# NSC HW 5

109550206 陳品劭

## Explain how you implement error control: 5%

雖然 UDP 應該已經會把 bit 錯的封包丟了，但測試過程中總有奇怪的狀況，因此有做一個簡易的 checksum: xor 封包中所有 bytes 成一個 byte 並放在 header ，對方收到時會去 check。

送方會去檢查所有已送出去的封包且已經超時 (self.recv_timeout = 0.5) ，就會重新再送一次該封包，並 reset time about packet。

收方會在收到封包時，傳 Ack 告知收到的封包。

封包送之前經過切分，收到時要重組，根據 header 依照順序重組成原貌，其中要注意重複的封包要處理掉。

## Explain how you implement flow control: 5%

每隔一段時間傳一個封包告知自己的 buffer 狀況 (self.cur_recv_buffer) ，對方收到時記下來 (self.buffersize)，每次送之前檢查該值不會太小。

太小時先暫停傳送，等到對方告知的值變大為止。

## Explain how you implement congestion control: 5%

原本的作法：

每次收到 ack 時，增加每秒傳出去 1000 byte。 (self.sending_waittime = 1500 / (1500 / self.sending_waittime + 1000))

當 loss 連續三個封包時減慢一半傳送速率。 (self.sending_waittime *= 2.0)

另外控制其 wait time 不要太久。

後來的作法：

設一個 window size 為 13 個 chunk (每個封包大小) ，當收到 ack window size + 1，當連續 loss 3 次，減一半 winsow size 。

send thread 會根據 send_queue 來送，send_queue 大小維持為 window_size - 送出去還沒收到 ACK 的封包數。

現象：

bit rate 太大其實看不出啥，小一點的 bit rate 1M ~ 500K 可以發現其貼合該數值。

原本的作法，平均送的速度太慢，即使網路很好，也送很慢。

## If you use two streams to send data simultaneously from the client to the server or in the other direction, what will happen if one packet of a stream gets lost? Is the behavior of QUIC different from that of TCP? Why? 10%

TCP 的情況，即使後面的 stream 送達也不會回傳、ACK。就會 block 等前一個 stream 送達才 ACK，因此傳有收到多個 ACK 相同段的封包要快速補送。

QUIC 則不會受此影響，當前一個 loss 卡住，其他可以先 ACK、回傳，前一個再自己慢慢補送。