Data Communication Networks

Lecture 2

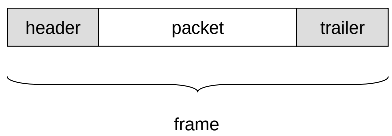
Saad Mneimneh
Computer Science
Hunter College of CUNY
New York

DLC

- We are not going to study the physical layer and how communication signals are sent and received
- We will assume that we are capable of sending **bits** over a link
- DLC is responsible for reliable transmission of **packets** over a link
  - every packet is delivered once,
  - only once,
  - without errors,
  - and in order
- To achieve this goal, we have:
  - Framing: determine start and end of packets
  - Error detection: determine when errors exist
  - Error correction: retransmit packets containing errors

Framing

- Recall that DLC adds its own header and trailer to the packet $\Rightarrow$ frame



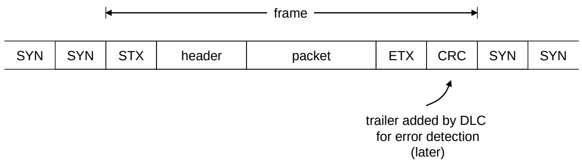
- The problem is to decide where successive frames start and end
  - in some cases, there is a period of idle fills between successive frames (e.g. synchronous bit pipe)
  - it is also necessary to separate idle fills from frames
  - even when idle fill are replaces by dead periods (intermittent bit pipe), problem is not simplified, e.g. often no dead periods

$$\ldots 01011010110010110101101110101110 \ldots$$

Where is the data?

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
- 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



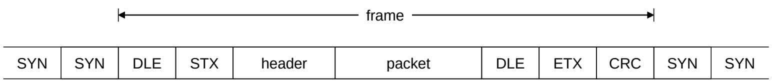
- Frame must contain integer number of characters
- Frame is character code dependent ⇒ how do we send binary data
  - e.g. packet is an arbitrary binary string and may possibly contain the ETX character for instance, which could be wrongly interpreted as end of frame

Character based framing (cont.)

- A special control character DLE (Data Link Escape) is inserted before any intentional use of communication control characters
  - e.g. DLE is not inserted before the possible appearance these characters as part of the binary data

But what if DLE appears itself in the data?

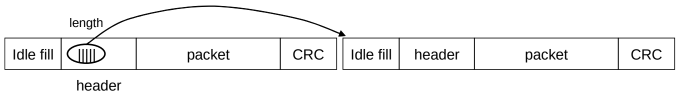
- Insert a DLE before each appearance of DLE in data
  - e.g. DLE ETX (but not DLE DLE ETX): end of frame
  - e.g. DLE DLE ETX (but not DLE DLE DLE ETX): DLE ETX in data



- Too much overhead: at least 6 characters/packet
- Primary framing method from 1960-1975

Length field

- The basic problem is to inform the receiving DLC where each idle fill ends and where each frame ends
  - idle fill: in principle, easy to identify because it is represented by a fixed string
  - frame: harder to indicate where it ends because it consists of arbitrary and unknown bit string
- Include a length field of certain number of bits in header (e.g. DECNET)



- Once synchronized, DLC can always tell where next frame starts
- Length field restricts packet size
  - length field must be $\lfloor \log_2 MaxFrameSize \rfloor + 1$ bits (that's the overhead)
- Difficult to recover from error in length field
  - e.g. resynchronization needed after error in length field

Maximum frame size

- How should transport layer choose maximum packet size?
  - not a big deal since IP fragments packets further if necessary
  - usually about 1500 bytes (from Ethernet)
  - but theoretically speaking?
- Let K_{max} be the maximum packet size. Assume V overhead bits. Let M be the message length. Then we have

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

bits to send

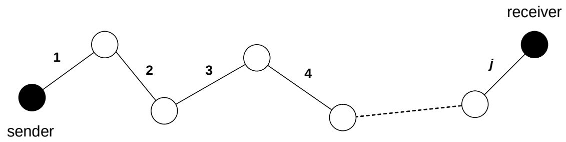
K_{max}	V	K_{max}	V	K_{max}	V	$Rest$	V

Therefore,

- $K_{max} \nearrow \Rightarrow$ small overhead per message
- $K_{max} \searrow \Rightarrow$ faster delivery of message (why?)

