Computer Networks

Day- 4

Recap

- Day-1 : Concept of Layering
- Day-2 : LAN Technologies (Ethernet)
- Day-3: Basic Wifi

Data Link Control

Error Detecting & Correction

Note

Data can be corrupted
during transmission.

Some applications require that
errors be detected and corrected.

Error Detecting & Correction

- **Error:** A condition when the receiver's information does not match with the sender's information
- Error Detection
  - Simple Parity check
  - Two-dimensional Parity check
  - Checksum
  - Cyclic Redundancy Code
- Error Correction
  - Hamming code

Error: A condition when the receiver's information does not match with the sender's information.

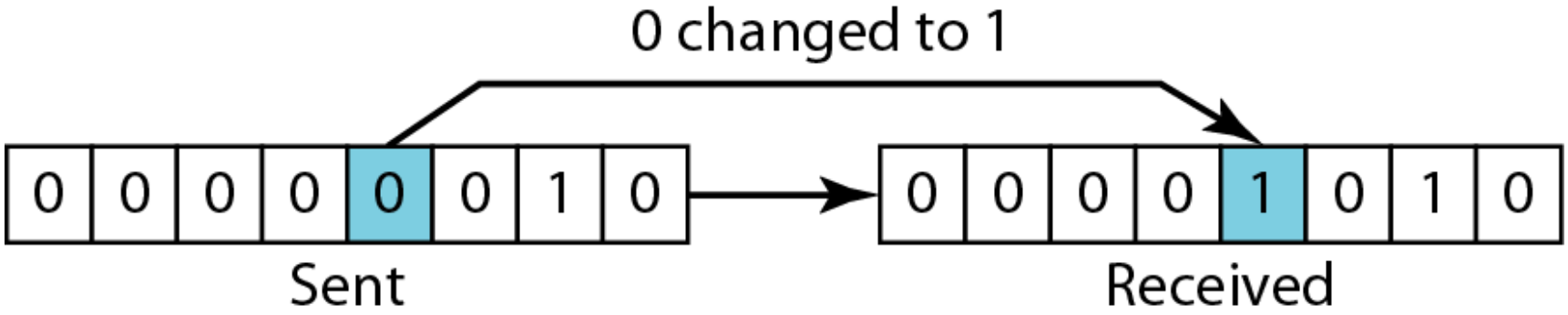
Types of Errors:
Redundancy
Detection Versus Correction
Forward Error Correction Versus Retransmission
Coding
Modular Arithmetic

Types of Errors

Note

In a single-bit error, only 1 bit in the data unit has changed.

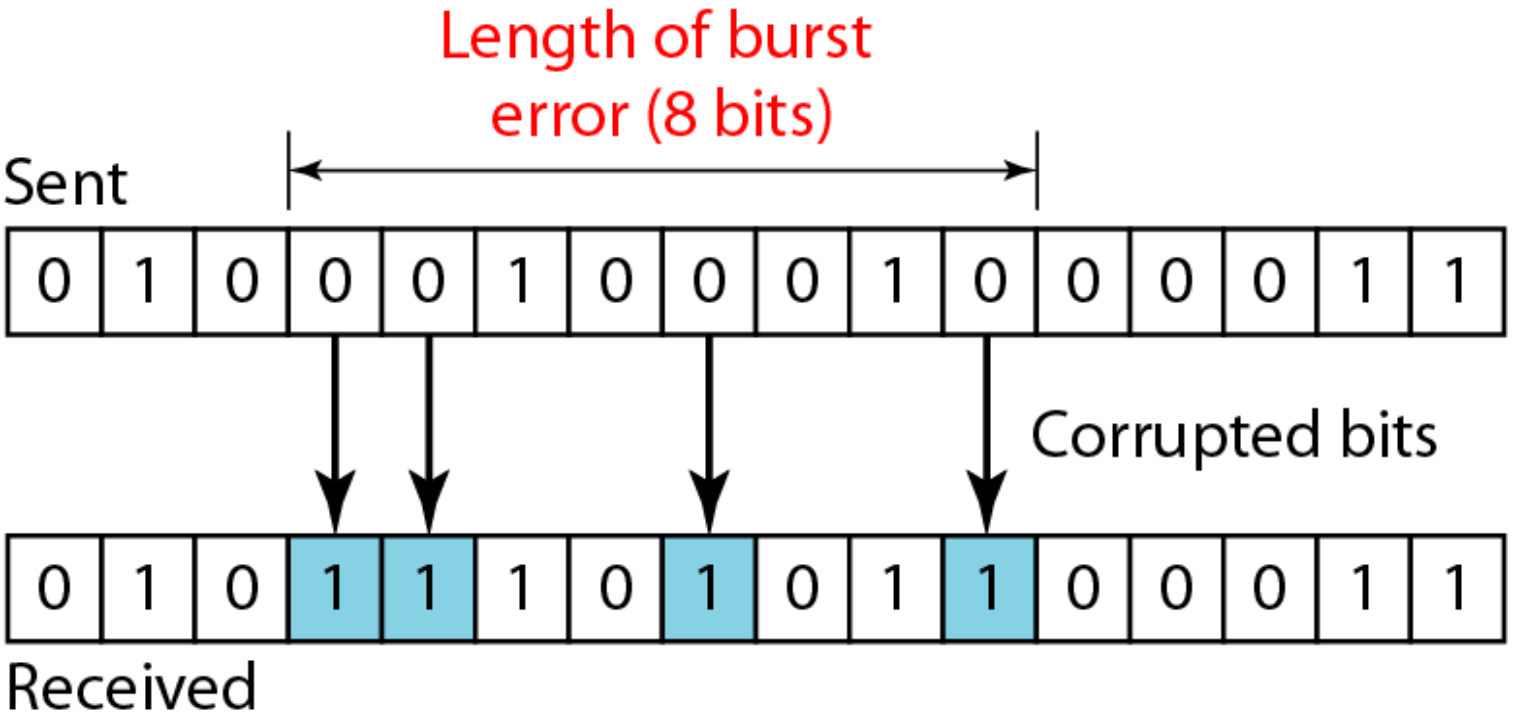
Single-bit error



Note

A burst error means that 2 or more bits in the data unit have changed.

