Designing the Protocol: Before you start coding, please design an application-layer protocol that meets the above specifications. Please submit the design along with your code. Here are some guidelines to help you get started:

1. ***What kinds of messages will be exchanged across the control channel?***

- The control channel will be used to exchange commands between the client and server. These commands include:
    - ls: sends the list of files in the current working directory on the server
    - get: transfers a file from the server to the client.
    - put: transfers a file from the client to the server
    - quit: close the connection between the client and server
    - If an invalid command is entered, then an error message will be raised.

2. ***How should the other side respond to the messages?***

- The server should respond to the client's commands based on their intended functionality.
    - For the ls command, the server will send a list of filenames in the current working directory to the client.
    - For the get command, the server will check if the requested file exists and respond with either an error message or a confirmation that the file transfer will begin.
    - For the put command, the server will wait for the client to send the file and respond with a status message indicating whether the transfer was successful or not.
    - For the quit command, the server will acknowledge the request and close the connection.

3. ***What sizes/formats will the messages have?***

- Messages will be encoded in utf-8 and have a maximum size of 1024 bytes.
- The messages sent across the control channel will be strings containing the command and any necessary arguments.
- The messages sent across the data channel will be the contents of the files being transferred.

4. ***What message exchanges have to take place in order to set up a file transfer channel?***

- The client sends a get or put command to the server, specifying the filename to transfer and the port number to use for the data channel.
- The server checks if the requested file exists and sends a confirmation message to the client (for get) or waits for the client to send the file (for put).
- The server creates a new socket for the data channel, binds it to a port, and sends the port number back to the client.
- The client creates a new socket for the data channel and connects to the server's data channel socket.

5. ***How will the receiving side know when to start/stop receiving the file?***

- For get commands, the server sends a confirmation message to the client and then begins sending the contents of the file over the data channel.

- The client continuously receives chunks of data until there is no more data to receive.
- For put commands, the client sends the file over the data channel and then waits for a status message from the server indicating whether the transfer was successful or not.

6. How to avoid overflowing TCP buffers?

- To avoid overflowing TCP buffers, the client and server can send and receive data in chunks rather than all at once.
- In our implementation, a chunk size of 1024 bytes is used for sending and receiving data.

7. You may want to use diagrams to model your protocol.

- ???