CS8581 NETWORKS LABORATORY

OBJECTIVES:

- To learn and use network commands.
- To learn socket programming.
- To implement and analyze various network protocols.
- To learn and use simulation tools.
- To use simulation tools to analyze the performance of various network protocols.

LIST OF EXPERIMENTS

1. Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute. Capture ping and traceroute PDUs using a network protocol analyzer and examine.
2. Write a HTTP web client program to download a web page using TCP sockets.
3. Applications using TCP sockets like:
    a. Echo client and echo server
    b. Chat
    c. File Transfer
4. Simulation of DNS using UDP sockets.
5. Write a code simulating ARP /RARP protocols.
6. Study of Network simulator (NS) and Simulation of Congestion Control Algorithms using NS.
7. Study of TCP/UDP performance using Simulation tool.
8. Simulation of Distance Vector/ Link State Routing algorithm.
9. Performance evaluation of Routing protocols using Simulation tool.
10. Simulation of error correction code (like CRC).

OUTCOMES:

Upon Completion of the course, the students will be able to:

- Implement various protocols using TCP and UDP.
- Compare the performance of different transport layer protocols.
- Use simulation tools to analyze the performance of various network protocols.
- Analyze various routing algorithms.
- Implement error correction codes.

SOFTWARE:

1. C / C++ / Java / Python / Equivalent Compilers
2. Network simulator like NS2/Glomosim/OPNET/ Packet Tracer / Equivalent

Exp. No.	: 01
Date	:

NETWORKS COMMANDS

Aim:
To Learn to use commands like tcpdump, netstat, ifconfig, nslookup and traceroute. Capture ping and traceroute PDUs using a network protocol analyzer and examine.

Procedure:

netstat
Displays active TCP connections, ports on which the computer is listening, Ethernet statistics, the IP routing table, IPv4 statistics (for the IP, ICMP, TCP, and UDP protocols), and IPv6 statistics (for the IPv6, ICMPv6, TCP over IPv6, and UDP over IPv6 protocols). Used without parameters, netstat displays active TCP connections.

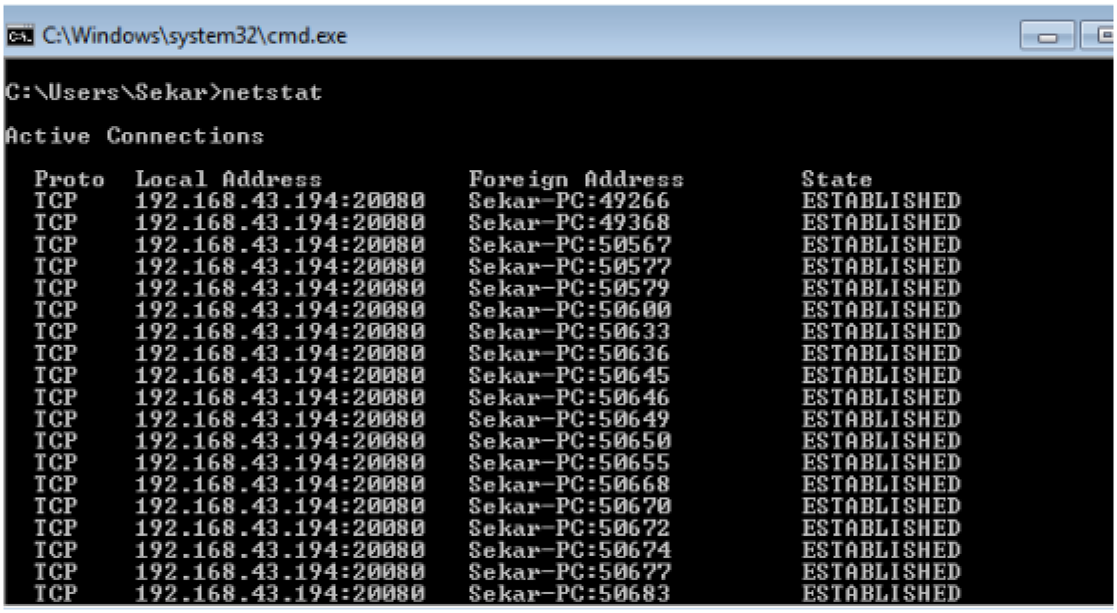
tracert
The tracert command is used to visually see a network packet being sent and received and the amount of hops required for that packet to get to its destination.

Users with Microsoft Windows 2000 and Windows XP who need additional information network latency and network loss should also consider using the pathping command.