Burst error of length 8



Redundancy

Note

To detect or correct errors, we need to send extra (redundant) bits with data.

Detection Vs Correction

- Detection: Checking whether error occurred
- Correction: Identifying the position of the errors : no. of errors, size

Forward Error Correction Vs Retransmission

- FEC: Receiver tries to guess the correct message using redundant bits
- Retransmission: detect the error and retransmit

Coding

Redundancy is achieved through various coding schemes
block codes and convolution codes

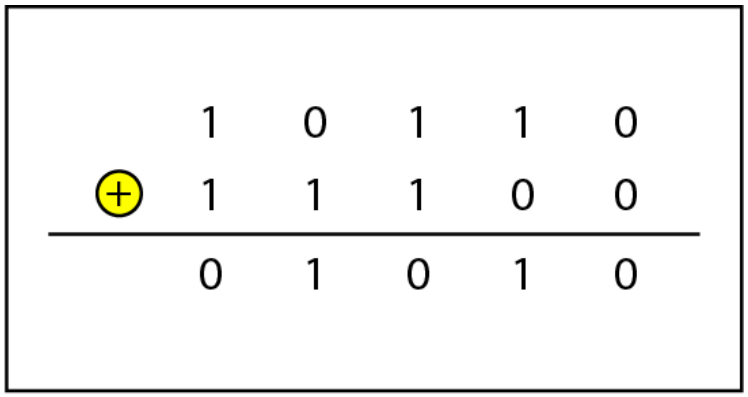
Note

In modulo-N arithmetic, we use only the integers in the range 0 to N −1, inclusive.

XORing of two single bits or two words

a. Two bits are the same, the result is 0.

b. Two bits are different, the result is 1.



c. Result of XORing two patterns

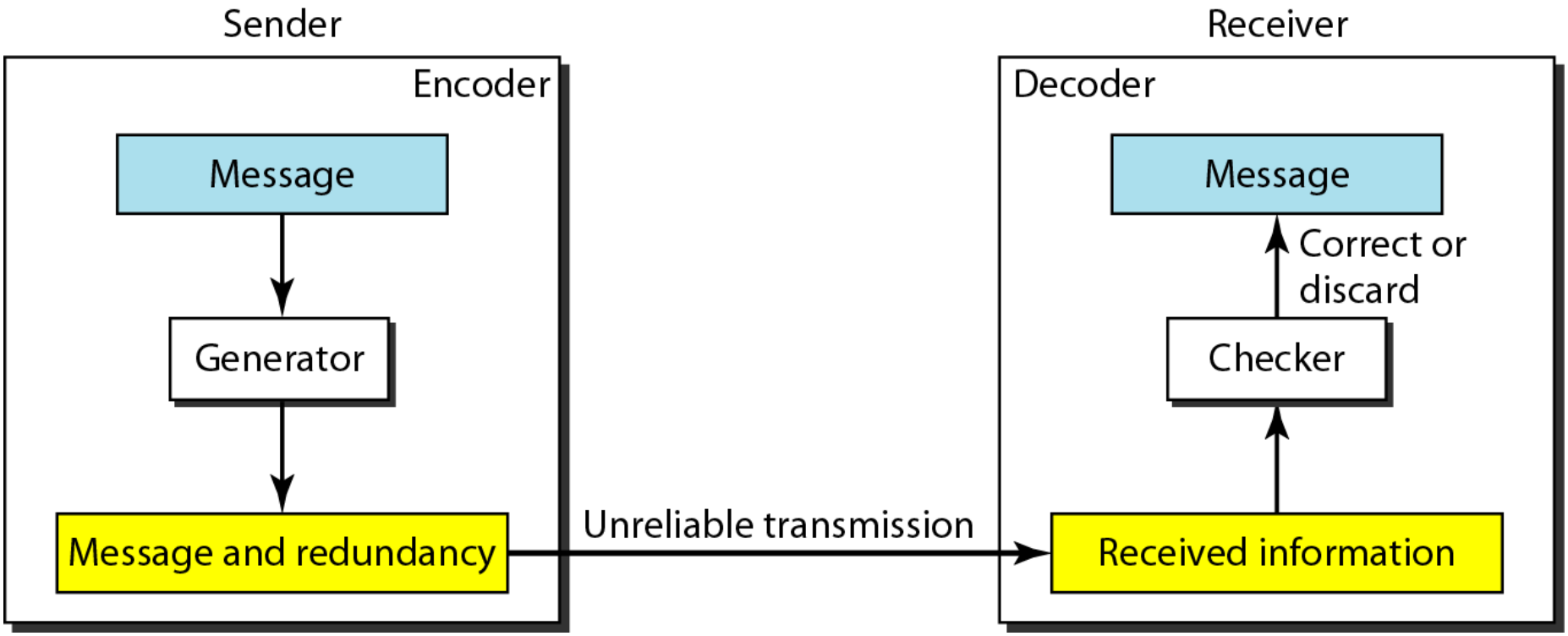
BLOCK CODING

In block coding, we divide our message into blocks, each of k bits, called datawords.

