Data Link Layer

Lecture 7 Framing and introduction to CRC

In the data link layer, we try to "make sense" of the incoming bits
↓
which bits together mean something in the stream of data.
↳ The resulting unit is called a frame.

In a frame, we first need to demarcate where the section begins/ends.

High-Level Data Link Control (HDLC) is used as Layer-2 technology in Wide Area Networks.
(WANs)

The (HDLC) frame has a

begin sequence,	8 bits
header,	16 bits
body,	variable
CRC, and	16 bits
end sequence.	8 bits

The begin/end sequence are the same: 01111110
If there is nothing to send, we continuously send this
It also helps in clock synchronization.

What if this sequence appears elsewhere?
We do bit stuffing. Say 01111110 is somewhere in the middle.
↳ we insert bits.

What HDLC does is:
→ If you see 5 consecutive 1s, insert a 0.

01111100011111100011111110
↓
011111**0**00111111**0**10001111110**0**110

At the receiver,

we somehow have to remove these stuffed bits.

Wherever you see 5 consecutive 1s, remove the subsequent stuffed 0.

The end sequence still has 6 consecutive 1s.

But what if there is some bit error?

11111 {
- $0 \rightarrow$ Remove (bit stuffing)
- $10 \rightarrow$ Assume end sequence
- $11 \rightarrow$ Assume error has occurred and discard the frame (We discard everything until we see the sequence again)

This is very barebones though, we need something better for errors.

Cyclic Redundancy Check (CRC)

We just append the k-bit CRC to the n-bit dataword to get a "codeword".
($k = 16$ here)

The space of datawords is the set of all 2^n bit words.

We keep it such that only 2^n of the 2^{n+k} $(n+k)$-bit strings are valid.

An issue only arises when the error is such that the erroneous string is a codeword as well.

$\Rightarrow$ We need to ensure that codewords are "far apart".

Given $v, w \in \{0,1\}^n$, the Hamming distance between v and w is

$$d(v,w) = \{i \in [n] : v_i \neq w_i\}.$$

(number of positions they are distinct)

For a "code" $C \subseteq \{0,1\}^n$: the Hamming distance of C is

$$\min \{ d(v,w) : v, w \in C, v \neq w \}.$$

We want this distance to be large.

$GF(2) = \mathbb{F}(2)$ is a finite field with elements $\{0,1\}$.

Addition ($+$) has identity 0. (Note that $a+b = a-b$)

Multiplication ($\times$) has identity 1.

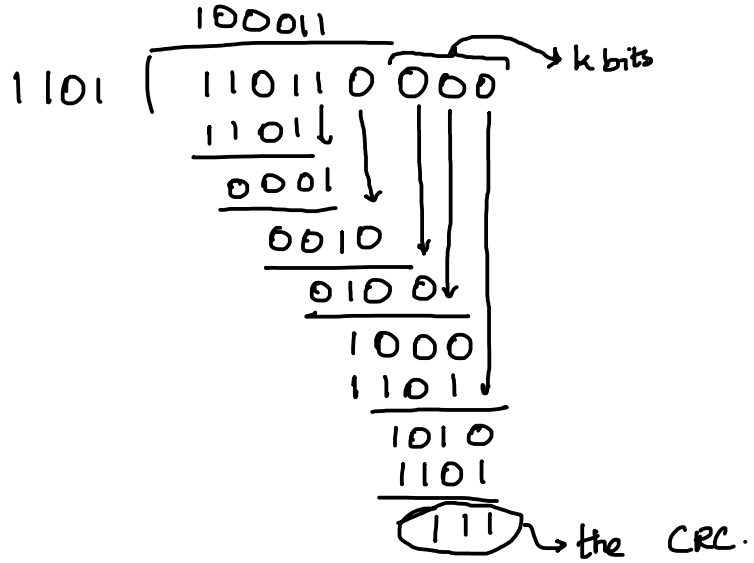
CRC is based on a cyclic code

If v is a codeword, cyclic shifts of v also result in codewords.

To generate a CRC, we use long division (in $\mathbb{F}_2$)

The divisor/generator is of length k+1 bits.

For example, say k=3 and the generator or "divisor" is 1101 and n=6 with dataword 110110



