Computer Networks
Framing

Saad Mneimneh
Computer Science
Hunter College of CUNY
New York

“Who **framed** Roger rab**bit**”?
A **detect**ive, a woman, and a rab**bit** in a **network** of trouble

1 Introduction

We will skip the physical layer and the detail of how communication signals are sent and received. In doing so, we are simply going to assume (a correct assumption) that we are capable of sending **bits over a single link**. Using this abstraction, the DLC layer becomes responsible for reliable transmission of **packets over a single link**. Reliable transmission of packets means that every packet is delivered once, only once, without errors, and in order. To achieve this goal, the DLC layer performs three operations:

- Framing: determines start and end of packets (all we see is a string of bits) and their order by inserting packet headers
- Error detection: determines when errors exist using packet trailers
- Error correction: retransmits packets containing errors

2 Framing

The main problem is to decide where successive packets start and end. Therefore, the DLC encapsulates the packet into a frame by adding its own header and trailer.

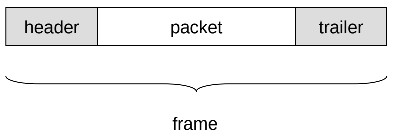

Figure 1: From packet to frame

Moreover, if there is a period of idle fills between successive frames (a synchronous bit pipe keeps on sending even if there is no data), it becomes also necessary to separate idle fills from frames. Even when idle fills are replaced by dead periods (intermittent bit pipe), the problem is not simplified, because often there are no dead periods.

For instance, where is the data in the following string of bits:

$$\ldots 0101101011001011010110111010111 0\ldots$$

2.1 Character based framing

Character based codes, such as ASCII (7 bits and 1 parity bit), provide binary representation for keyboard characters and terminal control characters. The idea in character based framing is that such codes can also provide representation for various communication characters:

- SYN: a string of SYN characters provide idle fill between frames when a sending DLC has no data to send (but a synchronous modem requires bits)
- STX: start of text
- ETX: end of text

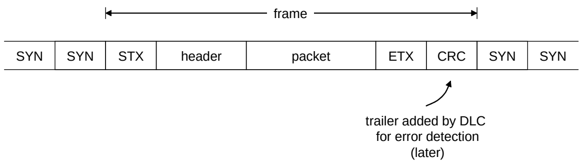

Figure 2: Character based frame

One disadvantage of character based framing is that the frame must contain an integer number of characters; otherwise, the characters cannot be read correctly. This also raises a question: how do we send binary data? For instance, a packet is a binary string of arbitrary length. Even more important is that a packet may possibly contain, say the ETX character, which could be wrongly interpreted as end of frame.

To solve these problems, a special control character DLE (Data Link Escape) is inserted before any intentional use of the communication control characters. The DLE character is not inserted before the possible appearance of these characters as part of the binary data. Of course this raises another question: what if DLE itself appears in the data? Well, then we can insert a DLE before each appearance of DLE in the data. For example DLE ETX (but not DLE DLE ETX) is end of frame, and DLE DLE ETX (but not DLE DLE DLE ETX) is DLE ETX in the data.

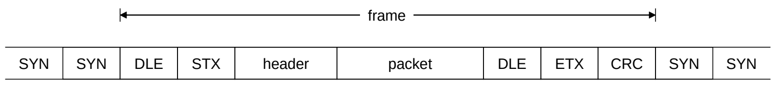

Figure 3: The DLE escape character

Character based framing adds too much overhead; for instance, at least 6 characters/packet need to be added for synchronous bit pipes: SYN SYN DLE STX . . . DLE ETX. This was the primary framing method from 1960 to 1975, e.g. ARPANET.

2.2 Length field

The basic problem as seen so far is to inform the receiving DLC where each idle fill ends and where each frame ends. In principle, an idle fill is easy to identify because it is represented by a fixed string. The end of a frame, however, is harder to indicate because the frame consists of arbitrary and unknown bit string.

Therefore, one solution is to include a length field of certain number of bits in the header (e.g. DECNET).