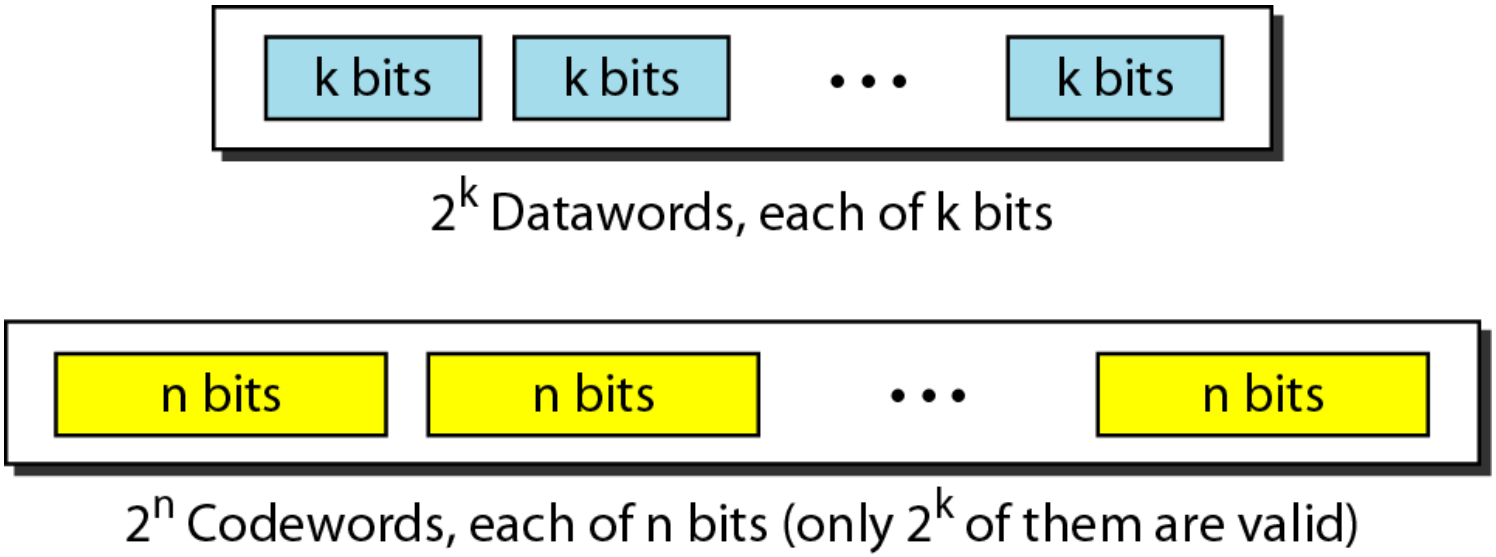
We add r redundant bits to each block to make the length n = k + r.

The resulting n-bit blocks are called codewords.

The structure of encoder and decoder



Datawords and codewords in block coding



4B/5B block coding:

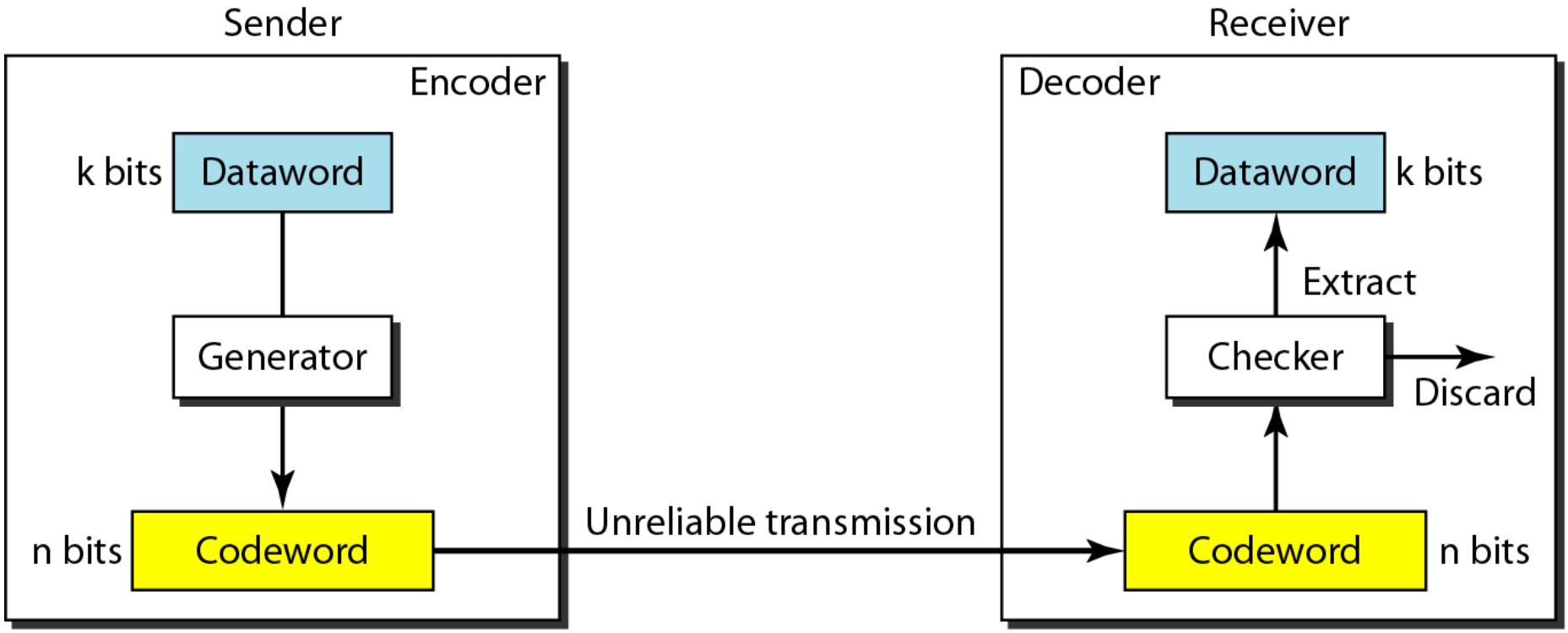
In this coding scheme,

k = 4 and n = 5.

2^k = 16 datawords and 2^n = 32 codewords.

16 out of 32 codewords are used for message transfer and the rest are either used for other purposes or unused.

Process of error detection in block coding



Let us assume that k = 2 and n = 3.
Table shows the list of datawords and codewords.

