CN Sheet 2

1. Datalink Protocol uses the following encoding

A: 01000111
B: 11100011
Flag: 01111110
ESC: 11100000

- Given the four-char. frame A B ESC FLAG
- Show the bit sequence if for framing

a) Character Count was used

- Put char. count including header at header of each frame: //have 1 frame

→ hence will put 1+4 = 00000101

bit sequence:

00000101	01000111	11100011	11100000	01111110
5	A	B	ESC	FLAG

b) Flag byte with byte stuffing

- let's start by stuffing the needed bytes (escape each special char. independently)

A B ESC ESC ESC FLAG

- Now put flag byte as header & trailer

FLAG A B ESC ESC ESC FLAG FLAG

//easily convert to bits

c) Flag byte with bit stuffing

- convert to bits
- stuff a 0 after each 5 bits in stream that are '1'

Original stream:

01000111 11100011 11100000 01111110

Add: 0 0 0

then prepend and append the flag byte 01111110

$\Rightarrow$ total overhead in this case is $2\times8+3$ which is $< 2\times8+2\times8$ as in b.

2. Can we use flag bytes only for headers?

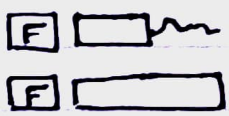
- So instead of

 ①

we do

 ②

- It won't work as when the sender is silent, there is still some noise in the channel
  - → In case ② we won't know if (the sender's frame has finished and what's in the channel is noise) or (its not noise, the sender's frame isn't over)

 receiver can't distinguish

- Likewise, if we decide to keep it at the trailer only then we don't know if the sender has started or its channel noise.

3. When bit stuffing is used can loss, insertion or modification cause an error?

- Clearly, if the bit we lose is the one we used to disambiguate a flag in the stream then yes (it will now be confused with real flag)

- likewise for modification, consider

FLAG 00000000 01111110 FLAG

↑ 0 (bit stuffing)

if this becomes 1 then its mistaken for FLAG

- For insertion 01111110 } in data bits

↑ 0 (bit stuffing)

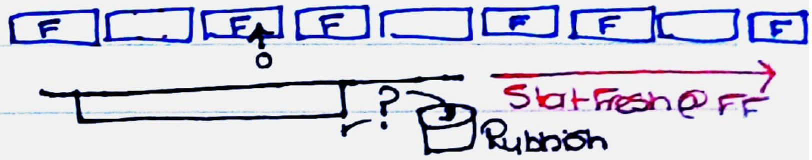
if 1 is inserted here, again will be mistaken for a flag

- Notice that any of the 3 is an error anyway but we perceived the question as asking for framing error

- In general, think of making false ESC/Flags or altering existing ones at header & trailer (we didn't do that here but should be okay)

* Sidenote

- Char. Count → error can propagate to all further frames
- Flag byte → only current and next frame (at most*)



• Sheet 2 Part 2

→ Single-bit Hamming Code

• For it to work, it must hold that $m+r+1 \leq 2^r$

• Payload = 1101 00 1100 1100 11 0101 //m = 16

hence, need $17 \leq 2^r - r$

$r=0$ $17 \leq 0$ ✗
$r=2$ $4-2$ ✗
$r=4$ $16-4$ ✗
$r=5$ $32-5$ ✓

$r = 5$ Parity bits

→ As far as I believe, there's no hamming code that can correct more than one error (regardless of r) (Sidenote)

⇒ Add the Parity bits and show the Code word to transmit assuming even Parity

//like lec.

• $r=5$, need P1, P2, P4, P8, P16

1	2	011 m_3	4	101 m_5	110 m_6	111 m_7	8	1001 m_9	1010 m_{10}	1011 m_{11}	1100 m_{12}	1101 m_{13}	1110 m_{14}	1111 m_{15}	16	10001 m_{17}	10010 m_{18}	10011 m_{19}	10100 m_{20}	10101 m_{21}
P1	P2	1	P4	1	0	1	P8	0	0	1	1	0	0	1	P16	1	0	1	0	1

$P_1 = m_3 \oplus m_5 \oplus m_7 \oplus m_9 \oplus m_{11} \oplus m_{13} \oplus m_{15} \oplus m_{17} \oplus m_{19} \oplus m_{21} = 0$ //8

$P_2 = m_3 \oplus m_6 \oplus m_7 \oplus m_{10} \oplus m_{11} \oplus m_{14} \oplus m_{15} \oplus m_{18} \oplus m_{19} = 1$ //5

