Practical Exercises 4

Development of a basic TCP Client and TCP Server. The Server offers a text transformation service with two functionalities: (i) removal of tabs and extra spaces; and (ii) removal of tabs and extra spaces with random re-ordering of the words.

Task 1: Client Specification (Client.java class)

- The client should accept as input parameters both the server's IP address and port number. For example: `java ClientTCP 192.168.166.121 12345`
- Once the client is connected to the server, it will receive a message that indicates that the client is connected to the service.
- Once the client receives this message, it should request two input textual lines to the user. The first line is the key of the service that the client requests and the second is the text to be transformed. The key of the service will be send using `print(key+"\r\n")` and the text using `println`.
- After each message delivery, the client should wait for the server's response, and print it in the output console. This response could be an error message or the transformed text. For example, if the client sends 'b' and "`Hello, how are you doing?`", the server could answer "`are you Hello, how doing?`".
- We assume that the information sent by the client is correct and the program does not need to check the text (it does not contain accents or ñ).
- When the user decides to finish the service, it should write '`F`'. Take into consideration that when the user introduce '`f`' as letter, the program will not read the following text.
- When the client reads '`f`' from the standard input, it sends '`f`' to the server using `println`, waits an answer from the server (`OK`) and closes the connection.
- The client should print out appropriate informative messages about its state during the session as it executes (e.g. `Connected to 192.168.167.2:12345, Waiting for a response, …`)
- If the client sends data, but it finds out the connection was closed by the server, then it finishes it execution.

Task 2: Server specification (Server.java class)

- The server should accept as input parameter the port number for accepting connections. For example: `java Server 12345`
- Once a connection to a client is established, the server will send the message "Welcome to the text transformation service" (without quotation marks).
- Immediately after, the sever tries to read from the "connected socket" the letter and the text (two lines that will be read using `readLine` two times).
- The answer of the server depends on the option selected by the client:
  - L: The server remove extra spaces and tabs. The result is a text with only one space between each word.
  - B: In addition to the functionality of the L option, this service randomly re-order the text.
  - F: The server does not read the second line, sends '`OK`' to the client and closes the connection.
  - Any other option: The server send "`Not supported option`".
- This is an iterative server, so after finishing one service provision, the server should wait for another service request.
- As the client, the server should print out appropriate informative messages about its state as it executes (e.g., Waiting for a new client …).
- The server should have only one waiting client (the queue length of pending clients must be 1).

Task 3: Capturing traffic (traces)

Execute the following scenarios with the help of a partner, and capture traffic with Wireshark.

Scenario 1 (trace 1 – **p4e1-7.pcapng**):

- Launch the server and afterwards the client.
- The client sends a pair letter and text, and right after `F` to finish the session.

Scenario 2 (trace 2 – **p4e8.pcapng**):

- Launch a client without a server active.

Scenario 3 (trace 3 – **p4e9-10.pcapng**):

- Launch the server in one machine.
- Then, execute 3 clients and try to connect with the server (in this experiment the clients could be in the same machine).
- Write `F` in the console of those clients that had success in connecting to the server.

Task 4: Analyzing our Echo protocol in TCP

Answer to the following questions after analyzing the above execution traces.

Trace 1 (**p4e1-7.pcapng**):

Exercise 1. Use the "Analize -> Follow -> TCP stream" option of the Wireshark to see the information interchange between the client and the server. Show a screenshot with this information.

Exercise 2. Does the client send the messages (letter and text) in the same TCP segment? Why?

Exercise 3. What is the port used by the client? What is the one of the server? What are the fields of the TCP header where they are stored?

Exercise 4. What is the initial sequence number used by the client? What is the initial sequence number used by the server?

Exercise 5. Take screenshots of the TCP segments responsible of the following activities.
a) Initiate a connection
b) Data sending
c) Finish a connection.

Exercise 6. How many sequence numbers are in each communication end (client and server) consumed during the initialization and closing of the connection?

Exercise 7. Observe the size of the sliding window that is transmitted in the segments that travel in both directions (client and server). Does the sliding window size change? What are its values in the client and in the server?

Trace 2 (**p4e8.pcapng**):

Exercise 8. Does the client receive any response after trying to connect to the server? If the answer is positive then, what are the special characteristics of this response?

Trace 3 (**p4e9-10.pcapng**):

Exercise 9. Did the 3 client manage to connect? If one has not been able to connect, does it receive a notification about full queue?

Exercise 10. Do the waiting clients (those queued ones) have the connection initialised? Or the connection initialisation is performed after the clients are pulled from the queue (in the `accept` method)?

Report guidelines

- It is recommended the use of the provided Word template.
- Include in the same report all the exercises corresponding to each part (part 4 and part 5).
- If the proposed exercise consists of implementing some code, the report should include an explicative schema detailing the most significant parts of it. You must deliver the source code (in independent files) jointly with the report.
- For each exercise include both the statement of the exercise as well as the solution. Define custom styles for the different parts of the document (e.g. title, statement, solution, etc.)
- For those exercises whose solution is the output of some process (i.e. capture of traffic with Wireshark, a command, …) make a screenshots (using +). In each screenshot the student must mark those parts corresponding to what is requested by the exercise (using a drawing utility). Also add a brief text explaining what is shown.
- The student must upload the report in PDF.
- Upload a .zip file with the report, the file .pcap with the trace of the captured packets and the source code of developed programs.