object.
4. Create a Mail Session.
5. Get POP3 store provider and connect to the store of the mail.
6. Get folder and open INBOX folder in the store.
7. Retrieve message from folder.
8. Stop listening until user enter “STOP” .
9. STOP

Source code:

```
import javax.mail.*;
import java.util.*;
import javax.activation.*;
import javax.mail.internet.*;
public class POP3Implementation {
   public static void main(String[] args) throws Exception
      {Scanner in = new Scanner(System.in);
      final String UserName = "username@gmail.com";
      System.out.print("Enter Password:");
      String Password = in.nextLine();
      Properties SessionProperties = new Properties();
      SessionProperties.put("mail.pop3.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
      SessionProperties.put("mail.pop3.socketFactory.fallback", "false");
      SessionProperties.put("mail.pop3.socketFactory.port", "995");
      SessionProperties.put("mail.pop3.port", "995");
      SessionProperties.put("mail.pop3.host", "pop.gmail.com");
      SessionProperties.put("mail.pop3.user", UserName);
      SessionProperties.put("mail.store.protocol", "pop3");
      Session ThisSession = Session.getDefaultInstance(SessionProperties,
         new javax.mail.Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication()
               {return new PasswordAuthentication(UserName, Password);
            }
         }
      );
      Store MailServerStore = ThisSession.getStore("pop3");
      MailServerStore.connect("pop.gmail.com",UserName,Password);
      Folder Inbox = MailServerStore.getFolder("INBOX");
```

```
        Inbox.open(Folder.READ_ONLY);
        Message[] MailMessages = Inbox.getMessages();
        for(Message mail : MailMessages){
            mail.writeTo(System.out);
            System.out.print("\nDo you want to continue?(yes/no):");
            String Choice = in.nextLine();
            if(Choice.equalsIgnoreCase("no")){
                System.out.println("Mails read successfully. Exitting...");
                break;
            }
        }
        Inbox.close();
        MailServerStore.close();
        in.close();
    }
}
```

Output:

```
> java POP3Implementation
Enter Password:[redacted]
MIME-Version: 1.0
Received: by 10.64.225.133; Sun, 26 Jan 2014 01:59:55 -0800 (PST)
Date: Sun, 26 Jan 2014 01:59:55 -0800
Message-ID: <CABhGCHJSKoaUZvupvsmLxErwKWv1i9En=96CtKwFkWcC07F8-g@mail.gmail.com>
Subject: Tips for using Gmail
From: Gmail Team <mail-noreply@google.com>
To: Monish A <[redacted]@gmail.com>
Content-Type: multipart/alternative; boundary=001a11c21946398f3804f0dca7aa

--001a11c21946398f3804f0dca7aa
Content-Type: text/plain; charset=windows-1252
Content-Transfer-Encoding: quoted-printable

  Tips for using Gmail                        * Hi Monish *        Tips for
using Gmail                    Chat right from your inbox    Chat with
contacts and start video chats with up to 10 people in Google+
Hangouts<http://www.google.com/+/learnmore/hangouts/?hl=3Den>
.           Bring your email into Gmail    You can import your email from
other webmail to make the transition to Gmail a bit easier. Learn
how.<https://support.google.com/mail/answer/1646407hl=3Den>
       Use Google Drive to send large files    Send huge files in Gmail
<https://support.google.com/mail/answer/2400713?hl=3Den> (up to 10GB)
using Google
Drive <https://drive.google.com/?hl=3Den>. Plus files stored in Drive stay
up-to-date automatically so everyone has the most recent version and can
access them from anywhere.           Save everything    With 10GB of space,
you-92ll never need to delete an email. Just keep everything and easily fin-
d
it later.           Find emails fast    With the power of Google Search
right in your inbox, you can quickly find the important emails you need
with suggestions based on emails, past searches and contacts.           Hap=
py
emailing,  The Gmail Team                =A9 2013 Google Inc. 1600
Amphitheatre Parkway, Mountain View, CA 94043

--001a11c21946398f3804f0dca7aa
Content-Type: text/html; charset=windows-1252
Content-Transfer-Encoding: quoted-printable

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.=
w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns=3D"http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv=3D"content-type" content=3D"text/html;charset=3DUTF-8"=
 />
    <title>Tips for using Gmail</title>
  </head>
  <body marginheight=3D"0" marginwidth=3D"0" text=3D"#444444" link=3D"#1155=
CC">
    <table border=3D"0" cellpadding=3D"0" cellspacing=3D"0" width=3D"100%" =
```

Result:

Hence, POP3 was implemented using java from a client to mail server.

Ex. No: 9	PING COMMAND
Date: 07/11/2022	

Aim:

To write a java program to implement ping command in raw sockets.

Algorithm:

1. Create a RAW socket in the client program.
2. Get the name of the host whose IP address is to resolve using ICMP.
3. Pass this name to the ICMP server through this socket.
4. The server will respond with the IP address of the host.
5. Receive the response and print it.

Source code:

PING SERVER

```
import java.io.*;
import java.net.*;
importjava.util.*;
classPingServer
{
public static void main(String[] args)
{
try
{
ServerSocketss=new ServerSocket(2156);
Socket s=ss.accept();
if(s.isConnected())
System.out.println("Connected ...");
System.out.println("Listening ...");
DataInputStream dis=new DataInputStream(s.getInputStream());
DataOutputStream dos=new DataOutputStream(s.getOutputStream());
int no=0;
String ip="";
if((dis.readUTF()).equals("P"))
{
System.out.println("Getting No. Of Packets ...");
no=dis.readInt();
}
if((dis.readUTF()).equals("A"))
{
System.out.println("Getting the Address ...");
ip=dis.readUTF();
}
Process p=Runtime.getRuntime().exec("ping -c "+no+" "+ip);
System.out.println("Running  ping -c "+no+" "+ip);
BufferedReaderbr=new BufferedReader(new InputStreamReader(p.getInputStream()));
```

```
String ipline=br.readLine();
while(ipline != null )
{
dos.writeUTF(ipline);
ipline=br.readLine();
}
dis.close();
dos.close();
}catch(Exception x)
{
x.printStackTrace();
}
}
}
```

PING CLIENT

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

public class PingClient
{
public static void main(String[] args)
{
try
{
Socket s=new Socket("localhost",2156);
BufferedReaderbr=new BufferedReader(new InputStreamReader(System.in));
if(s.isConnected())
System.out.println("Connected !!");
Scanner in=new Scanner(System.in);
DataInputStream is=new DataInputStream(s.getInputStream());
DataOutputStreamos=new DataOutputStream(s.getOutputStream());
System.out.println("How many Packets You want to send ? ");
int no=in.nextInt();
System.out.println("Address to be pinged :");
String ip=br.readLine();
os.writeUTF("P");
os.writeInt(no);
os.writeUTF("A");
os.writeUTF(ip);
String pingline=is.readUTF();
while(pingline != null )
{
System.out.println(pingline);
pingline=is.readUTF();
}
os.flush();
os.close();
is.close();
}catch(Exception x)
```

```
{
}
}
}
```

Output:

Server Side

```
❯  java PingServer
[INFO]: Server Started and listening for connections
[INFO]: Connected to client at/127.0.0.1
[INFO]: Waiting for inputs
[INFO]: Requested number of packets to ping: 3
[INFO]: Requested address to be ping: 1.1.1.1
[INFO]: Executing ping -c 3 1.1.1.1
[INFO]: Server Socket Closed and exitting
```

Client Side

