Data Communication
(DC)

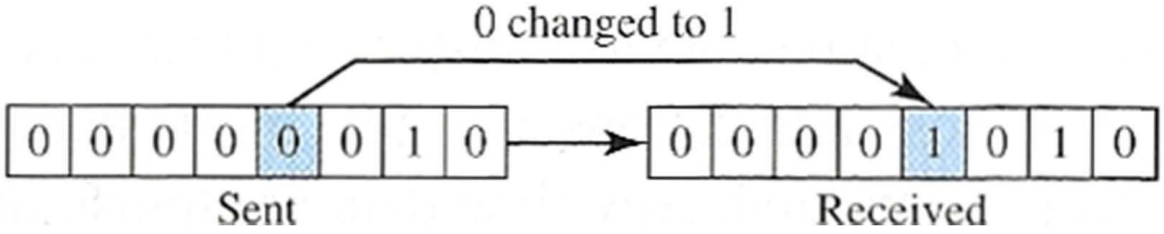
Lecture 3b

Overview of the contents

- **Types of errors**
- **Redundancy**
- **Detection versus correction**
- **Block coding**
- **Cyclic codes**

Data Link layer

Error detection and correction



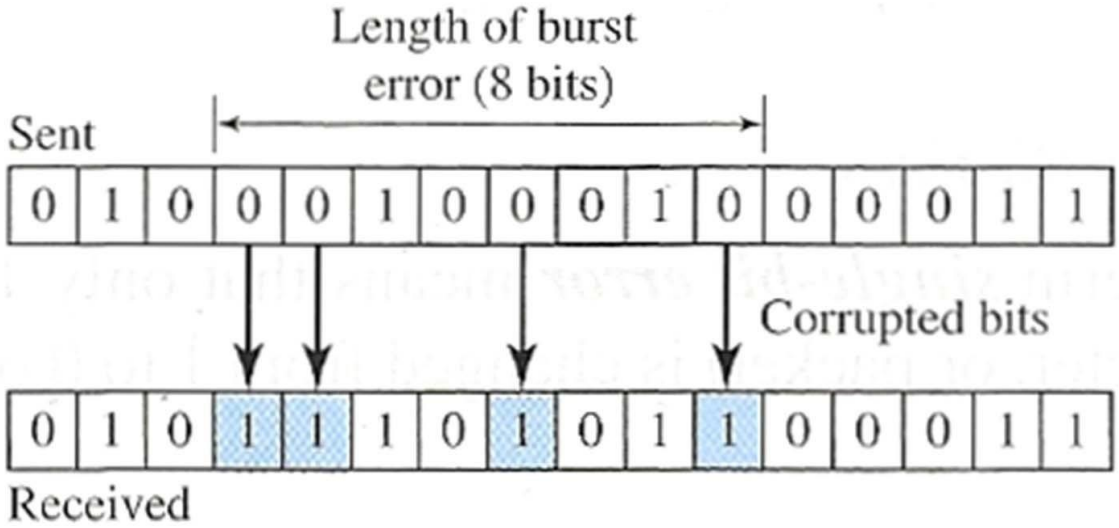
A **single-bit error** means only 1 bit of a given data unit (e.g., byte, character, or packet) is changed from 1 to 0 or from 0 to 1.

Single bit errors are less likely to occur.

e.g., a data rate of 1 Mbps, in such a case, will only have noise interference of a duration of less than 1µs

Data Link layer

Error detection and correction



A **burst error** means that 2 or more bits in a given data unithas changed from 1 to 0 or from 0 to 1.

Burst errors are more likely to happen in practice.

- If a data rate of 100 Kbps is affected by a noise of 1 ms, 100 bits are affected
- If a data rate of 100 Mbps is affected by a noise of 1 ms, 100,000 bits are affected

Data Link layer

Error detection and correction: Redundancy

The central concept in detecting or correcting errors is called redundancy. To be able to detect errors and possibly correct them, then we need to send some redundant bits along with our data stream.

- These redundant bits are added by the sender and removed by the receiver.
- Their presence allows the receiver to detect and possibly correct corrupted bits.

Data Link layer

Detecting errors is easier than correcting them

In case of **error detection**, we are only looking to see if any error has occurred. The answer to this question is either yes or no. So we do not consider about where these errors are and how many of them there are, but just whether they are there or not.

In case of **error correction**, we need to find not only the number of errors but also their exact locations.

- Correct 1 error in an 8 bit data unit (byte) = 8 different places.
- Correct 2 errors in an 8 bit data unit (byte) = 28 different places.
- Correct 10 errors in a 1000 bit data unit = $2.63 \cdot 10^{23}$ different places.
  (or 263.409.560.461.970.212.832.400 different places)

To find the number of different places by ***r*** errors in ***n*** bits:

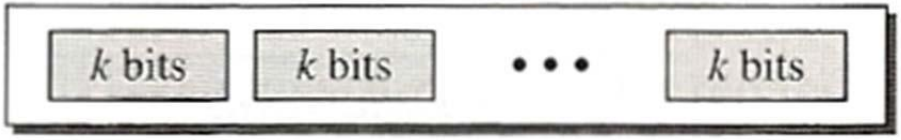
$$\binom{n}{r} = \frac{n!}{r!\,(n-r)!}$$