Maximum frame size (cont.)

- What is the time needed for the message to traverse j links?



- Assume capacity of link is c bps (bandwidth)

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

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

- Minimizing (take first derivative and set it to zero)

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

Fixed length packets/frames

- Length field is implicit (not needed)
  - e.g. ATM, all packets are 53 bytes
- Requires synchronization upon initialization
- Message length not multiple of packet size
  - last packet contains idle fill (efficiency?)

Bit oriented framing

- In character based framing, DLE ETX indicates the end of frame
  - avoided within frame by doubling each DLE character
- In bit oriented framing, a special binary flag indicates the end of frame
  - avoided within frame using a technique called *bit stuffing*
- The difference is that a flag can be of any length (later we see how to set length to minimize overhead)
- Standard protocols use 01111110, we denote it by 01^60
- The same flag can be used to indicate start of frame

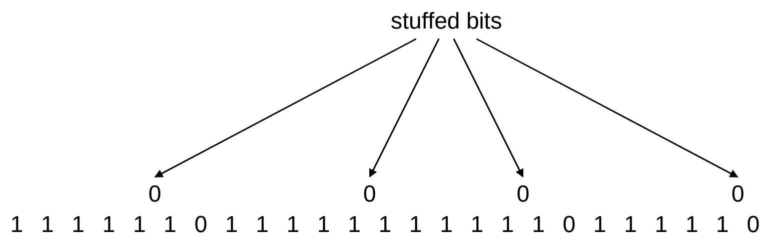
$$01111110 \ldots\ldots\ldots\ldots\ldots\ldots 01111110$$

- Standard DLCs have also an abort capability in which a frame can be aborted by sending 7 or more consecutive 1's (15 consecutive 1's $\Rightarrow$ link is idle)
- Therefore, 0111111, i.e. 01^6 is the actual bit string that must be avoided in data

Bit stuffing, 1970 by IBM

Bit stuffing

- Sender DLC
  - insert (stuff) a 0 after each appearance of five consecutive 1's
  - append the flag 01^60 (without stuffing) at the end of frame
- Receiver DLC
  - delete the first 0 after each string of five consecutive 1's
  - if six consecutive 1's are seen $\Rightarrow$ end of frame



Which ones of these stuffed bits can be avoided (provided receiver's rule for deleting stuffed bits is changed accordingly)?

Overhead of bit stuffing

- What is the overhead of bit stuffing (assume flag $= 01^j0$)?
  - if $j \nearrow \Rightarrow$ less stuffing but longer flag
  - if $j \searrow \Rightarrow$ more stuffing but shorter flag
- For the sake of analysis, assume a random string of bits with $p(0) = p(1) = 1/2$
  - insertion after i^{th} bit occurs with probability $(1/2)^j$

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

  - also insertion after 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 dur to yet longer strings of 1's
- The probability of stuffing after the i^{th} bit is approximately 2^{-j}, which is also $E[stuffed\ @\ i]$
- By linearity of expectation, $E[stuffed|K] \approx K2^{-j}$ (be careful at boundary), where K is the length of the frame

Overhead of bit stuffing (cont.)

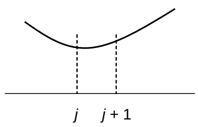
- From previous slide

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

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

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

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



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

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

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

Can we do better?

- 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. $p = \frac{1}{K_{max}}$, $H = \log_2 K_{max}$.

