Design TCP iterative client and server application to reverse given input word/sentence

TCP:

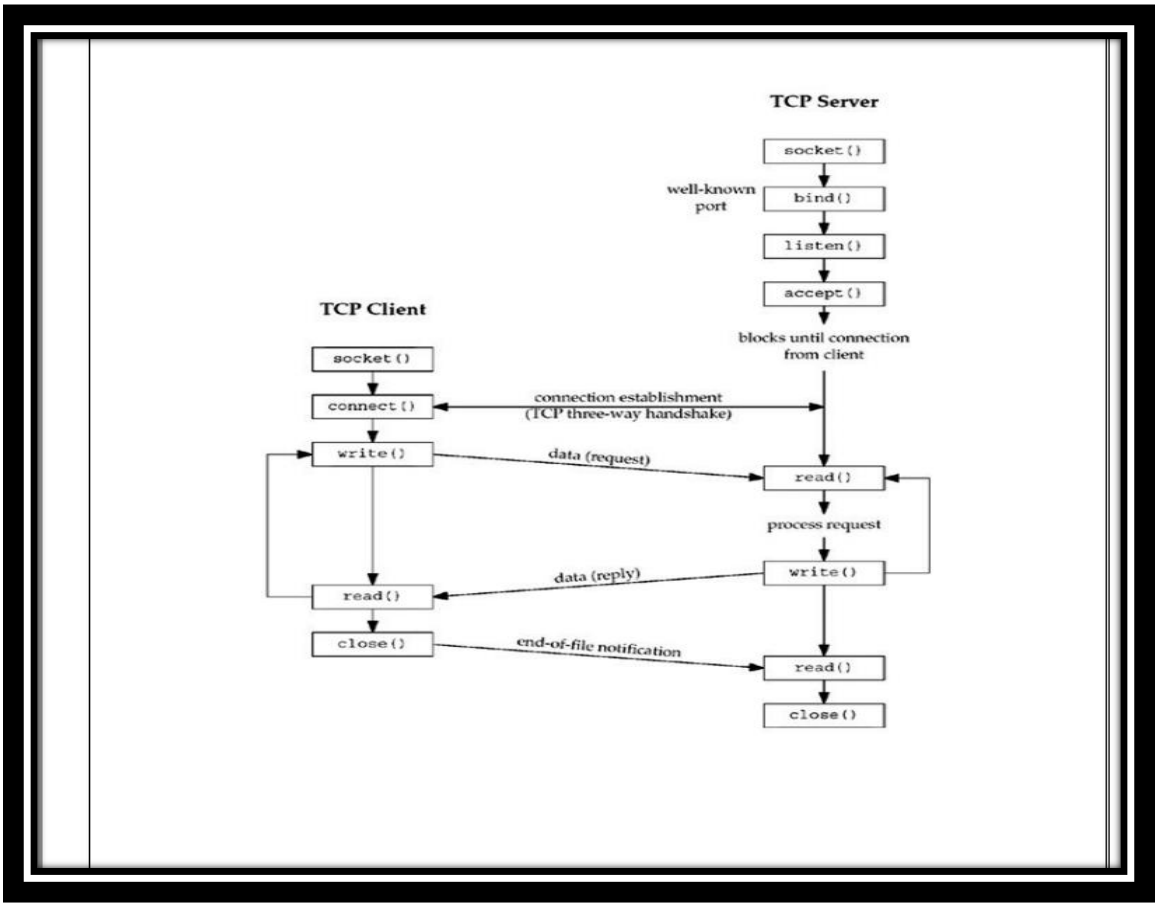
The Transmission Control Protocol (TCP) is a core protocol of the Internet Protocol Suite. It originated in the initial network implementation in which it complemented the Internet Protocol (IP).

NETWORK FUNCTION:

•The Transmission Control Protocol provides a communication service at an intermediate level between an application program and the Internet Protocol.

•It provides host-to-host connectivity at the Transport Layer of the Internet model. An application does not need to know the particular mechanisms for sending data via a link to another host, such as the required packet fragmentation on the transmission medium.

WORKING OF TCP:



DESCRIPTION OF FUNCTIONS:

- Socket function: #include int socket int family, int type, int protocol);
- Connect function: The connect function is used by a TCP client to establish a connection with a TCP server.
- Bind function: The bind function assigns a local protocol address to a socket.
- Close function: The normal UNIX close function is also used to close a socket and terminate a TCP connection.
- Listen function: The second argument to this function specifies the maximum number of connection that the kernel should queue for this socket.
- Accept function: The cliaddr and addrlen argument are used to ret urn the protocol address of the connected peer processes (client).