$P_4 = m_5 \oplus m_6 \oplus m_7 \oplus m_{12} \oplus m_{13} \oplus m_{14} \oplus m_{15} \oplus m_{20} \oplus m_{21} = 1$ //5

$P_8 = m_9 \oplus m_{10} \oplus m_{11} \oplus m_{12} \oplus m_{13} \oplus m_{14} \oplus m_{15} = 1$ //3

$P_{16} = m_{17} \oplus m_{18} \oplus m_{19} \oplus m_{20} \oplus m_{21} = 1$ //3

• Now by Plugging with the Parity bits

0111 1011 0011 0011 10101

bit no. 20 → becomes 1

• Now if recompute P16 P8 P4 P2 P1 we get

Flip

	P16	P8	P4	P2	P1
	0	1	0	1	0
→ XOR with real Parity ⊕	1	1	1	1	0
	1	0	1	0	0

• This is our syndrome vector → decimal = 20 means we detected error at 20th bit. (m_{20})

dbt.

2. Consider

→ datalink Protocol uses char. Count for framing

→ uses Hamming for error correction

→ Assume any frame has at most 1 error

Is it true that we can resync (avoid non-sync) after any error because Hamming can correct it

• For the receiver to start error det/corr. it must know m (to know r → know where are Ps)

→ if such error hits the char. count (m) then any correction we do based on that m is wrong

→ It is not true

3. Arrange the following in 16-bit words then Compute checksum

w_0 w_1 w_2 w_3

0001 F203 F4F5 F6F7

$$
\begin{array}{r}
\scriptstyle 1\,1 \\
(2)\ 0001 \\
+\ F203 \\
+\ F4F5 \\
+\ F6F7 \\
\hline
DDF0 \\
+\ (2) \\
\hline
DDF2
\end{array}
$$

- The internet checksum is based on 1-complement arithmetic (negative is flippiling all bits).
- Binary addition in 1-comp. arithmetic is equivalent to standard 1 bit addition but with wrapping around final carries

← we would add 1 again if a final carry occured.

- Now the checksum, which should give 0 when added to the four words, is the negative of the result (1's complement)

1101 1101 1111 0010

DDF2 ⟶ 220D

4. Remainder of dividing x^7+x^5+1 by x^3+1

→ Can turn into binary & solve with shift xor (but not so spac efficient as in lec.)

→ here, will rather divide them directly as polynomials

1. Divide highest terms
2. Multiply result
3. Subtract (xor) approp. terms

$$
\begin{array}{r}
x^4+x^2+x \\
x^3+1\,\overline{)\,x^7+x^5+1} \\
-\;x^7+x^4 \\
\hline
x^5+x^4+1 \\
-\;x^5+x^2 \\
\hline
x^4+x^2+1 \\
-\;x^4+x \\
\hline
x^2+x+1
\end{array}
$$

- repeat (divide highest, multiply, sub.)

x^2+x+1 ← Can't divide highest terms anymore

- This is the remainder

$x^2+x+1 \longleftrightarrow \underbrace{111}_{r \text{ bits}}$

(generator is r+1)

5. Bitstream: 10011101

$x^7+x^4+x^3+x^2+1$

Generator: x^3+1 (r=3)

$$
\text{Checksum} = \begin{array}{r}
x^4+1 \\
x^3+1\,\overline{)\,x^7+x^4+x^3+x^2+1} \\
-\;x^7+x^4 \\
\hline
x^3+x^2+1 \\
x^3+1 \\
\hline
x^2
\end{array}
$$

x^2 → remainder: 100

Append to message:

$10011101100 \equiv x^{10}+x^7+x^6+x^5+x^3+x^2 + x^8$

↓ invert

→ For there to be no error, remainder must be 0

$$
\begin{array}{r l}
 & x^7 + x^5 + x^3 \\
x^3+1 \,\big) & x^{10} + x^8 + x^7 + x^6 + x^5 + x^3 + x^2 \\
 & x^{10} + x^7 \\
\hline
 & x^8 + x^6 + x^5 + x^3 + x^2 \\
 & x^8 + x^5 \\
\hline
 & x^6 + x^3 + x^2 \\
 & x^6 + x^3 \\
\hline
 & x^2
\end{array}
$$

$\Rightarrow$ remainder at rec. isn't 0; error detected.

6. Why is CRC put in trailer rather than header?

→ As in the XOR-Shift implementation, the CRC (last r bits) are only used by the end of division

- This way we can start dividing while receiving (even before CRC bits come in)