HW3

B113040002 林之謙

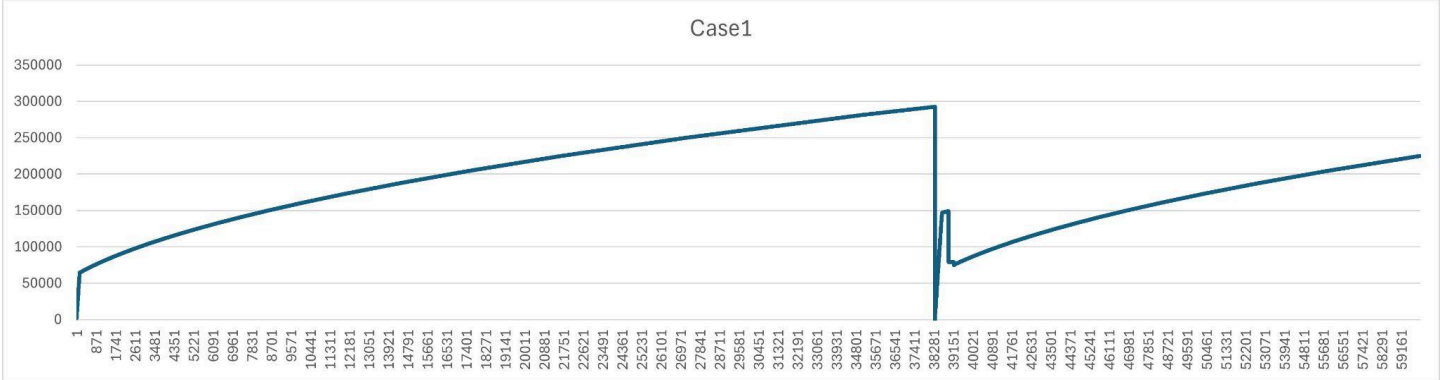
1. What happens when cwnd is too large?

一開始 goodput 會呈現線性增加的趨勢，當到達 20~30 pkts (30000 ~ 45000 bytes) 時，inflight packets 的數量逐漸塞滿整個帶寬，因此 goodput 的增加趨勢減緩並達到峰值。之後因 rbt 無法應付過多的 inflight packets，開始發生 drop，goodput 也開始下降，從 80 pkts (120,000 bytes) 開始到 10BDP，goodput 都在大約 14M 左右。

2. AIMD

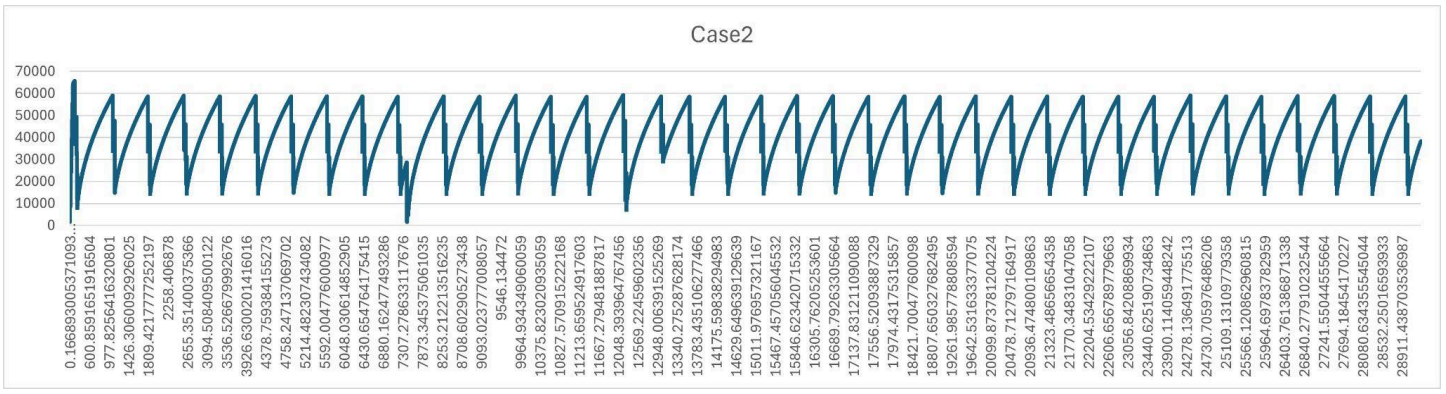
此為前 6M 次 get_cwnd() 的回傳結果
縱軸為 CWND 大小，橫軸為時間(sec) (Case 1 為次數)

Case 1 ~ Case 4 為 uplink buffer 大小的影響
Case 1.



