CISC 435 – Computer Networks – Fall 2019
Programming Assignment 1/3

Due October 10th, 2019 23:59

- 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"
- Use any programming language of your choice. Java is preferred for better support.
- 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 any).
    b. Difficulties you faced and how you handled them (if you faced no difficulties discuss the scalability of your app and how you designed for it)
    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

This assignment is meant to help you better understand the networking concept taught throughout the course. By the end of assignment 3, you will have demonstrated the TCP socket communication, DHCP and DNS application layer services/protocols, and the TCP flow control.

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 or UDP). 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 accepted, client will communicate his/her name to 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 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 name sent to server, client can send any string using CLI. Server on the other side should echo the same string with the work "ACK" attached.
9. Once a client finishes sending messages, client can send an "exit" message to terminate connection. On the server side, and upon receiving a connection closure request, server closes the connection to free resources for other clients.
10. As a bonus: server will have a repository of files. After accepting client's connection and upon sending the "list" message to server, server should reply with list of files in its repo. Client can send the name of any file to server, and the server should stream/serialize the file to 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 "Bried 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 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.