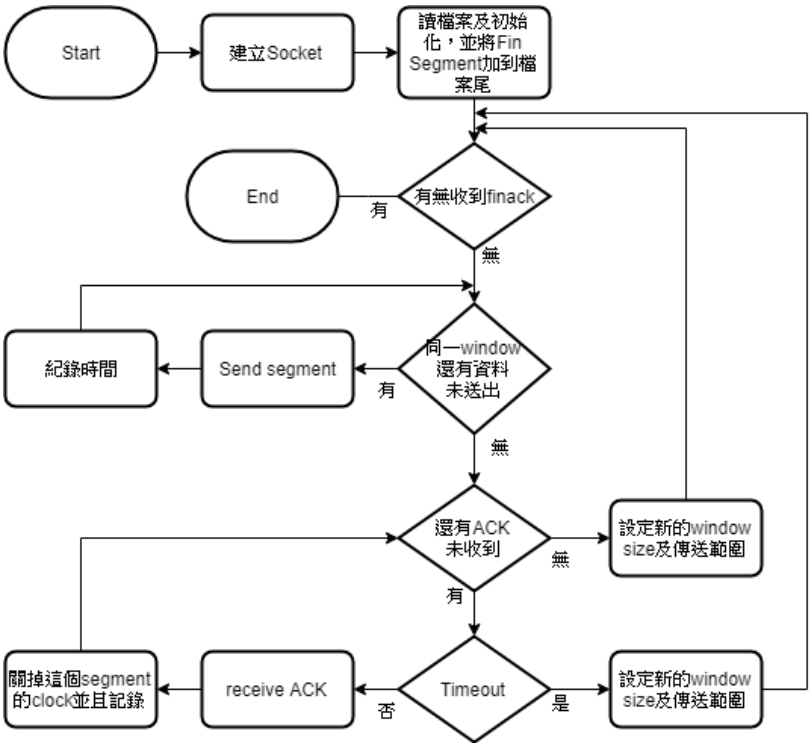
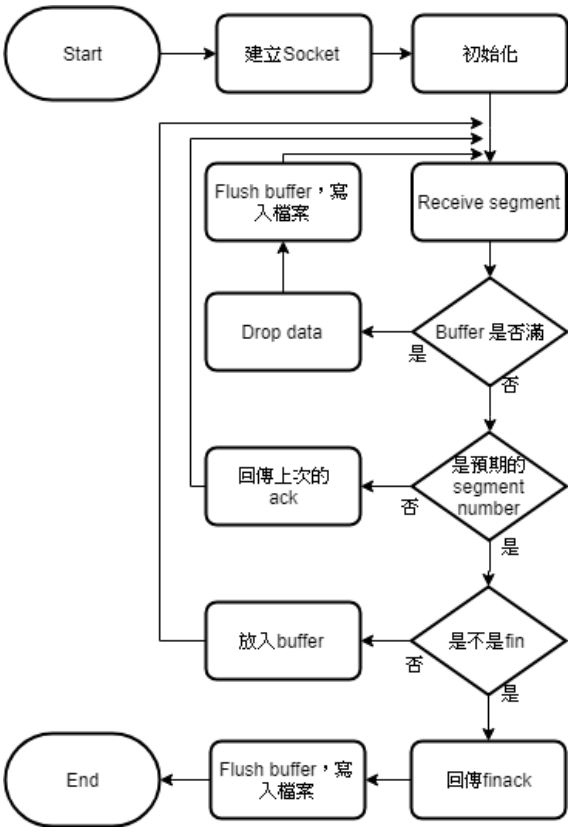
CNHW2- Retransmission + Congest control

