Report

(a) Video streaming

For server, after receive a “play <videofile>” command, server will try to open VideoCapture for the video file, if it fails, then tell the client that the file does not exist, otherwise tell the client that it exists. After a video is opened, server will find frame count, FPS and resolution of the video, then send this information to the client. Server then starts the following steps.

Step 1: send a frame to client.

Step 2: wait for ack from client.

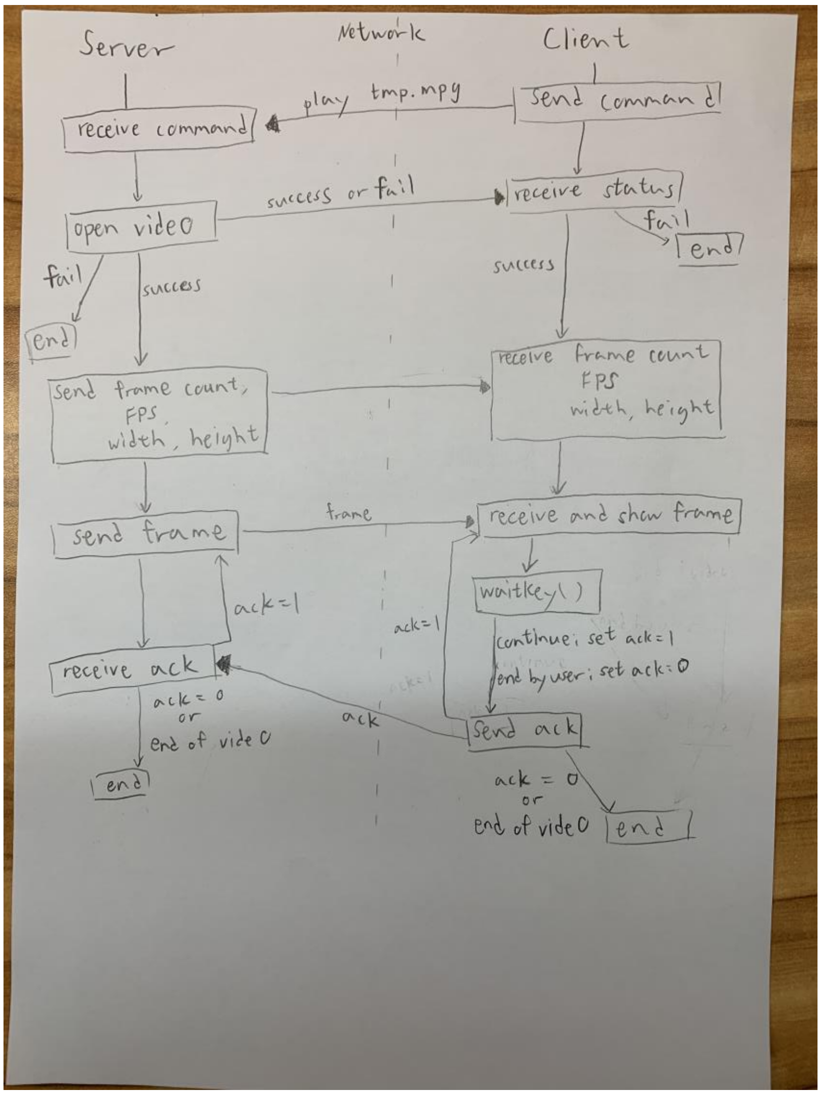
Step 3: If it is the last frame or ack is 0, then stop the streaming, else goto step 1.

For client, after send “play <videofile>” command, it wait for status from server, if the status is fail, print ”The ‘<videofile>’ is not a mpg file.” and stop the process of streaming. If the status is success, receive frame count, FPS and resolution of the video from the server, the starts the following steps.

Step 1: receive a frame from server.

