EXNO:3 C	APPLICATIONS USING TCP SOCKETS: FILE TRANSFER
DATE:	

AIM:

To write a java program for file transfer using TCP Sockets.

ALGORITHM:

Server:

Step 1.Import java packages and create class file server.
Step 2.Create a new server socket and bind it to the port.
Step 3.Accept the client connection
Step 4.Get the file name and stored into the BufferedReader.
Step 5.Create a new object class file and realine.
Step 6.If file is exists then FileReader read the content until EOF is reached.
Step 7.Stop the program.

Client:

Step 1.Import java packages and create class file server.
Step 2.Create a new server socket and bind it to the port.
Step 3.Now connection is established.
Step 4. The object of a BufferReader class is used for storing data content which has been retrieved from socket object.
Step 5. The connection is closed.
Step 6.Stop the program.

PROGRAM:

File Server :

```
import java.io.BufferedInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.OutputStream;
import java.net.InetAddress;
import java.net.ServerSocket;import java.net.Socket
public class FileServer {
public static void main(String[] args) throws Exception {
ServerSocket ssock = new ServerSocket(5000); Socket socket = ssock.accept();
InetAddress IA = InetAddress.getByName("localhost");
File file = new File("e:\\Bookmarks.html");
FileInputStream fis=new FileInputStream(file);
BufferedInputStream bis = new BufferedInputStream(fis);
OutputStream os = socket.getOutputStream();
byte[] contents;
long fileLength = file.length();
long current = 0;
long start = System.nanoTime();
while (current != fileLength) {
int size = 10000;
if (fileLength - current >= size)
current += size;
else {
size = (int) (fileLength - current);
current = fileLength;
}
contents = new byte[size];
bis.read(contents, 0, size);
os.write(contents);
System.out.print("Sending file ... " + (current * 100) / fileLength + "% complete!");
}
os.flush();
// File transfer done. Close the socket connection!
socket.close();
ssock.close();
System.out.println("File sent succesfully!");
}
}
```

File Client:

```
import java.io.BufferedOutputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.net.InetAddress;
import java.net.Socket;
public class FileClient {
public static void main(String[] args) throws Exception {
Socket socket = new Socket(InetAddress.getByName("localhost"), 5000);
byte[] contents = new byte[10000];
// Initialize the FileOutputStream to the output file's full path.
FileOutputStream fos = new FileOutputStream("e:\\Bookmarks1.html");
BufferedOutputStream bos = new BufferedOutputStream(fos);
InputStream is = socket.getInputStream();
// No of bytes read in one read() call
int bytesRead = 0;
while ((bytesRead = is.read(contents)) != -1)
bos.write(contents, 0, bytesRead); bos.flush();
socket.close();
System.out.println("File saved successfully!");

}
}
```

RESULT:

Thus the java application program using TCP Sockets was developed and executedsuccessfully.

EXNO: 4	SIMULATION OF DNS USING UDP SOCKETS
DATE:	

AIM:

To write a java program for DNS application

ALGORITHM:

Server:

Step 1.Start the program.

Step 2.Create UDP datagram socket

Step 3.Create a table that maps host name and IP address

Step 4.Receive the host name from the client

Step 5.Retrieve the client‘s IP address from the received datagram

Step 6.Get the IP address mapped for the host name from the table.

Step 7.Display the host name and corresponding IP address

Step 8.Send the IP address for the requested host name to the client

Step 9.Stop the program.

Client:

Step 1.Start the program.
Step 2.Create UDP datagram socket.
Step 3.Get the host name from the client
Step 4.Send the host name to the server
Step 5.Wait for the reply from the server
Step 6.Receive the reply datagram and read the IP address for the requested host name
Step 7.Display the IP address.
Step 8.Stop the program.

PROGRAM:

DNS Server.java

```
import java.io.*;
import java.net.*;
public class udpdnsserver {
private static int indexOf(String[] array, String str) {
 str = str.trim();
for (int i = 0; i < array.length; i++) { if (array[i].equals(str))
return i;
}
return -1;
}
public static void main(String arg[]) throws IOException
{
String[] hosts = { "yahoo.com", "gmail.com", "cricinfo.com", "facebook.com" };
String[] ip = { "68.180.206.184", "209.85.148.19", "80.168.92.140", "69.63.189.16" };
System.out.println("Press Ctrl + C to Quit");
while (true) {
DatagramSocket serversocket = new DatagramSocket(1362);
byte[] senddata = new byte[1021];
byte[] receivedata = new byte[1021];
DatagramPacket recvpack = new DatagramPacket(receivedata, receivedata.length);
serversocket.receive(recvpack);
String sen = new String(recvpack.getData());
InetAddress ipaddress = recvpack.getAddress();
int port = recvpack.getPort();
String capsent;
System.out.println("Request for host " + sen);
if (indexOf(hosts, sen) != -1)
capsent = ip[indexOf(hosts, sen)];
else
capsent = "Host Not Found";
senddata = capsent.getBytes();
DatagramPacket pack = new DatagramPacket(senddata, senddata.length, ipaddress, port);
serversocket.send(pack);
serversocket.close();
}
}
}
```

UDP DNS Client java

```
import java.io.*;
import java.net.*;
public class udpdnsclient {
public static void main(String args[]) throws IOException {
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
DatagramSocket clientsocket = new DatagramSocket();
InetAddress ipaddress; if (args.length == 0)
ipaddress = InetAddress.getLocalHost();
else
ipaddress = InetAddress.getByName(args[0]);
byte[] senddata = new byte[1024];
byte[] receivedata = new byte[1024];
int portaddr = 1362;
System.out.print("Enter the hostname : ");
String sentence = br.readLine();
Senddata = sentence.getBytes();
DatagramPacket pack = new DatagramPacket(senddata, senddata.length, ipaddress, portaddr);
clientsocket.send(pack);
DatagramPacket recvpack = new DatagramPacket(receivedata, receivedata.length);
clientsocket.receive(recvpack);
String modified = new String(recvpack.getData());
System.out.println("IP Address: " + modified);
clientsocket.close();

}
}
```

RESULT:

Thus the java application program using UDP Sockets to implement DNS wasdeveloped and executed successfully