



Mawlana Bhashani Science and Technology University

Santosh, Tangail-1902.

Lab Report

Department of Information and Communication Technology

Report No: 06

Report Name: Socket Programming (Time Protocol).

Course Title: Network Planning and designing Lab.

Course Code: ICT-3208

Submitted By	Submitted To
Name: Zafrul Hasan Khan & Hasibul Islam Imon ID: IT-18003 & IT-18047 Session: 2017-18 3rd Year 2nd Semester Dept. of Information & Communication Technology, MBSTU.	Nazrul Islam Assistant Professor, Dept. of Information & Communication Technology, MBSTU.

Objectives : The main objectives of this lab 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 creates 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 :

The answer will be different each Operating System. Unix offers System V IPC, which gives you message queues, shared memory, and semaphores . InterProcess Communication is a term we use for interactions between two processes on the same host. William Westlake mentions TCP/IP, which is used for interactions with another host. It can be used locally as well, but it is relatively inefficient. Unix domain sockets are used in the same way, but are only for local use and a bit more efficient.

Message queues are easy to use: processes and threads can send variable-sized messages by appending them to some queue and others can receive messages from them so that each message is received at most once. Those operations can be blocking or non-blocking. The difference with UDP is that messages are received in the same order as they are sent. Pipes are simpler, but pass a stream of data instead of distinct messages. Line feeds can be used as delimiters. Shared memory allows different processes to share fixed regions of memory in the same way that threads have access to the same memory. Unix allocates a certain amount of memory after which it can be accessed like private memory. A similar mechanism is the memory-mapped file: the difference is that the memory segment is initialised from a disc file and changes can be permanent. The size of a file can change, which complicates shared files. Since two threads updating the

same data can lead to inconsistencies, semaphores can be used to achieve mutual exclusion.

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 :

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)

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

>> In IPv6, the maximum length of packet size allowed is 64 kB, so, you can have UDP datagram of size greater than that.

>> 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)

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: These are used for:

htons() host to network short

htonl() host to network long

ntohs() network to host short

ntohl() network to host long

5) What is the difference between a datagram socket and a stream socket?

Ans:

The difference is given below :

Stream socket

A stream socket provides a bidirectional, reliable, sequenced, and unduplicated flow of data with no record boundaries. Stream sockets enable processes to communicate using TCP .After the connection has been established, data can be read from and written to these sockets as a byte stream. The socket type is SOCK_STREAM.

