Assignment 1
Socket Programming
CSE 322, July 2021

---

1. Overview

In this assignment, you will develop a special purpose file server where students (clients) will be able to upload, download and request for files. **You have to use JAVA programming language for the implementation.**

2. Core Functionalities

1. Each student has a specific IP address. A student can Login (i.e., connect to the server) with his/her student ID. If the student ID is already logged in from another IP address, new login is denied by the server and the connection is immediately terminated with proper message. So, each student is mapped to a single IP address at any instance. When a student gets connected for the first time, the server creates a directory with the student ID as directory name. All the files uploaded by this student will be stored in this directory.
   Update: For the ease of implementation and testing, you may drop the IP address constraint. Just ensue that no student is connected with multiple clients at the same time.

2. While connected, a student can:
   a. Look up the list of all the students who were connected at least once to the server. The students who are currently online should be distinguishable in the list.
   b. Look up the list of all the files (both private and public) uploaded by him/her and download any of these files. Private and public files should be distinguishable in the list.
   c. Look up the public files of a specific student and download any of these files.
   d. Request for a file. This request (short file description and request id) will be broadcasted to all the connected students.
   e. View all the unread messages.
   f. Upload a file.
      i. The file can be either public or private and it will be specified by the uploader.
      ii. If the file is uploaded in response to a request, it must be public and the uploader will provide valid request Id.
   g. When a requested file has been uploaded, the server sends a message to the person who requested for that file. Multiple students can upload the same requested file and the server will send as many messages.

3. Implementation Details

1. Configurable parameters for the server:
   a. MAX_BUFFER_SIZE
   b. MIN_CHUNK_SIZE
   c. MAX_CHUNK_SIZE
2. While uploading a file, sender first sends the filename and file size to the server. The server checks the total size of all chunks stored in the buffer plus the new file size. If it overflows the maximum size, the server does not allow the transmission. Otherwise, the server randomly

generates **Chunk Size** (between MIN and MAX CHUNK SIZE) and sends a confirmation message to the sender with the **Chunk Size** and a **fileID**, which is used as the file identifier for the rest of the file transmission. So, you need to maintain a link between the **fileID** and **fileName** and other info inside the server.

3. Now the sender splits the file into chunks depending on the **Chunk Size**. Let, a file size is 1235 KB and the server allows that client maximum 100 KB for each chunk. So, client splits the file into 13 chunks (first 12 chunks with 100KB, the last chunk with 35 KB) and sends all the chunks sequentially.
4. After receiving each chunk, the server sends an acknowledgement. The sender sends the next chunk only after receiving the acknowledgement. If the sender does not receive any acknowledgement within 30 seconds, it sends a timeout message to the server and terminates the transmission. After receiving the timeout message, the server should delete the chunk files for the corresponding fileID.
5. When the sender receives acknowledgement for the last chunk, it sends a completion message to the server. Then the server checks the file size by adding all the chunk sizes. If size of all chunks matches the initial file size mentioned by the sender, the server is done with the sender and sends a success message. Else, the server sends an error message indicating failure and deletes all the chunks.
6. A student goes offline if he/she logs out or gets disconnected. If the sender goes offline in the middle of a file transmission (between sender and server), discard the files from server.
7. Unlike upload, the server does not carefully monitor download of a file. It sends the file with the chunk size of MAX_CHUNK_SIZE and does not wait for the receiver's acknowledgement. When the download is completed the receiver gets a completion message from the server.

4. Bonus Tasks:

1. During, upload and download the client will be able browse other features of the server.
2. If an attempt to upload a file is likely to cause overflow of server's buffer, the sender will be put on queue. In the meantime, the sender will be able perform other tasks.
3. Implement GUI for the clients.

5. Marks Distribution

Task	Marks
Connect with the server	5
Lookup student list	5
Lookup own file list	5
Lookup public file list of others	5
Upload file	35
Download file	15
Message Management	20
File Request	5
Proper Submission	5
Bonus tasks	10+5+5=20
Total	**100 + 20**

6. **Submission Guideline:**
    a. Put your source codes in a folder named “1705xxx”, then zip it into “1705xxx.zip” and submit on Moodle. Your submission must not contain any other files except the source codes.
    b. Deadline: 11:55 PM, Saturday, November 27, 20201.
    c. **If you are involved in plagiarism, you will get Negative 100% Marks.**