```
❯ java PingClient
[INFO]: Connected to Server at localhost/127.0.0.1
[SYSTEM]> Enter Number of packets to send: 3
[SYSTEM]> Enter Address to ping: 1.1.1.1
[OUTPUT]: PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
[OUTPUT]: 64 bytes from 1.1.1.1: icmp_seq=1 ttl=53 time=48.5 ms
[OUTPUT]: 64 bytes from 1.1.1.1: icmp_seq=2 ttl=53 time=46.5 ms
[OUTPUT]: 64 bytes from 1.1.1.1: icmp_seq=3 ttl=53 time=80.3 ms
[OUTPUT]:
[OUTPUT]: --- 1.1.1.1 ping statistics ---
[OUTPUT]: 3 packets transmitted, 3 received, 0% packet loss, time 2003ms
[OUTPUT]: rtt min/avg/max/mdev = 46.521/58.449/80.343/15.501 ms
[INFO]: Completed ping process
[INFO]: Closing Socket and streams and exitting
```

Result:

Hence, ping command was implemented using raw sockets in a Java program.

Ex. No: 10	RIP AND OSPF
Date: 14/11/2022	

Aim:

To simulate the working of RIP and OSPF routing protocols in Cisco Packet Tracer.

Procedure:

RIP:

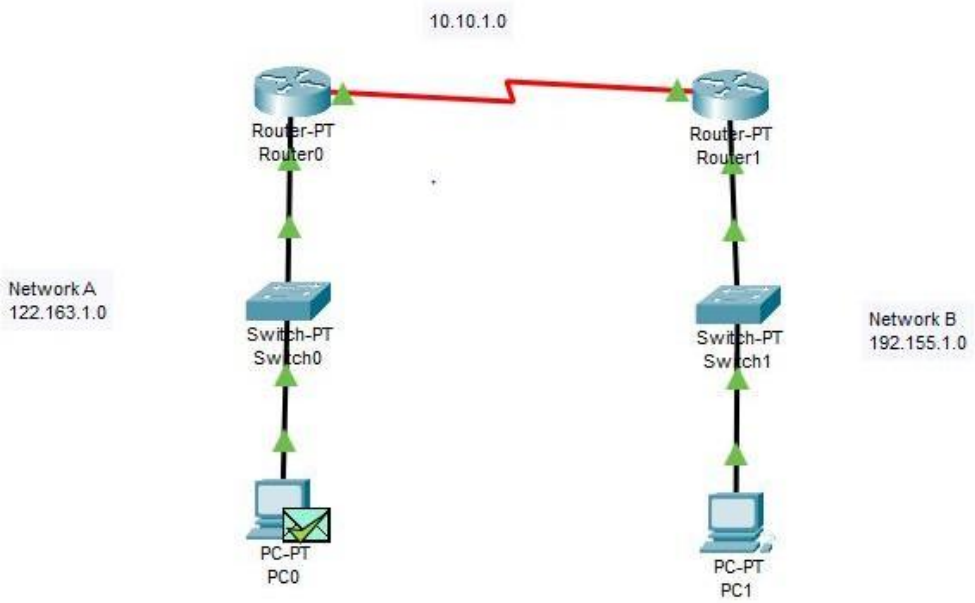
1. Download the Cisco Packet Tracer and Login,then open a new Window.
2. Select two End Devices(2 PC'S) and Drag them from the menu.
3. Select two Network Devices(PT routers) and two Switches (PT Switch) and Drag them from the menu.
4. Connect All of them as shown in the output.
5. Name the two networks as A and B and assign them a particular IP address.
6. Configure the two End Devices using the IP address assigned.
7. Similarly assign the IP address between the two connected Routers.
8. Send the Packet PC0 to Router0 and Similarly between PC1 to Router1.
9. Now try to send the packet between PC0and PC1,if it fails go to step 10.
10. Configure the Routers and add all the IP address to the RIP of Routers and save them.
11. Now send the packets from PC0 to PC1 and from Router0 to Router1.
12. The Packets are being successfully sent.

OSPF:

1. Login Cisco Packet Tracer and Open a new Window.
2. Drag in two PC'S from the End Devices.
3. Drag in Three Routers and Connect all the Devices.
4. Name the networks with appropriate IP address.
5. Configure the PC'S to their respective networks.
6. Connect the Routers in their appropriate Networks.
7. Now use the OSPF to make all the networks visible to each other.
8. See the Serial Routers are Connected it and Configure them.Use the Command line Interpreter to activate the OSPF.
9. Enable and Configure the terminal in the CLI.
10. Now Connect the Router to the Networks it is Connected.
11. The OSPF is activated hence making all the networks visible to each other.
12. Now send a Packet from PC0 to PC1 and Vice versa, the Routers choose the path to deliver the packet.

Output:

RIP:

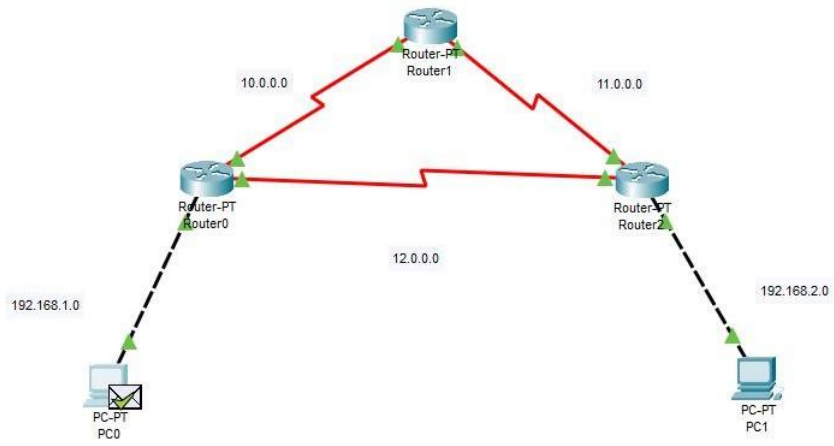


Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device	Type
	0.000	--	PC0	ICMP
	0.001	PC0	Switch0	ICMP
	0.002	Switch0	Router0	ICMP
	0.003	Router0	Router1	ICMP
	0.004	Router1	Switch1	ICMP
	0.005	Switch1	PC1	ICMP
	0.006	PC1	Switch1	ICMP
	0.007	Switch1	Router1	ICMP
	0.008	Router1	Router0	ICMP
	0.009	Router0	Switch0	ICMP
	0.010	Switch0	PC0	ICMP

OSPF:



Simulation Panel

Event List

Vis.	Time(sec)	Last Device	At Device	Type
	2.365	Router2	PC1	CDP
	2.365	Router2	Router1	CDP
	2.365	Router2	Router0	CDP
	2.490	--	Router0	OSPF
	2.491	Router0	PC0	OSPF
	2.496	--	Router0	OSPF
	2.497	Router0	Router1	OSPF
	2.498	--	Router0	OSPF
	2.499	Router0	Router2	OSPF
	2.499	--	Router2	OSPF
	2.500	Router2	Router0	OSPF
	2.501	--	Router1	OSPF
	2.502	Router1	Router2	OSPF
	2.504	--	Router2	OSPF
	2.505	Router2	PC1	OSPF
	2.512	--	Router2	OSPF
	2.513	Router2	Router1	OSPF
	2.532	--	Router1	OSPF
	2.533	Router1	Router0	OSPF
	3.025	--	PC0	ICMP
	3.026	PC0	Router0	ICMP
	3.027	Router0	Router2	ICMP
	3.028	Router2	PC1	ICMP
	3.029	PC1	Router2	ICMP

Result:

Hence Open Shortest Path First (OSPF) is simulated and the packet is successfully sent from the PC0 to PC1 and vice versa

Ex. No: 11	STUDY OF NS-2
Date: 14/11/2022	

Aim:

To study network simulator version 2 (ns2).

Program:

variables.tcl

```
set a 10
set b 15
puts $a
puts "the value of a = $a"
puts "the value of b = $b"
```

Output:

```
pradeshgv@pradeshgv-virtual-machine:~/ns2$ ns variables.tcl
10
the value of a = 10
the value of b = 15
```

expressions.tcl

```
set a 10
set b 15
set c [expr $a + $b]
puts "the value of c = $c"
set d [expr [expr $b - $a] * $c]
puts "the value of d = $d"
```

Output:

```
pradeshgv@pradeshgv-virtual-machine:~/ns2$ ns expressions.tcl
the value of c = 25
the value of d = 125
```

procedure.tcl

