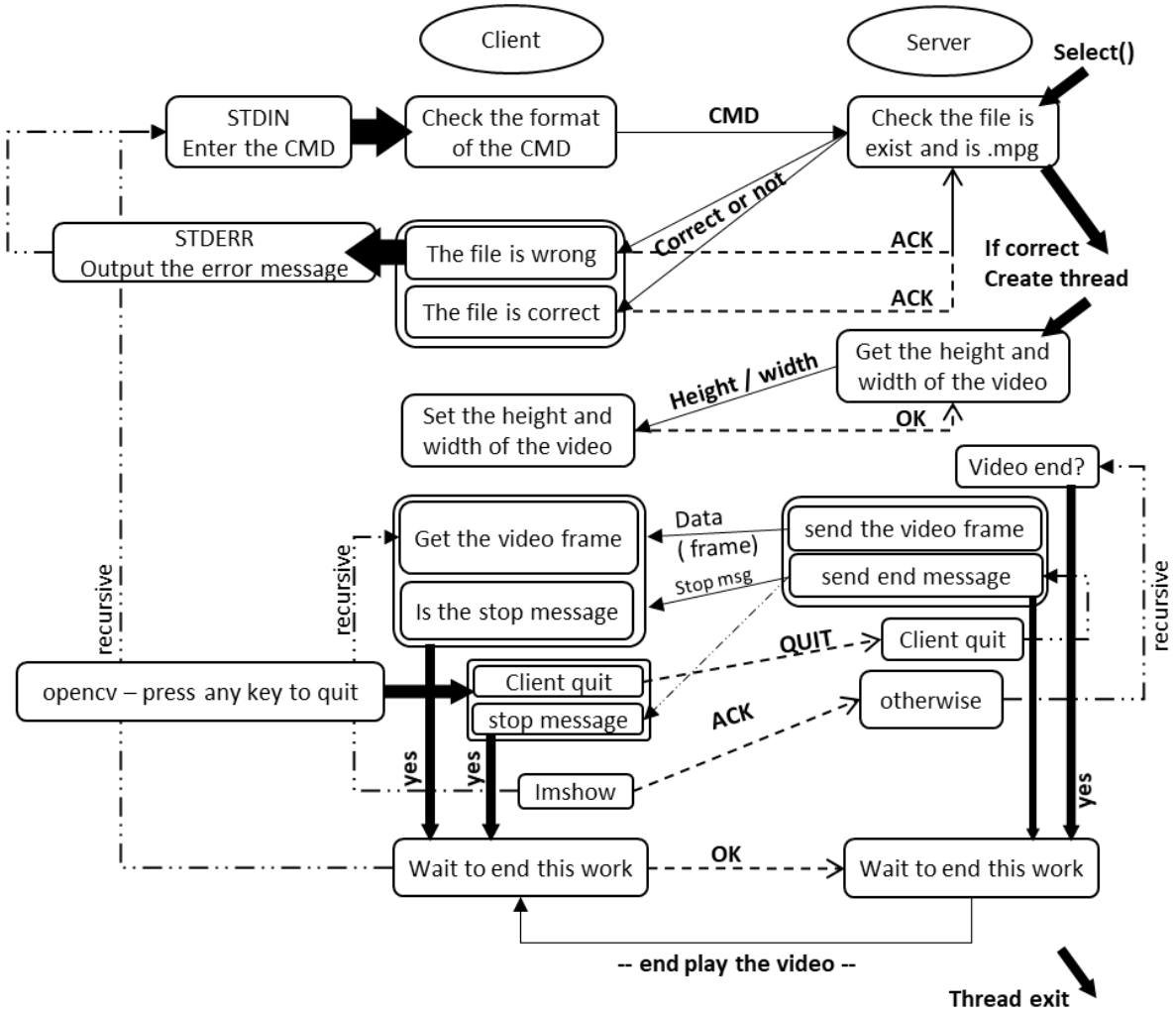
Computer Network - HW2 Report

B06902054 Tsai, you-shin

1 Video Streaming (Draw a flowchart of the video streaming and explains how it works in detail.)



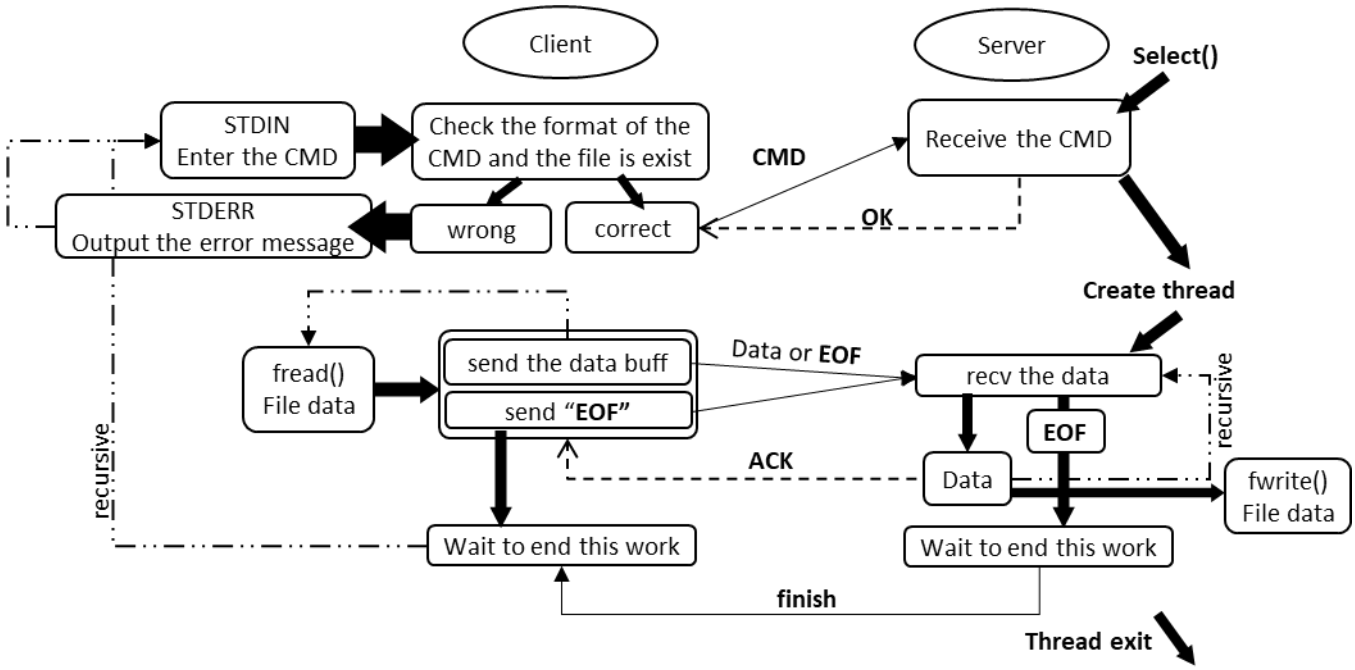
Client 收到輸入的指令之後會先做初步的分析確定格式正確，將指令傳給 server 端，server 則會去比對檔案是否存在及格式是否正確然後將結果回傳給 client 端。如果檔案不存在的話或格式錯誤的話，client 端就會將錯誤訊息 print 出來，回傳一個 **ACK** 的訊息給 client 然後回到等待使用者輸入指令的狀態。若檔案存在而且格式正確，client 端一樣會回傳一個 **ACK** 的訊息，接收到後 server 端會建立新的一個 thread 專門用來處理這個 client 的影片串流的事情。

首先 server 端會讀取影片格式，將影片的高度跟寬度的資訊傳給 client，client 讀取到之後會回傳一個 **OK** 表示收到，便開始兩邊影片串流的部分。每次 server 傳遞資料之前會先查看還有沒有資料要傳，有的畫 server 端會傳遞一個 frame 的 data 給 client，若沒有 server 則會傳遞一個 frame 但帶有終止訊息的資料給 client，並結束迴圈等帶 client 的回覆。client 端每次收到資料之後會先判對是不是結束訊息（在 recv 資料的時候使用 **MSG_WAITALL** 的 flag，以確保資料都有完整接收不會導致影片撥放畫面的跑位），如果是則結束迴圈傳送 **OK** 給 server，如果不是則會用 imshow 處理影片撥放。

