NOVA SCHOOL OF SCIENCE & TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

Programming with TCP sockets in Python (Lab-02 2023/24)

0. Introduction

As being discussed in lectures, processes running in distinct machines can communicate with each other using **sockets** which are devices allowing the access to Internet transport protocols like TCP and UDP.

In this class we will program with TCP sockets.

Learning Materials:

- Lecture slides available in CLIP
- Python basic concepts can be studied in several online texts namely *Python for Everybody: Exploring Data in Python3* by Charles R. Severance et al, available (including a tar file with the code of the examples used in the book) at https://www.py4e.com/book.php
- A Python socket tutorial is https://docs.python.org/howto/sockets.html
- Another Python socket tutorial https://realpython.com/python-sockets/

1. Python Program (example)

Get the source code of *TCPclient.py* and *TCPserver.py* available from CLIP. As with Lab01, the first exercise is to analyze the provided source code and run it.

2. Adding two numbers

In this assignment, you'll write a client that will use TCP sockets to communicate with a server that you will also write. Here's what your client and server should do:

Client

a) Accept a name and an integer between 1 and 100 from the keyboard;
b) Create a TCP socket and open a connection to the port agreed with the server;
c) Send to the server: (i) the entered name and (ii) the entered integer value and then wait for a server reply;
d) When the client receives the reply from the server it should process it and then display:
    I. its name and the server's name;
    II. its integer value, the server's integer value and the sum of both;

The client then terminates after closing any created sockets. As a note (and as a check that you are doing things correctly) you should make sure for yourself that the values and the sums are correct!

Server

a) Create a string containing a name (e.g., “Server of John Q. Smith”) and a random number;
b) Create a TCP socket, bind it to the agreed port, and wait for a connection;
c) On accepting a client connection, the server should:

   i. Display the client’s received name and the server’s name;

   ii. Display the client’s number, its number, and the sum of those numbers;

   iii. Reply to the client with the server name and the server number;

   iv. If your server receives an integer value that is out of range, it should terminate after closing any created sockets. You can use this to shut down your server.

3. File Transfer

You should write a file transfer program where:

- The server is invoked with the following command line:
  `python TCPserver.py port`
- The client is invoked with the following command line:
  `python TCPclient.py server:port:fileNameInServerFileSystem fileNameInClientFileSystem`
- The client connects to the server
- The client sends the file name to be transferred
- The server sends the file to the client in 1024 bytes blocks
- Client receives the blocks, sent by the server, and writes them to its local disk.
- Compare the contents of the two files (windows: fc command; linux and macOs: diff command).