Socket Programming Assignment
Computer Networks
Fall 2024
Due: Oct 21, 2024

- This is a group assignment – two students are allowed to each group.
- The allowed programming languages is Python.
- The source code and project report are to be uploaded to the assignment Dropbox before the deadline. Only one student needs to submit the report and code, but make sure both names are included on the code files as well as the report.
- Plagiarism in any form will be treated as per "Laurier University Academic Integrity guidelines".

1. The report should detail:
    a. A brief description of the code, outlining the steps, operation, and any special considerations (if any).
    b. Any difficulties faced and how they were handled (if any)
    c. Test result: Make sure to test all items in the rubric and provide the necessary screenshot of your testcase outputs in the report.
    d. Possible improvements: If you had given more time, what would you add or do differently?

2. Source file (Use comments to document/describe your code)

- Assignment demonstration is mandatory. You need to agree a timeslot with GTA. A shared excel file will be provided to book your demo time. Demonstration requires you to show the items listed in the Rubric.
- Submit the following three files in the Assignment 1 Dropbox folder
  Server.py
  Client.py
  Report.pdf

Description

Programming assignments are meant to help you better understand the networking concepts taught throughout the course. In this assignment, you will implement a simple client-server communication application. Detailed requirements are found below.

Requirements

1. Design a client-server communication application (chatting application between clients and the server).
2. Clients and the server are to communicate using **TCP sockets ONLY**. You will launch **only one server**, with as many clients as the server can serve.
3. Once created, clients are assigned a name in the form of ["Client" + incremental number starting with 1]. That is, the first client is "Client01", the second client is "Client02", and so on.
4. Once a connection is accepted, the client will communicate its name to the server.
5. The server will maintain an in-memory cache of accepted clients during the current session, along with the date and time the connection was accepted, and the date and time the connection was finished.

Cache will be only in memory, no need to use files for cache.

6. The server can handle a limited number of clients. you will hard code or configure this number, **let's assume 3 clients**.
7. Points 3 and 6 are easier to implement on the server side, but you can implement them on both sides if desired.
8. Once connected and after sending its name to the server, a client can send any string using CLI (Command Line Interface). The server on the other side should echo the same string back, appending the word "ACK".
9. A client can send a message "status" to the server to ask the server for the content of the cache. The server prints the content of the cache in response.
10. Once a client finishes sending messages, it can send an "exit" message to terminate the connection. Upon receiving a connection closure request, the server should close the connection to free resources for other clients.
11. **Bonus**: the server will have a repository of files. After accepting the client's connection, the client sends a "list" message to the server. The server should then reply with the list of files in its repository. The client can then request a file by name, and the server should stream the file to the client. You should handle all cases, like when a file name doesn't exist.

If any additional details are required beyond those outlined above, then it is up to you to decide on. However, all your design considerations must be detailed in "Brief Description" section of the report. In the "Possible Improvement" section, describe any things you might want to do if you are given more time.

Rubric

Project (No partial credit under each item, it is either fully met or receives no points)

1.	Program can create a Server and client (10 points) +demo (10 points)	20 points
2.	Each client created is assigned a name with correct number (5) +demo (5)	10 points
3.	Server can handle multiple clients (multi-threading) (10) +demo (10)	20 points
4.	Server limits the number of connected clients to 3 clients at a time (5) +demo (5)	10 points
5.	Server and client can exchange messages as described (10) +demo (10)	20 points
6.	A client can send "exit" and server cleanly disconnects the client for new clients	10 points
7.	Server maintains clients' connections details – sent to the client when client sends the status message to the server (10) +demo (10)	10 points
8.	Server lists files in its repository (10) +demo (10)	Bonus: 10 point
9.	Server streams files of choice to clients (10) +demo (10)	Bonus: 20 points