CISC 335 – Computer Networks – Winter 2021
Programming Assignment 1/2

Due March 1st, 2021, 11:59 pm

- This is an individual project – You can consult with peers, but codes have to be written individually. Plagiarism in any form will be treated as per "Queen's University Academic Integrity guidelines"
- Allowed programming languages are Python and Java. Once selected, you must use it to code the entire assignment. Your Assignment cannot be written in different languages.
- The source code and project report are to be uploaded to OnQ before the deadline.

Deliverables

1. Report, detailing:
    a. Brief description of the code steps, operation, and considerations (if there are any considerations).
    b. Difficulties you faced and how you handled them (if any)
    c. Possible improvements: If you had more time, what would you add or do differently?
2. Source file (Use comments to document/describe your code)

Description

Programming assignments are meant to help you better understand the networking concepts taught throughout the course.

As a start, in this assignment, you will be required to implement a simple client-server communication application (a chat-like service). Detailed requirements are found below.

Requirements

1. Design a client-server communication/chatting application.
2. Clients and server are to communicate using **sockets ONLY** (TCP). You will launch **only one server** and 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, first client name will be "Client#01", second client is "Client02", and so on.
4. Once the connection is accepted, the client will communicate his/her name to the server.
5. The server will maintain a cache of accepted clients during current session along with date and time accepted, and date and time finished. Cache will be only in memory, no need to use files.
6. A server can handle a limited number of clients (you will hard code or configure this number, **let's assume 3 clients**).
7. Points 3,6 are easier to be implemented on the server side. But you can choose either sides.
8. Once connected and its name sent to server, a client can send any string using CLI. The server on the other side should echo the same string with the word "ACK" attached.
9. Once a client finishes sending messages, it can send an "exit" message to terminate the connection. On the server side, and upon receiving a connection closure request, the server closes the connection to free resources for other clients.
10. As a bonus: the server will have a repository of files. After accepting the client's connection, the client sends the "list" message to the server. The server should then reply with the list of files in its repo. The client can send the name of any file to the server, and the server should stream/serialize the file to the client. You should handle all cases, like sending a file name that doesn't exist.

If there is a detail not included in the requirements above, then it is totally up to you to decide on. However, all your code design considerations must be detailed in "Brief Description" section of the report. In the "Possible Improvement" section, you can detail all the things you might want to do if you had the 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 — 2 points
2. Each client created is assigned a name with correct number — 1 point

3. Server can handle multiple clients (Multi-threading) 2 points
4. Server is able to force number of connected clients to 3 clients 1 point
5. Server and client can exchange messages as described 2 points
6. A client can send “exit” and server cleanly disconnects the client for new clients 1 point
7. Server maintains clients’ connections details 1 point
8. Server is able to list files in its repository Bonus: 1 point
9. Server streams files of choice to clients Bonus: 2 points

Sample test cases

Make sure to test all items in the rubric.