Step 2: waitKey(1000/FPS). If the key indicating that end by user, then set ack to 0, else set ack to 1

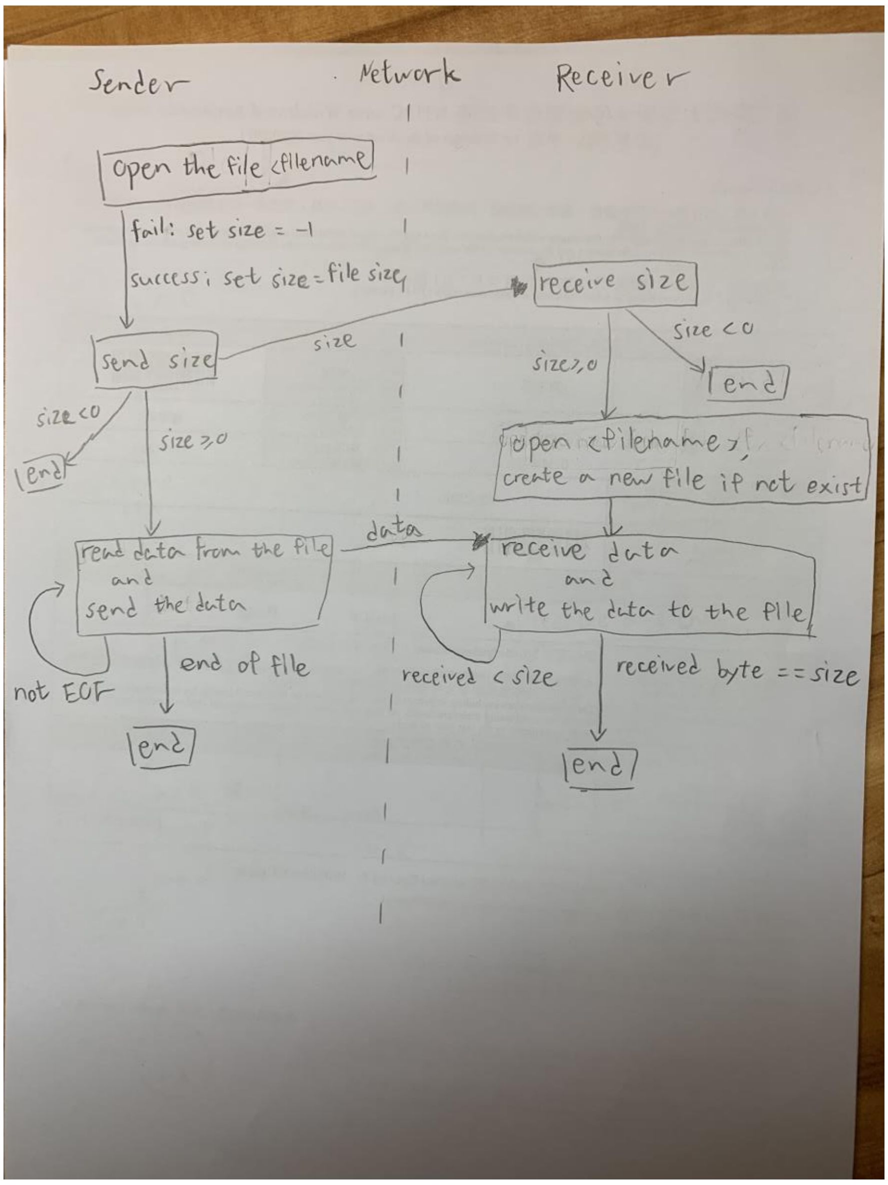
Step 3: If it is the last frame or ack is 0, then stop the streaming, else goto step 1.



(b) File transferring

For sender, try to open <filename>, if success, then set size to file size, else set size to -1. Then, the sender send size to the receiver. After size is sent, if size < 0, end the process of file transfer, else keep reading data from the file and send the data to the receiver until end of file.

For receiver, receive size from the sender. If size < 0, end the process of file transfer, else continue. Receiver then open <filename> or create a new file if nor exist. Then, receiver, keep receiving data from the sender and write the data to the opened file until total received bytes equals the size.



(c) SIGPIPE

SIGPIPE is a signal indicating that the process wrote to pipe with no readers. It can happen in my code. When client or server write to a socket, the socket could have been closed by the other end. In server, SIGPIPE is set to be ignored. When the return value of `recv()` is 0 or -1, server will know that a client has closed the socket. In client, it terminates when receive SIGPIPE, which is default action.

(d) I/O

Blocking I/O is not equal to synchronized I/O.

Synchronized I/O can be non-blocking. An example is that read() or write() on a file descriptor marked non-blocking, read() and write() can return with error, and no data is read or written.

Blocking I/O can be asynchronous. An example is that using select() to test whether a file descriptor is ready to be read or write. select() will block the process until there is a file descriptor that is ready, or until it timeout. However, the process is not reading or writing data while blocked by select().