Practical Exercises 5

Development of a basic UDP Client and UDP 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 (ClientUDP.java class)

- The client should accept as input parameters both the server's IP address and port number. For example: `java ClientUDP 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. These two lines should be concatenated and sent to the server.
- 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, ClientUDP finishes its execution (it does not send any information to the server).
- 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, …`)

Task 2: Server specification (ServerUDP.java class)

- The server should accept as input parameter the port number for accepting connections. For example: `java ServerUDP 12345`
- The server is always waiting for incoming datagrams that includes a letter (the transformation service selected) and the text to be transformed.
- The size of the reception buffer should be 20 bytes.
- The server sends the transformed text to the client that sent the datagram.
- 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.
  - Any other option: The server send "`Not supported option`".
- As the client, the server should print out appropriate informative messages about its state as it executes (e.g., Waiting for a new client …).

Task 3: Capturing traffic (traces)

Execute the following scenarios and capture traffic with Wireshark. The server and the client are running in the same machine, deploy the server in the loopback IP address (127.0.0.1). Capture the network traffic from the interface named "Adapter for loopback traffic capture".

Scenario 1 (trace 1 – **p5e1-5.pcapng**):

- Launch the server and afterwards two clients.

- Send some messages from the two clients.
- One of the clients send a long message (more than 20 characters).
- Finally, write `F` to finish the execution of both clients.

Scenario 2 (trace 2 – **p5e6-7.pcapng**):

- Launch a client without a server active.

Task 4: Analyzing our protocol in UDP

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

Trace 1 (**p5e1-5.pcapng**):

Exercise 1. How many messages are exchanged in the interaction between the client and the server? If the protocol had been TCP, how many messages would have been sent (you can suppose that the option and the text are in the same message)?

Exercise 2. What is the port used by the client? What is the one of the server? What kind of ports are they?

Exercise 3. Wireshark has the option “Follow UDP stream” but in UDP the concept of flow of messages does not exist. How can Wireshark decide that a UDP datagram is from a specific flow of messages?

Exercise 4. Analyze a message sent by the server and check the length field of the UDP header. Is this value the same as the size of the payload (application data) transported by the datagram? Why?

Exercise 5. Analyze the long message (more than 20 characters). What is the size of the payload in the datagram sent by the client? And in the datagram sent by the server? What happens with the data that cannot be included in the datagram because of the size of the reception buffer?

Trace 2 (**p5e6-7.pcapng**):

Exercise 6. Why is to sent a message possible when there is not a server running? Is an answer received? If the answer is yes, what is the meaning of this message.

Exercise 7. If the execution of the client is interrupted, will a closing message be sent? Why?

No trace:

Exercise 8. Try to open two times a server with the same input parameters and capture the Java exception thrown by the program. If the code was not prepared to capture that exception and show it (mehthod *getMessage()*), modify the code for doing so. What kind of error is notified? How can you modify the server to avoid this error and to make possible to have two servers with the same parameters running at the same time?

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.