Week 13 Lab - Exploring Connections & Remote Execution

COSC244

1 Assessment

This lab is worth 0.5%. The marks are awarded for completing the programming exercise and answering the questions.

Answers to the questions should be stored electronically in a plain text file with a .txt suffix in your ∼/244/13 directory.

Make sure you get your work signed off by a demonstrator before leaving the lab.

2 Purpose

This lab has two purposes:

1. To more fully understand the concept of a network connection.
2. To explore execution on more than one machine.

3 A Network Connection

A network connection is a communication path from a program running on one machine to a program running on another machine. In the previous labs, the two machines have in fact been the same machine. It was the machine you were using. It was referred to as *localhost* which is always the name for the machine you are sitting in front of and is denoted by the IP address, 127.0.0.1.

A network connection is identified by the IP address of the local machine, the port on the local machine, the IP address of the remote machine, and the port on the remote machine. It is designated as:

```
local_IP:local_port    remote_IP:remote_port
```

When you wish to establish a connection to a service running on a remote machine, you specify its IP address and the port number the service is listening on. You are doing this implicitly when you type something like:

```
ssh hextreme.otago.ac.nz
```

What's this, no port number specified? Did I just lie to you? Actually no. The *ssh* server daemon, called *sshd*, is listening on port 22. As you know from lectures, the low numbered ports are what are called *well-known ports*. The `/etc/services` file establishes a correspondence between these ports and their programs.

Therefore when you type the above command, you are explicitly invoking the *ssh* program on your machine and implicitly giving the port number on the remote machine (in this case, port 22). The remote machine's IP address is obtained by the DNS service. Hextreme's IP address is currently 10.81.164.35.

In most cases, the local port number is chosen by the operating system from the available and unused high numbered ports.

Logically the situation corresponds to Figure 1.

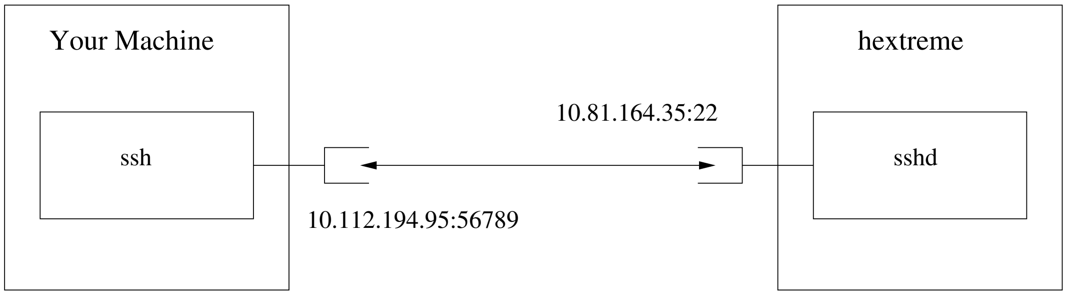

Figure 1: A Network Connection

Hextreme's end of the connection is identified by 10.81.164.35:22 indicating its IP address and port number. The IP address of your machine depends on which machine you are using. The port number used on your machine depends on the port number assigned by the operating system. In this example, I assumed you were using a machine with 10.113.194.95 for an IP address. I also assumed the operating system gave *ssh* port 56789 when starting up.

4 Preparation

In this lab, we will alter our familiar *Client.java* and *ChatServer.java* to output information about the IP address and port number of the local and remote sockets in use.

In a web browser, open the course web page: http://www.cs.otago.ac.nz/cosc244/

Click on *Resources*, then click on *Java API Documentation*. Click on the *Socket* class of *java.net*.

Q1 : What does the method `getLocalPort()` return?

Q2 : What does the method `getInetAddress()` return?

Q3 : What does the method `getPort()` return?

5 Program Exercise

For the lab part, you will need two machines. So find a place in the lab with two machines facing each other. If you are working remotely then you can use `hextreme.otago.ac.nz` and `hex.otago.ac.nz` and ports in the range 7700-7799.

Copy the files needed by typing:

```
cp -r /home/cshome/coursework/244/start/13 .
```

This gives you *ReadWriteThread.java*, *Client.java*, and *ChatServer.java*. These programs are the same as we have been using in the past labs. You will make minor modifications to both so they report some port and IP address information.