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

1. *The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.*

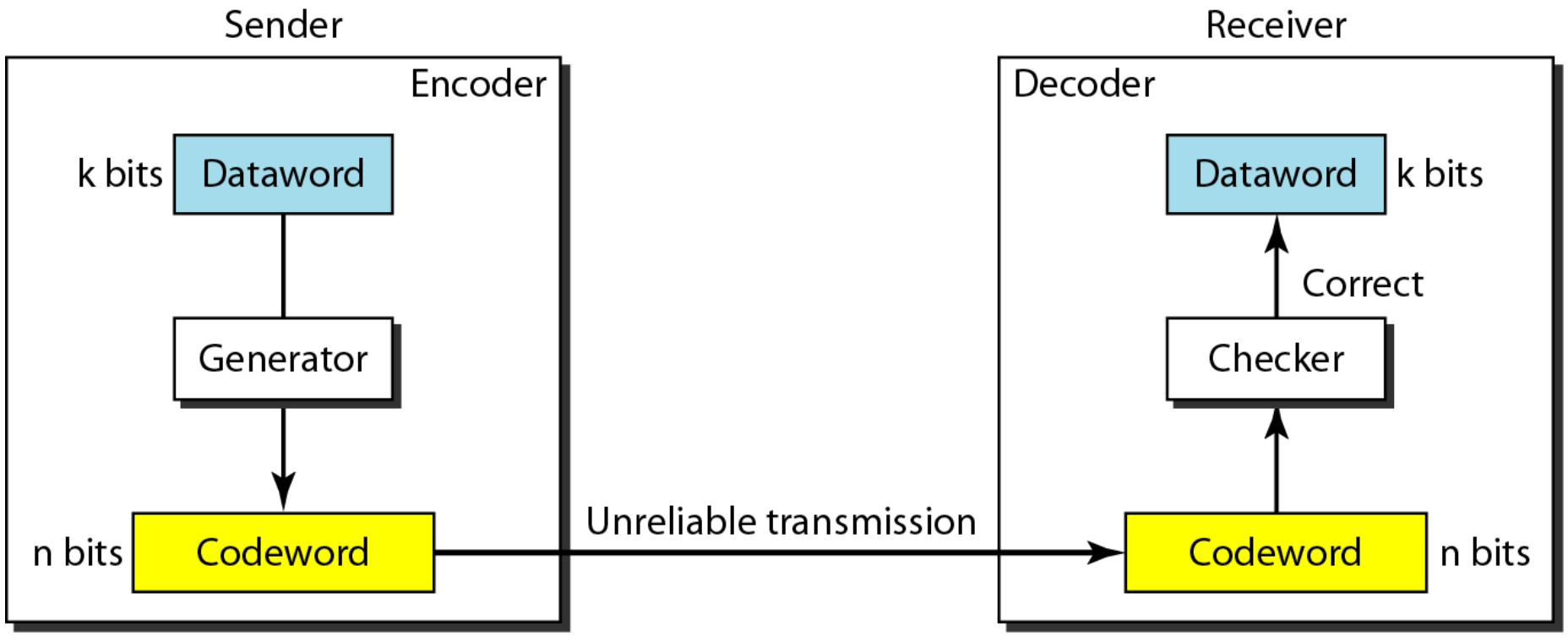
2. The codeword is corrupted during transmission, and 111 is received. This is not a valid codeword and is discarded.

3. The codeword is corrupted during transmission, and 000 is received. This is a valid codeword. The receiver incorrectly extracts the dataword 00. Two corrupted bits have made the error undetectable.

Note

An error-detecting code can detect only the types of errors for which it is designed; other types of errors may remain undetected.

Structure of encoder and decoder in error correction



Let us add more redundant bits to Previous Example to see if the receiver can correct an error without knowing what was actually sent. We add 3 redundant bits to the 2-bit dataword to make 5-bit codewords. Table shows the datawords and codewords. Assume the dataword is 01. The sender creates the codeword 01011.

Dataword	Codeword
00	00000
01	01011
10	10101
11	11110

The codeword is corrupted during transmission, and 01001 is received. First, the receiver finds that the received codeword is not in the table. This means an error has occurred. The receiver, assuming that there is only 1 bit corrupted, uses the following strategy to guess the correct dataword.

1. *Comparing the received codeword with the first codeword in the table (01001 versus 00000), the receiver decides that the first codeword is not the one that was sent because there are two different bits.*
2. *By the same reasoning, the original codeword cannot be the third or fourth one in the table.*
3. *The original codeword must be the second one in the table because this is the only one that differs from the received codeword by 1 bit. The receiver replaces 01001 with 01011 and consults the table to find the dataword 01.*

Note

The Hamming distance between two words is the number of differences between corresponding bits.

Let us find the Hamming distance between two pairs of words.

1. The Hamming distance d(000, 011) is 2 because

$000 \oplus 011$ is 011 (two 1s)

2. The Hamming distance d(10101, 11110) is 3 because

$10101 \oplus 11110$ is 01011 (three 1s)

Note

The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.

Find the minimum Hamming distance of the coding scheme

Datawords	*Codewords*
00	000
01	011
10	101
11	110

Solution

We first find all Hamming distances.

$d(000, 011) = 2$ $d(000, 101) = 2$ $d(000, 110) = 2$ $d(011, 101) = 2$
$d(011, 110) = 2$ $d(101, 110) = 2$

The d_{min} *in this case is 2.*

Find the minimum Hamming distance of the coding scheme C(5,2) with $d_{min} = 3$

Dataword	Codeword
00	00000
01	01011
10	10101
11	11110

Solution

We first find all the Hamming distances.

$d(00000, 01011) = 3$	$d(00000, 10101) = 3$	$d(00000, 11110) = 4$
$d(01011, 10101) = 4$	$d(01011, 11110) = 3$	$d(10101, 11110) = 3$

The d_{min} in this case is 3.

Note

To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s + 1$.

Example 10.7

The minimum Hamming distance for C(3,2)code scheme is 2.

Datawords	*Codewords*
00	000
01	011
10	101
11	110

This code guarantees detection of only a single error. For example, if the third codeword (101) is sent and one error occurs, the received codeword does not match any valid codeword. If two errors occur, however, the received codeword may match a valid codeword and the errors are not detected.

Our second block code scheme C(5,2) has $d_{min} = 3$.

Dataword	Codeword
00	00000
01	01011
10	10101
11	11110

This code can detect up to two errors. Again, we see that when any of the valid codewords is sent, two errors create a codeword which is not in the table of valid codewords. The receiver cannot be fooled.

However, some combinations of three errors change a valid codeword to another valid codeword. The receiver accepts the received codeword and the errors are undetected.