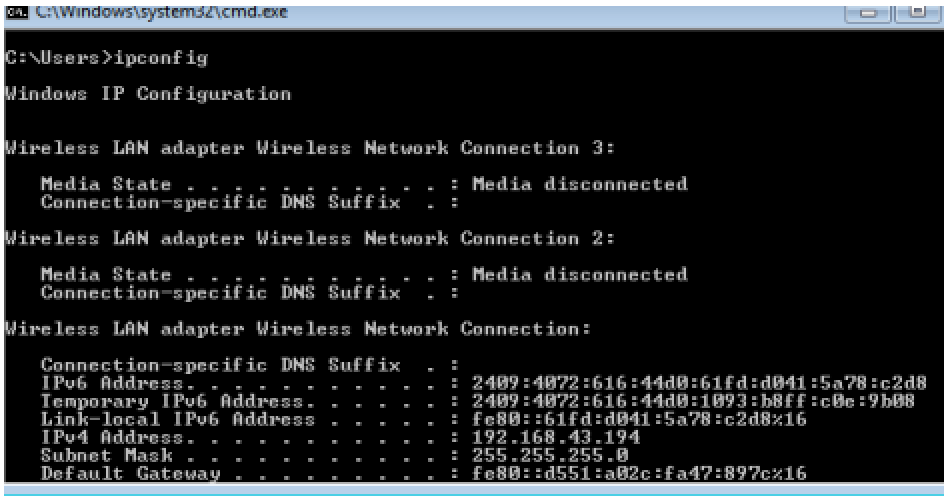
nslookup
Displays information that you can use to diagnose Domain Name System (DNS) infrastructure. Before using this tool, you should be familiar with how DNS works. The Nslookup command-line tool is available only if you have installed the TCP/IP protocol.

arp
Displays, adds, and removes arp information from network devices.

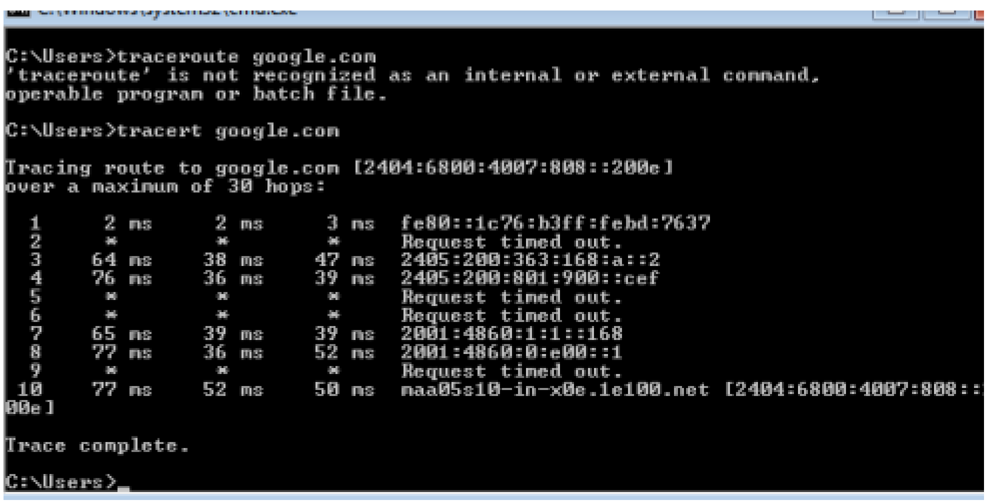
Sample Output:



```
C:\Users\Sekar>netstat

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    192.168.43.194:20080   Sekar-PC:49266         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:49368         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50567         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50577         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50579         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50600         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50633         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50636         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50645         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50646         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50649         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50650         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50655         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50668         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50670         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50672         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50674         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50677         ESTABLISHED
  TCP    192.168.43.194:20080   Sekar-PC:50683         ESTABLISHED
```



```
C:\Users>ipconfig

Windows IP Configuration


Wireless LAN adapter Wireless Network Connection 3:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wireless Network Connection 2:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Wireless LAN adapter Wireless Network Connection:

   Connection-specific DNS Suffix  . :
   IPv6 Address. . . . . . . . . . . : 2409:4072:616:44d0:61fd:d041:5a78:c2d8
   Temporary IPv6 Address. . . . . . : 2409:4072:616:44d0:1093:b8ff:c0e:9b08
   Link-local IPv6 Address . . . . . : fe80::61fd:d041:5a78:c2d8%16
   IPv4 Address. . . . . . . . . . . : 192.168.43.194
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : fe80::d551:a02c:fa47:897c%16
```



```
C:\Users>traceroute google.com
'traceroute' is not recognized as an internal or external command,
operable program or batch file.

C:\Users>tracert google.com

Tracing route to google.com [2404:6800:4007:808::200e]
over a maximum of 30 hops:

  1     2 ms     2 ms     3 ms  fe80::1c76:b3ff:febd:7637
  2     *        *        *     Request timed out.
  3    64 ms    38 ms    47 ms  2405:200:363:168:a::2
  4    76 ms    36 ms    39 ms  2405:200:801:900::cef
  5     *        *        *     Request timed out.
  6     *        *        *     Request timed out.
  7    65 ms    39 ms    39 ms  2001:4860:1:1::168
  8    77 ms    36 ms    52 ms  2001:4860:0:e00::1
  9     *        *        *     Request timed out.
 10    77 ms    52 ms    50 ms  maa05s10-in-x0e.1e100.net [2404:6800:4007:808::2
00e]

Trace complete.

C:\Users>_
```