```
proc display {} {
puts "this is the testing message" }
display
proc add {x y} {
set z [expr $x + $y]
puts "the value of $x + $y = $z" }
add 10 20
proc print { k } {
for {set i 0} {$i < $k} {incr i} {
puts "node_($i)"
puts "n$i" } }
print 10
```

Output:

```
pradeshgv@pradeshgv-virtual-machine:~/ns2$ ns procedure.tcl
this is the testing message
the value of 10 + 20 = 30
node_(0)
n0
node_(1)
n1
node_(2)
n2
node_(3)
n3
node_(4)
n4
node_(5)
n5
node_(6)
n6
node_(7)
n7
node_(8)
n8
node_(9)
n9
```

control.tcl

```
set x 10

while { $x > 0 } {
puts $x
set x [expr $x - 1]
}
```

Output:

```
pradeshgv@pradeshgv-virtual-machine:~/ns2$ ns control.tcl
10
9
8
7
6
5
4
3
2
1
```

arrays.tcl

```
set matrix(1,1) 140
puts $matrix(1,1)
```

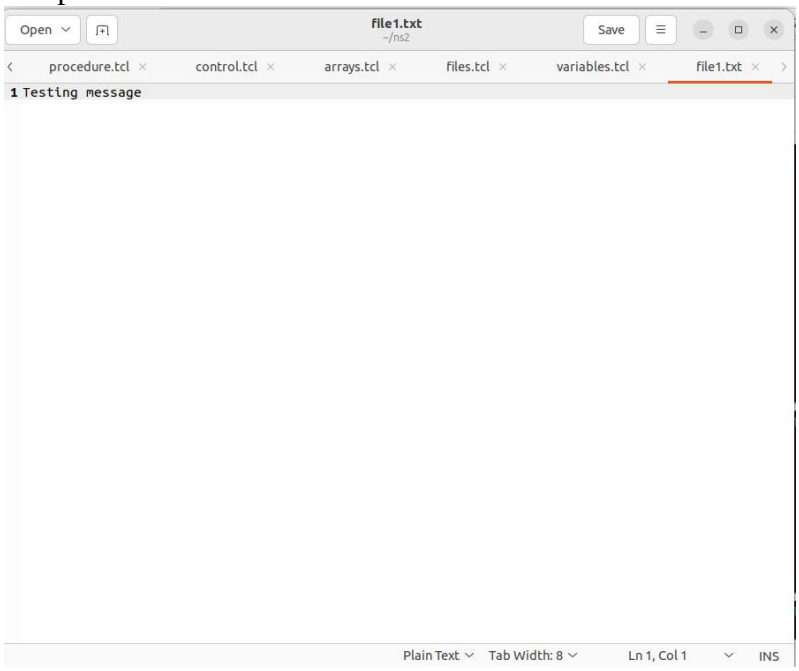
Output:

```
pradeshgv@pradeshgv-virtual-machine:~/ns2$ ns arrays.tcl
140
```

files.tcl

```
set test [open file1.txt w]
puts $test "Testing message"
```

Output:



Result:

Hence study on network simulator version 2 is done.

Ex. No: 12	NETWORK TOPOLOGY USING NS-2
Date: 14/11/2022	

Aim:

To generate a network topological node with suitable link characteristic using network simulator version 2.

Procedure:

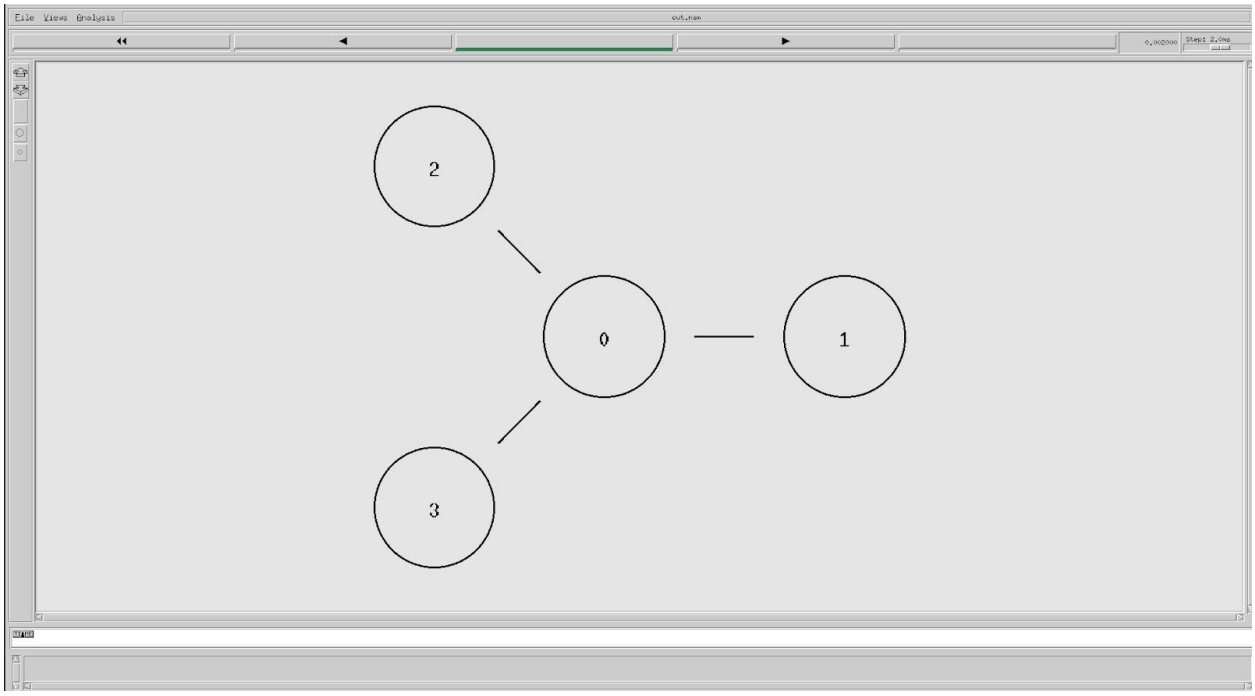
1. Create a simulator object which is an event scheduler using set ns [new Simulator]
2. Turn on tracing by opening the NAM trace file in writing mode and trace all the packets
3. Define a finish procedure to execute the NAM on the trace file
4. Create 6 topological nodes and assign corresponding variables
5. Create link between node n0 n1, n1 n2, n2 n3, n3 n4, n3 n5 which is a duplex link with queue as RED
6. Set queue limit to node n2 and n3 with 10 count
7. Provide orientation to the nodes linked with each other to be displayed on the NAM trace file
8. Create a TCP agent and attach tcp with the node n0
9. Create a traffic sink and attach sink with node n5
10. Then connect both the agents , tcp and the traffic sink
11. Set up a CBR over a TCP Connection by attaching cbr with tcp
12. Schedule events for CBR agent to start and stop
13. Call the finish procedure after 5 seconds of simulation time
14. Print CBR packet size and interval on the terminal
15. Finally run the network simulator using the command $ns run

Program:

```
set ns [new Simulator]
#Create a simulator object
# (Create event scheduler)
#create file for analysis mode
set tr [open out.tr w]
$ns trace-all $tr
#Open trace file
#create file for Animation Mode
set namtr [open out.nam w]
$ns namtrace-all $namtr
#Create Node
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
#Create Link    // wired network
$ns duplex-link $n0 $n1 10Mb 5ms DropTail
$ns duplex-link $n2 $n0 10Mb 5ms DropTail
$ns duplex-link $n3 $n0 10mb 5ms DropTail
#Create Orientation
$ns duplex-link-op $n0 $n1 orient right
$ns duplex-link-op $n0 $n2 orient left-up
```