Note

To guarantee correction of up to *t* errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$.

A code scheme has a Hamming distance $d_{min} = 4$. What is the error detection and correction capability of this scheme?

Solution

This code guarantees the detection of up to three errors ($s = 3$), but it can correct up to one error. In other words, if this code is used for error correction, part of its capability is wasted. Error correction codes need to have an odd minimum distance (3, 5, 7, . . .).

LINEAR BLOCK CODES

Almost all block codes used today belong to a subset called linear block codes.

A linear block code is a code in which the exclusive OR (addition modulo-2) of two valid codewords creates another valid codeword.

Note

In a linear block code, the exclusive OR (XOR) of any two valid codewords creates another valid codeword.

Datawords	*Codewords*
00	000
01	011
10	101
11	110

The scheme in Table is a linear block code because the result of XORing any codeword with any other codeword is a valid codeword. For example, the XORing of the second and third codewords creates the fourth one.

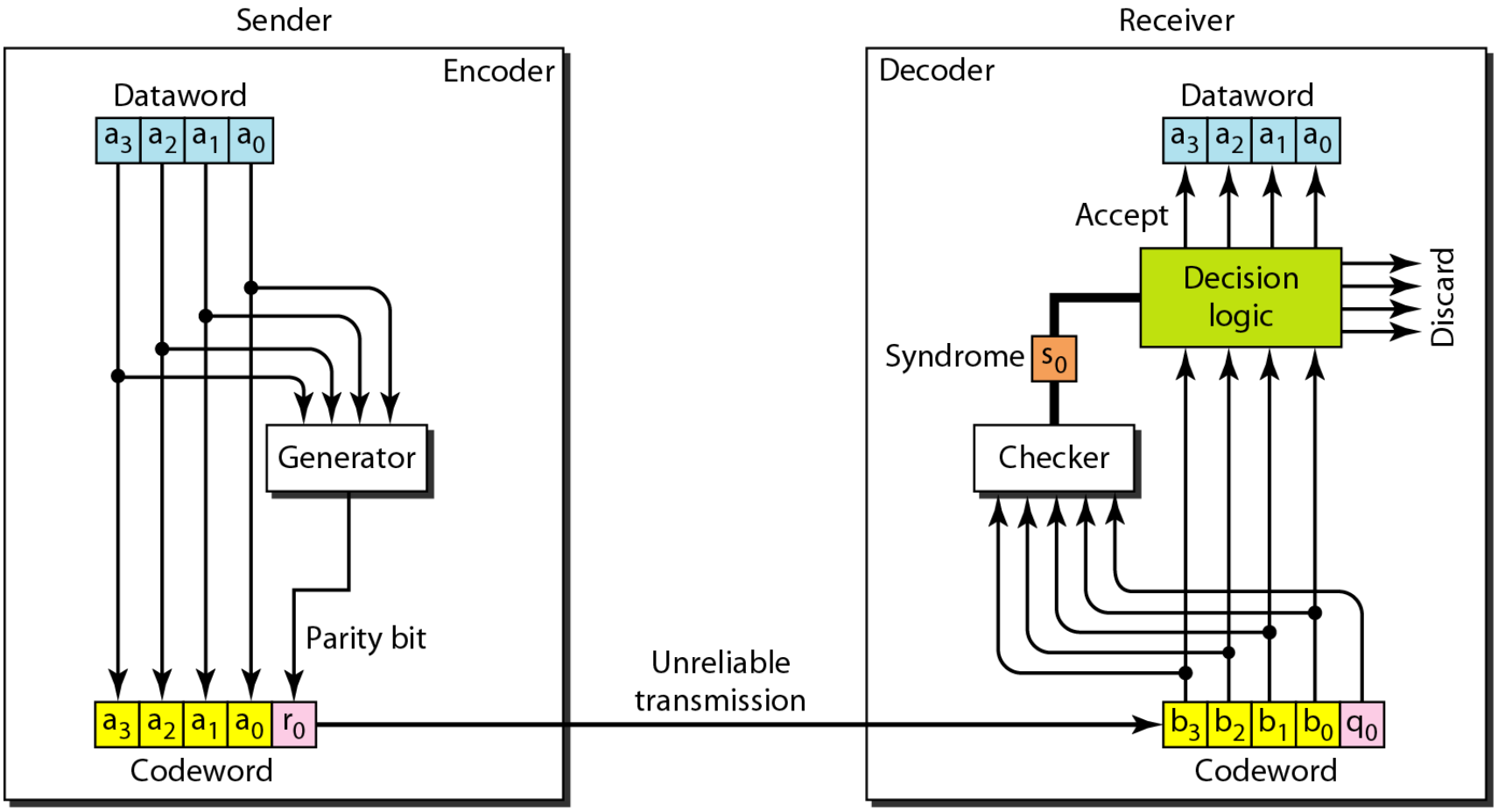
Note

A simple parity-check code is a single-bit error-detecting code in which $n = k + 1$ with $d_{min} = 2$.

Simple parity-check code C(5, 4)

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	00000	1000	10001
0001	00011	1001	10010
0010	00101	1010	10100
0011	00110	1011	10111
0100	01001	1100	11000
0101	01010	1101	11011
0110	01100	1110	11101
0111	01111	1111	11110

Encoder and decoder for simple parity-check code



Let us look at some transmission scenarios. Assume the sender sends the dataword 1011. The codeword created from this dataword is 10111, which is sent to the receiver. We examine five cases:

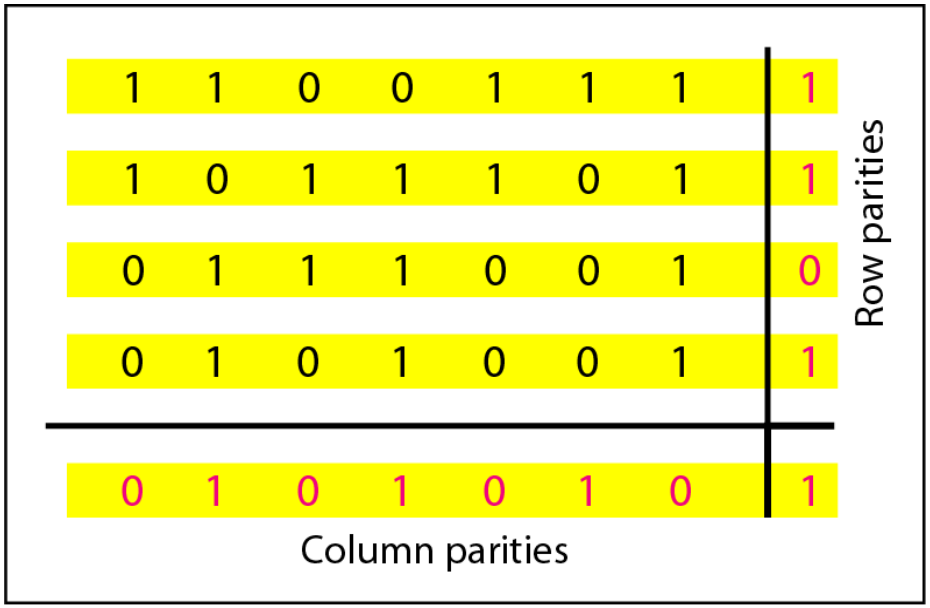
1. *No error occurs; the received codeword is 10111. The syndrome is 0. The dataword 1011 is created.*
2. *One single-bit error changes a_1. The received codeword is 10011. The syndrome is 1. No dataword is created.*
3. *One single-bit error changes r_0. The received codeword is 10110. The syndrome is 1. No dataword is created.*
4. *An error changes r_0 and a second error changes a_3. The received codeword is 00110. The syndrome is 0. The dataword 0011 is created at the receiver. Note that here the dataword is wrongly created due to the syndrome value.*
5. *Three bits—a_3, a_2, and a_1—are changed by errors. The received codeword is 01011. The syndrome is 1. The dataword is not created.*

This shows that the simple parity check, guaranteed to detect one single error, can also find any odd number of errors.

Note

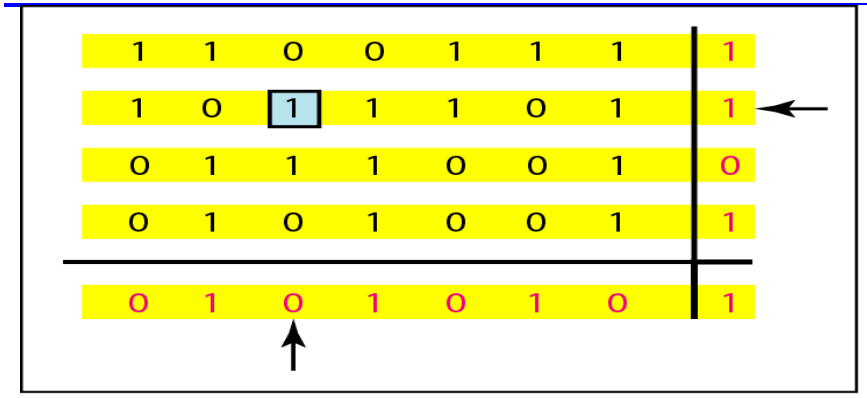
A simple parity-check code can detect an odd number of errors.