不過有一個例外狀況，要是 client 想要提早結束影片的話他可以在 opencv 的視窗裡點選任何按鍵結束影片，此時 client 會關閉視窗然後傳 **QUIT** 給 server，server 收到這個訊息則會回傳具有終止意義的 data 給 client，client 收到終止訊息後代表他已經將 socket 的資料清空，就可以結束迴圈傳送 **OK** 給 server。Server 收到最後的 OK 之後會再回傳"end playing the video"給 client 然後兩邊結束這次的任務。

2 File Transferring

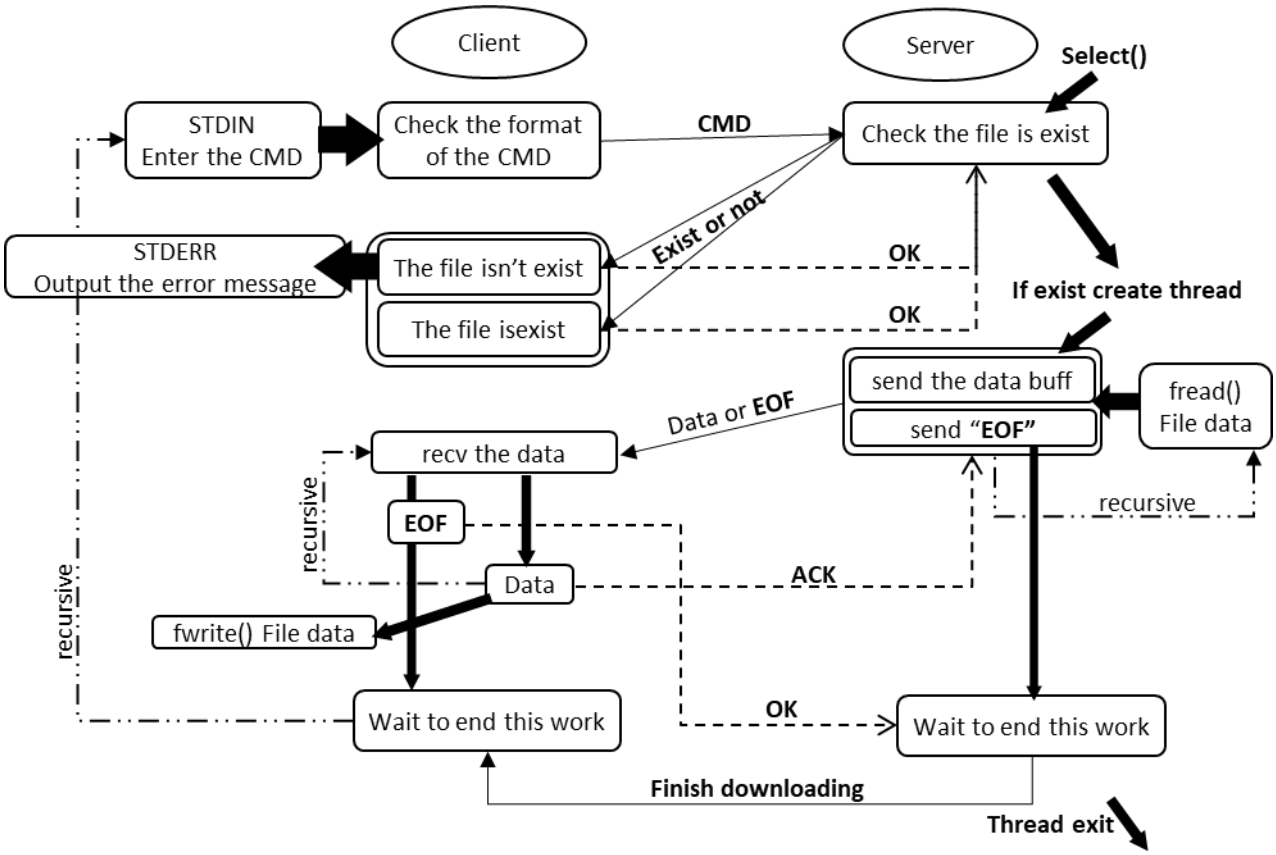
- Draw a flowchart of the file transferring and explains how it works in detail.
- **Case "Put"**



