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:

- What kinds of messages will be exchanged across the control channel?

Control channel is initial channel connecting client to server, and messages that would be exchanged would be “ls”, “get”, “put”, “quit”, “nonexistent”, and “continue”.

- How should the other side respond to the messages?

If Server receives the “quit” then it will close the connection between client and server.

If client wants to download a file and sends “get”, then the server responds by first checking whether file exists or not and sends back “nonexistent” to client side if it doesn’t. Otherwise, the server will respond by starting a file download.

If client wants to upload a file it will send the “put” message, and server will respond by receiving and downloading the file data on server side.

If client wants to list out the items with “ls” message, then the server will respond by listing out all the files on the server side to display to client.

- What sizes/formats will the messages have?

The buffer size of the data will be 1024 bytes, and the format is going to be utf-8, which is a byte encoding.

From the client side, the format for messages being sent from either client or server side will be encoded with utf-8 format. Receiving side of the message will have to decode the utf-8 format. It is important to mention that when reading a file to be transferred to the receiving side, the encoding will not be utilized due to format in which the file is being read (read in byte format). This is the same when the receiving side writes the received data to the new file. The receiving side will write the data in byte format, thus removing the decoding section. This method will facilitate the of files with .mp4, .png, and .mp3 extensions.

- What message exchanges have to take place in order to setup a file transfer channel?

Client side socket will have to begin connection with a “.connect()” function call message to establish connection to the server.

Server side socket will bind the port and hostname of the server and invoke a “.listen(1) “ function call, so that there is only one connection at a time. Then server-side socket will accept the connection once the client side sends the connect call, giving the client’s socket and address.

- How will the receiving side know when to start/stop receiving the file?

Sending side of file transfer will send the total file size amount along with the bytes to send each time. The receiving side will have the total file size and will send a message to client to continue with transfer and receive file content in 1024 byte data chunk sizes, while keeping track of the current data size against the total file size to download. The receiving side will only continue receiving content while the current data size is less than the overall file size to be transferred. Once it exceeds the file content amount, then the total file content will have been received successfully and it will know to stop.

- How to avoid overflowing TCP buffers?

To avoid buffer overflows, we will set the maximum buffer size to 1024 bytes to send and to be received. Our code will parse the partial messages and account for this in a loop we implement until we get the entire message. In this case, we are limiting the buffer size to some relatively small power of 2 as recommended for the ".recv()" function call and then continuously looping until we read the entire message and no data will be lost due to overflow.

Design Diagram: