File Copy Protocol Design

1. Overview

This document outlines the protocol we are going to use for file copy assignment. The protocol is going to use packets where each packet carries file data and metadata necessary for reliable transmission. The packets follow a specific structure and sequence to ensure correct file transfer.

2. Packet Structure

Each packet is defined using the following C++ struct:

```
struct Message {
     string command; // Command to execute, fixed as "COPY" for file copying
     string file_name;   // The name of the file being copied
     int byte_offset;    // The byte offset indicating where this packet's data belongs in the file
     string data;    // The file data being transmitted
     string hash;    // hash of data
};
```

Field Descriptions:

- **command**: This field tells the server to do a “CHECK”, “COPY”, “EQUAL”.
    - CHECK: This packet header is sent to the server to perform an end-to-end check on the file.
    - COPY: This packet header tells the server that we are sending a packet of the content of the file we want to copy.
    - EQUAL: This packet header tells the server the status of the end-to-end check.
- **file_name**: The name of the file being copied. This will be used to specify which file on the receiver’s end the data will be written to.
- **byte_offset**: The byte position in the file where the data chunk belongs. This allows the receiver to place the data at the correct position in the file, supporting out-of-order or retransmitted packets.

- **data**: A string that holds the actual content of the file being copied. The size of the data will depend on the maximum transmission unit (MTU) or the chunk size defined by the application.

3. Protocol Flow

3.1 File Transfer

The sender initiates the file copy by sending all the packets of the file. For now, the server will not send any confirmations and just take the information and write the new file. For chunking, the server will send a confirmation that the hashes of the chunk of packets is the same as the client's hashes. Then, the client will continue sending the next chunk.

3.2 Completion

When the sender has transmitted all file data, it stops sending packets. No special termination packet is required in this protocol, as the completion of file transfer is implicit when the sender sends a E2E check request.

4. Possible Improvements

Send everything multiple times if the end-to-end check fails.
Implement chunking when the client is sending information faster than the server can handle.