CIS 457 - Lab 3: Working with TCP and UDP sockets programming

Due by: 9/24/2019
Total Points: 12 Points

Submission format: one hardcopy report per a group of 2 students

Lab Objectives:

The purpose of this lab is to:

- Practice with the basic TCP and UDP sockets programming
- Be familiar with the java.net, java.io and java.util network programming packages

Notes:

- Copy the server and the client sides of the applications studied in this lab to a folder in your home directory.
- Ignore any Exception handling related issues. We focus on the concepts related to network programming more than the Java language.
- Run both client and server programs on the same machine.
- Hit "CTRL + C" to close a client or a server session.

Question 1 **(3 points)**

Compile the given Java programs *TCPEchoClient* and *TCPEchoServer* such as follows:

```
javac TCPEchoServer.java   and
javac TCPEchoClient.java
```

Then, test the application by sending a simple text after running the server and the client such as follows (use 2 separate terminals):

```
java TCPEchoServer PortNumber (In our example use port # 8888)
java TCPEchoClient localhost "Hi World" 8888
```

Then, answer the following questions:

a) Describe briefly, what this application is doing.
b) What happens if you use different port number at the client side to connect to the server from what the server uses?
c) Suppose you run TCPClient before you run TCPServer (try to do this if you wish). What happens?
d) Test your app 3 times by starting 3 clients connections from the same terminal and record the port numbers that the client used. Does the client use the same port number in each connection and why?

Keep the server up and running and solve the following questions:

e) Use the ps command with the appropriate switch to determine the Process ID for the TCPEcho application.
f) Use the netstat command (netstat -an) to determine the TCP connection state for this connection
g) Provide a screen capture for the PS and netstat commands (only as it is related to the TCPEcho application).

Question 2 **(3 points)**

Compile and run the given two java programs TimeClient and TimeServer. You can run this app such as follows:

@ The Server Side
java TimeServer 1234
@ The Client Side
java TimeClient localhost 1234

a. Describe briefly, what this application is doing.
b. What is the return type of the read () method in this program?
c. Why is there a single stream attached to either the client or the server side in this application?
d. What is the type of I/O streams that is attached to the sockets?
e. What are the limitations of using this type of I/O stream?
f. Provide a screen capture for the two terminals that are used in testing the application.

Question 3 **(2 points)**

Now, compile and run the same *Time Server* application using different implementation (TimeClient2 and TimeServer2).

a. How different is this implementation from the previous one?
b. What is the type of I/O stream that is attached to the sockets?
c. In TimeClient2 and TimeServer2, what does the following code do?

```
DataInputStream in = new DataInputStream(new
BufferedInputStream(server.getInputStream()));

DataOutputStream out = new DataOutputStream(new
BufferedOutputStream(client.getOutputStream()));
```

d. Which implementation (TimeClient and TimeServer) or (TimeClient2 and TimeServer2) is efficient and why?

Question 4 **(2 points)**

Compile the Java programs *UDPClient* and *UDPServer*. Run the server using the command *java UDPServer* and run the client using the command *java UDPClient*. Then test the application by sending a simple text. Make sure to change the "hostname" in the UDPClient file to the proper name.

a) Provide a screen capture for the testing process that shows the application's output in the 2 terminals.
b) Suppose you run UDPClient before you run UDPServer. What happens?

Suppose that in the UDPClient.java we replace the line

```
DatagramSocket clientSocket = new DatagramSocket();
```
with
```
DatagramSocket clientSocket = new DatagramSocket(5432);
```

c) Will it become necessary to change UDPServer.java? and why?

Question 5 **(2 points)**

One of the classes within the *java.net* package is called InetAddress, which handles Internet addresses both as host names and IP addresses.

a) Compile the given IPFinder.java program and describe its functionality.
b) Modify the given *IPFinder.java* program so that it takes a list of host names from the command line and prints the host name and the IP address (s) for each host specified in the user input. The program should be able to retrieve the IP address of the local machine. (**Hint**: you need to use the *getAllByName(), getHostName() and getHostAddress()* methods to complete this part). Feel free to use the supplied code in the file *IPFinder2.java* that walks you through the process and fill in any missing info **or** create your own code from scratch. The program output should look as it is shown in the figure below.
c) Provide a screenshot and the program code for the testing process. You can use any two domains that you may wish for the testing process.