Data Link layer

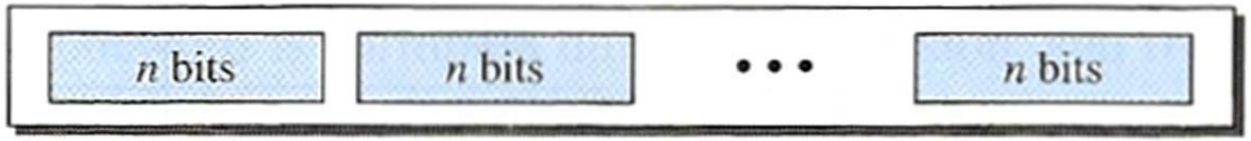
Block coding

Basic idea

- A message is divided into blocks of ***k*** bits, these are called ***Datawords***.
- We add ***r*** redundant bits to each block and then they are called ***Codewords***.
- Codewords have the size of ***n=k+r*** bits
- 2^k combinations of Datawords
- 2^n combinations of Codewords.
- Since ***n*** > ***k*** and a Dataword can be always mapped into a Codeword, we get: $2^n - 2^k$ invalid or illegal Codewords, used to detect errors.



2^k Datawords, each of k bits



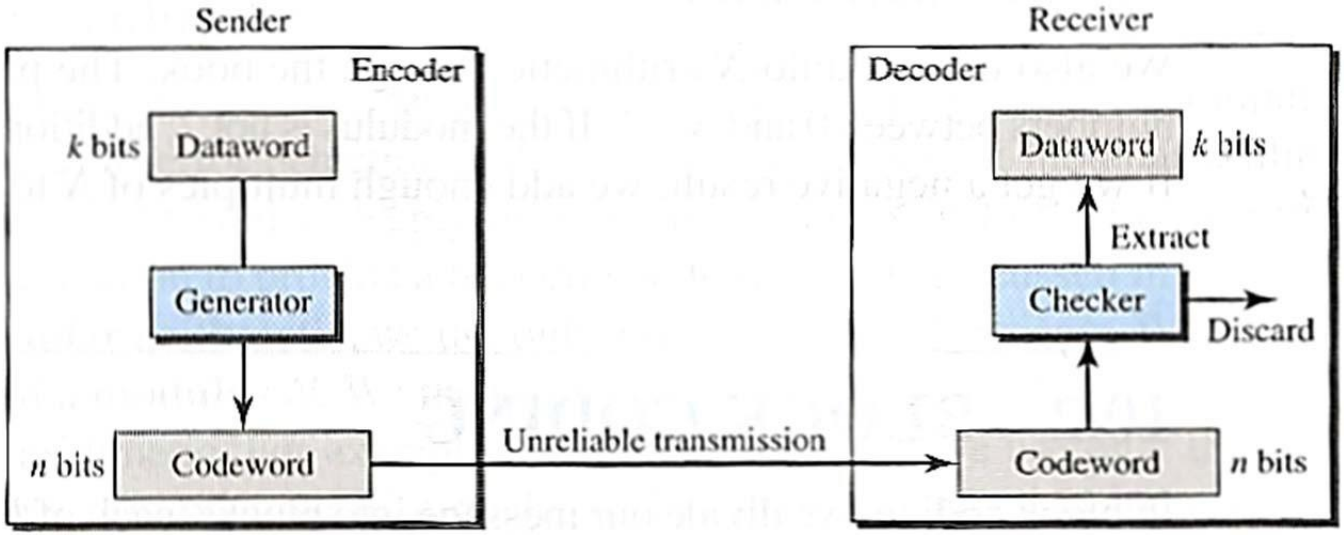
2^n Codewords, each of n bits (only 2^k of them are valid)

Data Link layer

Error detection

If the following two conditions are met, then the receiver can detect a change in the original Codeword:

1. The receiver has (or can find) a list of valid Codewords.
2. The original Codeword has changed to an invalid Codeword.



- The sender generates Codewords from Datawords using rules and procedures applied by generators
- The receiver checks if Codewords are in the list: accepted/rejected

Data Link layer

Example (10.1): $k = 2$ and $n = 3$.

Datawords	Codewords
00	000
01	011
10	101
11	110

We imagine that the sender encodes Dataword **01** to Codeword **011** and sends it to the receiver:

1. The receiver receives **011**. It is a valid Codeword and the receiver extract Dataword **01**.
2. Codeword has changed during transmission and is received as **111** (the leftmost bit is corrupted). This Codeword is not valid and is rejected.
3. Codeword has changed during transmission and is received as **000** (2 bits have been changed). It is a valid password and the receiver extracts Dataword **00**. this error is undetectable.

Conclusion: If the transmission changes the Codeword so that it corresponds to another valid Codeword, then the error will not be detected.

Data Link layer

Hamming distance

One of the key concepts in error detection is the idea of the Hamming distance.

Hamming distance between two datawords of the same size is the number of different bits when the datawords are compared bit by bit.

We indicate the Hamming distance between ***x*** and ***y*** by using ***d(x,y)***.
One can easily find the Hamming distance between two datawords by XOR operation on the two datawords with each other and then count the number of 1s.

E.g.,
D(000,011) = 2; because 000 XOR 011 = 011 (two 1s)
D(10101,11110) = 3; because 10101 XOR 11110 = 01011 (three 1s)

Data Link layer

Minimum Hamming distance

The minimum Hamming distance is used to design codes. In a set of codewords, the minimum Hamming distance will be the smallest Hamming distance between all possible pairs of codewords.

We use the notation d_{min} to specify the Minimum Hamming distance.