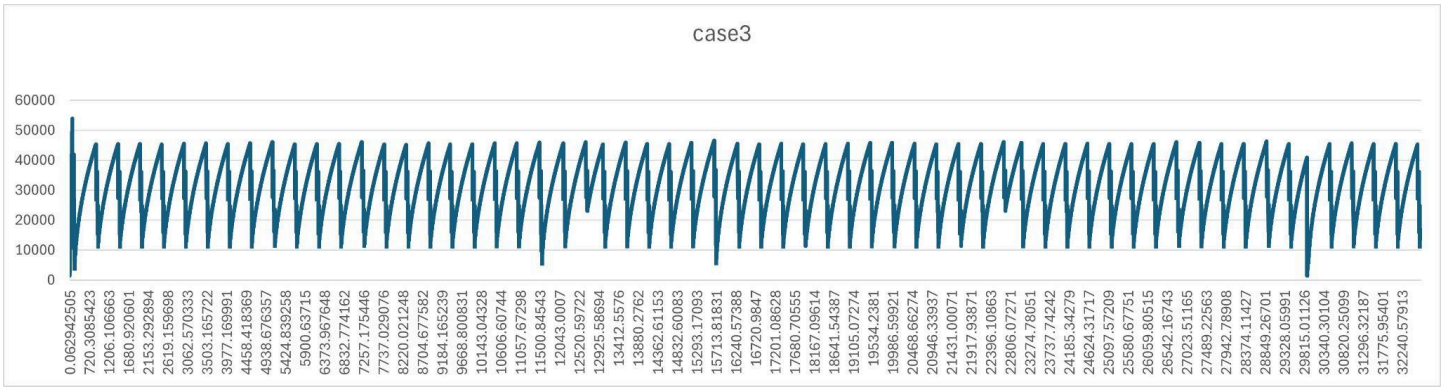
可以發現cwnd在很早期就突破初始 sstresh(64KB) 並進入 congestion avoidence，但是因為 uplink buffer、downlink buffer 大小都是無上限，使得 drop 不會發生，CWND 因此不斷增長，但 congestion avoidence 下 cwnd 增長速度極慢，到了 3M次才因 timeout 重新回到 slow start。

Case 2.



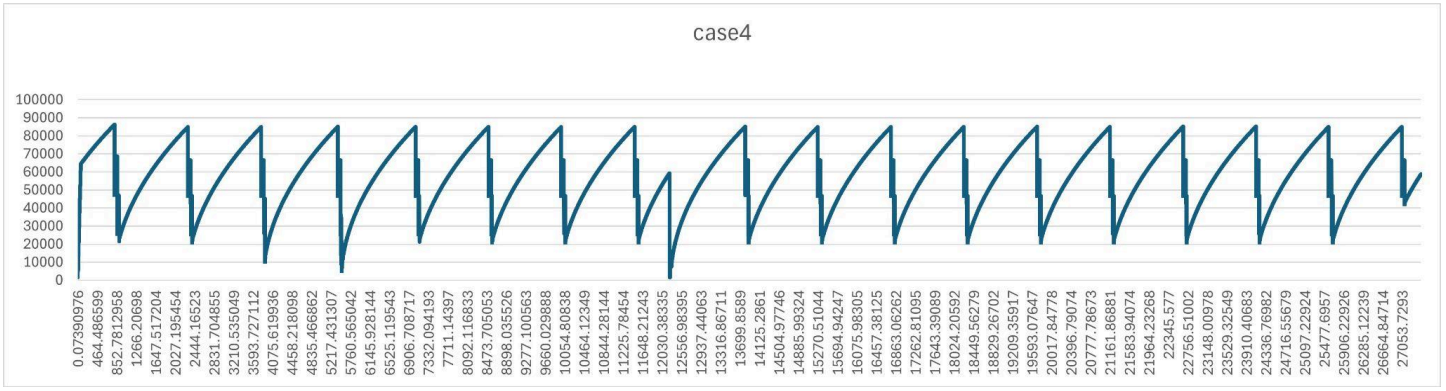
和剛才無上限相比，這裡開始出現 drop，以及因塞車而增加的 timeout 次數。但有趣的是，在此條件下 cwnd 的峰值從來沒有高於初始 sstresh。此外，除了一開始的狀況，sender 大多都在 conjestion avoidence 和 fast recovery間反覆跳動，只有週期性會變成 timeout。

Case 3.