Result:
Thus network command like tcpdump, netstat, ifconfig, nslookup and traceroute were tested successfully.

Exp. No. : 02
Date :

HTTP WEB CLIENT

Aim:
To Write a HTTP web client program to download a web page using TCP sockets.

Procedure:

1. Start the program.
2. Create a socket which binds the Ip address of server and the port address to acquire service.
3. After establishing connection send a request to server.
4. Receive and print the code for webpage from server.
5. Close the socket.
6. End the program.

Coding:

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

public class SocketHTTPClient
{
public static void main(String[] args)
            {

        String hostName = "www.krct.ac.in";
intportNumber = 80;

try {
           Socket socket = new Socket(hostName, portNumber);
PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
BufferedReader in = new BufferedReader(new
                                    InputStreamReader(socket.getInputStream()));
out.println("GET / HTTP/1.1\nHost: www.krct.ac.in\n\n");
           String inputLine;
while ((inputLine = in.readLine()) != null)
            {
System.out.println(inputLine);
           }
       }
            catch (UnknownHostException e)
            {
System.err.println("Don't know about host " + hostName);
System.exit(1);
            }
            catch (IOException e)
            {
```

```
System.err.println("Couldn't get I/O for the connection to " +
hostName);
System.exit(1);
            }
    }
}
```

Sample Output:

```
Server waiting for image
Client connected
Image size: 29kb
```

Result:

Thus a java program for a HTTP web client program to download a web page using TCP sockets was tested successfully.

Exp. No. : 03(a)
Date :

ECHO CLIENT SERVER APPLICATION

Aim:
To implement Echo client and echo server applications using TCP sockets

Procedure:

CLIENT SIDE
1. Start the program.
2. Create a socket which binds the Ip address of server and the port address to acquire service.
3. After establishing connection send a data to server.
4. Receive and print the same data from server.
5. Close the socket.
6. End the program.

SERVER SIDE
1. Start the program.
2. Create a server socket to activate the port address.
3. Create a socket for the server socket which accepts the connection.
4. After establishing connection receive the data from client.
5. Print and send the same data to client.
6. Close the socket.
7. End the program.

Coding:

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

public class EchoClient {
public static void main(String[] args) throws IOException {

        String serverHostname = new String ("127.0.0.1");

if (args.length> 0)
serverHostname = args[0];

                        System.out.println ("Attemping to connect to host " +
                        serverHostname + " on port 10007.");

        Socket echoSocket = null;
PrintWriter out = null;
BufferedReader in = null;

try {
            // echoSocket = new Socket("taranis", 7);
echoSocket = new Socket(serverHostname, 10007);
out = new PrintWriter(echoSocket.getOutputStream(), true);
in = new BufferedReader(new InputStreamReader(
echoSocket.getInputStream()));
        }
```

```
                    catch (UnknownHostException e)
                    {
System.err.println("Don't know about host: " + serverHostname);
System.exit(1);
      }
                    catch (IOException e)
                    {
System.err.println("Couldn't get I/O for "
                        + "the connection to: " + serverHostname);
System.exit(1);
      }

          BufferedReaderstdIn = new BufferedReader(
newInputStreamReader(System.in));
          String userInput;

System.out.print ("input: ");
          while ((userInput = stdIn.readLine()) != null) {
          out.println(userInput);
                    if (userInput.equals("bye"))
break;
          System.out.println("echo: " + in.readLine());
System.out.print ("input: ");

          }

          out.close();
          in.close();
          stdIn.close();
          echoSocket.close();
   }
}
```

Sample Output:

```
E:\pgms>java EchoServer
Waiting for connection.....
Connection successful
Waiting for input.....
Server: hello
Server: how are you

E:\pgms>java EchoClient
Attemping to connect to host 127.0.0.1 on port 10007.
input: hello
echo: hello
input: how are you
echo: how are you
input:
```

Result:

Thus a java programs to implement Echo client and echo server applications using TCP sockets were tested successfully.