Name: Rajat Rajeshsheth
CWID: 10[illegible]7484

Assignment-5

12 20 25 30 31

(12) Write a simple program for a server that accepts lines of input from a client & prints the lines onto the server's standard output. Compile & execute your program on any other machine that contains a web browser, set the proxy server in the browser to the host that is running your server program, also configure the port number appropriately.

⇒ TCP program code.

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

Class TCPServer
{
  public static void main (String argv[]) throws Exception
  {
    String clientMessage;
    ServerSocket messageSocket = new ServerSocket(6789);
    while(true)
    {
      Socket connectionSocket = messageSocket.accept();
      BufferedReader bd = new BufferedReader(new InputStreamReader(
          connectionSocket.getInputStream()));
      clientMessage = bd.readLine();
      System.out.println("client message recieved: "+clientMessage
          + "\n");
    }
  }
}
```

20] suppose you can access the cache in the local DNS servers of your department. Can you propose a way to roughly determine the web servers (outside your department) that are the most popular among the users in your department? Explain.

⇒ we can periodically take a snapshot of the DNS caches in the local DNS servers. The web server that appears most frequently in the DNS caches is the most popular server. This is because if more users are interested in a web server, then DNS requests for that server are more frequently sent by users. Thus, that web server will appear in the DNS cache more frequently.

---

(25) consider an overlay network with N active peers, with each pair of peers having an active TCP connection. Additionally, suppose that the TCP connections pass through a total of M routers. How many nodes & edges are there in the corresponding overlay network?

⇒ There are N Nodes in the overlay network.

There are $N(N-1)/2$ edges.

---

(29) (30). suppose that in UDPClient.py after we create the socket, we add the line: clientSocket.bind(("",5432))
will it become neccessary to change UDPServer.py? what are the port numbers for the sockets in UDPClient & UDP server? what were they before making this change?

⇒ In the orginal problem - UDPClient doesnot specify a port no when it creates the socket. In this case, the code lets the Underlying OS choose a port no. with the additional line, when UDP client is exhausted, a UDP socket is created with port no 5432.

UDPServer needs to know the client port no so that it can send packets back to the correct client socket. Glancing at UDP server we see that the client port no is not "hard-wired" into

into the server code; instead, UDP server determines the client port no by unraveling the datagram it receives from the client. Thus UDP server will work with any client port no, including 5432. UDP server ∴ does not need to be modified.

Before:

client socket : x (chosen by OS)

server socket = 9876

After:

client socket : 5432

---

30) can you configure your browser to open multiple simultaneous connections to a web site? what are the advantages & disadvantages of having a large no of simultaneous TCP connections?

⇒ Yes, you can configure many browsers to open multiple simultaneous connections to a web site. The advantage is that you will potentially download the file faster. The disadvantage is that you may be hogging the bandwidth, thereby significantly slowing down the download of other users who are sharing the same physical links.

---

31) we have seen the internet TCP sockets treat the data being sent as a byte stream but UDP sockets recognize message boundaries. what are one advantage & one disadvantage of byte-oriented API versus having the API explicitly recognize & preserve application-defined message boundaries?

⇒ For an application such as remote login (telnet & ssh) a byte-stream oriented protocol is very natural since there is no notion of messages & boundaries in the application.

In other applications, we may be sending a series of

messages that have inherent boundaries bet them. For example, when one SMTP mail server sends another SMTP mail servers several email messages back to back. since TCP doesnot have a mechanism to indicate the boundaries. The application must add the indicator itself, so that recieving side of the application can distinguish one message from the next. If each message were instead put into a distinct UDP segment, the recieving end would be able to distinguish the various messages without any indicators added by the sending side of the application.