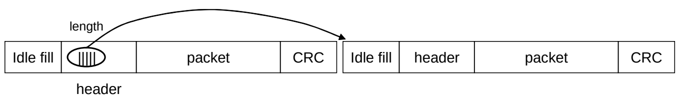

Figure 4: Length field in header

As it can be seen from the above figure, once synchronized, DLC can always tell where the next frame starts. However, one drawback of this technique is the difficulty to recover from errors in the length field; for instance, a resynchronization is needed after such an error.

Another drawback of this technique is that the length field restricts the packet size; the length field must contain at least $\lfloor \log_2 MaxFrameSize \rfloor + 1$ bits (that's the overhead). Therefore, if L is the size of the length field in bits, then $L > \log_2 MaxFrameSize$ and $MaxFrameSize < 2^L$ bits.

2.3 Fixed length packets/frames

In networks with fixed length packets, the length field is implicit (not needed) because all packets (and hence frames) have the same size. An example of such networks is ATM, where all packets are 53 bytes long. This framing technique requires synchronization upon initialization. Another drawback is that message length often not a multiple of packet size. In this case, the last packet of the message will contain idle fills (what happens to the overhead?)

2.4 Bit oriented framing

In character based framing, DLE ETX indicates the end of the frame, and it is avoided within the frame itself by doubling each DLE character in the data. In bit oriented framing, a special binary flag indicates the end of the frame, and it is avoided within the frame itself by using a technique called *bit stuffing*. Therefore, the concept is the same, but the main difference is that a flag, unlike DLE ETX, can be of any length (later we see how to set that length to minimize the overhead). Standard protocols use 01111110 as the binary flat, we denote it by $01^6 0$. The same flag can be used to indicate the start of the frame.

$$01111110 \ldots\ldots\ldots\ldots\ldots\ldots 01111110$$

Standard DLCs have also an abort capability in which a frame can be aborted by sending seven or more consecutive 1's (15 consecutive 1's mean the link is idle). Therefore, 0111111, i.e. 01^6 is the actual bit string that must be avoided in the data by bit stuffing.

Bit stuffing was invented by IBM in 1970, and works as follows: The sender DLC inserts (stuffs) a 0 after each appearance of five consecutive 1's, and appends the flag 01^60 (without stuffing) at the end of the frame. The receiver DLC deletes the first 0 after each string of five consecutive 1's. If six consecutive 1's are seen, this is interpreted as the end of the frame.

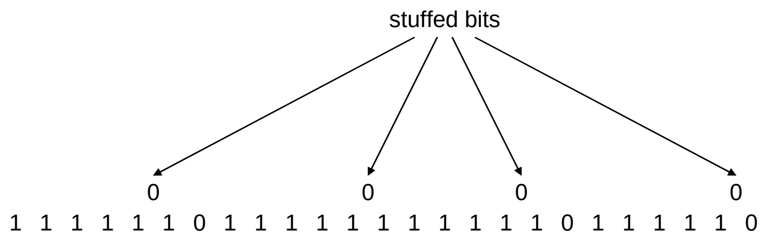

Figure 5: Bit stuffing

Something to think about: which ones of the above stuffed bits can be avoided (provided the receiver's rule for deleting stuffed bits is changed accordingly)?

Assume that the flag has the pattern 01^j0. How long should the flag be. Let us consider the overhead. On one hand, if j increases, we have less stuffing but a longer flag. On the other hand, if j decreases, we have more stuffing but a shorter flag. For the sake of our analysis, let us assume a random string of bits with $p(0) = p(1) = 1/2$. Therefore, insertion after the i^{th} bit occurs with probability $(1/2)^j$.

$$0\underbrace{1\ldots1}_{j-1}{}^{\downarrow}$$

Also another type of insertion after the i^{th} bit occurs with probability $(1/2)^{2j-1}$

$$0\underbrace{1\ldots1}_{j-1}\underbrace{1\ldots1}_{j-1}{}^{\downarrow}$$

Since $(1/2)^{2j-1} << (1/2)^j$, we can ignore such event and events of insertion due to yet longer strings of 1's. The probability of stuffing after the i^{th} bit is approximately 2^{-j}, which is also $E[\#\ stuffed\ bits\ @\ i]$, the expected number of insertions after the i^{th} bit. By the linearity of expectation, $E[\#\ stuffed\ bits\ |K] \approx K2^{-j}$ (be careful at boundary), where K is the length of the frame. Therefore,

