Mawlana Bhashani Science And Technology University

Lab-Report

Lab Report No: 06

Lab Report Name: Socket programming (Time protocol)

Group member ID: IT-18013 and IT-18028

Date of Performance: 30-06-2021

Date of Submission: 30-06-2021

Submitted by

Name: Anjom Nour Anika

ID: IT-18013.

3rd Year 2nd Semester

Session: 2017-2018

Dept. of ICT, MBSTU

Submitted To

Nazrul Islam

Assistant Professor

Dept. of ICT

MBSTU.

Objectives:

The objectives of this lab is how to know Sockets provide the communication mechanism between two computers using TCP and Java.net.ServerSocket class provides a mechanism for the server program to listen for clients and establish connections with them.

Theory:

A client program creats a socket on its end of the communication and attempts to connect that socket to a server. Sockets provide the communication mechanism between two computers using TCP. The Java.net.Socket class represents a Socket, and the Java.net.ServerSocket class provides a mechanism for the server program to listen for clients and establish connections with them.

Methodology:

1.Briefly explain the term IPC in terms of TCP/IP communication.

Ans:

In computer science, inter-process communication or interprocess communication (IPC) refers specifically to the

mechanisms an operating system provides to allow the processes to manage shared data. Typically, applications can use IPC, categorized as clients and servers, where the client requests data and the server responds to client requests. Many applications are both clients and servers, as commonly seen in distributed computing.

IPC is very important to the design process for microkernels and microkernels, which reduce the number of functionalities provided by the kernel. Those functionalities are then obtained by communicating with servers via IPC, leading to a large increase in communication when compared to a regular monolithic kernel. IPC interfaces generally encompass variable analytic framework structures. These processes ensure compatibility between the multi-vector protocols upon which IPC models rely.

An IPC mechanism is either synchronous or asynchronous. Synchronization primitives may be used to have synchronous behavior with an asynchronous IPC mechanism.

2. What is the maximum size of a UDP datagram? What are the implications of using a packet-based Protocol as opposed to a stream protocol for transfer of large files?

Ans:

It depends on the underlying protocol i.e., whether you are using IPv4 or IPv6.

- In IPv4, the maximum length of packet size is 65,536. So, for UDP datagram you have maximum data length as:

65,535 bytes - 20 bytes (Size of IP header) = 65, 515 bytes (including 8 bytes UDP header)

- In IPv6, the maximum length of packet size allowed is 64 kb. So, you can have UDP datagram of size greater than that.

NOTE: This size is the theoretical maximum size of UDP Datagram, in practice though, this limit is further constrained by the MTU of data-link layer (which varies for each data-link layer technology, but cannot be less than 576 bytes), considering that, maximum size of UDP datagram can be further calculated as (for IPv4):

- 576 bytes - 20 bytes(IP header) = 556 (including 8 bytes UDP header)

3. TCP is a reliable transport protocol, briefly explain what techniques are used to provide this reliability?

Ans:

A number of mechanisms help provide the reliability TCP. Each of these is described briefly below:

Duplicate Data Detection: It is possible for packets to be duplicated in packet switched network; therefore TCP keeps

track of bytes received in order to discard duplicate copies of data that has already been received.

Retransmissions: In order to guarantee delivery of data, TCP must implement retransmission schemes for data that may be lost or damaged. The use of positive acknowledgements by the receiver to the sender confirms successful reception of data. The lack of positive acknowledgements, coupled with a timeout period calls for a retransmission

Sequencing : In packet switched networks, it is possible for packets to be delivered out of order. It is TCP's job to properly sequence segments it receives so it can deliver the byte stream data to an application in order.

Timers: TCP maintains various static and dynamic timers on data sent. The sending TCP waits for the receiver to reply with an acknowledgement within a bounded length of time. If the timer expires before receiving an acknowledgement, the sender can retransmit the segment.

4) Why are the htons(), htonl(), ntohs(), ntohl() functions used?

Ans:

htons():

The **htons()** function is **used** to convert a short (2-byte) integer from the local host byte order to standard network byte order.

htonl():

The **htonl()** function is **used** to convert a long (4-byte) integer from the local host byte order to standard network byte order.

ntohs():

The **ntohs()** function is **used** to convert a short (2-byte) integer from the standard network byte order to the local host byte order.

ntohl():

The **ntohl()** function is **used** to convert a long (4-byte) integer from the standard network byte order to the local host byte order.

4. What is the difference between datagram socket and stream socket?

Ans:

A stream socket is like a phone call -- one side places the call, the other answers, you say hello to each other (SYN/ACK in TCP), and then you exchange information. Once you are done, you say goodbye (FIN/ACK in TCP). If one side doesn't hear a goodbye, they will usually call the other back since this is an unexpected event; usually the client will reconnect to the server. There is a guarantee that data will not arrive in a different order than you sent it, and there is a reasonable guarantee that data will not be damaged.

A datagram socket is like passing a note in class. Consider the case where you are not directly next to the person you are passing the note to; the note will travel from person to person. It may not reach its destination, and it may be modified by the time it gets there. If you pass two notes to the same person, they may arrive in an order you didn't intend, since the route the notes take through the classroom may not be the same, one person might not pass a note as fast as another, etc.

Time Protocol implementation:

```
public class GreetingServer implements Runnable {
   private ServerSocket serverSocket;

   public GreetingServer(int port) throws IOException {
      serverSocket = new ServerSocket(port);
      serverSocket.setSoTimeout(30000);
   }

   public void run() {
      while(true) {
         try {
            System.out.println("Waiting for client on port " +
                               serverSocket.getLocalPort() +
                               "...");
            Socket server = serverSocket.accept();

            System.out.println("Received connection from " +
                        server.getRemoteSocketAddress());
            DataInputStream in = new
                  DataInputStream(server.getInputStream());

            System.out.println("Client says: " +
in.readUTF());
            DataOutputStream out = new
                  DataOutputStream(server.getOutputStream());
            out.writeUTF("Thank you for connecting to " +
                         server.getLocalSocketAddress() +
                         ". Bye!");
            server.close();
```

```
            } catch (SocketTimeoutException s) {
                System.out.println("Socket timed out!");
                break;
            } catch (IOException e) {
                e.printStackTrace();
                break;
            }
        }
    }
```

The following GreetingServer program is an example of a server application that uses the Socket class to listen for clients on a port number specified by a command-line argument.

Conclusion

How to implement Time Protocol over the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP) – we have examined from this lab. Here, we have examined how to host connects to a server that supports the Time Protocol on port 8084. We have also learnt that how to communicate client and the server by writing to and reading from the socket and how to Sockets provide the communication mechanism between two computers using TCP. In this lab we also implemented how to java.net.ServerSocket class provides a mechanism for the server program to listen for clients and establish connections with them.