E.g.,

Minimum Hamming (e.g., table 10.1)

d(000,011) = 2 (entry 1 and 2)
d(000,101) = 2 (entry 1 and 3)
d(000,110) = 2 (entry 1 and 4)
d(011,101) = 2 (entry 2 and 3)
d(011,110) = 2 (entry 2 and 4)
d(101,110) = 2 (entry 3 and 4)

$d_{min} = 2$

Datawords (***k***)	Codewords (***n***)
00	000
01	011
10	101
11	110

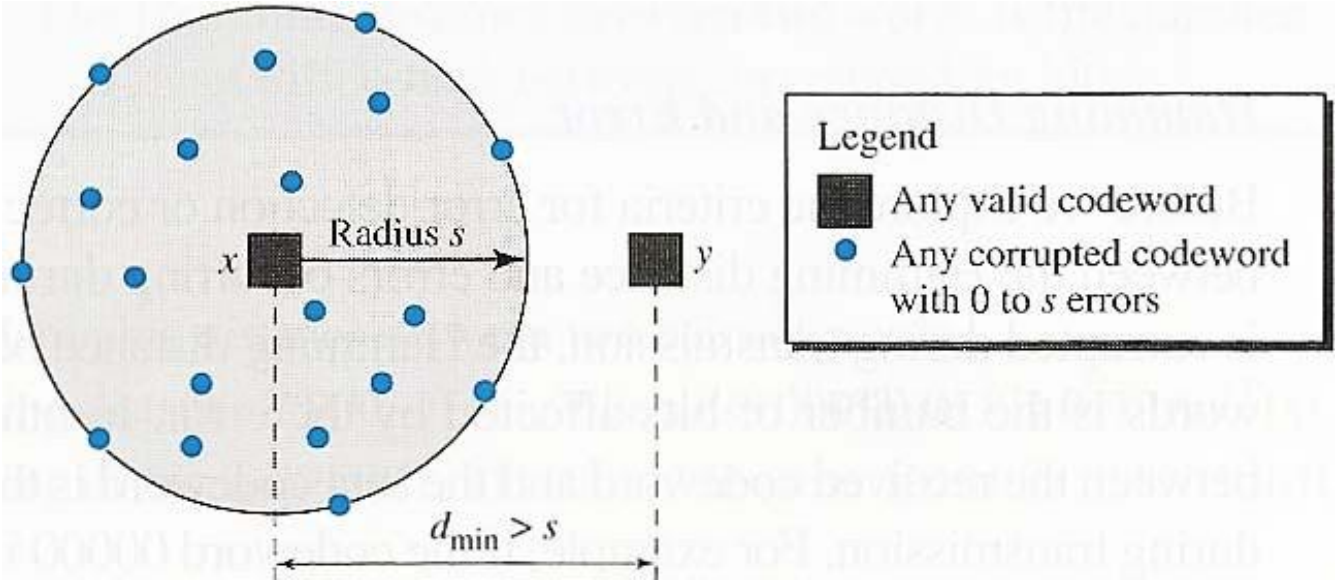
Code notation ***C(n,k)***
so ***C(3,2)*** with $d_{min} = 2$

Data Link layer

Minimum Hamming distance in case of error detection

What should the minimum Hamming distance be in a code scheme if we want to be able to detect **s** errors?

If **s** error bits are sent, then the Hamming distance is also **s**. So if our code scheme is to be able to detect up to **s** bit errors, then the minimum Hamming distance between valid codewords must be at least **s + 1**.
If the minimum Hamming distance between all pairs of codewords is **s + 1**, then a received password cannot be misinterpreted as another valid password.



Data Link layer

Minimum Hamming distance for linear block codes

Linear block coding is a code in which the XOR operation (addition modulo-2) of two valid codewords creates another valid codeword.

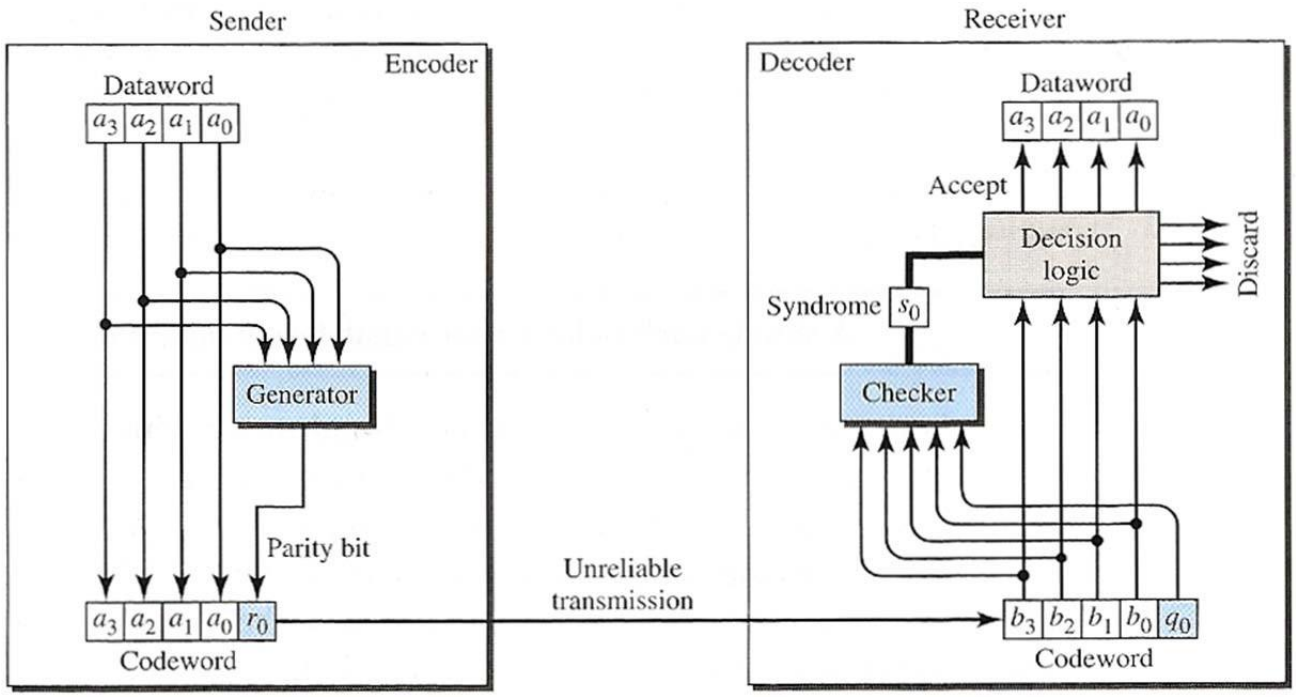
- If we look at Table 10.1, then we will see that it meets this criterion.
- It is simple to find the minimum Hamming distance for a linear block code. The minimum Hamming distance is the number of 1s in the nonzero valid codeword with the smallest number of 1s.