Case 3 的 uplink buffer 又比 Case 2 少了一半，因此 drop 更頻繁的發生使進入 fast recovery 的頻率也跟著生高。而 cwnd 的峰值也來不及在收到三次 duplicate packet 前達到 5M，相教於 Case 2 來說情況更嚴峻。

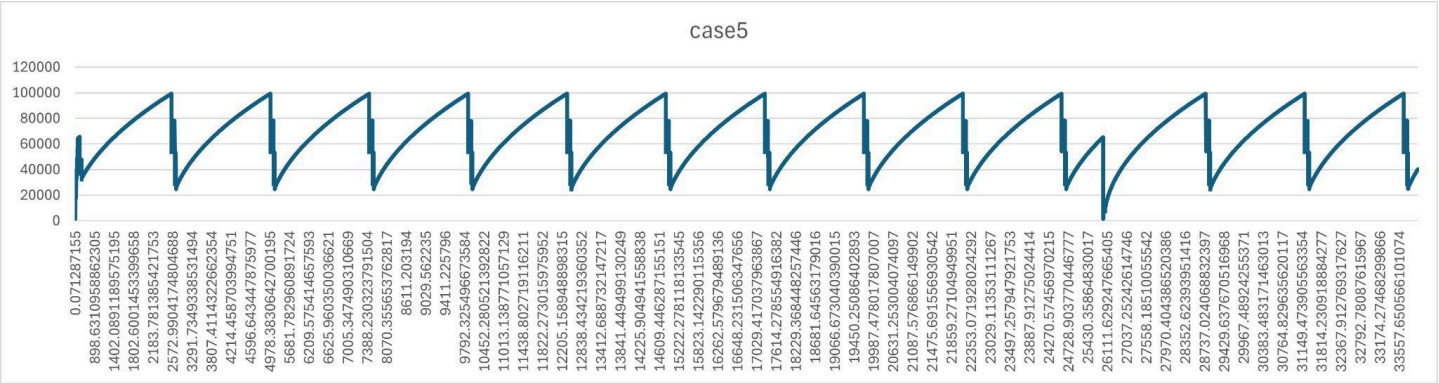
Case 4.



反之，Case 4 的 uplink buffer 是 Case 2 的兩倍，Case 3 的 4 倍，因此其 drop 週期更長，使 cwnd 有時間成長到 9M，也得以在 timeout 前傳送更多封包。

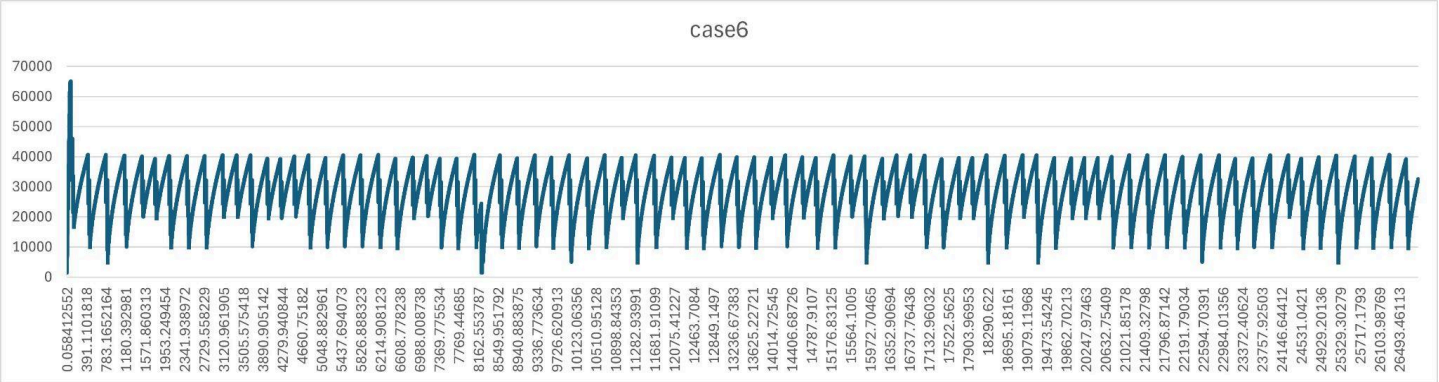
Case 2、5 、6 為 RTT 長短的影響

Case 5.