```
$ns duplex-link-op $n0 $n3 orient left-down
$ns at 10.0 "$ns halt"
$ns run
```

Output:



Result:

Hence a network topological node with suitable link has been created using ns2.

Ex. No: 13	WIRED NETWORK USING TCP
Date: 21/11/2022	

Aim:

To generate a wired network using TCP in ns2 .

Algorithm:

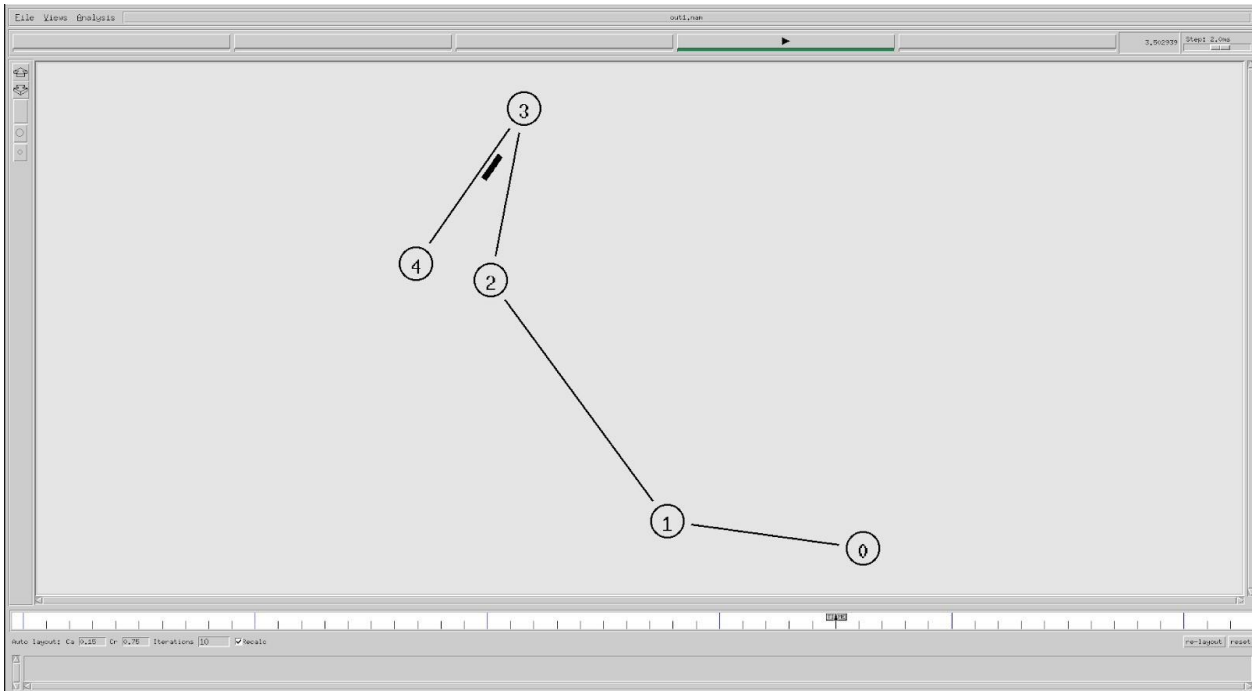
1. START
2. Declare a new Simulator ns and nam nf.
3. Define a procedure finish to execute the namfile.
4. Set the colors for Nodes.
5. Create five Nodes n0,n1,n2,n3,n4 and links with each other.
6. Set the Orientation of the nodes with respect to the nodes with respect to the nodes they are linked with.
7. Set up a TCP Agent and Create Application, FTP
8. Start the Traffic and stop at 1 second and call the finish procedure.
9. Run the Simulator
10. STOP

Program:

```
set ns [new Simulator]
set tracefile [open out1.nam w]
$ns namtrace-all $tracefile

set n0 [$ns node]
set n1 [$ns node]

set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

$ns duplex-link $n0 $n1 6MB 60ms DropTail
$ns duplex-link $n1 $n2 2MB 50ms DropTail
$ns duplex-link $n2 $n3 10MB 20ms DropTail
$ns duplex-link $n3 $n4 1MB 100ms DropTail

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp

set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink

set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP
$ftp set packet-size_ 1000

proc finish {} {
global ns tracefile
```

```
$ns flush-trace
close $tracefile
exec nam out.nam &amp;
exit 0

}

$ns at 0.1 "$ftp start";
$ns at 5.0 "$ftp stop";
$ns at 5.5 "finish";
$ns run
```

Output:



Result:

Hence, wired network using TCP is simulated in ns2.

Ex. No: 14	WIRELESS NETWORK USING UDP
Date: 21/11/2022	

Aim:

To create a wireless topological network simulation using UDP in ns2.

Algorithm:

1. START
2. Initialize Necessary variables for Wireless Connection.
3. Create Tracing and Animation Files.
4. Set Temporary and Load the values of the Variables.
5. Create nodes and mark their Location.
6. Create Channel (i.e) Communication Path for the nodes.
7. Specify mobility codes if anu of the nodes are moving.
8. Set CBR Traffic.
9. Run the simulator.
10. STOP

Program:

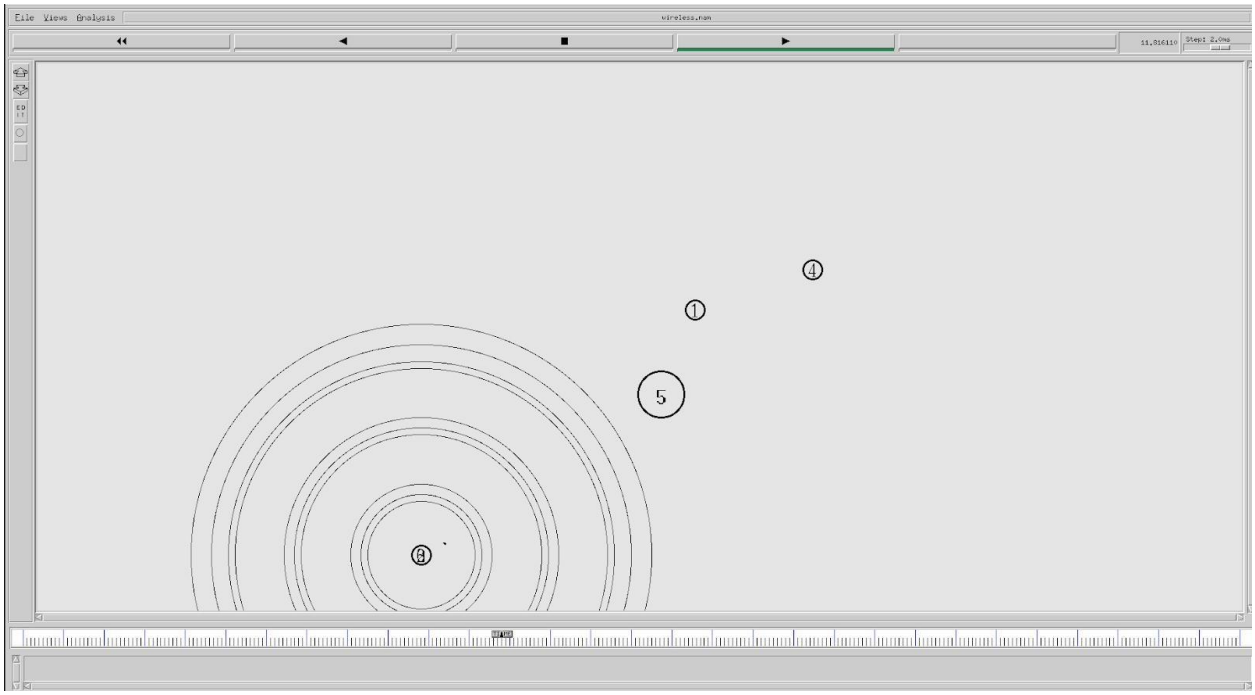
```
set val(chan)          Channel/WirelessChannel    ;#Channel Type
set val(prop)          Propagation/TwoRayGround  ;# radio-propagation model
set val(netif)         Phy/WirelessPhy            ;# network interface type WAVELAN DSSS 2.4GHz
set val(mac)           Mac/802_11                 ;# MAC type

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

set val(ll)            LL                         ;# link layer type
set val(ant)           Antenna/OmniAntenna        ;# antenna model
set val(ifqlen)        50                         ;# max packet in ifq

set val(nn)            6                          ;# number of mobilenodes

set val(rp)            AODV                       ;# routing protocol
set val(x)  500  ;# in metres
set val(y)  500  ;# in metres
#Adhoc OnDemand Distance Vector

