Dylan Huang : 2789356H

COMPSCI2024 Networks and Operating Systems Essentials 2 - 2023 - 24

Assessed Exercise 1

Description of the Application - Level Protocol

At the start of both the server and client file we import the functions file(1) from our parent folder in order to access the functions we will be using on both our client-side and server-side.

The protocol then creates a TCP socket and initialises the host and port numbers on the server-side. Upon a successful creation of a socket along with the host and port being initialised it binds the host and port together and can begin listening in for new connections. Here we used a backlog of 5(2). Should the socket encounter any errors, an exception error will be raised and output to the user(3). This is where the main loop begins and will print that it is actively waiting for clients to connect. Should there be a successful connection the client address is output.

On the client-side after the creation of its own TCP socket it reads input from the command line taking in the server address, request to perform and file to be used depending on the request. Should a connection not be established based on the server address an error is then raised to try again otherwise it goes on to read the request.

After breaking down the request the client socket sends the server a message indicating that a request of this type and of this name(4) is coming its way(5). Upon receiving this message, the server outputs that it is commencing this request.

If the message received from the client is a “PUT” request, the server begins to attempt to download the file. If the file already exists on the server, it sends the client a response informing the client that the file already exists on the server’s folder and an error message is output on both the client-side and server-side. If the file doesn’t exist, the client calls the send file function and the server calls the receive file function(6) to start and end the file transfer.

If the client receives “GET” request it checks if the filename exists in its folder, if the file exists it sends the server a response to cancel the file transfer and an error is output on both sides. If the filename doesn't exist on the client side the server will call the send file function and the client will call the receive file function[6].

If the client requests a "LIST" a message is sent to the server to call the send listing function and the client, then calls the receive listing function.
If the response is invalid, a message is output on both client and server side that it is an invalid operation.

At the end of the request the client socket is then closed terminating its side of the connection.
The server then closes its connection to the client socket however it continues to listen out for a new connection.

Design of the Application - Level Protocol

(1) For good file management we separate the client and server files into separate folders and functions as a neutral file in a parent folder.
(2) We use a backlog of 5 for efficiency and performance, 1 is too low but 20 would be a waste of resources.
(3) Use of sys.exit(1) to indicate the program quit with an error and sys.exit(0) to indicate there was no error to help identify errors.
(4) To allow for the sending of special characters in the filename such as "#" and "$" we have to use .encode("utf-8") as well as .decode("utf-8").
(5) For "PUT" we send the request type and filename beforehand to allow for the file's existence to be checked on the server to prevent it being duplicated or overwritten.
(5) For "GET" we check if the file exists first on the client side, taking pressure off the server side.
(6) For our functions we use a buffer size of 1024 as opposed to larger ones as we don't need a higher buffer size for the requirements we need to meet.
(7) Request handling on both the client-side and server-side are placed within Try: Exception: blocks to handle any unexpected connection drops from either side, raising an error message.