Error detection

- All framing techniques are sensitive to errors
  - error in DLE ETX
  - error in length field (re-sync needed)
  - error in flag
    - flag ruined (frame disappears)
    - flag created by error (extra frame appears)
  - error in data itself
- Flag approach is least sensitive to errors because a flag will eventually appear
  - the only thing is that an erroneous packet/frame is created
  - but this can be removed by error detection techniques
- Error detection is used by receiving DLC to determine if a frame contains errors
- If frame contains errors, receiver requires the transmitter to resend the frame

How to detect errors?

- The problem (simply stated)
  - Assume that DLC knows where frames begin and end (solved earlier)
  - Determine which frames contain errors
- Error cannot be detected by analyzing the packet itself (why?)
- Therefore, extra bits must be used
  - Parity check
    - single parity
    - multiple parity (e.g. horizontal and vertical)
  - Cyclic Redundancy Check CRC

Single parity check

- Add one parity bit
- Parity bit is 1 if frame contains ODD number of 1's, and 0 otherwise
  - e.g. 1001010 [1]
  - e.g. 0111010 [0]
- Therefore, a frame always contains an even number of 1's
- Receiver counts number of 1's
  - ODD number of 1's: an error must have occurred
  - EVEN number of 1's: *interpret* as no error (why?)
    - even number of errors cannot be detected!
    - $p(undetected\ error) = \sum_{i\ even} \binom{k}{i} p^i (1-p)^i$ assuming independent errors (simplification), where k is the length of the frame, p is the probability of error (binary symmetric channel)

Horizontal and vertical parity checks

- Data is visualized as a rectangular array

1	0	0	1	0	1	0
0	1	1	1	0	1	0
1	1	1	0	0	0	1
1	0	0	0	1	1	1
0	0	1	1	0	0	1

- Parity bit is computed for every row and every column
- If an even number of errors is confined to a single row, each of them can be detected by the corresponding column parity checks (and vice-versa)

Horizontal and vertical parity checks

- Data is visualized as a rectangular array

1	0	0	1	0	1	0	1	
0	1	1	1	0	1	0	0	
1	1	1	0	0	0	1	0	horizontal checks
1	0	0	0	1	1	1	0	
0	0	1	1	0	0	1	1	
1	0	1	1	1	1	1	0	← always consistent with both checks (why?) (addition modulo 2)

vertical checks

- Parity bit is computed for every row and every column
- If an even number of errors is confined to a single row, each of them can be detected by the corresponding column parity checks (and vice-versa)

Horizontal and vertical parity checks

- Data is visualized as a rectangular array

1	0	0	1	0	1	0	1	
0	1	1	1	0	1	0	0	
1	1	[1]	0	0	[0]	1	0	horizontal checks
1	0	0	0	1	1	1	0	
0	0	[1]	1	0	[0]	1	1	
1	0	1	1	1	1	1	0	← always consistent with both checks (why?) (addition modulo 2)

vertical checks

- Parity bit is computed for every row and every column
- If an even number of errors is confined to a single row, each of them can be detected by the corresponding column parity checks (and vice-versa)
- Some errors are still undetected
  - e.g. any 4 errors forming a rectangle

Arbitrary parity check codes

- Parity check code is simply addition modulo 2

$$\underbrace{s_1\ s_2\ \dots\ s_k}_{K\text{bit frame}}\ \underbrace{c_1\ c_2\ \dots\ c_L}_{L\text{bit parity check}}$$

- every c_i is the sum of some bits in $s_1 \dots s_K$

$$c_i = \sum_{j=1}^{K} \alpha_{ij} s_j$$

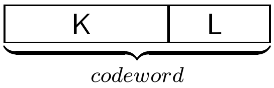
where α is an $L \times K$ 0-1 matrix

s_1	s_2	s_3	c_1	c_2	c_3	c_4
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	1	1	1
0	1	1	1	0	1	0
1	0	0	1	1	1	0
1	0	1	0	0	1	1
1	1	0	1	0	0	1
1	1	1	0	1	0	0

