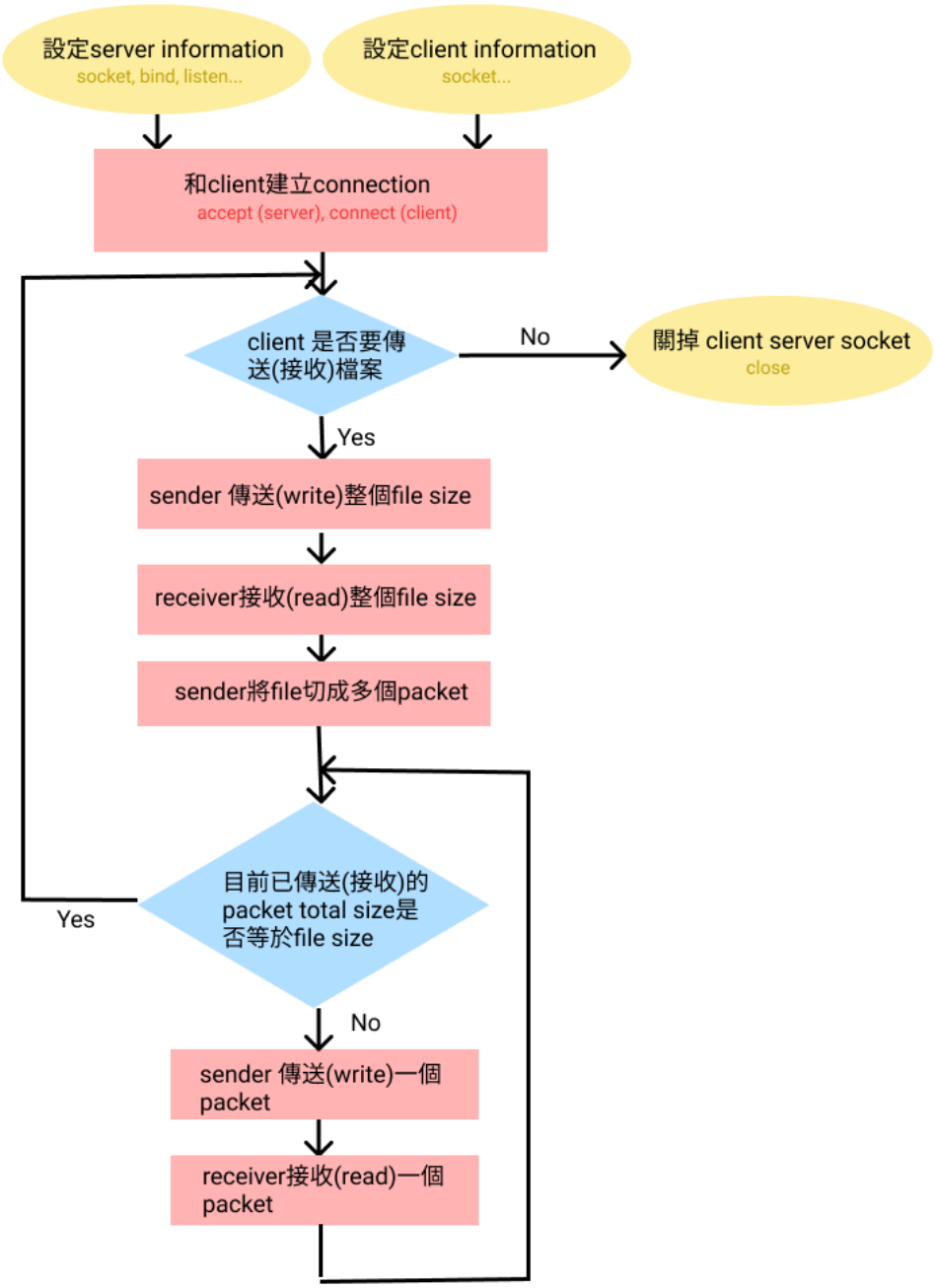
HW02 Report

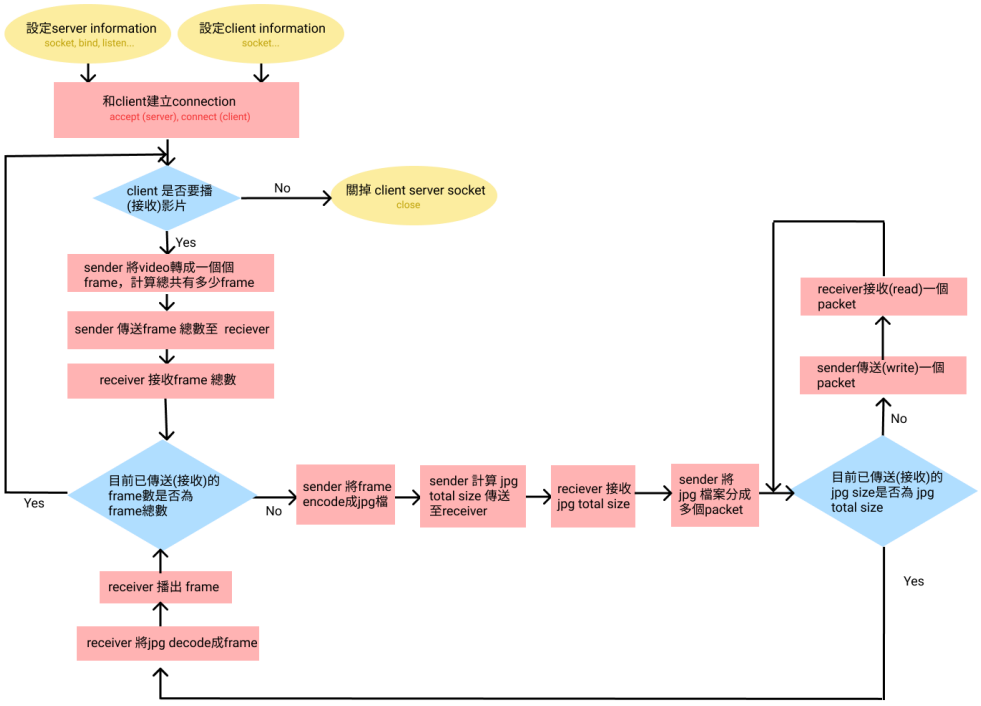
ntnu40771107H 簡郁宸

1. Draw a flowchart of the file transferring and explains how it works in detail.



先設置server、client socket 的information，sender先傳送要傳送的file總大小(因為file最多2GB所以直接用long long int是夠的)給receiver，緊接著將file分成數個2^{15} bytes(或以下)的packet去傳送，直到已傳送的所有packet總size和file的總size相等表示傳送完畢。

2. Draw a flowchart of the video streaming and explains how it works in detail.



影片傳輸使用openCV 的樣子，sender先使用VideoCapture 抓影片出來，再算出影片總共有多少frame，並把frame的總數量傳給receiver，緊接著sender讀取一個frame使用imencode將frame轉成jpg並存在std::vector 中。因為傳送是char *去傳送的所以將vector 轉存在 std::string，sender再計算string有多大，並將string的大小傳給receiver，而後將string 分成多個2^{15} bytes(或以下)的packet傳給receiver直到整個string傳完。

receiver 接收到2^{15} bytes packet後append存至std::string中，當已接收的string size和total string size相同時，則string轉存至std::vector並使用imdecode 轉成frame，最後再使用imshow() 播出frame。