工海三 B05505004 朱信靂

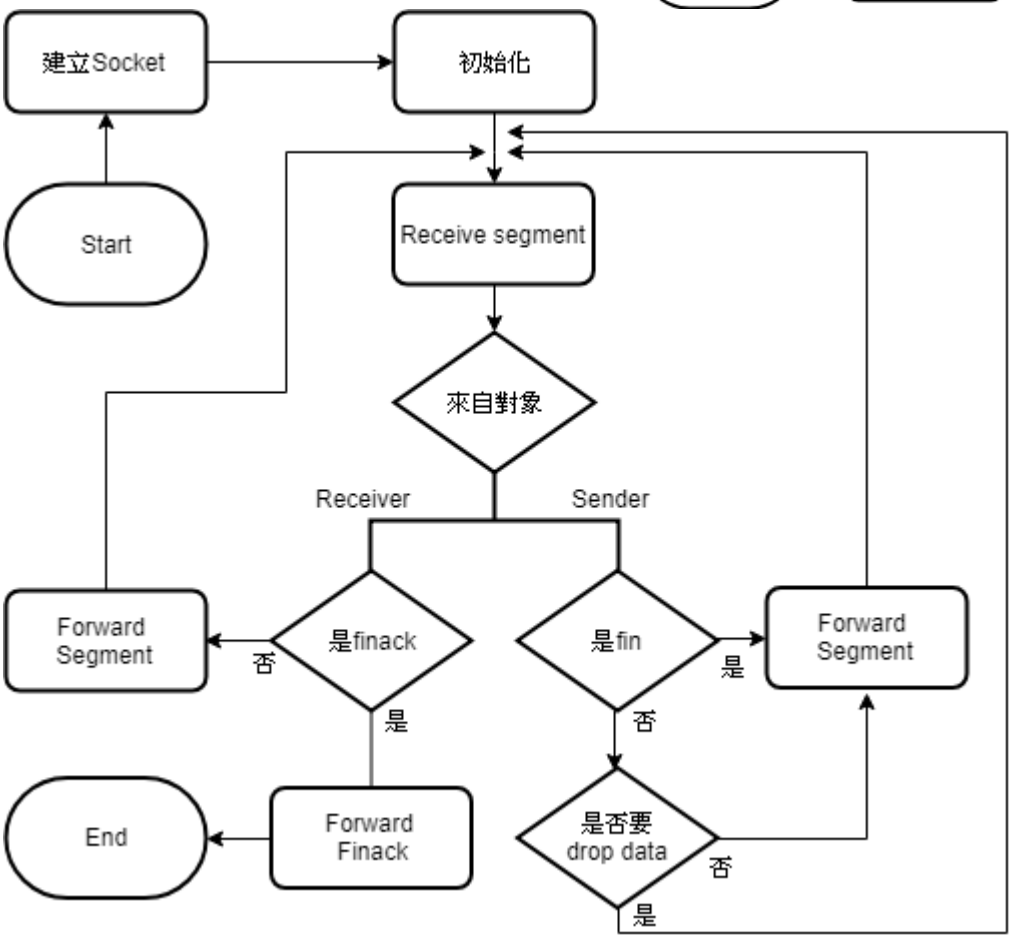
- Sender:



- Receiver:



- Agent:



- How to execute your program

  1. 編譯

     gcc sender.c －o sender

     gcc receiver.c －o receiver

  2. 執行

     ./sender <agent IP> <agent port> <sender port> <filename>

     ./receiver <agent IP> <agent port> <receiver port> <output filename>

- Explain your program structure

  - Sender: sender 的部分我是會將所有的 data 一次讀進來，接著就可以知道說總共需要分成多少個 segment，也可以避免如果 data 被 drop 掉還要一直從硬碟讀取所耗費的時間。接著會有一個 while loop，每次會傳送一個 window 大小的 segment，接著再等待相對應的 ACK 數，如果 timeout 發生就重新設定 window size、threshold，以及下次要從哪個 segment 開始傳。當資料都傳完後 sender 就會接著傳 fin，如果正確收到 finack 就結束。
  - Receiver: Receiver 則是會在 while loop 中不斷 receive segment，當收到後會先判斷 buffer 是否滿了，如果沒有則繼續判斷是否是預期的 segment number，如果不是則回傳上次正確收到的 segment number，代表有 data drop 的情況發生，接著判斷如果是 fin 則回 finack，不是則回該收到的 segment number ACK，並且將資料寫道 buffer。Buffer 只有在滿了又有 data 進來時以及收到 fin 時才會 flush。整個過程會在收到 fin，flush data 後結束。

- Difficulties and Solutions

  在這次作業當中我首先面對的問題是不知道每個 segment 的 timer 要如何維護，我原先的想法是希望可以用 signal 的方式每個 segment 都設定獨立的 alarm，當時間到了再做相對應的動作。然而在 C 裡面中 alarm 其實只能設定一次。所以最後我是維護一個 clock array，並且把接收 ack 的 socket 設成 nonblocking，用 while loop 的方式不斷檢查最前面的 segment timeout 有無發生，但這部分有點 busy waiting 的情況，這也是之後可以改進的地方。

  另外原先我送資料的方式是如果傳完 data 的最後一個 segment 且 window size 允許的話，我會直接接著傳送 fin segment 給 receiver。可是因為我實作 receiver 是一收到 fin 就會回傳 finack，如此就會發生有可能同一個 window 在 fin 前面的資料被 drop 掉，可是 agent 轉傳完 finack 後就結束，導致 sender 無法 retransmit 剛剛被 drop 的資料。解決方法的話就是 receiver 收到 fin 時還是必須要先檢查其 segment number 是否是預期的 segment number，如果不是就應該按照 go back N 的方式回傳上個正確收到的 segment number，而非 finack。