$c_1 = s_1 + s_3$
$c_2 = s_1 + s_2 + s_3$
$c_3 = s_1 + s_2$
$c_4 = s_2 + s_3$

$$\alpha = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Effectiveness of a code



The effectiveness of a code is usually measured by three parameters

- minimum distance of the code d: the smallest number of errors that can convert one code word into another
- burst detecting capability
  - burst = number of bits from first error to last error (inclusive)
  - defined as: largest integer B such that a code can detect all bursts $\leq B$
- probability that a random string is accepted as error free
  - useful when framing is lost, e.g. check code is random with respect to received frame
  - We have 2^K codewords (why?) and 2^{K+L} random strings
  - therefore, the probability is 2^{-L}

Effectiveness of a code (cont.)

- Minimum distance d
  - single parity:
  - horz. and vert. parity:
- Burst detecting capability B
  - single parity:
  - horz. and vert. parity (assumes data sent by rows):

Cyclic Redundancy Check

- For convenience, denote the data bits as

$$s_{K-1}, s_{K-2}, \ldots, s_0$$

- Represent the string as a polynomial

$$s(x) = s_{K-1}x^{K-1} + s_{K-2}x^{K-2} + \ldots + s_1 x + s_0$$

- Similarly, we can represent the CRC (with L bits) as

$$c(x) = c_{L-1}x^{L-1} + c_{L-2}x^{L-2} + \ldots + c_1 x + c_0$$

- The whole frame can be represented as a polynomial

$$f(x) = s(x)x^L + c(x) = \underbrace{s_{K-1}} x^{L+K-1} + \ldots + \underbrace{s_0} x^L + \underbrace{c_{L-1}} x^{L-1} + \ldots + \underbrace{c_0}$$

- Why this polynomial representation? because we are going to obtain c as $c(x)$ by dividing $s(x)x^L$ by some polynomial $g(x)$

Obtaining $c(x)$

- We know s_i (data) for $i = 0 \ldots K-1$
- How do we compute c_i (CRC) for $i = 0 \ldots L-1$?
  - let $g(x) = x^L + g_{L-1}x^{L-1} + \ldots + g_1 x + 1$ be given ($g_L = g_0 = 1$)
  - then

$$c(x) = Remainder\left[\frac{s(x)x^L}{g(x)}\right]$$

$\nwarrow$ division modulo 2

  - result is a degree $L-1$ polynomial $\Rightarrow$ L bits
- Example: $s = 101$ ($K = 3$) and $g(x) = x^3 + x^2 + 1$ ($L = 3$)
  - $s(x) = ?$
  - $s(x)x^L = ?$
  - divide $s(x)x^L$ by $g(x)$ and obtain remainder (i'll do it on the board?)

Another example

$s = 110101$

$g(x) = x^3 + 1$

$s(x) = x^5 + x^4 + x^2 + 1$

$s(x)x^L = x^8 + x^7 + x^5 + x^3$

$$
\begin{array}{l|l}
x^8 + x^7 + x^5 + x^3 & \underline{x^3 + 1} \\
\underline{x^8 + x^5} & x^5 + x^4 + x + 1 \\
x^7 + x^3 & \\
\underline{x^7 + x^4} & \\
x^4 + x^3 & \\
\underline{x^4 + x} & \\
x^3 + x & \\
\underline{x^3 + 1} & \\
x + 1 &
\end{array}
$$

$c(x) = 0.x^2 + 1.x + 1 \ (L = 3) \Rightarrow c = 011$

110101	011

Using bits only

$s = 110101$

$g(x) = x^3 + 1 \ (L = 3)$

$$
\begin{array}{lcccc}
g = & 1 & 0 & 0 & 1 \\
 & \downarrow & \downarrow & \downarrow & \downarrow \\
 & x^3 & x^2 & x^1 & x^0
\end{array}
$$

$$
\begin{array}{lccccccccc}
 & & & & & & & \multicolumn{3}{c}{\overbrace{\hphantom{0 \quad 0 \quad 0}}^{L}} \\
s(x)x^L \Rightarrow & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
 & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
 & x^8 & x^7 & x^6 & x^5 & x^4 & x^3 & x^2 & x^1 & x^0
\end{array}
$$

Using bits only (cont.)