-

Put <filename>，我將指令想成 client 端要把資料 upload 到 server 上面去，因此原檔案許要存在在 client 的資料夾裡。一開始 client 收到輸入的指令之後會先判斷指格式及檔案是否存在，如果有錯就直接回傳錯誤訊息到 stderr 給使用者，如果正確就會傳送指令給 server，server 收到後會回傳 **OK**，建立新的 thread 準備用來處理檔案傳送。Client 端收到 **OK** 之後也準備開始傳送資料，每次傳送前 client 會先看看有沒有資料要傳，有的話傳送 data 並等待 server 的回應，沒有的話傳送一則 **EOF** 的訊息且結束傳送等待 server 回復。Server 收到資料後第一步也是先檢查資歷是不是 **EOF** 訊息，如果不是的話會將資料寫進相對應的檔案，並回傳一個 ACK，是的話則會結束迴圈傳送一個 **finish** 的訊息給 client。結束兩邊的資料傳遞。

- **Case “get”**



Get <fiename>，我將指令想成要將 server 上的資料 download 下來，因此原檔案要存在在 server 的資料夾裡。一開始 client 收到輸入指令之後會先做最基本的 format 的判別，再將指令傳給 server，server 收到之後會先檢查檔案是否存在，回傳結果給 client，client 會再回傳一個 **OK** 表示了解。如果指令正確檔案存在，client 跟 server 兩端就會開始做資料傳送，server 每一次會先看看有沒有資料要傳，如果還有就傳送 data 等待回覆，如果沒有則傳送 **EOF**，結束迴圈等待 client 回覆。Client 每次收到資料之後會先看看是不是 **EOF** 訊息，如果不是就將資料寫進相對應的檔案裡並回傳一個 **ACK**，如果是 EOF 則是傳送 **OK** 給 server，結束迴圈等待回覆，server 在收到 **OK** 之後會再回傳 finish downloading 結束兩邊的資料傳遞。

3 SIGPIPE

- **What is SIGPIPE? Is it possible to happen to your code? If so, how do you handle it?**
  SIGPIPE 是一個 signal。當 process 嘗試將資料寫進或傳送到 pipe/socket 的時候，若沒有人在另外一端打開讀取的話，process 就會收到 SIGPIPE 導致自身程式也跟著結束。而在我們 clinet/server 傳送檔案或資料的過程中，若是 client 端突然結束導致沒有人接收 socket 的資料，若這時 server 再傳資料給 client 的話就會導致 SIGPIPE 的發生。為了處理這個問題，我們可以透過設定在程式的一開始將 SIGPIPE 這個指令設成 ignore（補充說明：不同的 signal，系統有預設不同的處理方始，SIGPIPE 預設是 terminate 會導致程式結束），然後在每一次重送資料的時候判別回傳值，若回傳值為-1 就代表 client 端已經離線，可以將他從服務中的 client 名單剔除關閉一些開啟的檔案結束服務。

4 Blocking I/O and synchronized I/O

- **Is blocking I/O equal to synchronized I/O? Please give me some examples to explain it.**
  Blocking I/O 跟 synchronized I/O 是不一樣的。Blocking I/O 說的是要到 I/O 做完才能 return，synchronized I/O 則是說明要等到 I/O 做完才能繼續做其他事。舉一個例子，假設今天中午我一次叫了五間外送，要去校門口取餐回來吃：
  1. blocking I/O 是指我會去校門口等到五間外送都送到之後，我才提著這些食物回來系館。
  2. synchronized I/O 則是說我要等到我拿到 5 份外送回系館之後，我才可以開始念書。

  Synchronous Blocking: 我去校門口等，等到我領到全部的外送之後才回系館吃然後開始唸書。
  Synchronous Non-Blocking: 我去校門口發現外送沒有都送到，於是我提著部份收到的外送或什麼都沒有就走回系館。回來後我同學就說不管啦我要吃其他的，所以我又走去校門口了。
  Asynchronous Blocking: 我在校門口等外送，然後覺得計網的期中考快到了不能浪費時間，所以我拿出計網課本做在校門口念書等外送。
  Asynchronous Non-blocking: 我決定直接在系館念書，直到我收到所有外送員的電話都告訴我食物到了之後我再去校門口一次取回來吃。