Data Link layer

Parity-Check Code

Parity-Check Code is the most common error-detecting code in the category of linear block codes. A k-bit dataword is changed to an n-bit codeword where n = k + 1. The extra bit, called the parity bit, is selected to make the total number of 1s in the codeword even.

- The minimum Hamming distance is 2, used for single-bit error detection.



The parity bit calculated as: $r_0 = (a_3 + a_2 + a_1 + a_0)$ modulo-2

The receiver decodes all receiving bits and finds the **syndrome**: $S_0 = (b_3 + b_2 + b_1 + b_0 + q_0)$ modulo-2

S_0=0 → Accept and S_0=1 →Discard (A simple parity check can only detect an odd number of errors)

Data Link layer

Cyclic codes

Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.

Cyclic Redundant Check (CRC)

This method is used in both LAN and WAN networks. An CRC example is given below with C(7,4):

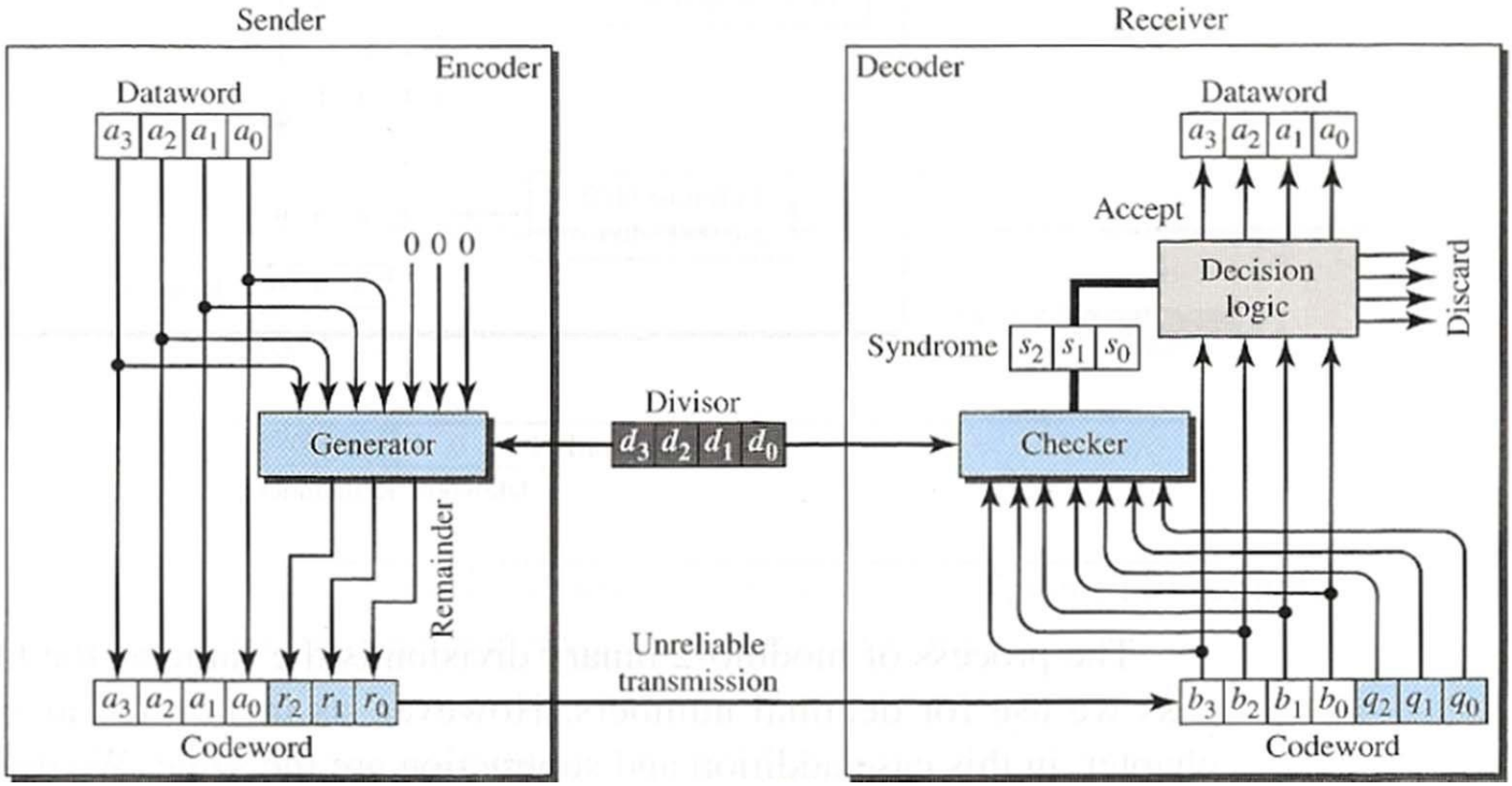
	Datawords	Codewords	Datawords	Codewords
	0000	0000000	1000	1000101
A	0001	0001011	1001	1001110
B	0010	0010110	1010	1010011
C	0011	0011101	1011	1011000
	0100	0100111	1100	1100010
D	0101	0101100	1101	1101001
	0110	0110001	1110	1110100
E	0111	0111010	1111	1111111

We can see that both linear and cyclic properties are present in this coding scheme.

