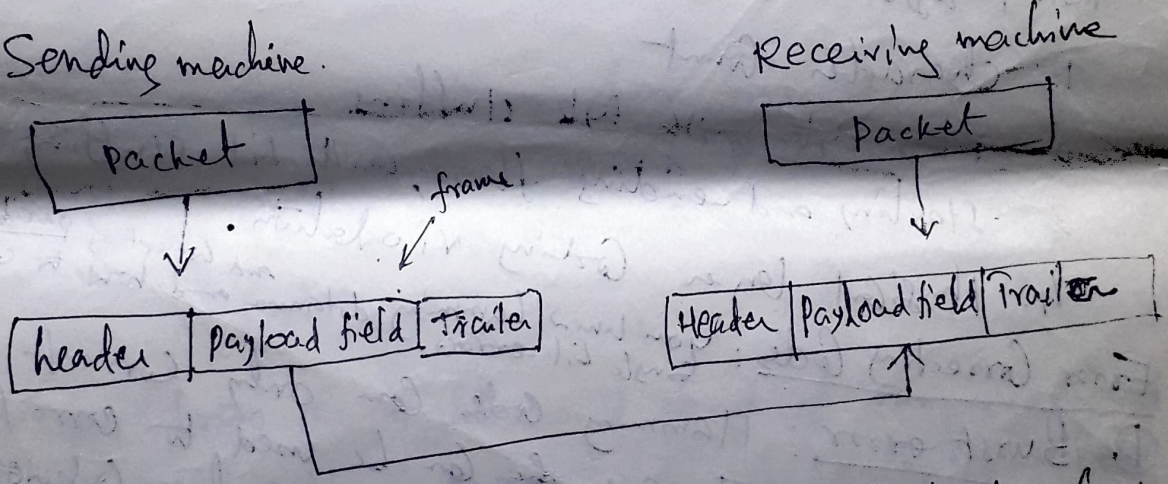
Data link layer design issues:

1. Providing a well-defined service interface to the network layer
2. Dealing with transmission errors
3. Flow Control: regulating the flow of data so that slow receivers are not swamped by fast senders.

To accomplish these goals, the data link layer takes the packets from the network layer and encapsulates them into frames for transmission. Each frame contains a frame header, a payload field for holding the packet, and a frame trailer.



The principle function of data link layer is to transferring data from the network layer on source machine to the network layer on the destination machine.

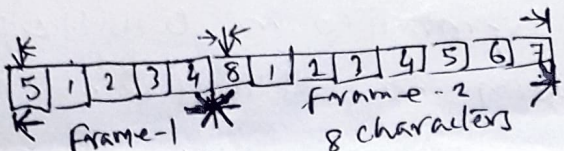
The data link layer can be design to offer various service — Unacknowledged or acknowledged / Connectionless or Connection oriented.

Framing: To provide services to the network layer, the
(2) link layer must use the service provided to it by the
physical layer. What the physical layer does is accept
a raw bit stream and attempt to deliver it to the
destination. This bit stream is not error free. The usual
approach is for the data link layer to break the bit
stream into discrete frames and compute the checksum
for each frame. When frame arrives at the destination,
the checksum is recomputed. If newly-computed
checksum is different from the one contained in the
frame, the data link layer knows that an error
has occurred and takes steps to deal with it.

Breaking the bit stream into frames can be done in
many different ways—

1. Character Count
2. Flag byte with byte stuffing
3. Starting and ending flags, with bit stuffing
4. Physical layer Coding violation.

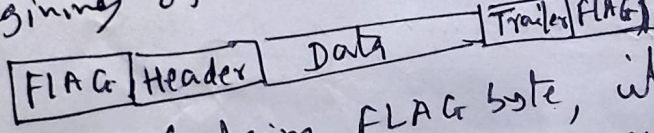
Character Count: This framming method uses a field in the header to specify the number of characters in the frame. When data link layer at the destination machine sees the character count, it knows how many characters follow and hence where the end of the frame is.



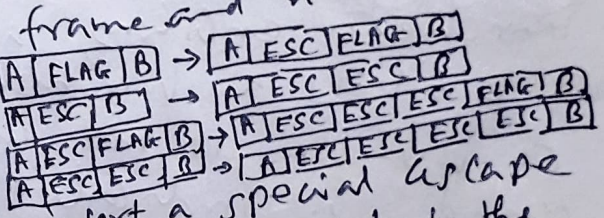
* If count is garbled by transmission error, the receiver will set out of synchronization and will be unable to locate the start of the next frame.

* Frames must be integer number of characters.
* character-code dependent.

Flag byte with byte stuffing: Each frame starts and ends with a special byte known as flag byte. If receiver loses synchronization, it can search for the flag byte to find the end of the current frame and hence the begining of the next frame.