3. What is SIGPIPE? It is possible that SIGPIPE is sent to your process? If so, how do you handle it?

(1) SIGPIPE表示為要結束process的一個訊號。

(2)有可能的，當server close client connection時，若client再往這個伺服器傳送資料時，系統會發出一個SIGPIPE訊號給client process，告訴process這個connection已經斷開了，不能再寫入且直接結束client process。

(3) 沒有特別做處理，不過可以直接使用signal(SIGPIPE, SIG_IGN)讓client 忽略SIGPIPE 。

4. Is blocking I/O equal to synchronized I/O? Please give some examples to explain it.

不完全相同，兩者的差別主要為：

blocking I/O: syscall不會return 直到kernel在某處紀錄了data，但此**紀錄不保證能順利地留著**，若中途遇到斷電或硬體故障，這份紀錄會消失。

> A synchronous I/O operation causes the requesting process to be blocked until that I/O operationcompletes;

synchronized I/O: 做"真正的I/O操作"時才會阻塞process，syscall不會return 直到在kernel上**保證確實**紀錄data。包含blocking IO，non-blocking IO，IO multiplexing。

例如 : 執行recvfrom 的system call，若kernel的資料仍未準備好，non-blocking I/O 不會block process，而blocking I/O 會直接block process。若kernel資料準備好，recvfrom的資料會複製到使用者的記憶體中，此時不管是blocking還是non-blocking I/O 皆會block process，而兩者皆屬於為synchronized I/O。