CS117: FileCopy Assignment

Jack Zhang(jzhang34), Ivian Zhang(yzhang41)

Module 1. END TO END CHECK

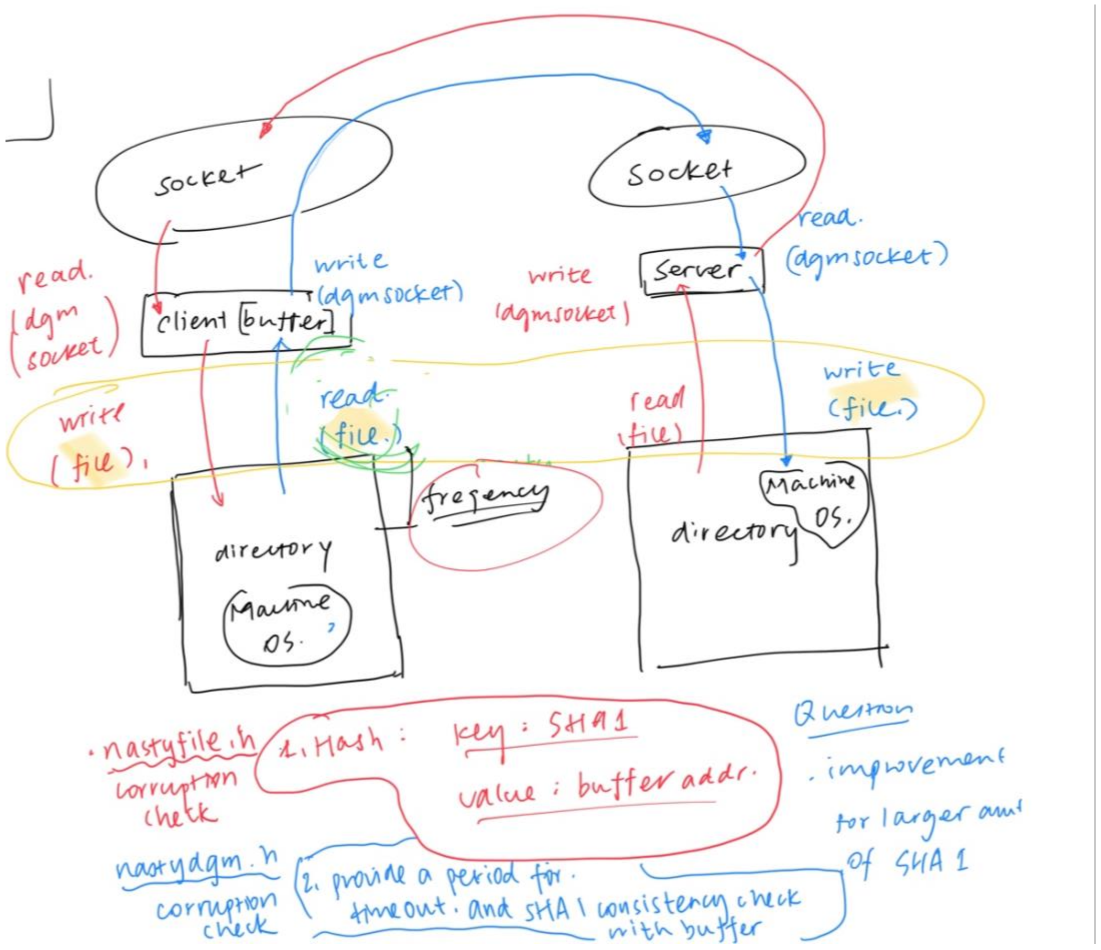

Figure 1. filecopy operation logic demonstration

1. End-to-End check on NASTY_FREAD

- **Purpose**: the goal for end-to-end check here is to confirm that all bytes read in from the file system, or target directory, into the program sockets are correct.
- **Data Structure**: Unordered map [hash table] is used to store SHA1 that is generated at each read. The key for this hash table is [SHA1], and the value is [Content Buf].
- **Reliability**: the design is to rely on the low collision rates of nasty reads SHA1: compared to the chance of reading in the content repeatedly correctly, there is a much smaller chance for multiple nasty reads on one file to have the same SHA1 given the large number of bytes per file. Hence, we choose the buffer that has generated a SHA1 previously seen and found in the table to be the correct content buffer.

- **Implementation**: A HashMap is generated for each file read in. One iteration of read from the local file system produces one content buffer, and we generate a SHA1 for the current content buffer. If the SHA1 is found identical to what is available in the table, we decided that the correct read is found and generate a fileProp object with the read in contents, its filename and SHA1 information. If the SHA1 is not found identical to anything in the map, another read with another C150::Nastyfile object takes place. This iteration is allowed up to 1e6 times, accounting for high file nastiness scenario.

2. End-to-End check on NASTY_SOCKET

- **Data Structure**: N/A
- **Reliability**: the design is to send all information for a file in order and then the content in unit of 10 packets per transmission; the process only moves forwards to send next piece of information if the client side receives an "ACK" from the server. If server didn't receive the next piece of information within the expected time period (set at 30 seconds for first module submission), it will instead send a "REJ" message. Whenever the client receives a "REJ", it will repeat the process of sending information for this file again by resending the Status (starting from the top). For file contents, the process begins by sending 10 packets (each consists of 512 bytes) at once and then sends a SHA1 of the previously sent bytes. When the server has received 10 packets, it also calculates the SHA1 and compares it with the received SHA1. If they are consistent, an "ACK" is sent back to the client, indicating that the server is ready for the next 10 packets. If they are inconsistent, a "REJ" is returned and the client will retry sending the previous 10 packets.
- **Implementation**: the order of the information on a file is sent in the order as followed:[Status → Filename → FilenameSHA1 → FileContentSize → FileContentSHA1 → FileContent].
  1. After the filenameSHA1 is received, two ACKs are expected: one to confirm that SHA1 is received and one to confirm that the calculated SHA1 aligns with the received SHA1 for filename. Two ACKs are designed to confirm correctness of information received.
  2. After the file content is received at the server end, three ACKs are sent: one to confirm that all file contents are received within the timeout period, one to confirm that the length of contents received is what we expect with the received content length, and last ACK to confirm that the content SHA1 calculated is the same as the SHA1 we received.