Lab 10: Socket Programming

What you will do:

- Implement Server and Client java programs.
- Use Wireshark to capture/explore/analyze communication packets.

Things that you will need to know or learn:

- Everything that you learned in previous labs you will need to complete this lab.
- Understand sockets and ports.
- Create comprehensive network applications using sockets.

What you need to submit and when:

- Complete both the Server and Client programs and submit them on BrightSpace before due time.

Required Equipment and Software:

- Equipment requirements:
  - Wireshark installed and working on your laptop (done in Lab 01)
  - Lab 10 documents downloaded to your laptop
  - Two laptops connected by a Linksys Router

    NOTE: You can use one laptop with both client+server in case you have no partner.
- Programming IDE and software:
  - Eclipse for Java or any JAVA IDE (or a notepad and command line interface)
  - Wireshark

References and Resources:

- Lab 01 - 9
- Cisco Modules learned during the semester
- Socket programming reference document

TCP/IP SOCKET PROGRAMMING

The two key classes from the **java.net** package used in creation of server and client programs are:

- ServerSocket
- Socket

A server program creates a specific type of socket that is used to listen for client requests (server socket), in the case of a connection request, the program creates a new socket through which it will exchange data with the client using input and output streams. The socket abstraction is very similar to the file concept: developers have to open a socket, perform I/O, and close it.

A simple client program normally needs the following steps to communicate with a server program:

1. Create a Socket Object:

```
Socket client = new Socket(server, port_id);
```

2. Create I/O streams for communicating with the server.

```
is = new DataInputStream(client.getInputStream());
os = new DataOutputStream(client.getOutputStream());
```

3. Perform I/O or communication with the server:

Receive data from the server - for example: **`String line = is.readUTF();`**

Send data to the server - for example: **`os.writeUTF("Hello\n");`**

4. Close the socket when done:

```
client.close();
```

Task 1: Write the Server/Client program

In this task, you will write two simple server/client programs:
TextServer and TextClient. TextServer creates a socket on port **18990**.

TextClient establishes a connection to the server, read the text message sent by the server and display it on the console. The program should keep running until a "goodbye" message received. The messages sent from the server are the messages the user input at the console.

The programs should meet the following requirements:

- After done with one client, the server keeps running and another client may connect to it.
- Resources should be closed before the programs stops.

Task 2: Wireshark Capture Analysis

In this task you will capture the traffic resulting from socket communication between the server and the client. In particular you will focus on the TCP PDU.

1. Start a Wireshark capture on your laptop.
   NOTE: if you are running both the Server and the Client programs on one laptop, choose the network interface "Adapter for loopback traffic capture" in Wireshark.
2. Run the code TextServer on the server laptop.
3. From the Command Prompt window, enter the following command: **`netstat -a | findstr 18990`**
   Take a screen capture of the output and save it as **Netstat_TCP.png**
4. Run the code TextClient on the client laptop.
5. Filter Wireshark capture by the tcp port number.
6. Stop and save your Wireshark capture as **WS-task2.pcapng**
7. Examine the Wireshark capture to ensure that the desired traffic has been captured: 3-way handshake, client-server communication, and teardown. take a screenshot with all these frames and the full frames values from Wireshark and save as: **WS-task2.png**

Task 3: Demo, Cleanup and Other Tasks

1. Demo your program to your lab instructor during your lab session.
2. Carefully pack the classroom equipment and place in the closet.
3. **Indicate with a comment in your Java file your name and the name of your partner in this lab.**
4. Submit to Brightspace: Only the .java code you wrote (Do not submit your partner's code) as well as the two screenshots you took: **Netstat_TCP.png ; WS-task2.png.**