A XOR B = C; A << 1 = B; A << 2 = D; C << 1 = E

Data Link layer

Cyclic codes



Note: The divisor is predefined and agreed in advance, it is used by both sender and receiver.

Data Link layer

Cyclic codes: Sender side

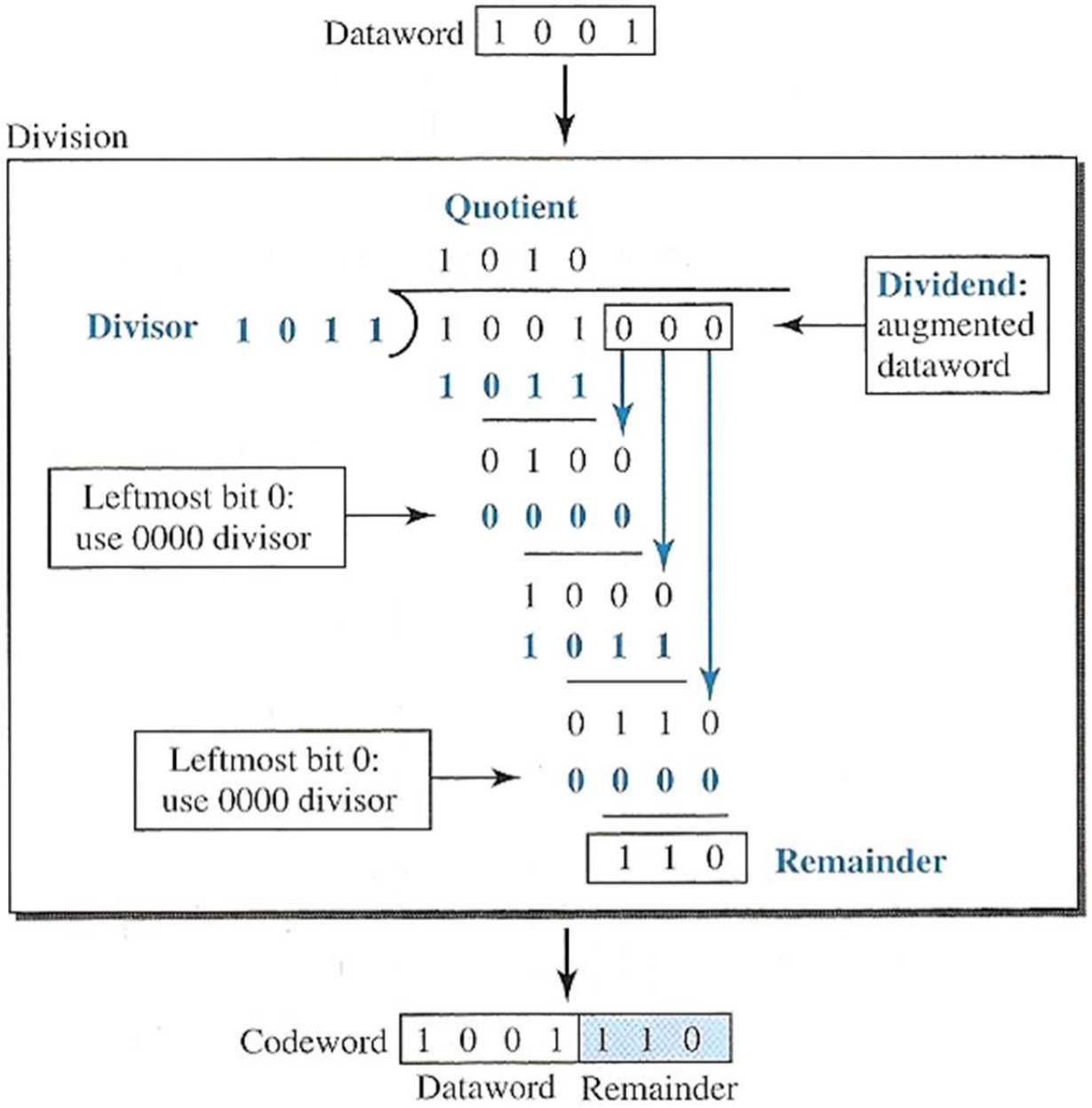
If the leftmost bit in the dividend is 1, then 1011 is used.

If the leftmost bit in the dividend is 0, then 0000 is used.

In both cases, the subtraction of the two 4-bit dividend and divisor implemented by XOR and a 0 is moved down.