#creation of Simulator
set ns [new Simulator]

#creation of Trace and namfile
set tracefile [open wireless.tr w]
$ns trace-all $tracefile

#Creation of Network Animation file
set namfile [open wireless.nam w]
$ns namtrace-all-wireless $namfile $val(x) $val(y)

#create topography
```

```
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

#GOD Creation - General Operations Director
create-god $val(nn)

set channel1 [new $val(chan)]
set channel2 [new $val(chan)]
set channel3 [new $val(chan)]

#configure the node
$ns node-config -adhocRouting $val(rp) \
  -llType $val(ll) \
  -macType $val(mac) \
  -ifqType $val(ifq) \
  -ifqLen $val(ifqlen) \
  -antType $val(ant) \
  -propType $val(prop) \
  -phyType $val(netif) \
  -topoInstance $topo \
  -agentTrace ON \
  -macTrace ON \
  -routerTrace ON \
  -movementTrace ON \
  -channel $channel1

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$n0 random-motion 0
$n1 random-motion 0
$n2 random-motion 0
$n3 random-motion 0
$n4 random-motion 0
$n5 random-motion 0

$ns initial_node_pos $n0 20
$ns initial_node_pos $n1 20
$ns initial_node_pos $n2 20
$ns initial_node_pos $n3 20
$ns initial_node_pos $n4 20
$ns initial_node_pos $n5 50

#initial coordinates of the nodes
$n0 set X_ 10.0
$n0 set Y_ 20.0
$n0 set Z_ 0.0

$n1 set X_ 210.0
```

```
$n1 set Y_ 230.0
$n1 set Z_ 0.0

$n2 set X_ 100.0
$n2 set Y_ 200.0
$n2 set Z_ 0.0

$n3 set X_ 150.0
$n3 set Y_ 230.0
$n3 set Z_ 0.0

$n4 set X_ 430.0
$n4 set Y_ 320.0
$n4 set Z_ 0.0

$n5 set X_ 270.0
$n5 set Y_ 120.0
$n5 set Z_ 0.0
#Dont mention any values above than 500 because in this example, we use X and Y as 500,500

#mobility of the nodes
#At what Time? Which node? Where to? at What Speed?
$ns at 1.0 "$n1 setdest 490.0 340.0 25.0"
$ns at 1.0 "$n4 setdest 300.0 130.0 5.0"
$ns at 1.0 "$n5 setdest 190.0 440.0 15.0"
#the nodes can move any number of times at any location during the simulation (runtime)
$ns at 20.0 "$n5 setdest 100.0 200.0 30.0"

#creation of agents
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 1.0 "$ftp start"

set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n2 $udp
$ns attach-agent $n3 $null
$ns connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$ns at 1.0 "$cbr start"

$ns at 30.0 "finish"

proc finish {} {
 global ns tracefile namfile
 $ns flush-trace
 close $tracefile
```

```
 close $namfile
 exit 0
}

puts "Starting Simulation"
$ns run
```

Output:



Result:

Hence wireless network is simulated using UDP in ns2.

Ex. No: 15	PERFORMANCE ANALYSIS USING XGRAPH
Date: 21/11/2022	

Aim:

To analyze performance of TCP and UDP using Xgraph **.**

Algorithm:

1. START
2. Create a new simulator ns and open out.nam as nf.
3. Create five nodes and establish Link with appropriate Nodes.
4. Declare a Procedure attach-expoo-traffic and define it to simulate a UDP traffic.
5. Create sink agents and attach it with nodes.
6. Declare and Define Finish Procedure to be called at the End.
7. Decalre a Procedure record to Track the Bytes that are received by the traffic Links.
8. Calculate the bandwidth and write it to the file.
9. Reset the Bytes value on the Traffic sinks.
10. Reschedule the Procedures by calling each defined functions in the appropriate order.
11. Run the Simulator.
12. STOP

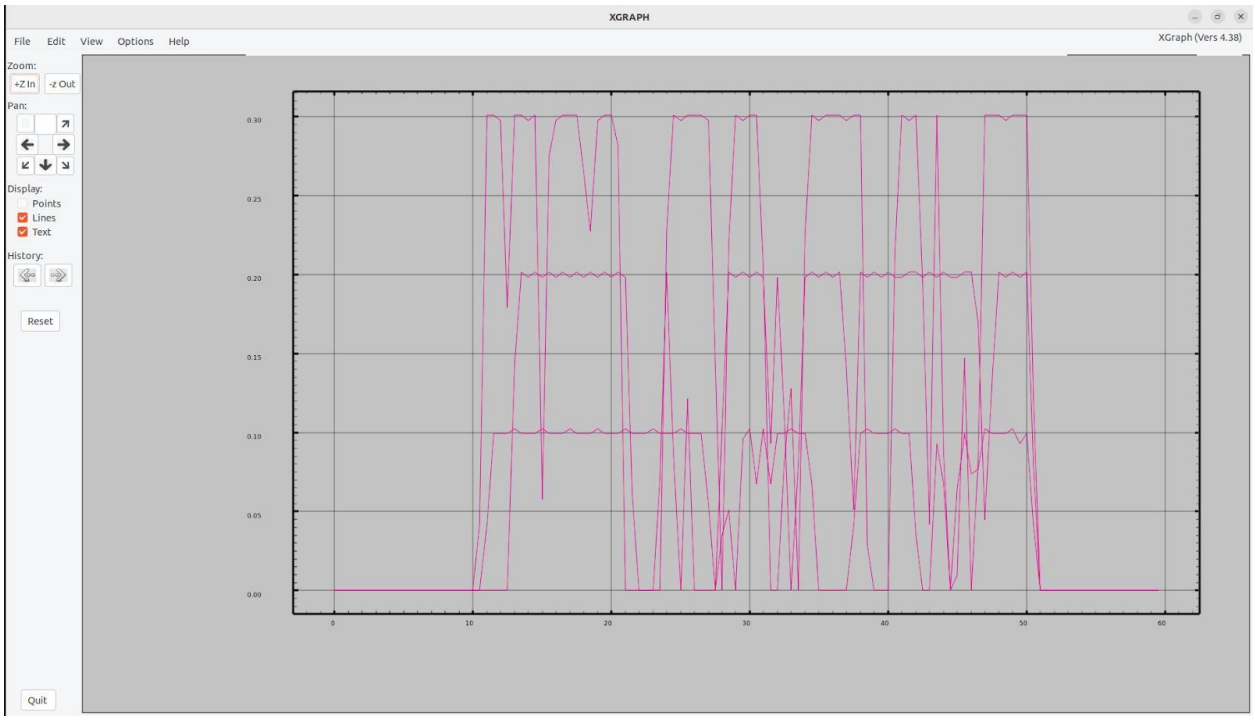
Program:

```
set ns [new Simulator]

set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]

$ns duplex-link $n0 $n3 1Mb 100ms DropTail
$ns duplex-link $n1 $n3 1Mb 100ms DropTail
$ns duplex-link $n2 $n3 1Mb 100ms DropTail
$ns duplex-link $n3 $n4 1Mb 100ms DropTail

proc attach-expoo-traffic { node sink size burst idle rate } {
	#Get an instance of the simulator
	set ns [Simulator instance]

	#Create a UDP agent and attach it to the node
	set source [new Agent/UDP]
	$ns attach-agent $node $source

	#Create an Expoo traffic agent and set its configuration parameters
	set traffic [new Application/Traffic/Exponential]
	$traffic set packetSize_ $size
	$traffic set burst_time_ $burst
	$traffic set idle_time_ $idle
	$traffic set rate_ $rate
```