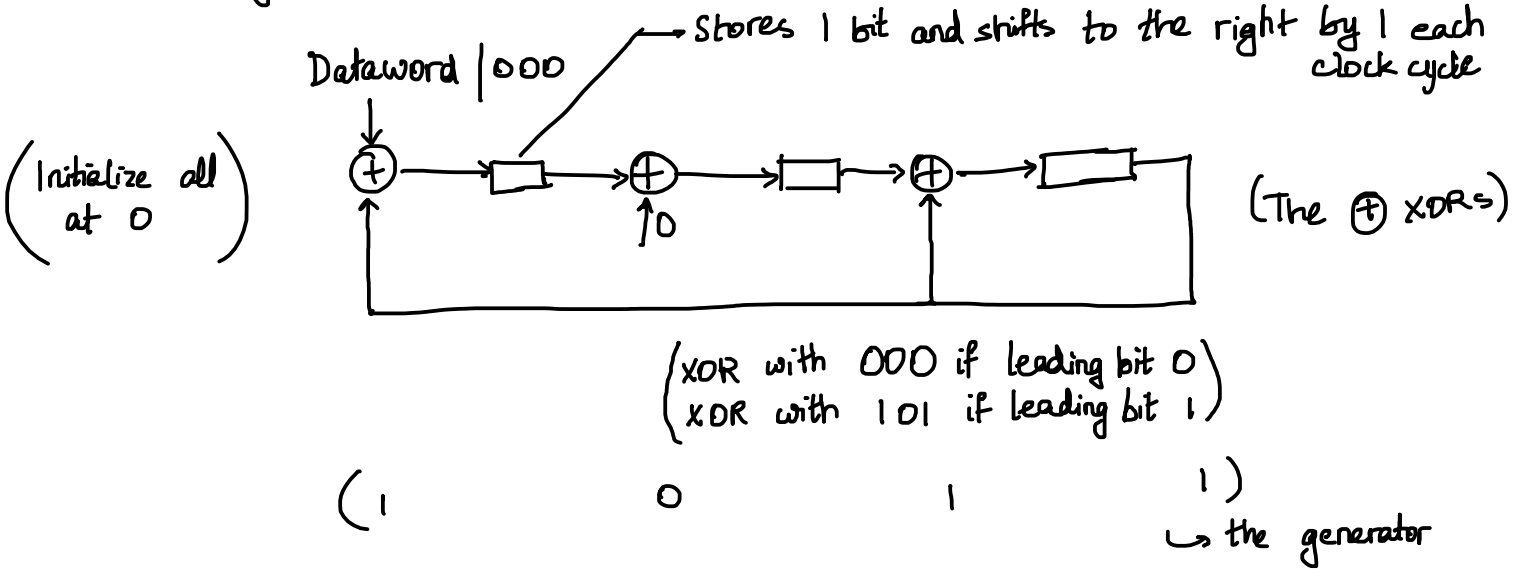
The codeword is then 110110 111.

dataword CRC

At the sender,

Long division for a given generator can be implemented using shift registers.

For generator 1101,



After Dataword | 000 is emptied, the CRC is left in the shift registers.

At the receiver, either

1. pass the dataword with k 0s through the CRC circuit and verify that the CRCs match or
2. pass the entire received word through the CRC circuit and verify that you get all 0s.

Lecture 8 CRC Polynomial Arithmetic

What bit errors can the CRC detect?

We will represent the divisor as a polynomial

For example, $1101 \equiv 1 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = x^3 + x^2 + 1$.

$1101 \times 11 = 10111$

↳ $x^4 + x^2 + x + 1$.

$(x^3 + x^2 + 1)(x+1) = x^4 + 2x^3 + x^2 + x + 1 = x^4 + x^2 + x + 1$

$(2 = 0$ in $\mathbb{F}_2)$

So say we transmit some codeword with equivalent polynomial $P(x)$. Let the error bitstring be $E(x)$.

The received word is then $P(x) + E(x)$.

We divide the received polynomial by $C(x)$ and if the resultant is 0, we say there is no bit error

↳ poly. corresponding to generator.

We want

$$\frac{P(x) + E(x)}{C(x)} \neq 0 \quad \text{if } E(x) \neq 0.$$

→ Single bit errors.

$E(x) = x^i$ for some i.

$$\frac{P(x) + E(x)}{C(x)} = \frac{P(x)}{C(x)} + \frac{E(x)}{C(x)} = \frac{E(x)}{C(x)}.$$

($\frac{P(x)}{C(x)} = 0$)

If $C(x) = x^k + \underbrace{\cdots}_{\text{0s or 1s}} + 1$, single bit errors can be detected.

$$E(x) = \underbrace{C(x) \cdot D(x)}_{\text{will have at least two non-zero powers of } x.}$$

(So for example, 1101 can detect single bit errors)

→ 2-bit errors.

$E(x) = x^j + x^i = x^i(x^{j-i} + 1)$ (suppose $j > i$)

Write each polynomial as a product of irreducible polynomials.

$$\frac{E(x)}{C(x)} = \frac{g_1(x) \cdots g_t(x)}{f_1(x) \cdots f_m(x)}$$

If C is of the form $x^k + \cdots + 1$, no $f_r(x)$ is of the form x^B.

$\Rightarrow$ no f_r will divide the x^i (if $i > 0$).

However, we could have

$(x^k + \cdots + 1)(\cdots) = x^\gamma + 1$ for some "large" γ.

So if $j-i$ is large (the errors are far apart), the bit errors might not be detected

The smallest γ such that $C(x)$ divides $x^\gamma + 1$ is called its order.

It is known how to find $C(x)$ of the form $x^k + \cdots + 1$ such that it has order $2^k - 1$.

So while this limits the length of the codeword that can be transmitted, it gets the job done quite well.

So if we have a 16 bit CRC, we can transmit data as large as $2^{16} - 1 - 16$ while detecting 2-bit errors.

→ Odd-bit errors

$E(x) = x^{i_1} + x^{i_2} + \cdots + x^{i_{2s+1}}$.

If $C(x)$ has $(1+x)$ as a factor, it cannot divide x of this form. Indeed,

$(1+1)\,D(1) = 0$ but $E(1) = 1$.

↑ substituting $x=1$

$D(x) \rightarrow 0001111000$

$xD(x) \rightarrow 0011110000$

0010000100 → each string of 1s in D two 1s in $(1+x)D(x)$

If $C(x)$ has an even number of terms (x^i), it can detect any error $E(x)$ with an odd number of terms

HDLC uses CRC-16-IBM

$C(x) = x^{16} + x^{15} + x^2 + 1$

CRC-32 has

$C(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1.$