$$E[\#\ stuffed\ bits] = E[K]2^{-j}$$

$$E[overhead] = \underbrace{E[K]2^{-j}}_{stuffed} + \underbrace{j+2}_{flag}$$

Minimizing with respect to j, we need smallest j such that

$$E[K]2^{-j} + j + 2 < E[K]2^{-(j+1)} + (j+1) + 2$$

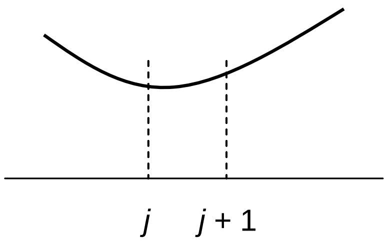

Figure 6: Minimizing overhead

$$E[k]2^{-(j+1)} < 1$$

The smallest j that satisfies the above inequality is $j = \lfloor \log_2 E[k] \rfloor$. We can show that (homework?):

$$\log_2 E[K] + 2.914 \leq E[overhead] \leq \log_2 E[K] + 3$$

2.5 What is the best overhead?

Length field based framing and bit oriented framing are comparable in their overhead,

$$\approx \log_2 K$$

where K is the length of the frame. Can we do better? Information theory tells us NO. Essentially, we are encoding information about the length of the frame at the sending DLC and transmitting it to the receiving DLC. At least we need a number of bits equal to the entropy

$$H = \sum_{k=1}^{K_{max}} p(k) \log_2 \frac{1}{p(k)}$$

When distribution is uniform, i.e. $p = \frac{1}{K_{max}}$, $H = \log_2 K_{max}$.

2.6 Maximum packet size

Recall that the transport layer is the one responsible for breaking messages into packets. So how should the transport layer choose the maximum packet size? Practically speaking, this is not a big deal since IP fragments packets further if necessary. Usually packet size is about 1500 bytes (from Ethernet). But theoretically speaking, for a given overhead of V bits, what is the best maximum packet size K_{max}?

Let M be the message length. Then we have

$$M + \lceil \frac{M}{K_{max}} \rceil V$$

bits to send

K_{max}	V	K_{max}	V	K_{max}	V	$Rest$	V

The overhead per message is $\lceil \frac{M}{K_{max}} \rceil V$. If K_{max} increases, we have a small overhead per message; therefore, less bits to transmit. But since the message might need to traverse many links, a larger packet size may result in a slower delivery time, because each intermediate node must wait until it completely receives a packet before relaying it.

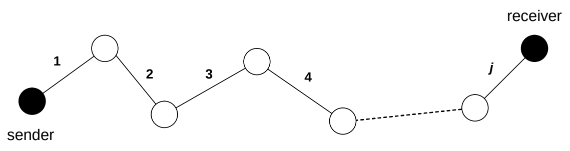

Figure 7: Traversing links

This additional total waiting time can be computed as:

$$\frac{(K_{max} + V)(j-1)}{c}$$

where c is the capacity of a link in bps (bandwidth). Therefore,

- $K_{max} \nearrow \Rightarrow$ small overhead per message, larger wait
- $K_{max} \searrow \Rightarrow$ bigger overhead, but smaller wait

Which value of K_{max} achieves the best tradeoff? Let us minimize the time to deliver the message. The time needed for the message to traverse j links is:

$$T = \frac{(K_{max} + V)(j-1)}{c} + \frac{M + \lceil \frac{M}{K_{max}} \rceil V}{c} + P + Q$$

where P and Q are the propagation delay and the queuing delay respectively. Therefore, the expected value of T is approximately given by:

$$E[T] = \frac{(K_{max} + V)(j-1)}{c} + \frac{E[M] + \frac{E[M]}{K_{max}} V}{c} + P + Q$$

and we need to minimize

$$K_{max}(j-1) + \frac{E[M]}{K_{max}} V$$

By taking the first derivative with respect to K_{max} and setting it to zero, we get

$$K_{max} = \sqrt{\frac{E[M]V}{j-1}}$$

References

Dimitri Bertsekas and Robert Gallager, Data Networks
Larry Peterson and Bruce Davie, Computer Networks: A Systems Approach