```
	# Attach traffic source to the traffic generator
	$traffic attach-agent $source
	#Connect the source and the sink
	$ns connect $source $sink
	return $traffic
}

set sink0 [new Agent/LossMonitor]
set sink1 [new Agent/LossMonitor]
set sink2 [new Agent/LossMonitor]
$ns attach-agent $n4 $sink0
$ns attach-agent $n4 $sink1
$ns attach-agent $n4 $sink2

set source0 [attach-expoo-traffic $n0 $sink0 200 2s 1s 100k]
set source1 [attach-expoo-traffic $n1 $sink1 200 2s 1s 200k]
set source2 [attach-expoo-traffic $n2 $sink2 200 2s 1s 300k]

set f0 [open out0.tr w]
set f1 [open out1.tr w]
set f2 [open out2.tr w]

proc finish {} {
	global f0 f1 f2
	#Close the output files
	close $f0
	close $f1
	close $f2
	#Call xgraph to display the results

	exit 0
}

proc record {} {
	global sink0 sink1 sink2 f0 f1 f2
	#Get an instance of the simulator
	set ns [Simulator instance]
	#Set the time after which the procedure should be called again
	set time 0.5
	#How many bytes have been received by the traffic sinks?
	set bw0 [$sink0 set bytes_]
	set bw1 [$sink1 set bytes_]
	set bw2 [$sink2 set bytes_]
	#Get the current time
	set now [$ns now]
	#Calculate the bandwidth (in MBit/s) and write it to the files
	puts $f0 "$now [expr $bw0/$time*8/1000000]"
	puts $f1 "$now [expr $bw1/$time*8/1000000]"
	puts $f2 "$now [expr $bw2/$time*8/1000000]"
	#Reset the bytes_ values on the traffic sinks
	$sink0 set bytes_ 0
	$sink1 set bytes_ 0
	$sink2 set bytes_ 0
```

```
	#Re-schedule the procedure
	$ns at [expr $now+$time] "record"
}

$ns at 0.0 "record"
$ns at 10.0 "$source0 start"
$ns at 10.0 "$source1 start"
$ns at 10.0 "$source2 start"
$ns at 50.0 "$source0 stop"
$ns at 50.0 "$source1 stop"
$ns at 50.0 "$source2 stop"
$ns at 60.0 "finish"