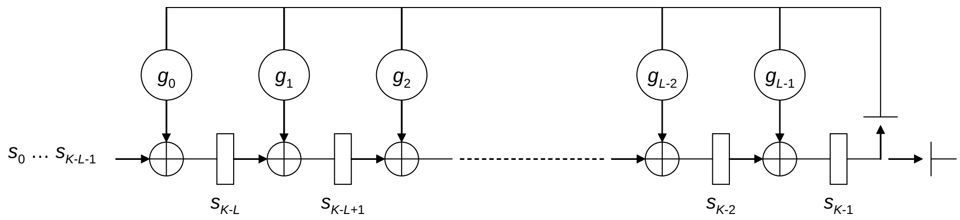
```
1 1 0 1 0 1 0 0 0 | 1 0 0 1
1 0 0 1           |----------
-------             1 1 0 0 1 1
  1 0 0 0
  1 0 0 1
  -------
    0 0 1 1
    0 0 0 0
    -------
      0 1 1 0
      0 0 0 0
      -------
        1 1 0 0
        1 0 0 1
        -------
          1 0 1 0
          1 0 0 1
          -------
            0 1 1
```

Using bits only (cont.)

```
1 1 0 1 0 1 0 0 0 | 1 0 0 1
1 0 0 1           |----------
-------             1 1 0 0 1 1
  1 0 0 0
  1 0 0 1
  -------
    0 0 1 1
    0 0 0 0
    -------
      0 1 1 0
      0 0 0 0
      -------
        1 1 0 0
        1 0 0 1
        -------
          1 0 1 0
          1 0 0 1
          -------
            0 1 1
```

- Multiply g by first bit
- Add
- Shift
- Can be implemented using a feedback shift register

Feedback shift register



- Register is initialized with first L bits of s
- After K shifts, switch is moved and CRC is read

How does $c(x)$ help?

$$s(x)x^L = g(x)z(x) + c(x)$$

$$s(x)x^L + c(x) = g(x)z(x) + \underbrace{c(x) + c(x)}_{0} \text{ (modulo 2)}$$

$$s(x)x^L + c(x) = g(x)z(x)$$

$$f(x) = g(x)z(x)$$

- Polynomial representation of the frame is multiple of $g(x)$
- Assume $f(x)$ received as $y(x)$
- Receiver DLC computers

$$Remainder\left[\frac{y(x)}{g(x)}\right]$$

  - if remainder is not zero $\Rightarrow$ error in frame
  - if remainder is zero, declare the frame error free

Undetected errors

- Assume error is $e(x)$, i.e. $y(x) = f(x) + e(x)$
- Then, $\frac{y(x)}{g(x)} = \frac{f(x)}{g(x)} + \frac{e(x)}{g(x)}$
- Therefore, we have undetected errors iff $e(x) \neq 0$ divisible by $g(x)$
- Single errors are always detected
  - assume undetected, i.e. $e(x) = x^i = g(x)z(x)$ for some i
  - since $g(x) = x^L + \ldots + 1$, multiplying $g(x)$ by any $z(x) \neq 0$ cannot produce x^i (must produce at least 2 terms)
- g(x) can be chosen such that
  - all odd number of errors are detected
  - all double errors are detected (if $K + L < 2^L$)
  - therefore, minimum distance $d = 4$
  - burst detecting capability $B = L$
  - probability of random string accepted is 2^{-L}
    - e.g. (L=16) $g(x) = x^{16} + x^{15} + x^2 + 1$ CRC-16
    - e.g. (L=16) $g(x) = x^{16} + x^{12} + x^5 + 1$ CRC-CCITT
    - e.g. (L=32) $g(x) =$ ... (see book page 64)