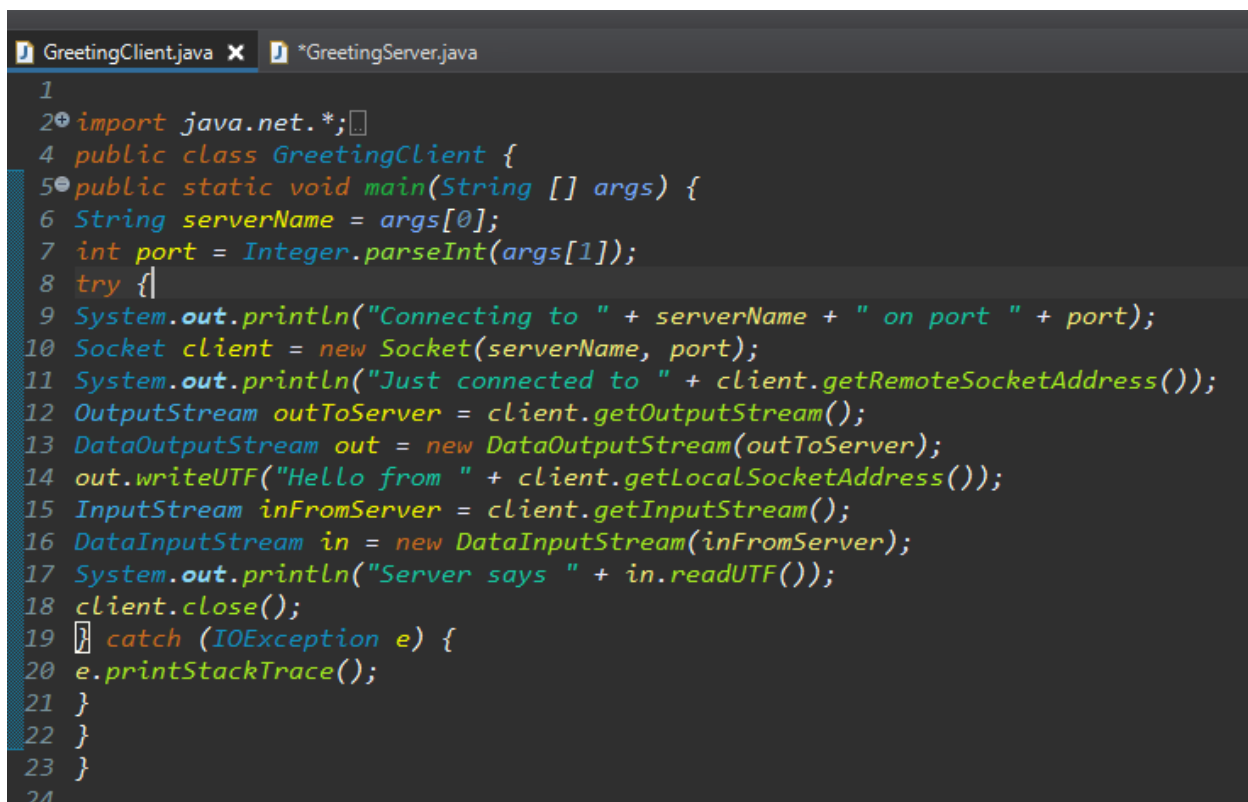
Datagram socket

A datagram socket supports a bidirectional flow of messages. Datagram sockets enable processes to use UDP to communicate. A process on a datagram socket

might receive messages in a different order from the sending sequence. A process on a datagram socket might receive duplicate messages. Messages that are sent over a datagram socket might be dropped. The socket type is SOCK_DGRAM. Record boundaries in the data are preserved.

Time Protocol implementation:

A java program where the following GreetingClient is a client program that connects to a server by using a socket and sends a greeting, and then waits for a response.



```
1
2 import java.net.*;
3
4 public class GreetingClient {
5     public static void main(String [] args) {
6         String serverName = args[0];
7         int port = Integer.parseInt(args[1]);
8         try {
9             System.out.println("Connecting to " + serverName + " on port " + port);
10            Socket client = new Socket(serverName, port);
11            System.out.println("Just connected to " + client.getRemoteSocketAddress());
12            OutputStream outToServer = client.getOutputStream();
13            DataOutputStream out = new DataOutputStream(outToServer);
14            out.writeUTF("Hello from " + client.getLocalSocketAddress());
15            InputStream inFromServer = client.getInputStream();
16            DataInputStream in = new DataInputStream(inFromServer);
17            System.out.println("Server says " + in.readUTF());
18            client.close();
19        } catch (IOException e) {
20            e.printStackTrace();
21        }
22    }
23 }
24
```

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 –

```
GreetingClient.java  *GreetingServer.java X
1 import java.net.*;
2 import java.io.*;
3 public class GreetingServer extends Thread {
4     private ServerSocket serverSocket;
5     public GreetingServer(int port) throws IOException {
6         serverSocket = new ServerSocket(port);
7         serverSocket.setSoTimeout(10000);
8     }
9     public void run() {
10        while(true) {
11            try {
12                System.out.println("Waiting for client on port " + serverSocket.getLocalPort() + "...");
13                Socket server = serverSocket.accept();
14                System.out.println("Just connected to " + server.getRemoteSocketAddress());
15                DataInputStream in = new DataInputStream(server.getInputStream());
16                System.out.println(in.readUTF());
17                DataOutputStream out = new DataOutputStream(server.getOutputStream());
18                out.writeUTF("Thank you for connecting to " + server.getLocalSocketAddress()
19                    + "\nGoodbye!");
20                server.close();
21            } catch (SocketTimeoutException s) {
22                System.out.println("Socket timed out!");
23                break;
24            } catch (IOException e) {
25                e.printStackTrace();
26                break;
27            }
28        }
29    }
30 }
```

```
35 } catch (IOException e) {
36     e.printStackTrace();
37 }
38 }
39 }
40
<
Console X
<terminated> GreetingServer [Java Application] C:\Program Files\Java\jdk-11.0.2\bin\javaw.exe (F
$ java GreetingClient localhost 8084
Connecting to localhost on port 8084
Just connected to localhost/127.0.0.1:8084
Server says Thank you for connecting to /127.0.0.1:8084
Goodbye
```

Conclusion : From this lab , I learnt that how to Time Protocol implement over the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP).I have known that how to communicate client and the server by writing to and reading from the socket. I have also known that how to Sockets provide the communication mechanism between two computers using TCP. I learnt that how

to `java.net.ServerSocket` class provides a mechanism for the server program to listen for clients and establish connections with them. I have implement how to host connects to a server that supports the Time Protocol on port 8084.