Network Distributed Computing

Student Lab Manual

Mark Cummins, ITB

Description

In this lab we'll have a quick look at sockets programming with Java. A socket is basically an interface between the user and the network, so when we want to send data across a network between two programs we open a socket, send or receive the data, and close the socket.

Message destinations are specified as socket addresses; each socket address is a communication identifier that consists of an internet address and a port number. Make sure to study the sample code in detail, you will have to write your own applications later in the term., so try to understand what is happening. There are four steps to programming a client or a server using sockets:

- Opening a socket
- Creating a data input stream
- Creating a data output stream
- Closing the I/O streams
- Closing sockets

Activities:

1 **Study the source code for both of the given programs.**
(SimpleClient.java & SimpleServer.java)

2 **Copy the code onto your computer, compile and run it.**
Make sure you open a separate Command Prompt Shell for each program, and run the Server program first then the client.

3 **What happens if you start the Client Program before the server? Why?**

4 **The Server can respond with two different messages, depending on what you type on the client. Study the code and make sure you can get both responses.**

5 **The Client needs the address of the server and a port to connect to.**
(You may need to change the coded IP address to get it working)
Change the IP address in the code for the keyword 'localhost"

6 **Try connecting two PC's together and connecting a Client program running on one PC to a Server program running on the other.**

7 **After the Client connects to the Server and sends a message, both the Server and Client programs end. By changing the code slightly you should be able to create a basic chat application.**

8 **What happens if you try to connect a second or third Client to your new chat Server?**

SimpleServer.java

```
1  import java.io.*;
2  import java.net.*;                      //Supports Socket & DatagramSocket Class
3
4  public class SimpleServer {
5     public final static int TESTPORT = 5000;
6     public static void main(String args[]) {
7
8        // declare a server socket and a client socket for the server
9        // declare an input and an output stream
10       ServerSocket checkServer = null;
11       String line;
12       BufferedReader is = null;
13       DataOutputStream os = null;
14       Socket clientSocket = null;
15
16       // Try to open a server socket on port TESTPORT
17       try {
18          checkServer = new ServerSocket(TESTPORT);
19          System.out.println("SimpleServer started.....");
20       }  catch (IOException e) {
21          System.out.println(e);
22       }
23
24       // Create a socket object from the ServerSocket to listen and accept connections.
25       // Open input and output streams
26       try {
27          clientSocket = checkServer.accept();
28          is = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
29          os = new DataOutputStream(clientSocket.getOutputStream());
30       }  catch(Exception ei) {
31          ei.printStackTrace();
32       }
33
34       // receive client's input
35       try {
36          line=is.readLine();
37          System.out.println("we received: "+line);
38          if (line.compareTo("Greetings") == 0) {
39             os.writeBytes("...and saluation...");
40          } else {
41             os.writeBytes("sorry, you don't speak my protocol");
42          }
43       }  catch (IOException e) {
44          System.out.println(e);
45       }
46
47       // close the i/o streams and the connection
48       try {
49          os.close();
50          is.close();
51          clientSocket.close();
52       }  catch(IOException ic) {
53          ic.printStackTrace();
54       }
55    }
56 }
```

SimpleClient.java

```
1  import java.io.*;
2  import java.net.*;
3
4  public class SimpleClient {
5     public final static int REMOTE_PORT = 5000;
6     public static void main(String argv[]) throws Exception {
7
8        Socket cl = null, cl2=null;
9        BufferedReader is = null;
10       DataOutputStream os = null;
11       BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
12       String userInput = null;
13       String output = null;
14
15       // Open connection to the server on port REMOTE_PORT
16       try {
17          cl = new Socket("192.168.1.1",REMOTE_PORT);
18          is = new BufferedReader(new InputStreamReader(cl.getInputStream()));
19          os = new DataOutputStream(cl.getOutputStream());
20       }  catch(UnknownHostException e1)
21          System.out.println("Unknown Host: "+e1);
22       }  catch (IOException e2) {
23          System.out.println("Erorr io: "+e2);
24       }
25
26       // write the url to the compute engine
27       try {
28          System.out.print("Please input a keyword: ");
29          userInput = stdin.readLine();
30          os.writeBytes(userInput+"\n");
31       }  catch (IOException ex) {
32          System.out.println("error writing to server..."+ex);
33       }
34
35       // receive results from the compute engine
36       try {
37          output = is.readLine();
38          System.out.println("Got from server: "+output);
39       }  catch(IOException e) {
40          e.printStackTrace();
41       }
42
43       // close input stream, output stream and connection
44       try {
45          is.close();
46          os.close();
47          cl.close();
48       }  catch (IOException x) {
49          System.out.println("Error writing...."+x);
50       }
51    }
52 }
```