SOCKET COMMUNICATION:

DIFFERENCE BETWEEN ITERATIVE SERVER AND CONCURRENT SERVER:

ITERATIVE SERVER	CONCURRENT SERVER
1. It process one request at a time	1. It can process multiple request at a time.
2. It doesn't use resources in an efficient manner	2. It uses resources in an efficient manner
3. Used when requests are guaranteed to be completed within a small amount of time.	3. Used in most cases where the time taken to complete a request cannot be limited.
4. Less complex.	4. More complex
5. It is easy to build & implement.	5. It is difficult to design & build.

CODE:

- **CLIENT**

```
import java.lang.*;
import java.io.*;
import java.net.*;
class Client
{
	public static void main(String args[])
	{
	try
	{
		Socket skt = new Socket("localhost", 1234);
```

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

                DataOutputStream outToServer = new
DataOutputStream(skt.getOutputStream());

                BufferedReader inFromServer = new BufferedReader(new
InputStreamReader(skt.getInputStream()));

                String sentence = br.readLine();

                outToServer.writeBytes(sentence + '\n');

                        System.out.print("Received string: '");

               while (!inFromServer.ready()) {}

              System.out.println(inFromServer.readLine()); // Read one line and output it

              System.out.print("'\n");

              br.close();

        }

        catch(Exception e) {

                        System.out.print("Whoops! It didn't work!\n");

        }

    }

}
```

- **SERVER:**

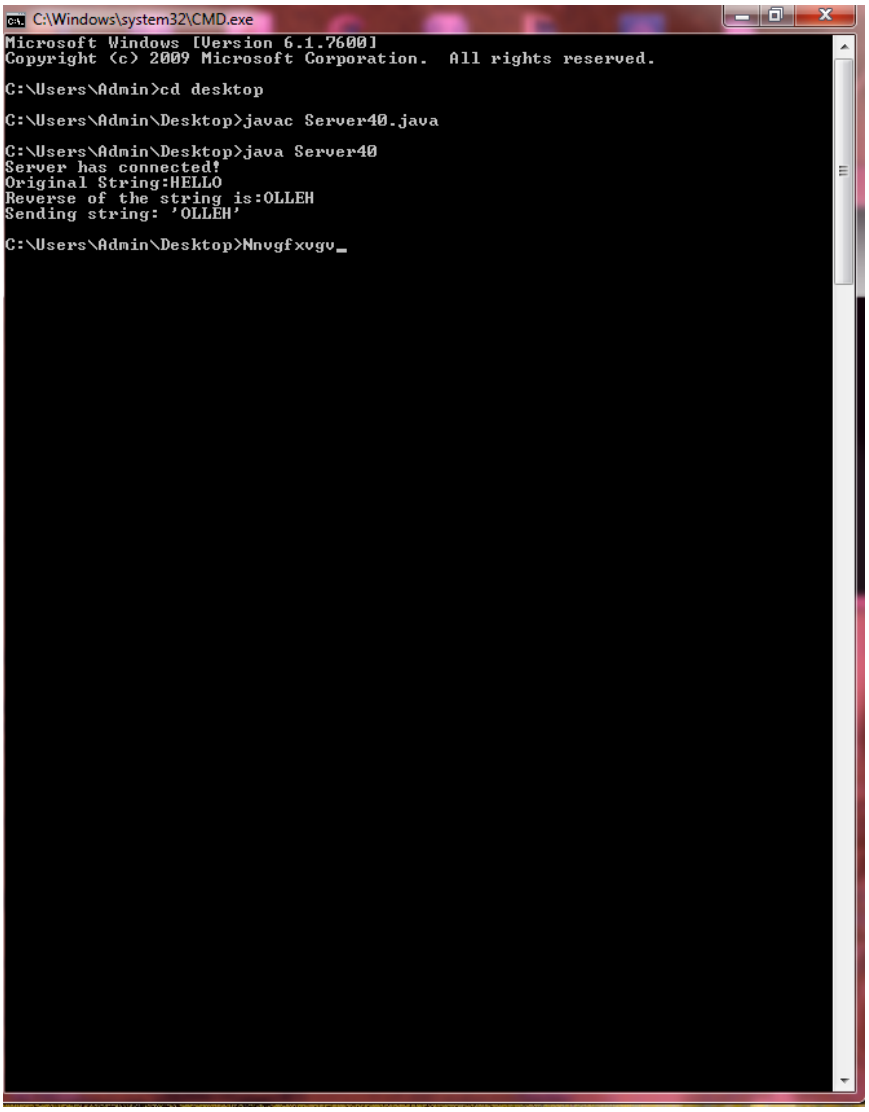
```
import java.lang.*;

import java.io.*;

import java.net.*;

import java.util.*;


class Server40
```

```
{
    public static void main(String args[])
    {
            try {
            ServerSocket srvr = new ServerSocket(1234);
            Socket skt = srvr.accept();
                    BufferedReader inFromClient = new BufferedReader(new
InputStreamReader(skt.getInputStream()));
            DataOutputStream outToClient = new DataOutputStream(skt.getOutputStream());
            String clientSentence = inFromClient.readLine();
                    System.out.print("Server has connected!\n");

