Distributed Systems – Labs

If you think you need to, do the tasks from Introduction to Java Lab notes.

Week 2 – Introduction to Network Programming in Java

Learning Outcomes:

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

Tasks

1. Extract, compile and run `T1/IPFinder.java` and `T1/MyLocalIPAddress.java.` Go through the code to understand what is going on. Remember, to compile...

```
:/> javac -classpath . IPFinder.java
:/> javac -classpath . MyLocalIPAddress.java
 and to run...
:/> java -classpath . IPFinder
:/> java -classpath . MyLocalIPAddress
```

Notice the use of the import statements to use the IO and networking packages.

2. Extract and compile `T2/TCPEchoClient.java` and `T2/TCPEchoServer.java`.

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

3. 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.

4. Extract and compile `T4/UDPEchoClient.java` and `T4/UDPEchoServer.java`.

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

5. Extract and compile `T5/UDPClient.java` and `T5/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 DOS box and run the client in another DOS box.
6. Extract and compile `T6/TCPClient.java` and `T6/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 DOS box and run the client in another DOS box.

7. Create a 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.

IP Multicast

8. Extract and run `T8/MulticastServer` as a process from one command prompt. This process will broadcast the time to the clients every few seconds.

Extract and run `T8/MulticastClient` as a process from a second command prompt. The client will listen for the time messages, and print them out when received.

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

Further Reading
Java Tutorial sections on:

- http://docs.oracle.com/javase/tutorial/networking/sockets/
- http://docs.oracle.com/javase/tutorial/networking/urls/definition.html

- Chapter 1 and 2 of Introduction to Network Programming in Java by Graba (Third Edition)
- https://docs.oracle.com/javase/tutorial/networking/datagrams/broadcasting.html
- http://docs.oracle.com/javase/8/docs/api/index.html