Università della Svizzera italiana

Faculty of Informatics

Distributed Systems **Autumn 2024/25**

Code Assignment 1 Chat Server Part 1: Establishing a Connection

The deadline to uploading your source code to iCorsi is on the 10th of October, at 23:59. Late submission policy will apply; see the Intro slides for details. Upload a single Python project, with separate main files for each exercise. You can use the provided `template.py` as a starting point. If you are using a language other than Python, include a Dockerfile that builds and runs your project.
Please take extra care to only submit source code (and not build artifacts).

Exercise 1 - (2 points)

Make a simple server that accepts a TCP connection on port 8080 from a client, receives a text message, prints the message, and then terminates.

You can test your server by running it and then typing `telnet 127.0.0.1 8080` on a separate terminal. In order to run the server, use your IDE or the `python3` command-line utility.

Exercise 2 - (3 points)

Time to create our own client. Make a program that reads a user text input and sends it to the server. The client should also receive the reply for the server and display it.

To test the interaction between client and server you need to run each one in a separate terminal. Pay attention to the running order, as a client cannot connect to the server if the server is not running yet.

How can we allow more than one connection at the same time? The following exercises will help with that.

Exercise 3 - (2 points)

Make a program that spawns three threads, gives each thread an identifier, and makes them greet the user by saying "Hi, I'm thread *id*", with *id* being the identifier of the thread. After the greeting, each thread should sleep for a random amount of time and then display a farewell message.

To test your threads, simply run the source file containing them.

Exercise 4 - (3 points)

Modify the simple server created in Exercise 1 so that it spawns one thread to handle each connecting client. Each thread should receive messages and reply until the “end” message is received. The client code will also need a small update to enable the sending of multiple messages and termination after sending “end”.

To test your updated server, interact with it with multiple clients.

Is this the only way to handle multiple connections at the same time? Reflect on possible alternatives.