Section 5. Serial Communication

Some Coding Concepts in Computer Communication

Error Control Coding

Goal: detect "errors" (e.g., flipped bits) in transmitted segment

Block code

*a set of code words of fixed length n, with each code word being an n-tuple over a finite field: **S**, **V** = all n-tuples*

Linear code

*If **S** forms a subspace of **V**, then **S** is called linear code*

codeword: a word in **S** is called codeword and otherwise noncodeword

***Hamming* weight (*w*)**: # of nonzero components of $\underline{X} = (x1, x2, \ldots xn)$

Hamming distance (d): # of positions in which the two words differ

Minimum Distance:

- the minimum of the distances between all pairs of code **C**, it is also the distance of the code.

examples:

x=(10011), y=(01010), w(x)=3, w(y)=2, d(x,y)=3

C= (001, 010, 100,110, 101, 011)

distance of **C** is 1

Coding Theory

Theorem 1:

It is necessary and sufficient that the distance of a code is at least d in order to detect any error pattern of weight d-1 or less

How do we design a code that has distance *d?*

A linear code can be represented by a matrix G, or H:

$$G = \begin{pmatrix} g_{0,0}\ g_{0,1} & \cdots & g_{0,n-1} \\ \cdots\cdots & & \\ g_{k-1,0}\ g_{k-1,1} & \cdots & g_{k-1,n-1} \end{pmatrix}$$

Where n is the code length, k is number of information bits, and n-k is number of parity bits

Representation of Linear Code

Matrix G is called generating matrix

Matrix H, defined as a null space of G, is called parity matrix

G or H uniquely defines a linear code

Code words of the linear code are generated by multiplying all possible information words to G

If $G = \begin{pmatrix} I_k & P_{n-k} \end{pmatrix}$

then $H = \begin{pmatrix} P^T & I_{n-k} \end{pmatrix}$

codewords =

$$u * \begin{pmatrix} g_{0,0}\ g_{0,1} & \cdots & g_{0,n-1} \\ \cdots\cdots & & \\ g_{k-1,0}\ g_{k-1,1} & \cdots & g_{k-1,n-1} \end{pmatrix}$$

For u = 0……00, 0……01,…,1……1

For Binary Numbers, It is simple:

Define binary addition:

0 + 0 = 0;

0 + 1 = 1;

1 + 0 = 1;

1 + 1 = 0.

Exclusive-OR

For example:

$$G = \begin{pmatrix} 1\,0\,1\,1\,0 \\ 1\,1\,0\,1\,1 \\ 0\,1\,0\,1\,0 \end{pmatrix}$$ Through canonical reduction $$G = \begin{pmatrix} 1\,0\,0\,0\,1 \\ 0\,1\,0\,1\,0 \\ 0\,0\,1\,1\,1 \end{pmatrix}$$

$$H = \begin{pmatrix} P^T & I_{n-k} \end{pmatrix} = \begin{pmatrix} 0\,1\,1\,1\,0 \\ 1\,0\,1\,0\,1 \end{pmatrix}$$

$U * H^T = \underline{0}$

Two Orthogonal Matrices

$$G = \begin{pmatrix} 1\,0\,\ldots\ldots 0\ p_{1,1}\ p_{1,2} & \cdots & p_{1,n-k} \\ 0\,1\,\ldots\ldots 0\ p_{2,1}\ p_{2,2} & \cdots & p_{2,n-k} \\ \cdots\cdots & & \\ 0\,0\,\ldots\ldots 1\ p_{k,1}\ p_{k,2} & \cdots & p_{k,n-k} \end{pmatrix}$$

$$H = \begin{pmatrix} p_{1,1}\ p_{2,1} & \cdots & p_{k,1}\ 1\,0\,\ldots\ldots 0 \\ p_{1,2}\ p_{2,2} & \cdots & p_{k,2}\ 0\,1\,\ldots\ldots 0 \\ \cdots\cdots & & \\ p_{1,n-k}\ p_{2,n-k} & \cdots & p_{k,n-k}\ 0\,0\,\ldots\ldots 1 \end{pmatrix}$$

For any i and j, we have

$$(0\,0..1\,\ldots\ldots 0\ p_{i,1}\ p_{i,2} \cdots p_{i,n-k}) *$$

$$* (p_{1,j}\ p_{2,j} \cdots p_{k,j}\ 0\,0..1\,\ldots\ldots 0)^T$$

$$= p_{i,j} + p_{i,j} = \underline{0}$$

Distance and Error Control

How do we relate distance of a code to G?

Theorem 2: For any codeword $\underline{u}$ of weight d in a linear code $\underline{C}$, d columns of its H matrix are linear dependent

Theorem 3: A linear code $\underline{C}$ has distance at least d iff every $d-1$ or fewer columns of its H matrix are linearly independent

recall that distance of a code is also the minimum weight.

