End-to-end FileCopy Protocol Design

Erli Cai, Xudong Dai

Mailto: erli.cai@tufts.edu, xdai03@cs.tufts.edu

1. General

The general process of our file-copy protocol would be divided into several steps like the following,

1. Start the server and client, then the client will launch a connection to the server.
2. For each file in the source directory
   a. The client would send the name, length and number of packets of the file to the server.
   b. The server would send back a confirmation containing the 3 entries back to the client.
   c. While copying the file
      i. The client would send each *slice* of the file in a UDP packet.
      ii. If the server receives the packet, it will send back a confirmation.
   d. After the sending is done, the client would send a packet carrying the name and checksum of the file.
   e. The server would compute the checksum of the received file.
      i. if it finds the checksum is different from the received one, go back to step 2.
      ii. Otherwise, the server sends a confirmation response to the client.

2. File Copy

1. Anti-network-nastiness Design

Since the network would randomly drop packets due to its nastiness, we would use a retry-and-timeout design.

Each sender, whether it is the server or the client, would send each packet 5 times. The receiver would keep a queue of packets.

- Every time it receives a new packet, it would send back the confirmation 5 times too. If it receives a packet already in the queue, it would simply drop it.
- If the sender doesn't receive the confirmation, it will retry till receiving the confirmation.

After all the packets of a file are received, it will be temporarily kept in the buffer and a checksum

would be computed for the next step.

2. Anti-file-nastiness Design

1. Reading the file

Since the file would be randomly corrupted, we would copy the file 3 times. Each time the datastream would be kept in a separate buffer. Then 3 buffers would generate 3 checksums. Therefore, we can hold a vote,

a. if 2 of them agree with each other, we pick one of the 2 as the correct one.

b. If none of them agree, go back to read the file 3 times again.

2. Writing the file

After dumping the buffer into the disk, we would also repeat the latter step to acquire a checksum of the file, then compare it to the checksum remained in the buffer.

a. If both of them meet, then it is correct.

b. Otherwise, dump the buffer again then go back to a.

3. Dividing the file into packets

We would read the length of the file. Then the file would be divided into several 400-byte sections to fit in the UDP packet. If the last one is not that long, leave the rest blank.

3. End-to-end check

1. Sending Files

To ensure that any packet could arrive at the receiver, we would like to introduce a Multiple-Sending-and-Retry, MSAR mechanism.

a. Each packet would be sent 5 times at the beginning, regardless of other conditions.

b. The receiver would keep the queue of arrived packets. If a repeated packet arrives, it will be dropped.

c. The receiver sends the confirmation 5 times.

d. If the sender does not receive the confirmation, go back to a. Else, it could safely drop the packet from its buffer.

2. Receiving Files

For each file stream, the receiver would initialize a buffer for it since it would know the length of the file. When it confirms a packet is received, it copies the content of the packet to its place.

3. Format of packets

- Pre-sending connection request: `<filename> !!!! <file length> !!!! <packets of the file>`
- Pre-sending confirmation response `<filename> @@@@ <file length> @@@@ <packets of the file>`
- Data packet: `<filename> #### <4-byte packet ID> #### <data>`
- Data packet confirmation: `<filename> $$$$ <4-byte packet ID> $$$$ `
- Checksum packet: `<filename> %%%% <20-byte checksum>`
- Checksum confirmation packet: `<filename> ^^^^ <20-byte checksum> ^^^^ <true|false>`