Assignment(Lab4): File Transfer Using Sockets

Objective

Design and implement a program that allows a client to upload a file to a server using TCP sockets. The server will receive the file, save it locally, and acknowledge the successful upload.

Problem Statement

1. **Client Responsibilities:**
   - Connect to the server.
   - Send a file to the server using TCP sockets.
   - Ensure the file is sent in chunks to handle large files efficiently.
2. **Server Responsibilities:**
   - Accept connections from clients.
   - Receive the uploaded file in chunks and save it locally.
   - Detect the end of file transfer and acknowledge the client.

Requirements

1. **File Transmission:**
   - The client must send the file in chunks (e.g., 1024 bytes per chunk).
   - The server must reassemble the file from the chunks and save it locally.
2. **Protocol:**
   - The client first sends the file name to the server.
   - The client sends the file content in chunks.
   - The client sends an `EOF` marker to indicate the end of file transfer.
   - The server sends an acknowledgment (`ACK`) after successful file receipt.

Deliverables

1. **Server Script:** Python code to receive and save the file.
2. **Client Script:** Python code to send a file to the server.
3. **ReadMe File:** Instructions for running the server and client programs.

Example Interaction

Client Console Output:

```
Connected to the server.
Sending file: example.txt
File example.txt sent successfully.
```

Server Console Output:

```
Server is listening on port 8080...
Connected to client at ('127.0.0.1', 54321)
Receiving file: example.txt
File example.txt received successfully.
```

Marking Scheme (10 Marks)

1. **Functionality (5 Marks):**
   - Full functionality with no errors: 5 marks.
   - Partial functionality (e.g., handles file transfer but lacks proper error handling): 3-4 marks.
   - Basic functionality only (e.g., sends file but no EOF handling): 1-2 marks.
2. **Code Quality (3 Marks):**
   - Clear, modular code with proper functions and comments: 3 marks.
   - Some modularity but lacks sufficient comments: 2 marks.
   - Poorly structured, hard-to-follow code: 1 mark.
3. **Error Handling and Robustness (2 Marks):**
   - Handles all possible errors gracefully: 2 marks.
   - Handles some errors (e.g., file not found): 1 mark.
   - No error handling: 0 marks.

Instructions for TAs

- Ensure that students' submissions include both server and client scripts.
- Test the functionality with a sample file.
- Verify the handling of edge cases such as missing files, empty files, and network interruptions.
- Ask them not to forget submitting the zip file (client and server script) to the email address that was shared earlier.

Deliverables

1. **Server Script:** Python code for the server.
2. **Client Script:** Python code for the client.

Submission Guidelines

- Submit all deliverables in a zip file named as `<RollNo>_LAB<LabNo>.zip` to **csecsc307@gmail.com**
- Ensure your scripts are well-documented with comments.

Note that the TAs will give marks only if they are satisfied with your answers to their questions. Otherwise, 5 marks will be deducted straightaway, and you will be evaluated only out of 5 marks.