Computer Network Hw2 Report b00902062 陳俊瑋

我將 sender 跟 receiver 的功能都寫在 transmitter 裡面，不打參數就是 receiver，打參數的話，參數的數量就會是要連結的 agent 數量，因此作業要求 3 個 muti-path 就打三個 agent 所在的 hostname 即可。agent 部分則是需要知道 sender 和 receiver 所在的 hostname，以及 loss rate 共三個參數。

我的寫法是先在 transmitter 裡面的 sender 功能(以下都稱為 sender)，開 PATH_SIZE 個 socket，分別連到該 hostname 所在的 agent，然後 sender 開始讀輸入檔名的資料，按照 window size 的變化一次傳 window size 個 package 給不同 PATH 的 agent，順序是 0->1->2->0->1....(if 有三個 PATH)，達到 multi-path 的要求，接著 package 會傳送到 agent。而 send 完 winSize 個 package 以後，在接收 winSize 個 ack，由於 package 有可能會在傳送的過程中 loss 或者碰到 fflush 的時候被 drop，因此有可能 recv 不到 winSize 個 ack，利用 setsockopt 使得 recv 有三秒的 timeout，也就當要接收第一個 ack 的時候，就計時三秒內 recv，若超過時間，則會出現 timeout，將 threshold= winSize /2，winSize=1，並且開始 slow start 將還沒收到的最小 ack 重傳(winSize=1)，然後 winSize 若超過 thread 就線性成長(+1)，若沒有則是指數成長(*2)。透過 timeout 和 winSize 和 ack 達到 reliable 和 congestion control 的需求。直到讀到該資料的檔案結尾並且 sender 端 handle 的 ack 已經是所有的 ack(等於所讀到的最大的 sequence_num)，那麼就傳送一個 FIN 的 package 告知 receiver 傳送結束。

Agent 的部分，因為 recv 是 blocking 的，所以主要是依靠 select 判斷 recv sender 的 package 有資料，還是 recv transmitter 的 receiver 功能(以下都稱為 receiver)的 ack 有資料，也有可能兩者都有收資料，如果有資料的話就 recv，達到 non-blocking 不會塞車。而為了讓 sender 端和 receiver 端都可以一個接收 fd 能 recv 到來自不同的 agent 的 package，因此 agent 必須知道 sender 和 receiver 的 hostname，並且開好 socket，以便之後的傳送。Agent 會將從 sender 端收到的 package，先用機率(只支援到小數點下第四位)判斷這個 package 要不要傳到 receiver 去，如果要的話就正常將 package 導到 receiver，若不要的話則印出 drop 就 select 其他的資料，使得這個 package 消失於這次傳送中。

接著 package 來到 receiver，由於不同的 agent 都有建立對 receiver 的 socket，因此 receiver 可以利用一個 recv 的 fd 就能接收來自 agent 的所有 package，接收到的 package 先檢查它的 sequence_num 有沒有在 receiver handle 的最低拿過 ack 以上，有的話就先把他塞進 recv_buffer 裡面，以下的話就 ignore 因為已經拿過了，又或者這個 package 來的時候 buffer 已經滿了，就將這個 package drop 掉並且將 package flush 至檔名為 file0(fileX, X 是這次連線開始傳送的第 X 個檔案)，如果沒有 drop 的話就將 ack 重新送回 agent。若收到 FIN package，則是將 recv_buffer 中的資料 flush 至檔案中，然後結束。最後來自 receiver 的 ack 再傳到 agent 藉由之前建立好的 socket 傳回 sender。