CS 535 HW2 | Jiankun Dong

3. Max overhead in byte stuffing:

Assume we have data consists only of FLAG or ESC bytes.

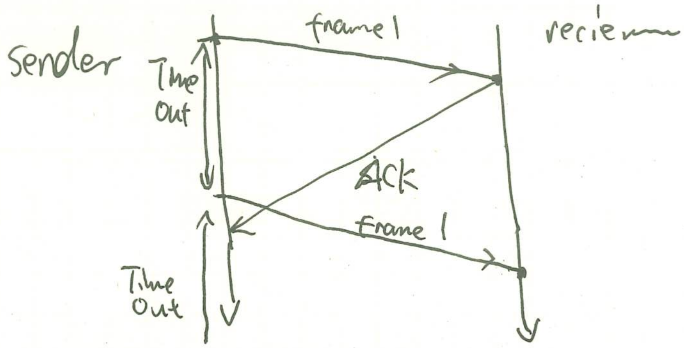
Then, we need 1 byte of ESC ~~op~~ for each byte of data.

∴ Max overhead would equal to the data length (payload), ~~and~~ (100%)

4. Recieved: 0110 0111 1100 1111 0111 1101

Remove '0' for every 5 '1's: $\boxed{0110011111011110111111}$

27. If the implemented 'Timeout' on the sender is too soon, consider this:



then reciever might recieve multiple copies of the same frame when there's no lose of the frames.

35. $T_{oneway} = 3\times10^3 km \times 6\times10^{-6} sec/km = 1.8\times10^{-2} sec = 1.8\times10^4 \mu sec.$

~~Based on the code: MAX_SEQ = 7 $2^3 = 8$~~

~~∴ the bits required for sequence is 3~~

T_1 carrier: 1.544 Mbps (from book). ⇒ 192 bits / 125 μsec.

∴ Bits in channel $= 1.8\times10^4 \mu sec \times \frac{192 bits}{125 \mu sec} = 2.7792\times10^4 bits.$

Frames in channel = 144 frames $> 2^7$

∴ ~~from~~ 8 frames.

∴ Seq_number $= \log_2 8 = \boxed{3}$

~~40~~ ~~a.~~

37. Without the else statement, the ack_timer will not be started. This results in the separate acknowledgement frame not being sent. ~~Since this only~~ Since the sender doesn't recieve the acknowledgement frame. Without this timer, when reverse traffic is light, it's possible that sender gets blocked when its window reaches the maximum. It makes ~~waste of bandw.~~ the protocol incorrect.

38. the purpose of the while-loop is that when Ack n, it checks n-1, n-2...
Without it, the timer for those frames will time out, and ack ~~of~~ for those will be resent at a later time (when ack-timer out). ~~It affects the performance of the protocol.~~ Also, it processes the ACK coming in. Without it, we would get stuck transmitting same frames. It BREAKS the code.

40. Checksum case is related to damaged frames. Removing it will results in accepting damaged frames and ~~results~~ affects the correctness of the protocol

44. a. Stop-and-wait:

$$t_{frame} = \frac{length}{channel\ speed} = \frac{1 \times 10^3 b}{1 Mbps} = 1ms.$$

$$t_{arrive} = t_{frame} + 270ms = 271ms$$

$$t_{ack-sent} = t_{arrive} + t_{frame} = 272ms$$

$$t_{ask-get} = t_{asksent} + \cancel{t_{arri}}\ 270ms = \underline{542ms}$$

$$\therefore U_{S\&W} = \frac{frames \cdot t_{frame}}{t_{ask-get}} = \frac{1 \cdot 1ms}{542ms} = \frac{1}{542} \approx \boxed{0.18\%}$$

44b) Protocal 5: (go-back-n)

In protocal 5, 7 frames is the max-seq value

$\therefore$ 7 frames are sent

$$\therefore U_{GBN} = \frac{7 \times 1ms}{542ms} \approx \boxed{1.29\%}$$

c) Protocal 6:

$$\cancel{n_{frame\_sent} = 2^{max\_seq} - 1}$$

$$max\_seq = 2^{n_{frame\_sent}} - 1 = 7$$

$$\therefore n_{frame\_sent} = 3$$

$$\therefore U_{protocol\ 6} = \frac{3 \times 1ms}{542ms} \approx \boxed{0.74\%}$$

45. I don't think it's possible to have only NACK.

The reciever doesn't know the TOTAL NUMBER of frame for the entire transmittion. NACK can tell when a frame in the middle is missing, but if say I transmitte 10 frames, recieverer only got the 1st frame. there's no way for it ~~to~~ to know there's 9 other frames lost and send me NACK ~~of~~ for those frames.