由於 timeout interval 受到 RTT 影響，因此 Case 5 的 timeout 頻率並沒有上升，且因 ACK 來得慢，不但使 cwnd 的變化週期拉長，進入 fast recovery 的週期也變長

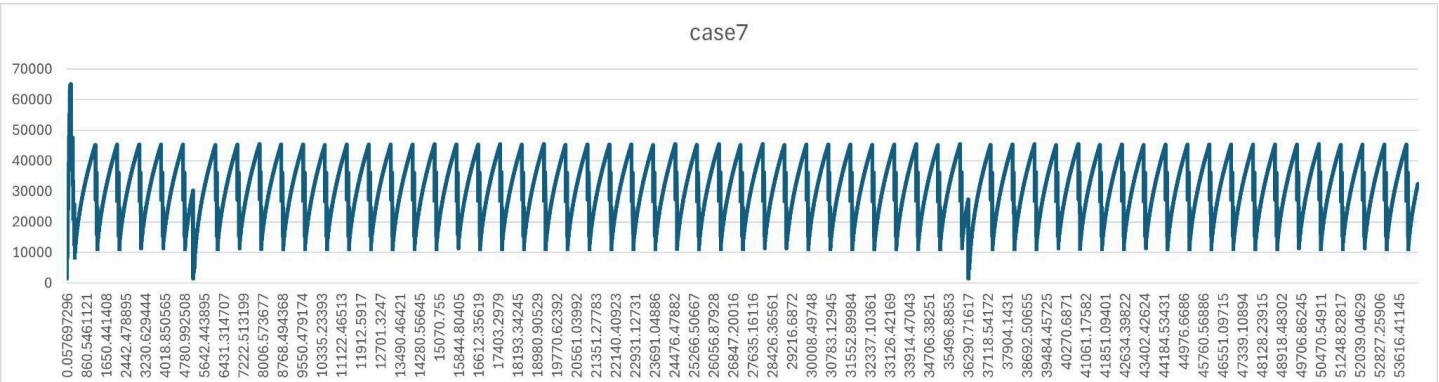
Case 6.



反之 RTT 變成 6ms 後 timeout 頻率上升，ACK 來得快使 cwnd 的快速變化，進入 fast recovery 的週期緊縮。cwnd 的峰值也就 case 5 低很多。波型看起來跟 Case 3 有點像，但 6M次取樣所需的時間較短，峰值也比 Case 3 稍少了些。

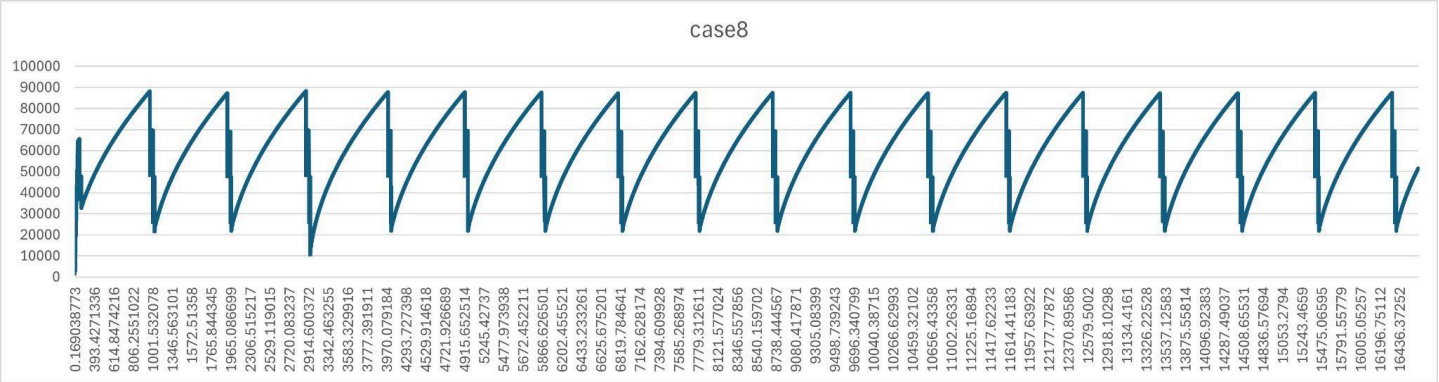
Case 2、7 、8 為 帶寬大小的影響

Case 7.