$ns run
```

Output:



Result:

Hence performance analysis is executed using xgraph in NS-2.

Ex. No: 16	SLAAC AND DHCP
Date: 21/11/2022	

Aim:

To study about SLAAC and DHCP.

Procedure:

SLAAC:

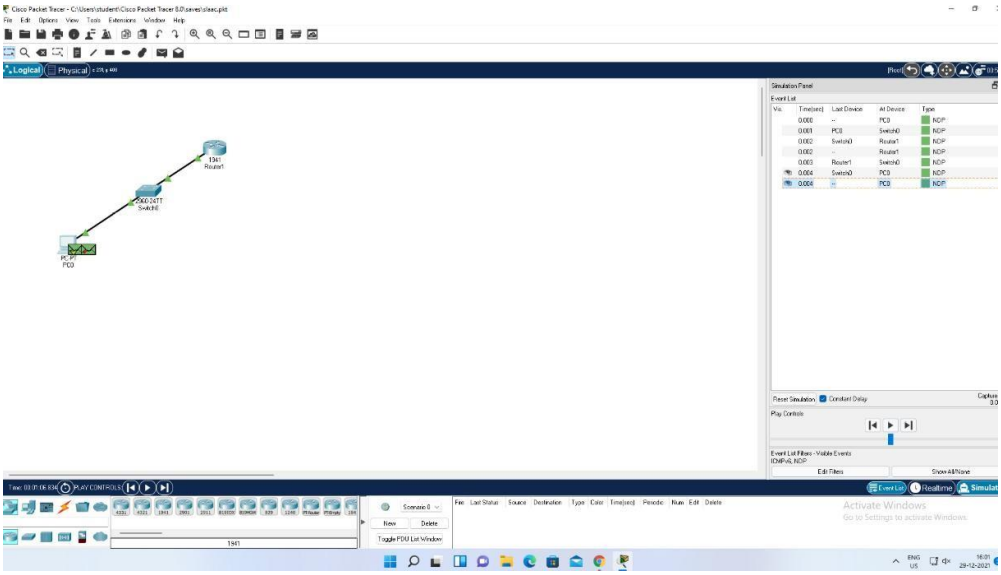
1. START
2. Login into the Cisco Packet Tracer and open a New Window.
3. Drag and Drop Router from the menu bar.
4. Drag and Drop two switches from the menu bar.
5. Drag and Drop the End Devices from the menu bar such as a PC/laptop.
6. Connect all the end devices with the switch .
7. Configure the router for the two switches and the switches also.
8. Set up the Ipv6 address for the End systems connected.
9. Stop

DHCP:

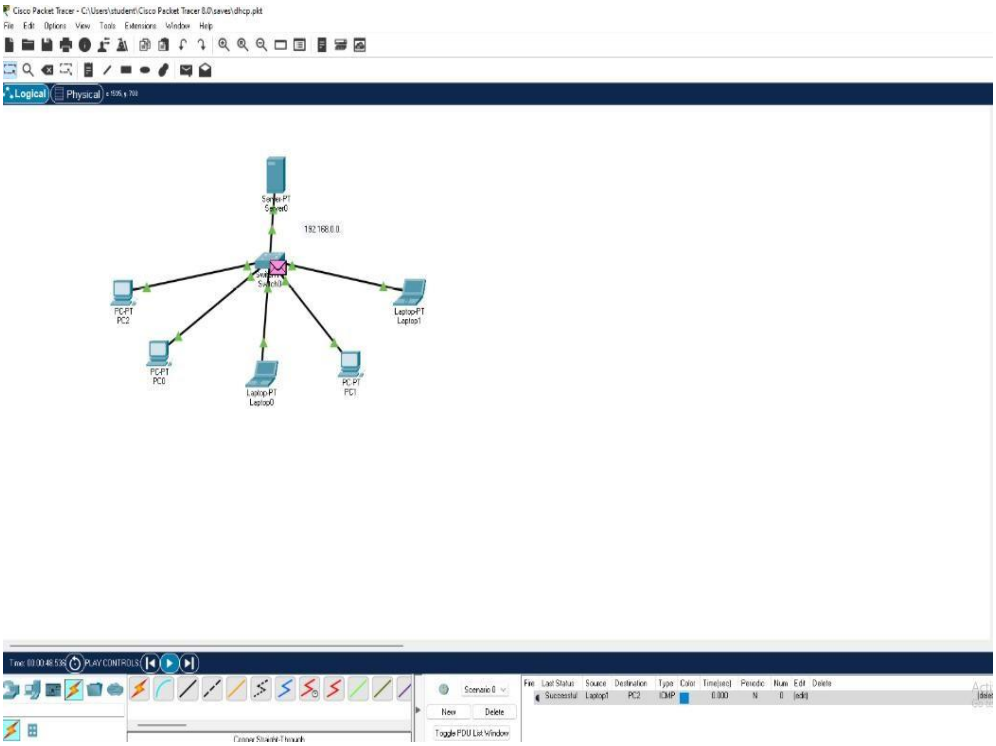
1. Start
2. Login into Cisco Packet Tracer and open a new Window.
3. Drag and Drop a Router from the menu bar.
4. Drag and Drop two Switches from the menu bar.
5. Drag end devices, three laptops and one PC System from the menu bar.
6. Connect the end devices to the switches using a fast ethernet cable .