Two-dimensional parity-check code

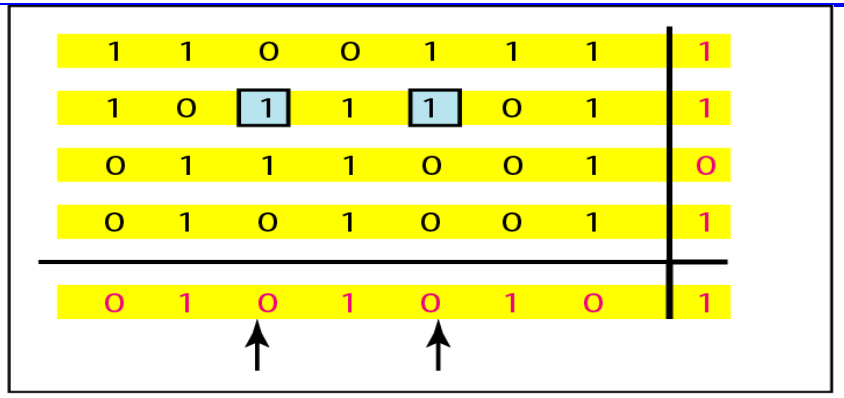


a. Design of row and column parities

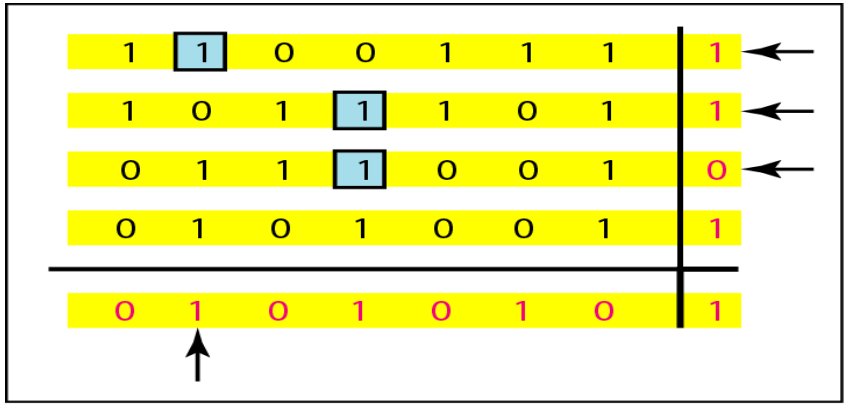
Two-dimensional parity-check code



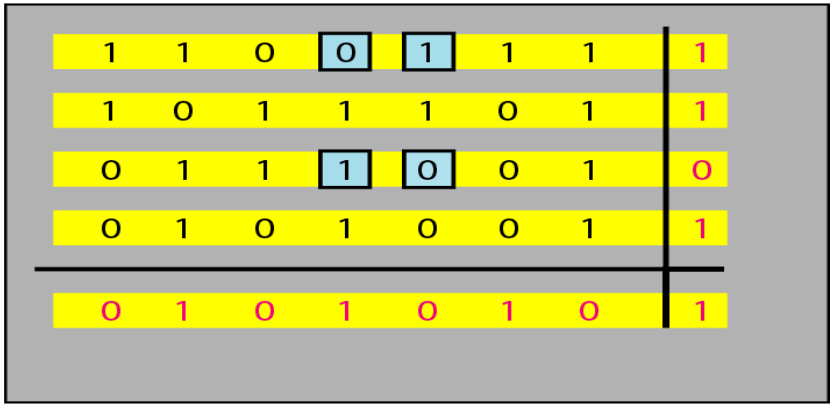
b. One error affects two parities



c. Two errors affect two parities



d. Three errors affect four parities



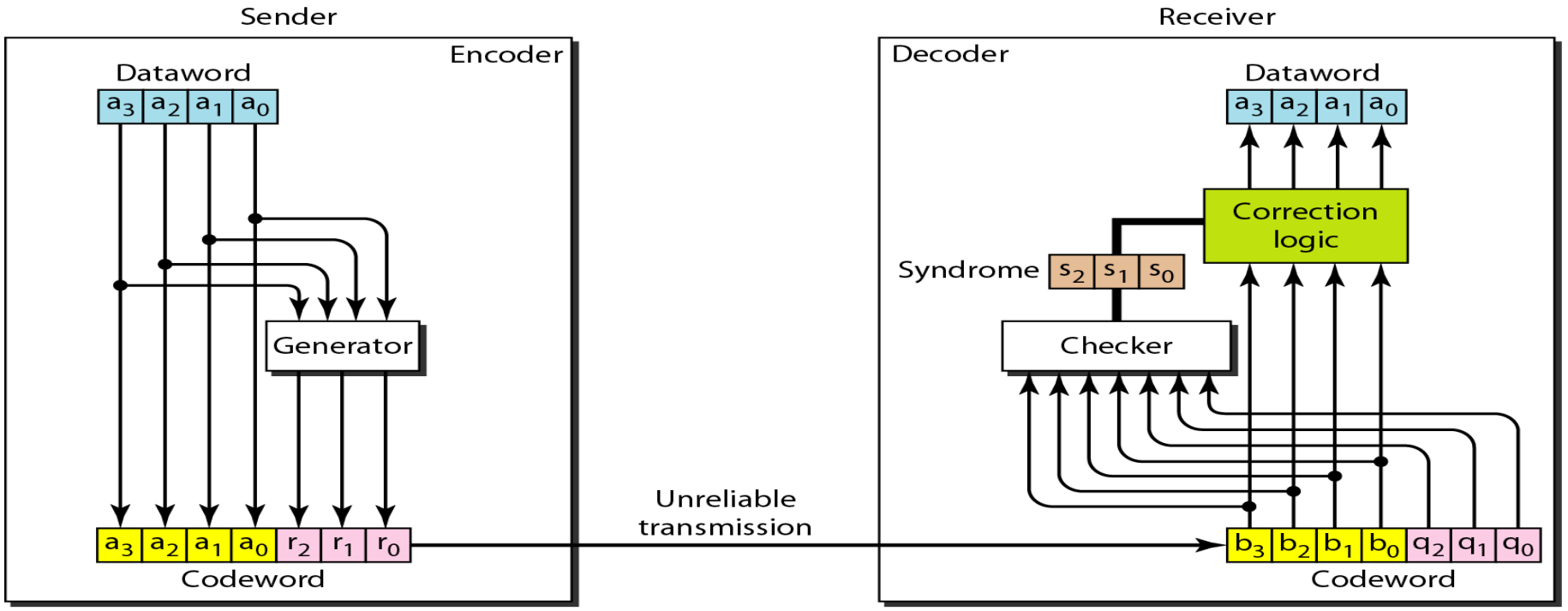
e. Four errors cannot be detected

Hamming Code

- The Hamming distance between two words is the number of differences between corresponding bits.
- To guarantee the detection of up to s errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = s+1$.
- To guarantee correction of up to *t* errors in all cases, the minimum Hamming distance in a block code must be $d_{min} = 2t + 1$.
- k – no. of data bits, r – no. of redundant bits, n – No. of code bits
- n = k + r
- $2^r \geq k + r + 1$ where, r = redundant bit, k = data bit
- Eg: k = 4; then r = 3

Hamming code C(7, 4)

Datawords	*Codewords*	*Datawords*	*Codewords*
0000	0000000	1000	1000110
0001	0001101	1001	1001011
0010	0010111	1010	1010001
0011	0011010	1011	1011100
0100	0100011	1100	1100101
0101	0101110	1101	1101000
0110	0110100	1110	1110010
0111	0111001	1111	1111111