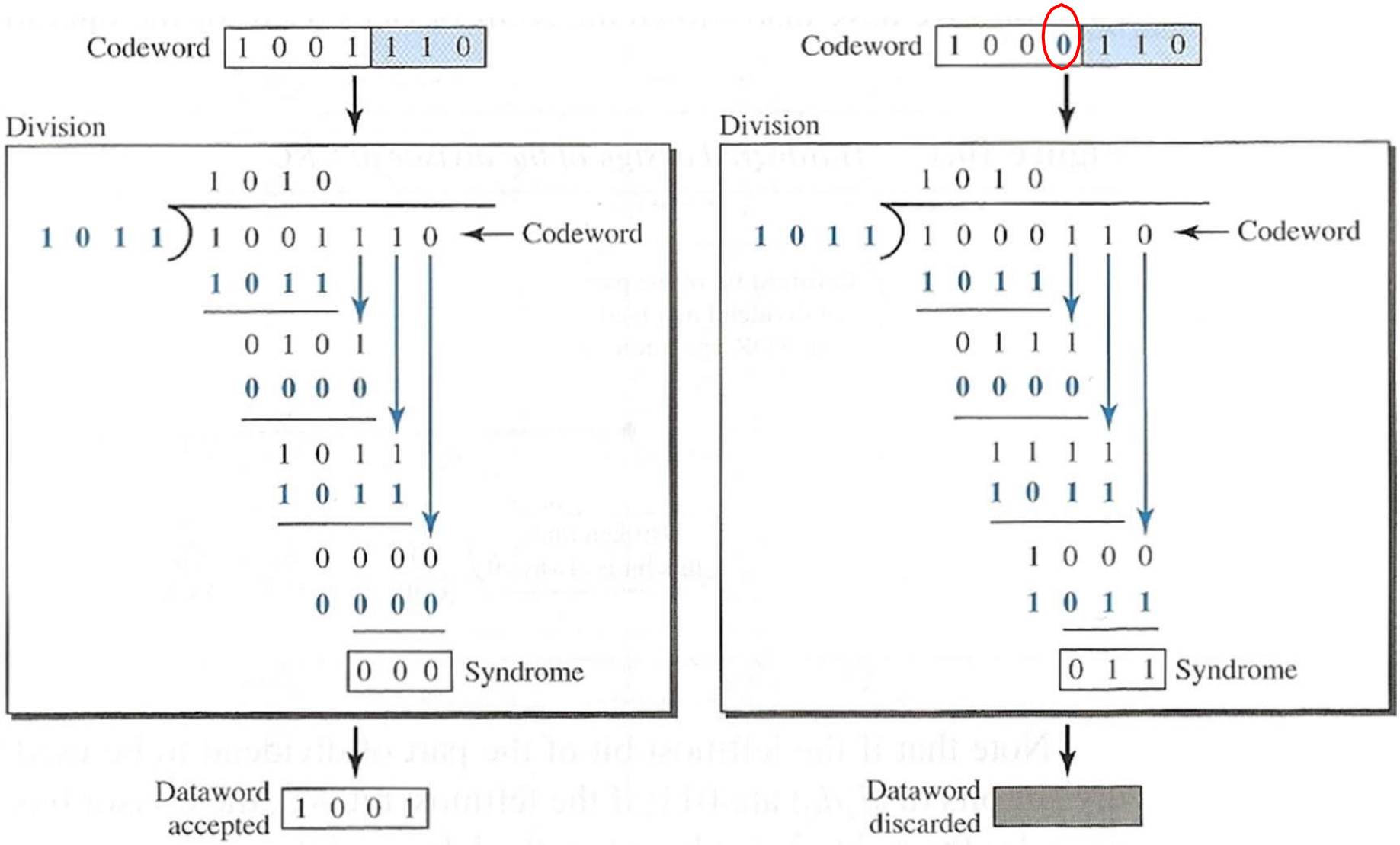
The last 3 bits (the remainder) are put in the back of the original dataword.

The quotient is just thrown away!



Data Link layer

Cyclic codes: Receiver side

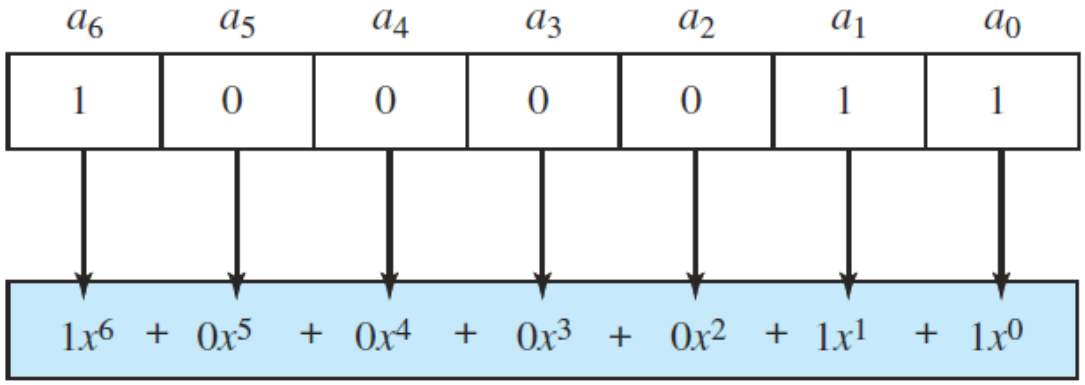


Data Link layer

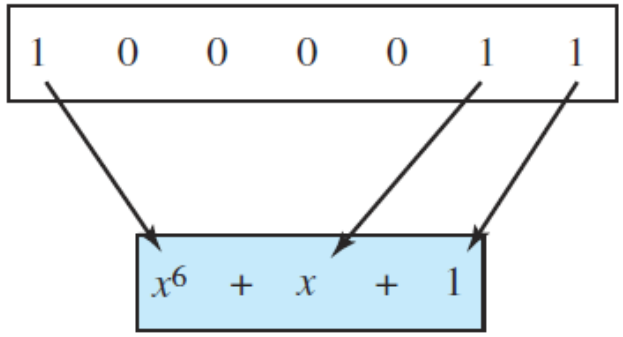
Polynomials

A better way to understand cyclic codes and how they can be analyzed is to represent them as **polynomials**

A pattern of 0s and 1s can be represented as a polynomial with coefficients of 0 and 1. The power of each term shows the position of the bit; the coefficient shows the value of the bit.