Rename *Client.java* to *Client2.java*, open it in an editor, and change it so it looks like the code given below.

```
 1  // Client2.java
 2
 3  import java.net.*;
 4
 5  public class Client2 {
 6
 7     public static void main(String[] args) {
 8        try {
 9           int port = Integer.parseInt(args[0]);
10           String host = args[1];
11           Socket socket = new Socket(host, port);
12           System.err.println("Connected to " + socket.getInetAddress() +
13                              " on port " + socket.getPort() +
14                              " using local port " + socket.getLocalPort());
15           new ReadWriteThread(System.in, socket.getOutputStream()).start();
16           new ReadWriteThread(socket.getInputStream(), System.out).start();
17        } catch (Exception e) {
18           e.printStackTrace();
19           System.err.println("\nUsage: java Client2 <port> <host>");
20        }
21     }
22
23  }
```

The class name is changed from `Client` to `Client2` in three places (One change is a comment at the beginning of the file). The other change is the insertion of the statement given in lines 12 - 14.

Q4 : Based on your preparation for the lab, what does this new statement do?

Compile the program.

Rename *ChatServer.java* to *ChatServer2.java*, open it in an editor, and change it so it looks like the code given below.

```
// ChatServer2.java

import java.io.*;
import java.net.*;
import java.util.*;

public class ChatServer2 {

   private static List<ClientHandler> clients = new LinkedList<ClientHandler>();

   public static void main(String[] args) {
      try {
         new ChatServer2().startServer(Integer.parseInt(args[0]));
      } catch (Exception e) {
         e.printStackTrace();
         System.err.println("\nUsage: java ChatServer2 <port>");
      }
   }

   public void startServer(int port) throws Exception {
      ServerSocket serverSocket = new ServerSocket(port);
      System.err.println("ChatServer2 started, listening on port " +
                         serverSocket.getLocalPort());
      while (true) {
         ClientHandler ch = new ClientHandler(serverSocket.accept());
         System.err.println("Clients ID is " + ch);
         synchronized (clients) {
            clients.add(ch);
         }
         ch.start();
      }
   }

   public static void sendAll(String line, ClientHandler sender) {
      System.err.println("Sending '" + line + "' to : " + clients);
      synchronized (clients) {
         for (ClientHandler cl : clients) {
            cl.send(sender + ": " + line);
         }
      }
   }

   public static class ClientHandler extends Thread {

      private BufferedReader input;
      private PrintWriter output;
      private String id;
      private static int count = 0;

50       public ClientHandler(Socket socket) throws Exception {
51          input = new BufferedReader(
52                new InputStreamReader(socket.getInputStream()));
53          output = new PrintWriter(socket.getOutputStream(), true);
54          id = "client_" + ++count;
55          System.err.println("Accepted connection:");
56          System.err.println("Clients host " + socket.getInetAddress() +
57                            " Clients port " + socket.getPort());
58          System.err.println("Local port " + socket.getLocalPort());
59       }
60
61       public void send(String line) {
62          output.println(line);
63       }
64
65       public String toString() {
66          return id;
67       }
68
69       public void run() {
70          try {
71             send("Welcome! You are " + this + ".");
72             String line;
73             while ((line = input.readLine()) != null) {
74                sendAll(line, this);
75             }
76          } catch (IOException e) {
77             e.printStackTrace();
78          } finally {
79             synchronized (clients) {
80                clients.remove(this);
81             }
82             System.err.println(this + " closed connection!");
83          }
84       }
85    }
86 }
```

The class name is changed from `ChatServer` to `ChatServer2` in five places (One of these changes is the comment at the beginning of the file).

The other changes are as follows:

1. The statement following line 21 is changed to the statement shown in lines 22 - 23.
2. The statement in line 26 is modified.
3. In the constructor for *ClientHandler*, lines 55 - 58 are new.

Q5 : Based on your preparation for the lab, what do lines 55 -58 do?

Compile the program.

If you have not already done so, log into your second machine. Bring up a single terminal window and `cd` to the directory you are working in.

Type `hostname` and note the name of the machine.

Type `java ChatServer2 7777` to start the server.

On your original machine, have three terminal windows open and `cd` each to the directory you are working in.

In each window, start a client by typing `java Client2 7777` *server* where *server* is the machine name you noted above.

Q6 : Using the data displayed in the server window and each client window, draw a figure showing the server and each client. The figure should look similar to Figure 1. Your figure will have three clients instead of one. Label each socket with its IP address:port number. You get this data from the information displayed in the windows.

Once you have completed your figure, leave the clients and server running so the demonstator can check your figure for correctness.

Marking

Let a demonstrator know via Teams the IP address of the machine running ChatServer2 and the port number it is listening on. Once you know the demonstator has connected to the server, type a message in one of your clients to request that your lab work be marked off.

Before leaving the lab, make sure you log off all machines.