                    String original=clientSentence, reverse = "";
                    System.out.println("Original String:"+original);
                    Scanner in = new Scanner(System.in);

                    int length = original.length();

                    for ( int i = length - 1 ; i >= 0 ; i-- )
                    {
                            reverse = reverse + original.charAt(i);
                    }
                    System.out.println("Reverse of the string is:"+reverse);
            PrintWriter out = new PrintWriter(skt.getOutputStream(), true);
            System.out.print("Sending string: '" + reverse+ "'\n");
```

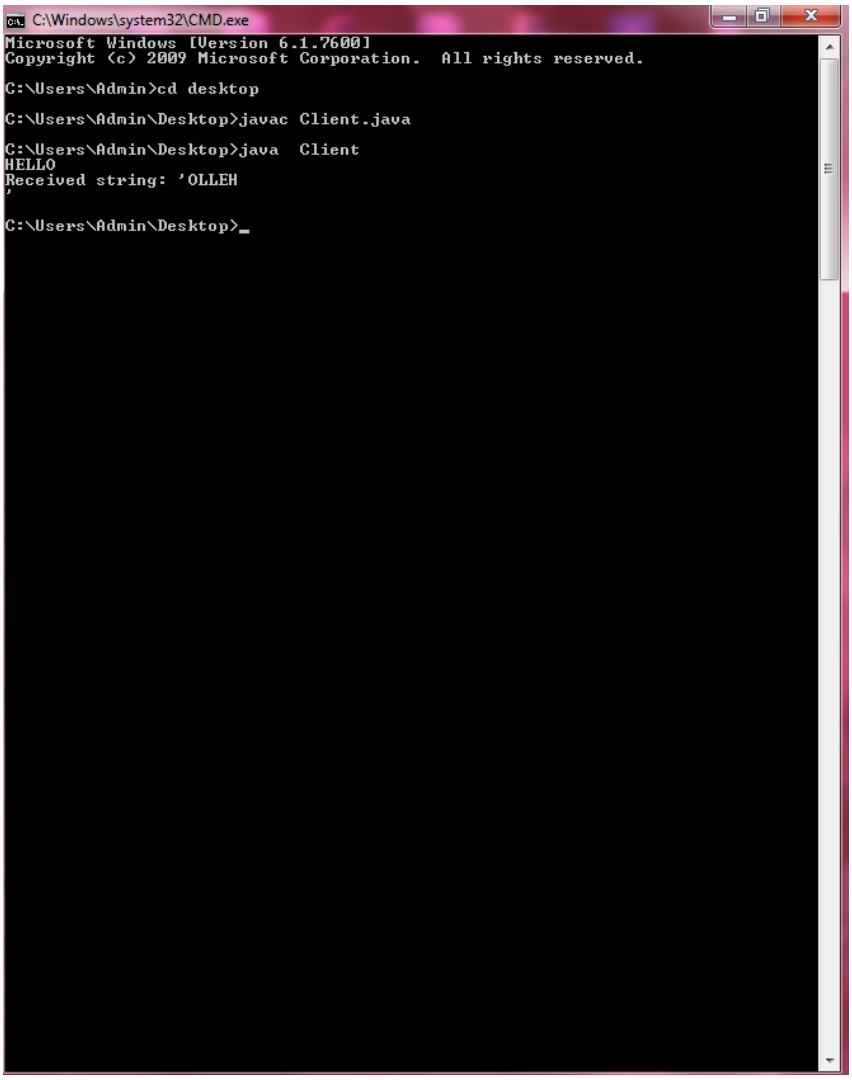
```
            out.print(reverse);
            out.close();
            skt.close();
            srvr.close();
        }
        catch(Exception e)
            {
            System.out.print("Whoops! It didn't work!\n");
        }
    }
}
```

OUTPUT:

SERVER SIDE OUTPUT



CLIENT SIDE OUTPUT


C:\Windows\system32\CMD.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Admin>cd desktop
C:\Users\Admin\Desktop>javac Client.java
C:\Users\Admin\Desktop>java Client
HELLO
Received string: 'OLLEH
'
C:\Users\Admin\Desktop>