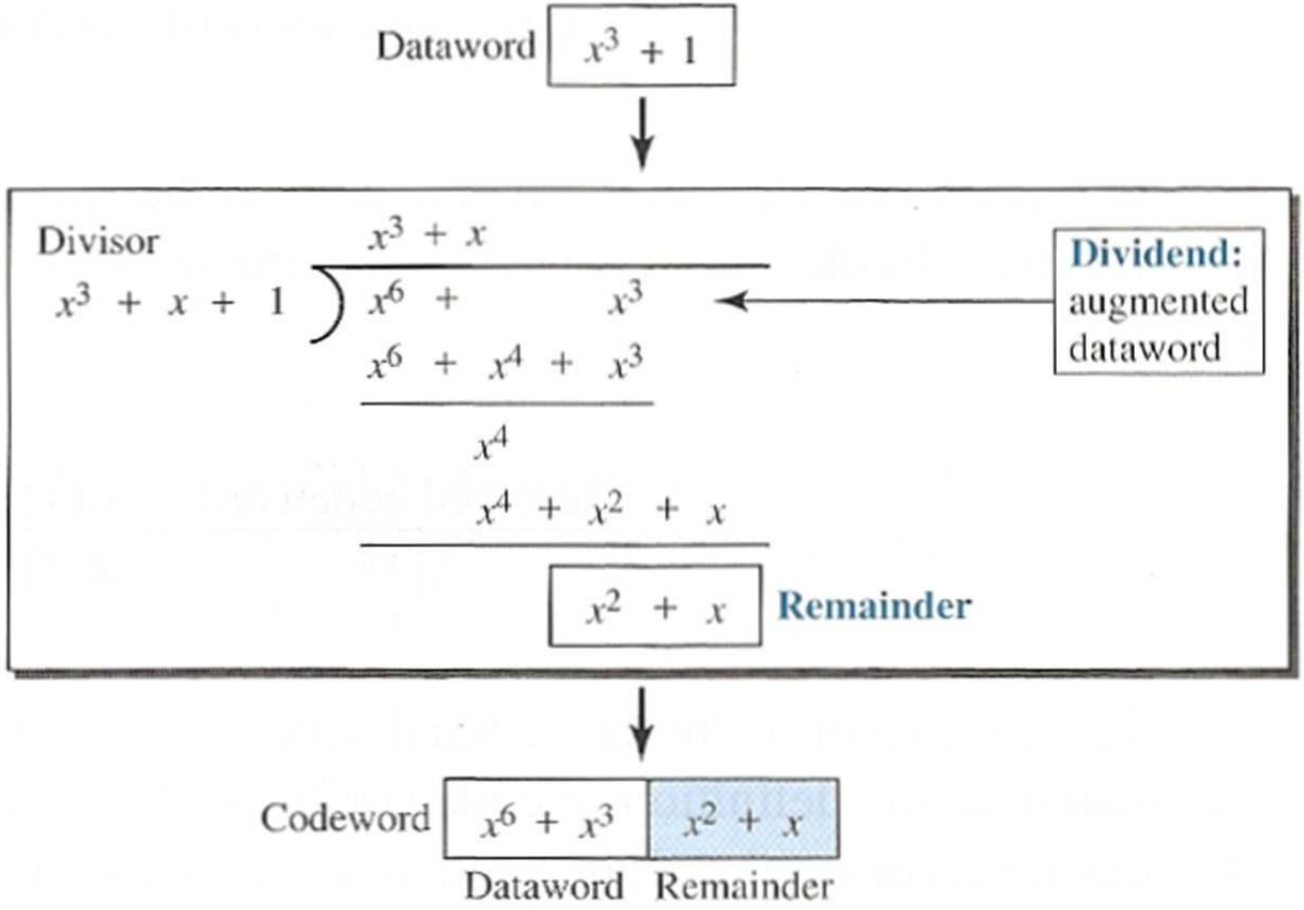
a. Binary pattern and polynomial



b. Short form

Data Link layer

Polynomials



The dividend is shifted 3 bits to the left (i.e., 000 was put at the end of the dataword in the binary version) What the divisor must be multiplied by is the quotient polynomial, which get the same degree as the dividend polynomial at each step, e.g., $(x^3+x+1)\cdot x^3 = x^6+x^4+x^3$ and $(x^3+x+1)\cdot x = x^4+x^2+x$

Data Link layer

Cyclic code analysis

We can now analyze a cyclic code and explore its possibilities using f(x) type polynomials with binary coefficients. We define the following polynomials:

Dataword: $d(x)$
Codeword: $c(x)$
Generator: $g(x)$
Syndrome : $s(x)$
Error: $e(x)$

For cyclic code:

1. If $s(x) \neq 0$, then one or more bits are faulty.
2. If $s(x) = 0$, then:
    a) No bit is corrupted.
    b) or some bits are corrupted but the decoder failed to detect them.