7. Connect the Switches to the Router using the Fast Ethernet cable.
8. Using the Command line for the Router and configure the router first for the two fast ethernet cable connections.
9. Configure the I P address of end devices and set the configuration to DHCP .
10. DHCP is thus Configured.
11. Stop

Output:

SLAAC:



DHCP:



Result:

Hence SLAAC and DHCP is studied, and packets are sent successfully.

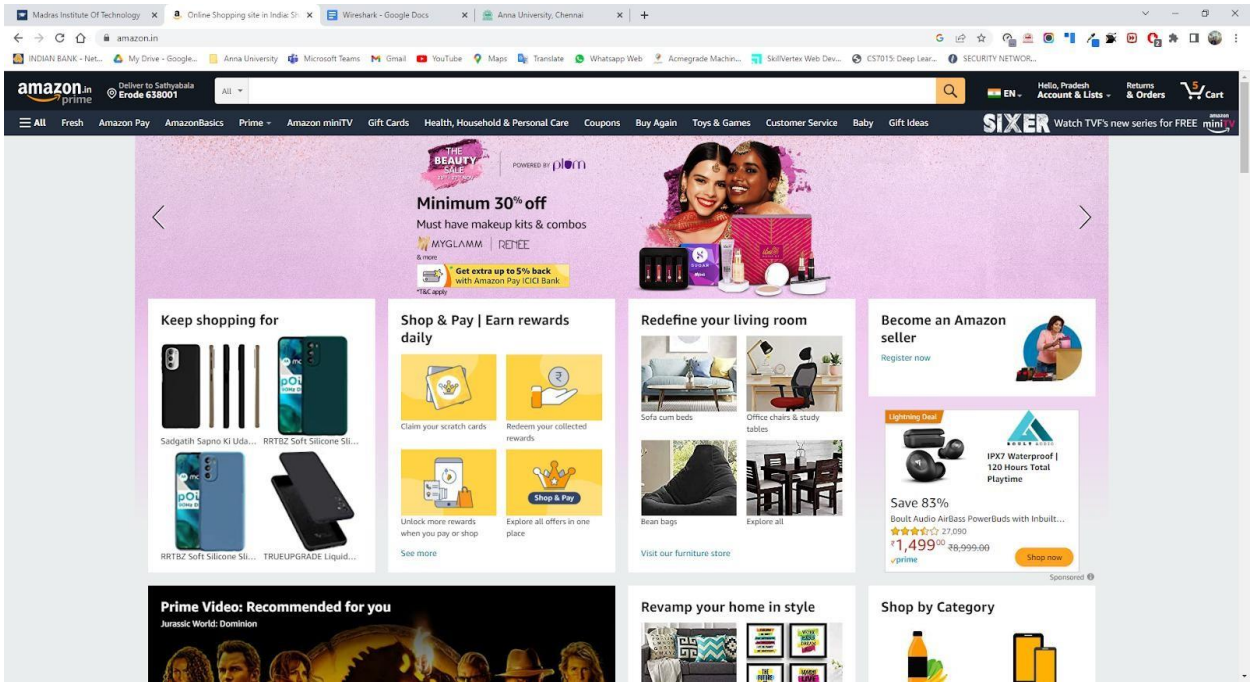
Ex. No: 17	NETWORK ANALYSIS USING WIRESHARK
Date: 21/11/2022	

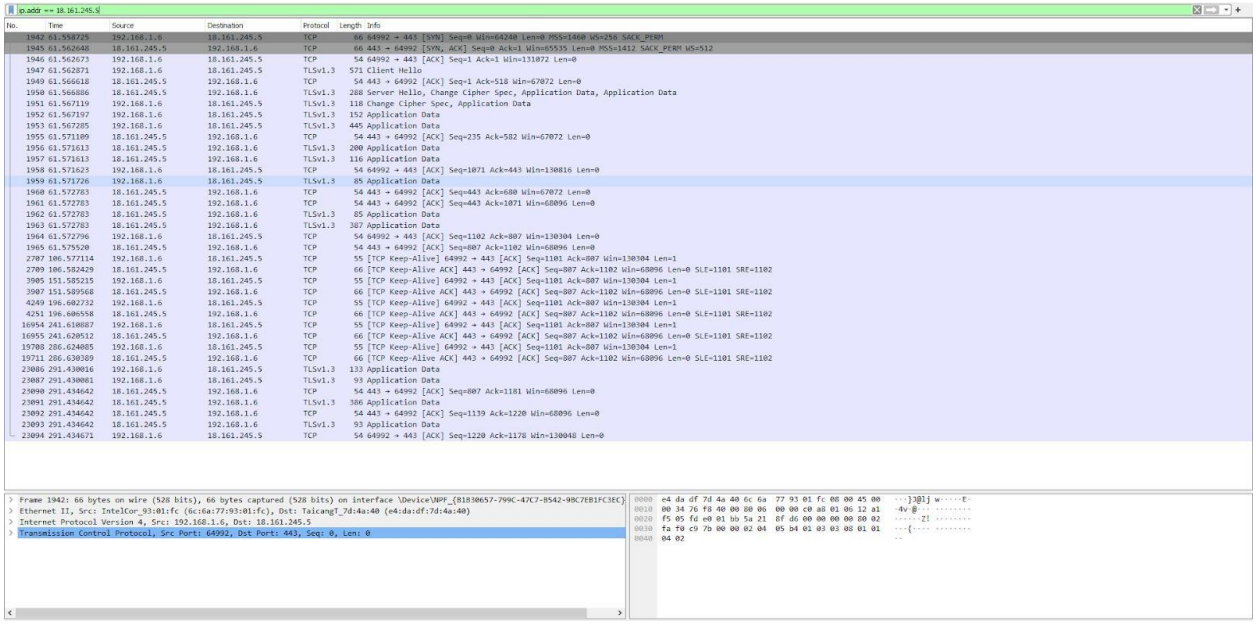
Aim:

To monitor different websites and their network protocols using Wireshark (network analysing tool)

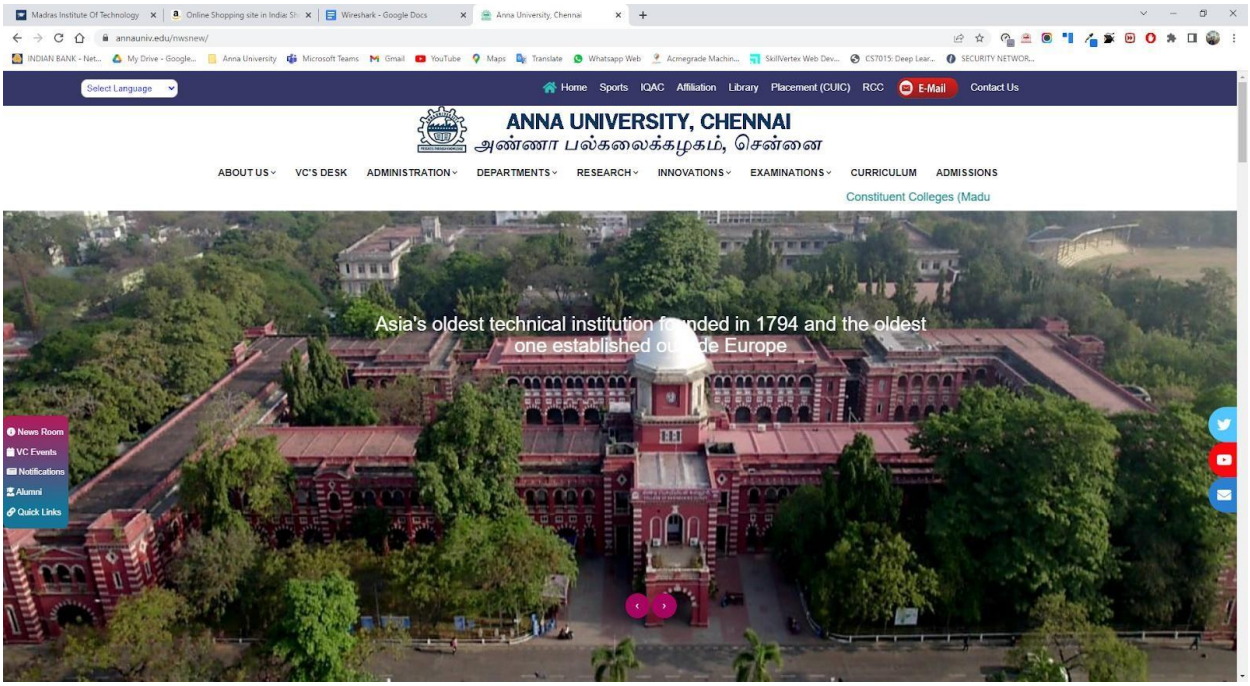
Output Screenshots:

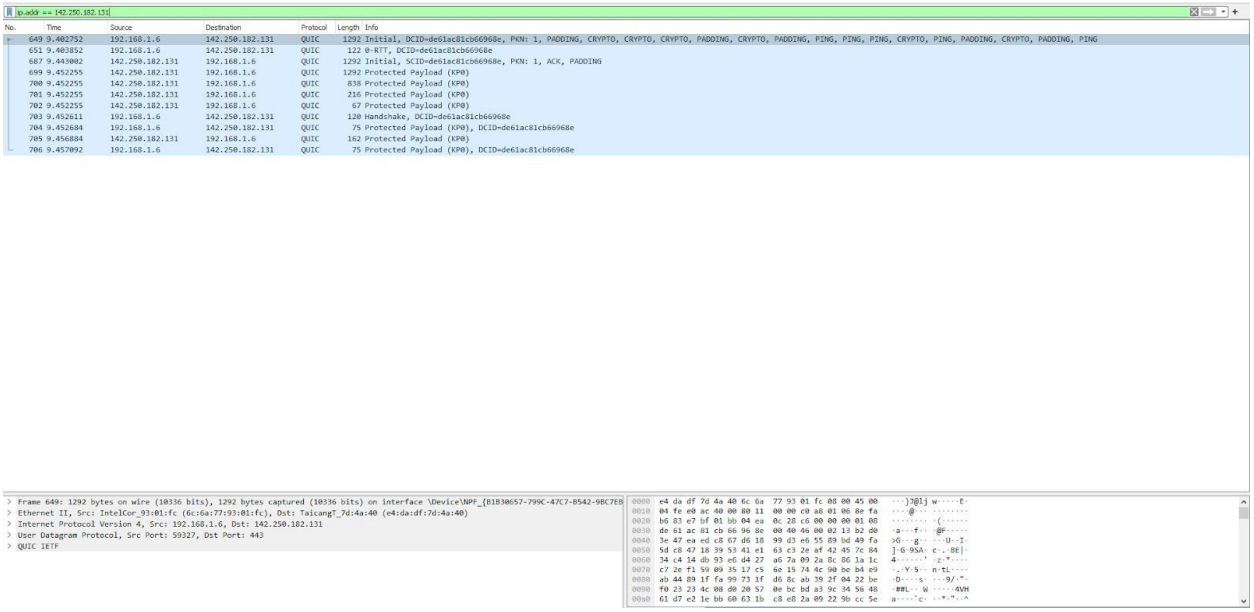
https://www.amazon.in/



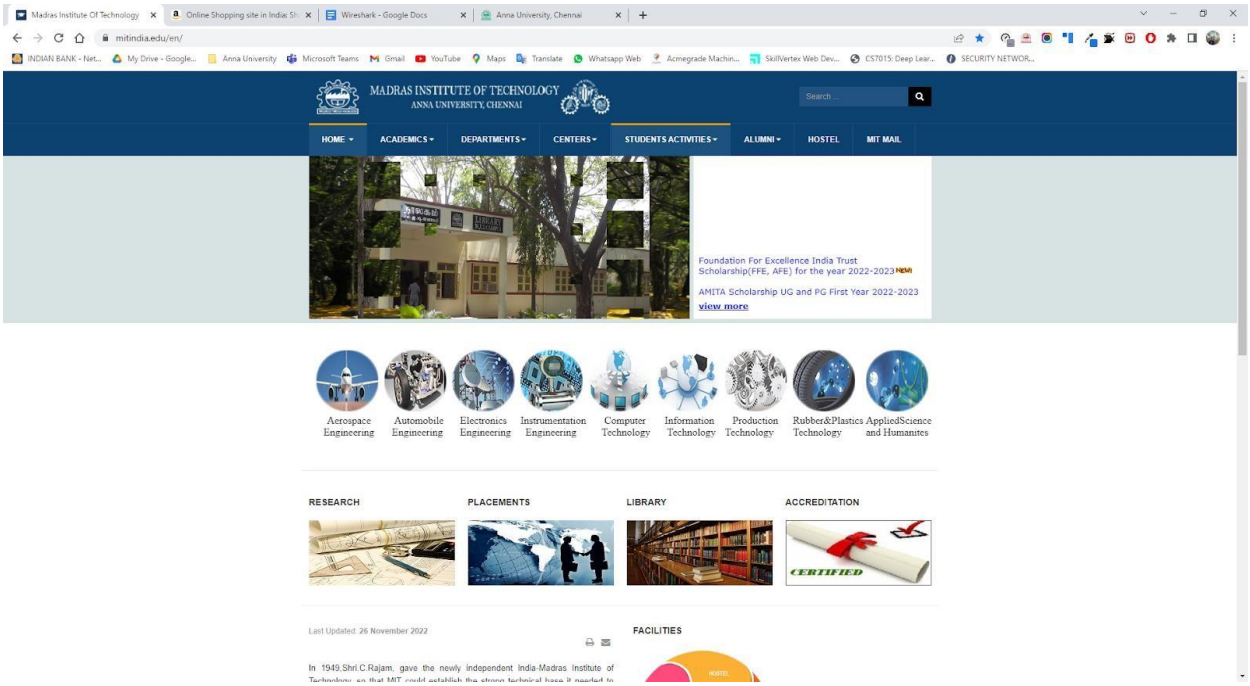


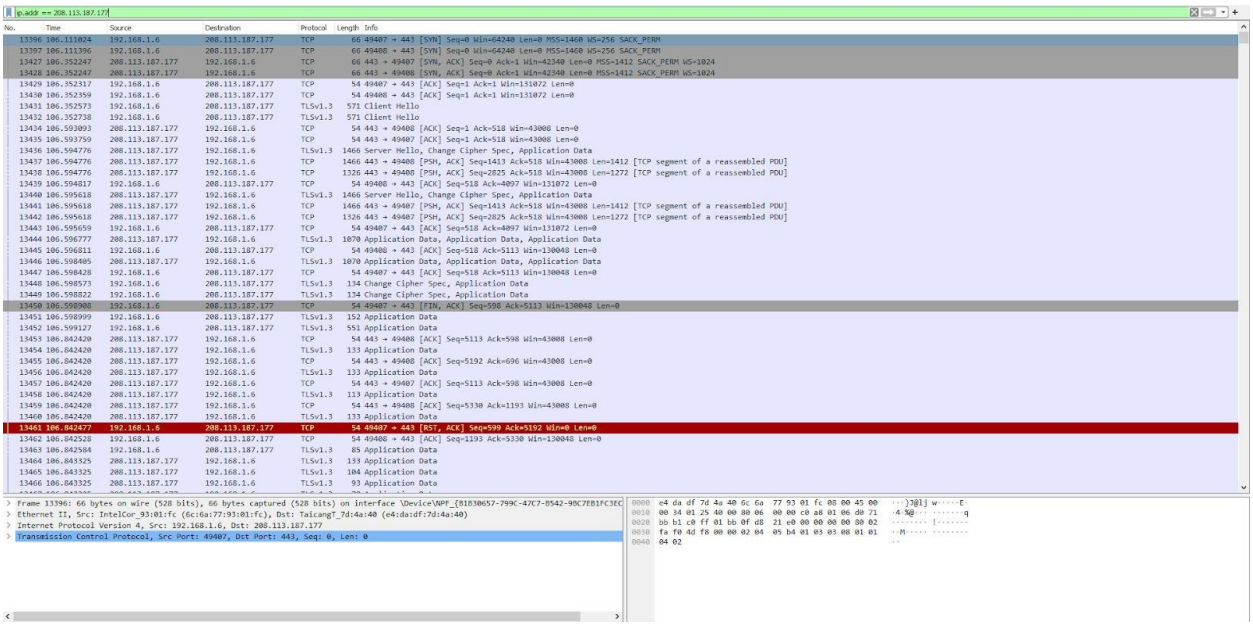
https://www.annauniv.edu/





https://www.mitindia.edu/en/





Result:

Hence websites are monitored and respective network protocol is identified using Wireshark.