If data contains FLAG byte, it insert a special escape (ESC) byte before each occurance of FLAG byte in the data. If data also contains ESC byte, it inserts another ESC byte before it. Then a framming FLAG byte can be distinguished from the one in data by the presence and absence of ESC byte before it.



Similarly, a single ESC byte is part of framming, whereas, a double one indicates that a single ESC occurred in the data. Disadvantage is that it is closely tied to 8-bit character encoding. Not all character codes use 8-bit characters. For example, UNICODE uses 16-bit characters. To solve this problem, bit stuffing was developed which allows arbitrary sized characters.

Optimal value of $k = \log_2 L$.

Bit Stuffing: Each frame begins and ends with a special bit sequence (01111110). Whenever, the receiver encounters five consecutive 1s in the data, it stuffs a 0-bit into the outgoing stream. When the receiver sees five consecutive 1s, followed by a 0 bit, it destuffs the 0-bit. Thus 01111110 in the data is transmitted as 011111010 but stored in the receiver memory as 01111110.

0110 111 111 11 01 → 0110 111 11011 101

Applicable only if each signal element encodes only one bit.

Physical Layer Coding violation: In encoding like Manchester encoding a 1-bit is a high-low pair and a 0-bit is a low-high pair. The combination high-high or low-low is not used for data. Thus can be used to indicate the beginning and end of a frame.

A packet is split into 10 frame, each of which has an 80% chance of arriving undamaged. If no error control is done, how many times the message needs to be sent to get the entire packet through?

1 packet = 10 frame, each as prob. $\cdot 8$ for success.

P(success of the whole packet) $= (\cdot 8)^{10} = P = 0.107$

Expected no. of times $= \sum_{i=1}^{\infty} i\, P(1-P)^{i-1} = P \sum_{i=1}^{\infty} i\,(1-P)^{i-1}$

$= P \cdot \frac{1}{\{1-(1-P)\}^2} = 1/P = \frac{1}{0.107} = 9.3.$

mean = 96/5 + 16 T.O

Consider a bit stuffing method when both start and end of a frame is indicated by the flag 01^K0 where 1^K denotes K consecutive ones.

a) what should be the bit stuffing rules at the transmitter?

Ans: In general with a a flag 01^K0, the bit stuffing is required whenever 01^{K-1} appears in the original data.

~~b) If 0 is preceded by 01^{K-1}~~

b) what should be the bit destuffing rules at the receiver?

Ans: If 0 is preceded by 01^{K-1} in the bit stream, remove it. If 0 is preceded by 01^K, it is the final bit of the flag.

c) Is it necessary to stuff a 0 in $01^{K-1}0$?

Ans: Let K=6, if 0 is not stuffed in 0111110, then $0\,111111\,0111 \rightarrow 0\,111110\,111$ and $0\,111111\,111 \rightarrow 0\,111110\,111$

How the receiver will differentiate? So, it is necessary to stuff a 0 in $01^{K-1}0$.

d) Assume all bit patterns are equally likely, Compute the expected overhead for a data packet of length L. Your expression for overhead calculation should include the flag length as well as the stuffed bits.

Ans: For a packet of length L, the string 01^{K-1} will occur about $L/2^K$ times on expectation.

So expected overhead is $L/2^K + (K+2)$ (# of stuffed bits; length of the flag)

This overhead is minimum when $K = \log_2 L$.

(2) The following character encoding is used in a data link protocol: A 01000111; B 11100011; FLAG 01111110; ESC 11100000
Show the bit sequence transmitted (in binary) for the 4-character frame: A B ESC FLAG when each of the following framing methods are used. 1. character count 2. flag bytes with byte stuffing 3. starting and ending flag bytes with bit stuffing.

Ans: 1. character count: 00000100 01000111 11100011 11100000 01111110
4 A B ESC FLAG

2. flag byte with byte stuffing
⇒ (insert ESC before flag)

01111110 A B ESC ESC ESC FLAG 01111110
FLAG FLAG

3. starting and ending flag bytes with bit stuffing.

01111110 01000111 11010011 11100000 01111101 0 01111110
FLAG A FLAG

(3) Consider the framing method starting and ending flag bytes with bit stuffing where 01111110 is used as the flag byte. Given an original data size of 64 bits, what is the largest and smallest number of bits that may need to be transmitted? Remember to include both begin and end flag bytes and the stuffed bits.

Ans: 8 bits for start flag + 8 bits for end flag. If every bit is a 1, we will need to stuff a 0 every 5th bit. This adds another 64/5 = 12 bits. So max = 12+16+64 = 92 bits.
If there are no sequence of 5 1's, we do not have to stuff at all. So min = 16+64 = 80 bits.
If you include a header of 32 bit then
min = 64+32+16 = 112 bits.
max = 96/5 + 16 + 64 + 32 = 19+16+96 = 131 bits.