Data Link layer

Cyclic code analysis

The codeword received can be described as:

$$received\ codeword = c(x) + e(x)$$

The receiver divides the received password with the generator polynomial, this can be written as:

$$\frac{received\ codeword}{g(x)} = \frac{c(x)}{g(x)} + \frac{e(x)}{g(x)}$$

Since the division **c(x)/g(x)** by definition has a remainder of zero, we can say that the syndrome can be written as:

$$s(x) = \frac{e(x)}{g(x)}$$

Conclusion:

In a cyclic code, those e(x) errors that are divisible by g(x) are not caught.

Data Link layer

Cyclic code analysis

Single-bit error

- A single bit error can be written as follows: $e(x)=x^i$, where **i** is the position of the erroneous bit.

- If a single bit error is caught, then x^i is not divisible by $g(x)$ (so there is a remainder which means that $s(x)\neq 0$)

- if $g(x)$ has at least two terms (which is normal) and $x^0 \neq 0$, then $e(x)$ (which is x^i) cannot be divided by $g(x)$ (which means that $s(x)\neq 0$)

Conclusion:

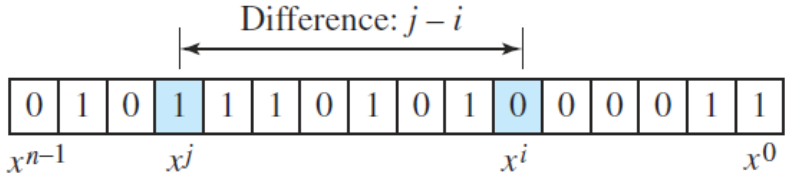
If the generator polynomial g(x) has more than one term and g(x)'s smallest term $x^0=1$, then all single bit errors are detected.

Data Link layer

Cyclic code analysis

Two isolated Single-bit errors

This type of error can be written as:



j and **i**'s values describe the positions of the errors and the difference **j - i** defines the distance between the two errors. We can rewrite x^j+x^i as follows:

if ***g(x)*** has more than one term and ***g(x)'s*** smallest term $x^0=1$, then it cannot divide x^i (as we have just seen).
so if ***g(x)*** is to be able to divide ***e(x)***, it must be able to divide $x^{j-i}+1$.
Therefore we must demand that ***g(x)*** cannot divide x^t+1 where ***t*** is between 2 and n-1

- 2 corresponds to the minimum distance between two errors (= two isolated single-bit errors)
- n-1 corresponds to maximum distance between two errors (first and last bit of the transmitted data)

Conclusion:
If the generator polynomial g(x) cannot divide x^t+1 (where t is between 2 and n - 1), then all isolated double errors are detected.

Data Link layer

Cyclic code analysis

An odd number of errors

A generator polynomial ***g(x)***, containing a factor of ***x+1***, can detect all odd-numbered errors.

This means that we need to make ***x+1*** a factor of any generator

it is not the same as saying the generator should be ***x+1,*** actually if it is ony ***x+1***, it cannot catch the two adjacent isolated errors.

Data Link layer

Cyclic code analysis

Burst error

Burst errors are the most important to include in this analysis as they are the most common errors. A Burst error has the form:

$$e(x) = (x^j + \cdots + x^i)$$

We can isolate x^i and write:

$$e(x) = x^i \cdot (x^{j-i} + \cdots + 1)$$

If $g(x)$ can detect single-bit errors, then we do not have to worry about x^i. What we need to ensure is that:

$$\frac{(x^{j-i} + \cdots + 1)}{(x^r + \cdots + 1)} \neq 0$$

(Generator polynomial g(x) is what is shown as denominator)

Data Link layer

Cyclic code analysis

Burst error

We can have the following outcomes:

1. If $j - i < r$, then the division remainder can never be 0. we can write $j - i = L - 1$, where L is the length of the burst error. This means that $L - 1 < r$, or $L < r + 1$, or $L \leq r$. Which means that all burst errors with a length less than or equal to the degree of the generator polynomial will be detected.

2. In rare cases $j - i = r$, or $L = r + 1$. Here the syndrome $s(x)$ can become 0 and the error will not be detected. It can be shown that the probability of an undetected burst error of length L is: $(½)^{r-1}$.
   If our $g(x)$ is $x^{14}+x^3+1$, then r=14, and if the burst error is L=15, then the probability of errors not being detected is $(½)^{14-1}$, which corresponds to 1 out of approximate 10,000.

3. In rare cases, $j - i > r$, or $L > r + 1$. Here the syndrome $s(x)$ can become 0 and the error will not be detected. It can be shown that the probability of an undetected burst error of length L is: $(½)^{r}$.
   If our $g(x)$ is $x^{14}+x^3+1$, then r=14, and if the burst error is of L≥15, then the probability of not being detected is $(½)^{14}$, which corresponds to 1 out of approximate 16,000.

Data Link layer

Cyclic code analysis

Burst error

- **All burst errors of length $L \leq r$ will be detected.**
- **All burst errors with length $L = r + 1$ will be detected with a probability of $1 - (½)^{r-1}$.**
- **All burst errors with length $L > r + 1$ will be detected with a probability of $1 - (½)^{r}$.**

Data Link layer

Cyclic code analysis

A good generator polynomial should have the following characteristics:

1. **The pattern have at least two terms.**
2. **The coefficient of the term x^0 should be 1.**
3. **It should not divide $x^t + 1$, for t between 2 and n-1.**
4. **It should have the factor x + 1.**

Name	Polynomial
CRC-8	$x^8 + x^2 + x + 1$ (100000111)
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$ (11000110101)
CRC-16	$x^{16} + x^{12} + x^5 + 1$ (10001000000100001)
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (100000100110000010001110110110111)