An Example

Design a 5-bit linear code that can detect 2 bits errors

- come up with a H matrix with distance 3
- derive G from H
- generate all codewords
- encoder and decoder circuit.

$$H = \begin{pmatrix} 0\,0\,0\,1\,1 \\ 0\,1\,1\,0\,0 \\ 1\,0\,1\,0\,1 \end{pmatrix} \quad \text{Through canonical reduction} \quad H = \begin{pmatrix} 0\,1\,1\,0\,0 \\ 1\,1\,0\,1\,0 \\ 1\,1\,0\,0\,1 \end{pmatrix} \qquad G = \begin{pmatrix} 1\,0\,0\,1\,1 \\ 0\,1\,1\,1\,1 \end{pmatrix}$$

Defining CRC

Definition: A linear code $\underline{C}$ is said to be a cyclic code if for any code word $\underline{u}=(u_0, u_1, \ldots, u_{n-1})$ in $\underline{C}$, the word $\underline{u'} =(u_{n-1}, u_0, u_1, \ldots, u_{n-2})$ obtained by a shift of the bits to the right cyclically is also a code word in C

In cyclic code, we use polynomials to represent codeword, e.g 1101 is represented using $X^3 + X^2 + 1$

It is the algebra of polynomials modulo $x^n + 1$, $x^n = 1 \bmod (x^n + 1)$

For example, $X^7 + 1$ can be factorized as
$X^7 + 1 = (X+1)(X^3 + X + 1)(X^3 + X^2 + 1)$
Any factor can be a generator of a cyclic code.

An Example

Assume C is generated by $g(x) = (X^3 + X + 1)$, then we have

$0*g(x) = 0$,	0000000
$1*g(x) = X^3 + X + 1$	0001011
$x*g(x) = X^4 + X^2 + X$	0010110
$(x+1)*g(x) = X^4 + X^3 + X^2 + 1$	0011101
$X^2 * g(x) = x^5 + x^3 + x^2$	0101100
$(x^2+1)*g(x) = x^5 +x^2+x+1$	0100111
$(x^2+x)*g(x) = x^5+x^4+x^3+x$	0111010
$(x^2+x+1)*g(x) = x^5 +x^4 +1$	0110001
$X^3 * g(x) = x^6 +x^4 +x^3$	1011000
$(x^3+1)*g(x) = x^6 +x^4 +x +1$	1010011
$(x^3+x)*g(x) = x^6 +x^3+x^2+x$	1001110
$(x^3+x+1)*g(x) = x^6+x^2+1$	1000101
$(x^3+x^2) *g(x) = x^6+x^5+x^4+x^2$	1110100
$(X^3+x^2+1)*g(x) = x^6+x^5+x^4+x^3+x^2+x+1$	1111111
$(x^3+x^2+x)*g(x) = x^6+x^5+x$	1100010
$(x^3+x^2+x+1)*g(x) = x^6+x^5+x^3+1$	1101001

A few important points

Every proper divisor, $g(x)$, of $(x^n + 1)$ generates an (n,k) cyclic code, where r=n-k is the degree of $g(x) = x^r + x^{r-1} + \ldots..+1$

Every codeword polynomial is a multiple of g(x), since g(x) generate the code

d(x)g(x) generates codewords that are usually nonsystematic.
To generate systematic CRC code words, we divide $d(x)x^r$ by g(x), the remainder r(x) is added (concatenated) to the data part. i.e.
$U(x) = d(x)x^r + r(x)$

Algorithm:

- Append r 0's to d(x) -->k+r bits: $x^r d(x)$
- divide g(x) into $x^r d(x)$: $x^r d(x)/g(x)$
- subtract(add) the remainder (which is always r or fewer bits) from $x^r d(x)$ using modulo 2
- transmit the frame.

Which factor to chose as G?

Decoding: Rem {(T(x)+E(x))/g(x)} = Rem{E(x)/g(x)},
if **no remainder===> no error**!
Key, all errors that do not have g(x) as a factor can be detected.

- If $E(x)= x^i$: any g that has 2 or more terms will detect it

- If $E(x) = x^i + x^j = x^i(1 + x^{j-I})$, any g that has a constant term, 1, x is not a factor, it is sufficient to detect if $1 + x^{j-i}$ can not be divided by g.
  e.g. $x^{15} + x^{14} + 1$ will not divide $1 + x^k$ for k upto 32,768.

- If E(x) has odd number of bits:
  no polynomial with odd number of terms has x+1 as a factor
  By making x+1 a factor of g(x), we can detect all odd number bits errors.

Selecting g(x) for cyclic code

Proof:

Assume E(x) has odd # of terms and has (x+1) as a factor.

Then we have

$$E(x) = (x+1)Q(x)$$

substituting x by 1 we have

$$E(1) = (1+1)Q(1) = 0$$

But $E(1) = \underbrace{1 + 1 + \ldots\ldots + 1}_{\text{Odd \# of 1's}} = 1.$

Contradiction! Proved.

Selecting g(x) for cyclic code

- **r check bits detect all bursty errors of length <= r**

Assume $x^j (x^{k-1+\ldots\ldots} + 1)$ $0 < k <= r$, j gives position if $g_0 = 1$, x^j is not a factor of g(x)

if $k <= r$, $(x^{k-1+\ldots\ldots} + 1)$ /g(x) is never divisible ie remainder is never 0.

Example g(x):

CRC-12 = $(x^{12} + x^{11} + x^3 + x^2 + x + 1)$
CRC-16 = $(x^{16} + x^{15} + x^2 + 1)$
CRC-CCITT = $(x^{16} + x^{12} + x^5 + 1)$

An Example

$g(x) = (X^3 + X^2 + 1)$,

To generate the CRC code, we multiply d(x) by X^3 : $d(x) X^3$

Then divide $d(x) X^3$ by $g(x) = (X^3 + X^2 + 1)$,
add remainder to $d(x) X^3$ to obtain the transmitted word