Lab 8A

GUI server

[Optional]

Author: Johannes Schmidt

This is the first of four optional labs. The optional labs can give together up to 10 *bonus points*. If all regular labs are passed by the general deadline, the accumulated bonus points will count as a bonus on the exam. The scale is that 10 bonus points correspond to 10% of the exam points. This *bonus rule* is only applicable to the first exam (December), and only if the exam is passed without the bonus points.

In order for this lab to qualify for bonus points, you must present and submit your solution before the first exam.

The 10 bonus points are distributed among the optional labs as follows:
Lab8A - 3 points
Lab9A - 1 point
Lab9B - 3 points
Lab11A - 3 points

In this lab you have a chance to further advance your skills in socket programming and GUI programming.

You will program a server with a graphical user interface.

As this is an optional lab, the instructions are less detailed than usually.

0.1 Requirements

The requirements on your GUI server are the following:

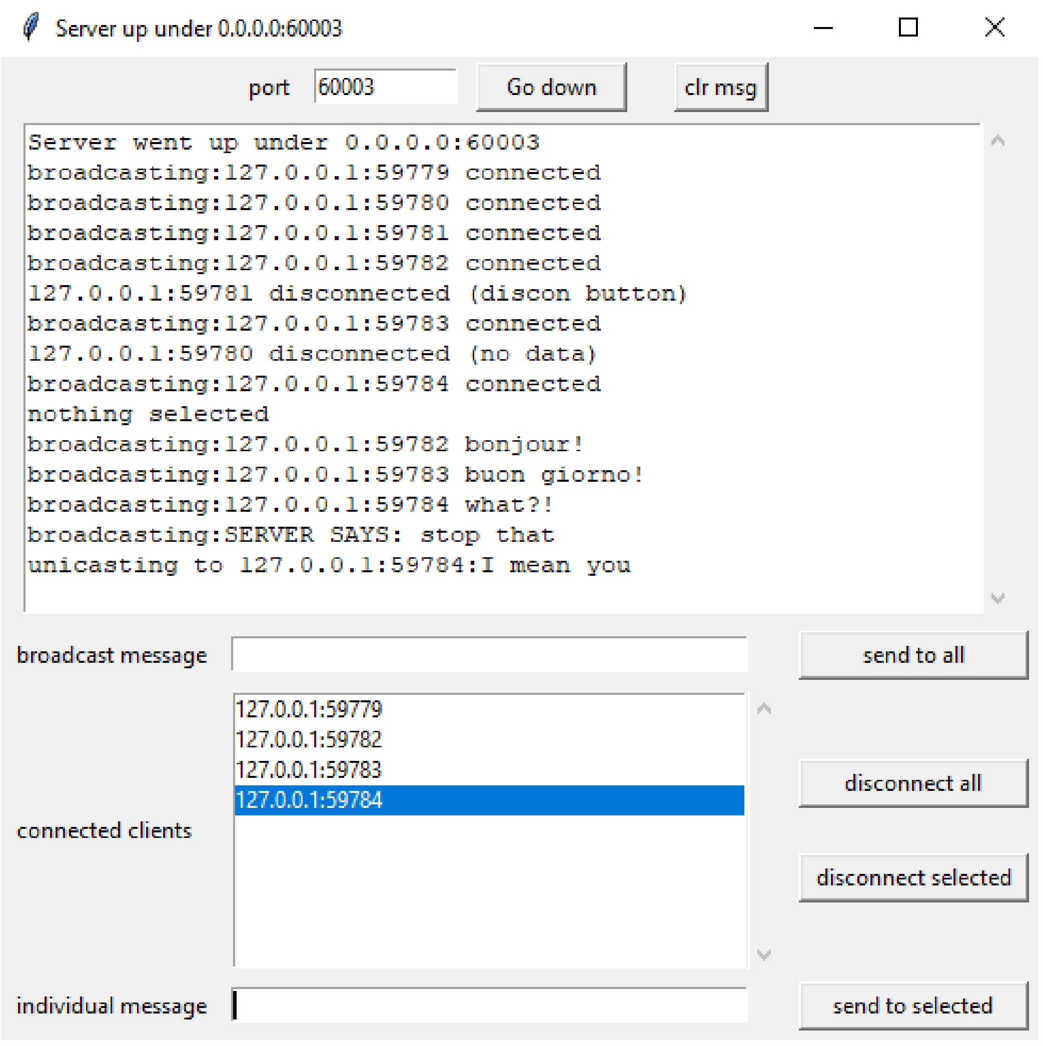
1. From a client's perspective the server works and behaves **exactly** as the server from lab 7.
2. It is possible to set the server up (listening for incoming connections) and to take it down (not listening, all clients are disconnected). E.g. provide buttons 'go up' and 'go down'.
3. The gui displays all the messages that are sent and received, just as the client gui from lab 8 does.
4. It is possible to send broadcast messages (from the server to all clients).
5. It is possible (provide a corresponding button) to disconnect all connected clients (while the server continues to be 'up').
6. The gui shows a connected-clients-list (scrollable text field) with all the currently connected clients. It updates as clients connect or disconnect.
7. In the connected-clients-list it is possible to select (mark/highlight) a certain client.
8. It is possible to disconnect the currently selected client.
9. It is possible to send an individual message to the selected client.

Hints

- The server needs to do multiple things simultaneously: listen for incoming connections, serve all the connected clients when needed, take care of user input (from the gui). As outlined in the last lab there are multiple solutions to this. A solution we have not explicitly proposed yet is to use a select statement as in lab 7, but in non-blocking mode. You do this by providing as a third argument a timeout of 0.0: `select(listOfSockets, [], [], 0.0)` This could be an elegant solution to implement *polling*.
- In figure 1 is a screen shot of my solution.

0.2 Submission and presentation

Submit your python file on Canvas and present your code and program to your lab assistant. Be prepared to explain your code.



```
Server went up under 0.0.0.0:60003
broadcasting:127.0.0.1:59779 connected
broadcasting:127.0.0.1:59780 connected
broadcasting:127.0.0.1:59781 connected
broadcasting:127.0.0.1:59782 connected
127.0.0.1:59781 disconnected (discon button)
broadcasting:127.0.0.1:59783 connected
127.0.0.1:59780 disconnected (no data)
broadcasting:127.0.0.1:59784 connected
nothing selected
broadcasting:127.0.0.1:59782 bonjour!
broadcasting:127.0.0.1:59783 buon giorno!
broadcasting:127.0.0.1:59784 what?!
broadcasting:SERVER SAYS: stop that
unicasting to 127.0.0.1:59784:I mean you
```

```
127.0.0.1:59779
127.0.0.1:59782
127.0.0.1:59783
127.0.0.1:59784
```

Figur 1: A screen shot of Johannes' gui server in action.