帶寬小使 drop 多次發生，drop的次數也變多，但多虧 RTT 較長，cwnd 的峰值得以累積得比類似波型的 Case 6 高，雖然看似跟 Case 6、3 相像，但取樣所需的時間明顯多很多。看來不只是 RTT ， 帶寬也和速度有很大關係。

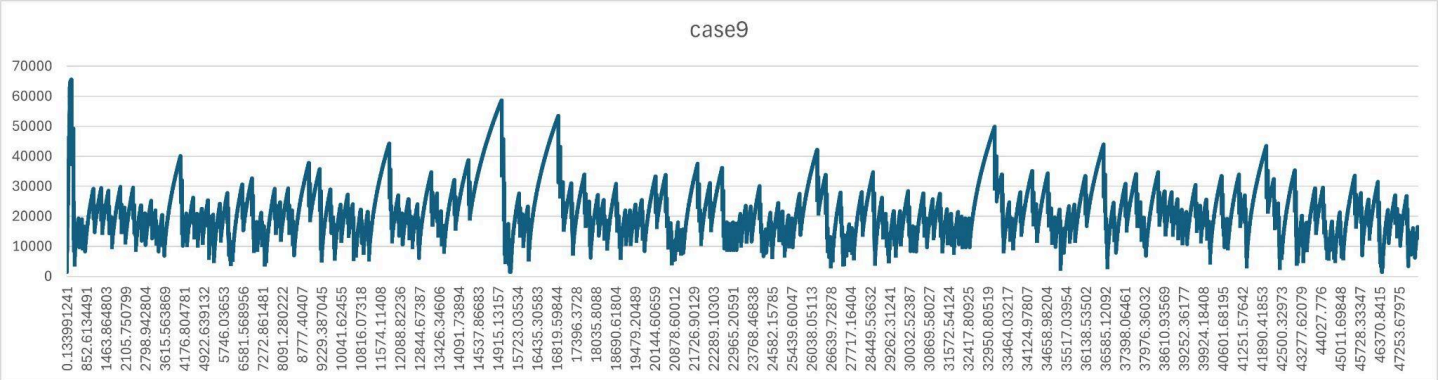
Case 8.



帶寬變大使一次可以送出的封包量大增，雖然波型和 Case 4 相近但時間上是大大縮短，timeout 的次數也因傳輸量大而在整個傳輸過程中被稀釋了。看來比起 uplink buffer，rdt 的帶寬影響傳輸速度的幅度更大。

Case 2 、*9* 比較 *loss* 的影響

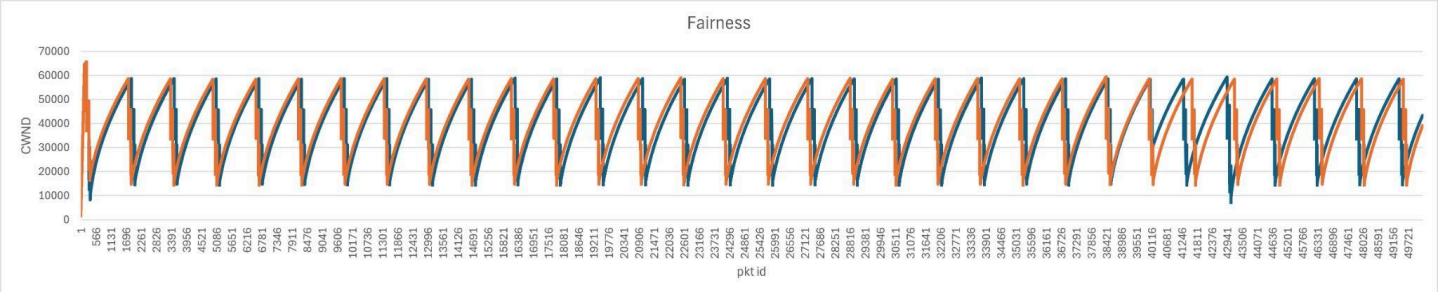
Case 9.



Loss 在所有因素中對整體傳輸品質造成的壞影響最大，因為 simloss 是看機率決定要不要 loss 的，因此 drop 發生的頻率也不定，但可以看出即使是 simloss == 0.01 就可以在 timeout 前大幅增加進入 fast recovery 的次數。可見處理 loss 真的是非常重要的問題。

3. Fairness

縱軸為 CWND 大小，橫軸為 pkt 寄送次數



這是我同時執行兩對 sender、receiver 的結果，兩邊分別用不同 port 。可以發現只要 AIMD 做得恰當，兩邊的 cwnd 變化基本上是重疊的。因此 TCP 確實存在公平性。