CMPU4021 Distributed Systems – Labs

Week 2 – Socket Communication in Java

Learning Outcomes:

1. Be able to open TCP and UDP sockets with Java.
2. Be able to implement a simple distributed system: i.e. client server applications in Java.
3. Describe what is meant by multicasting, and demonstrate a clear understanding of how Java provides support for IP multicast.

Tasks

T1. Examine and compile `JT1/TCPEchoClient.java` and `T3/TCPEchoServer.java`.

Run the server in one CMD box and run the client in another CMD box.

T2. Find out your IP address (type `ipconfig` at the command prompt) or host name. Edit your code so that your client will take in the IP address or host name of the server, and not just use localhost. Use this client to access the server of the person sitting at the PC beside you.

T3. Examine and compile `JT3/UDPEchoClient.java` and `T5/UDPEchoServer.java`.
Run the server in one CMD box and run the client in another CMD box.

T4. Examine and compile `JT4/UDPClient.java` and `JT4/UDPServer.java`.

UDP server repeatedly receives a request and sends it back to the client

UDP client sends a message to the server and gets a reply.

Run the server in one CMD box and run the client in another CMD box.

T5. Examine and compile `JT5/TCPClient.java` and `JT5/TCPServer.java`.

TCP server makes a connection for each client and then echoes the client's request: TCP server opens a server socket on its server port (7896) and listens for *`connect`* requests. When one arrives, it makes a new thread in which to communicate with the client. The new thread creates a *`DataInputStream`* and a *`DataOutputStream`*

from it socket's input and output streams and then waits to read a message and write it back.

TCP client makes connection to server, sends request and receives reply:
TCP client's `main` method supplies a message and the DNS hostname of the server. The client creates a socket bound to the hostname and server port 7896. It makes a `DataInputStream` and a `DataOutputStream` from the socket's input and output streams and then writes the message to its output stream and waits to read a reply from its input stream.

Run the server in one CMD box and run the client in another CMD box (i.e. on Windows machines, start/run cmd.exe)

IP Multicast

T6_1. Examine, compile and run `JT6_1/MulticastNetworkInterfaces.java`

This program display names of specific network interfaces that support multicasting. You may need to run it to decide on which network interface to use the multicast.

T6. Examine, compile and run `JT6/MulticastServer` as a process from one command prompt. This process will broadcast the time to the clients every few seconds.

```
:/> java -classpath . MulticastServer portnum
```

Examine, compile and run `JT6/MulticastClient` as a process from a second command prompt. The client will listen for the time messages, and print them out when received.

```
:/> java -classpath . MulticastClient portnum
```

Run more clients to observe how they all receive the one message broadcast from the server.

T7. Create a Java client-server application using UDP. The aim of the application is to guess a number. When the server is started it stores a random number between 1 and 100...

```
int randomNumber = (int)(Math.random() * 100);
```

The client reads numbers in from the user and sends these to the server. The server responds with a string saying HIGHER, LOWER to CORRECT.
The client can continue entering values until it gets CORRECT returned.

If you do not finish this task in the lab – you can continue it next week or finish at home.

Further Reading

- https://docs.oracle.com/en/java/javase/22/docs/api/java.base/java/net/package-summary.html