r0 = a2+a1+a0 mod 2
r1=a3+a2+a1 mod 2
r2=a1+a0+a3 mode 2

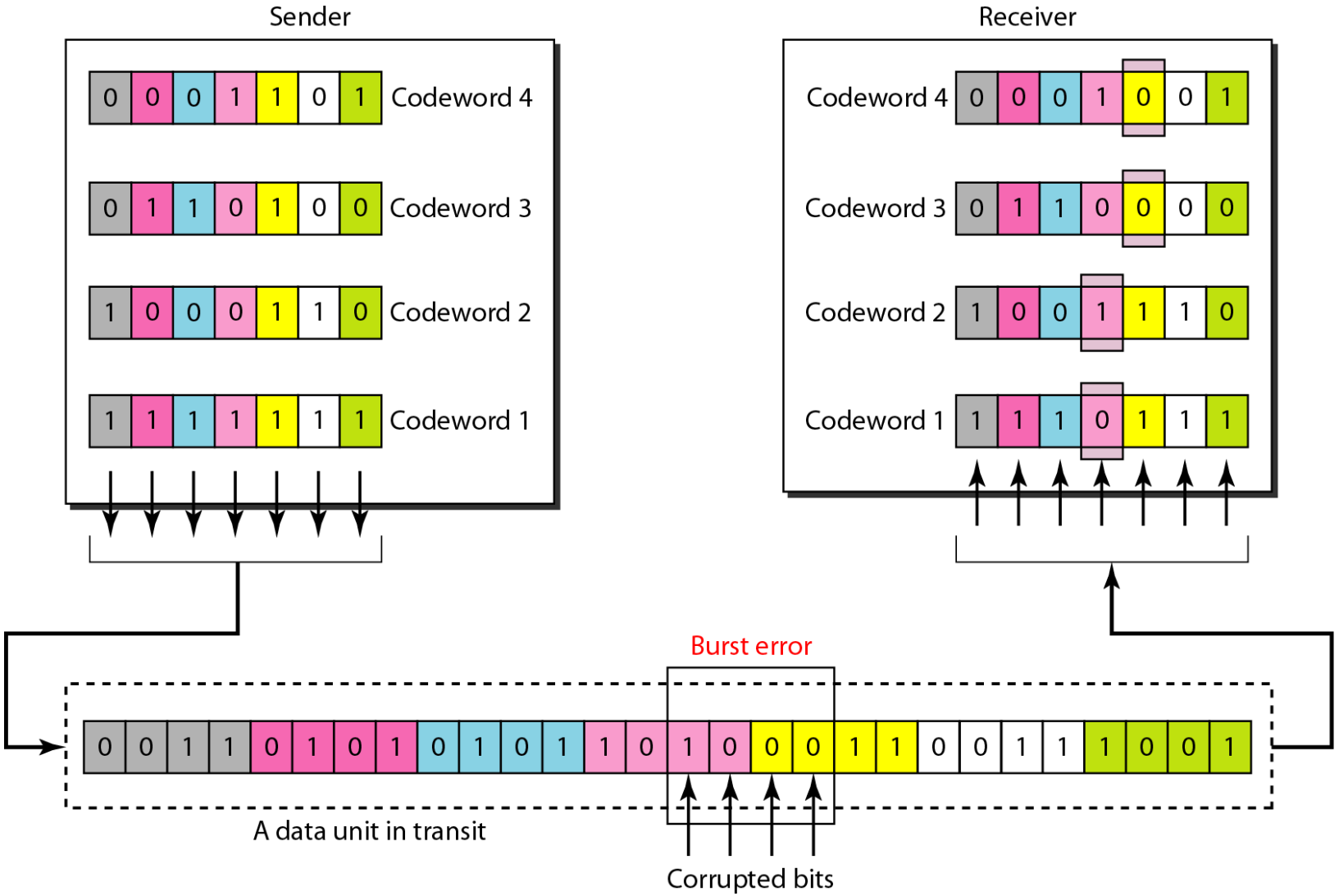
s0 = b2+b1+b0+q0 mod 2
s1 =b3+b2+11+q1 mod 2
s2 =b1+b0+b3+q2 mod 2

Syndrome	000	001	010	011	100	101	110	111
Error	None	q_0	q_1	b_2	q_2	b_0	b_3	b_1

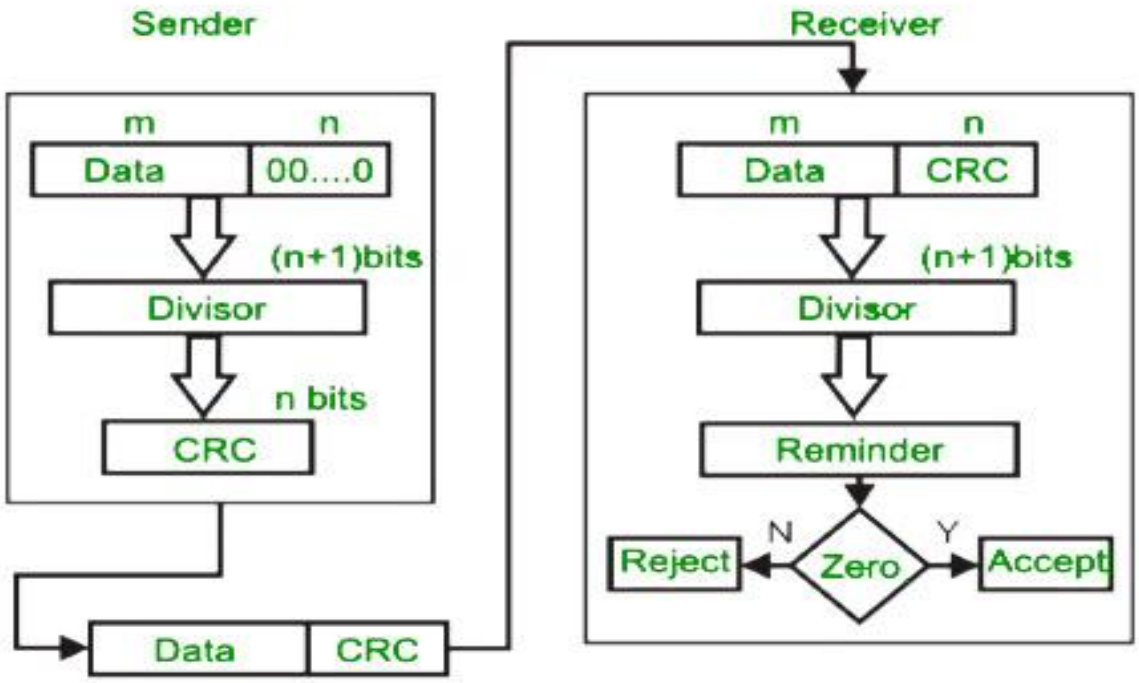
Let us trace the path of three datawords from the sender to the destination:

1. *The dataword 0100 becomes the codeword 0100011. The codeword 0100011 is received. The syndrome is 000, the final dataword is 0100.*
2. *The dataword 0111 becomes the codeword 0111001. The codeword 0011001 is received. The syndrome is 011. After flipping b_2 (changing the 1 to 0), the final dataword is 0111.*
3. *The dataword 1101 becomes the codeword 1101000. The syndrome is 101. After flipping b_0, we get 0000, the wrong dataword. This shows that our code cannot correct two errors.*

Burst error correction using Hamming code



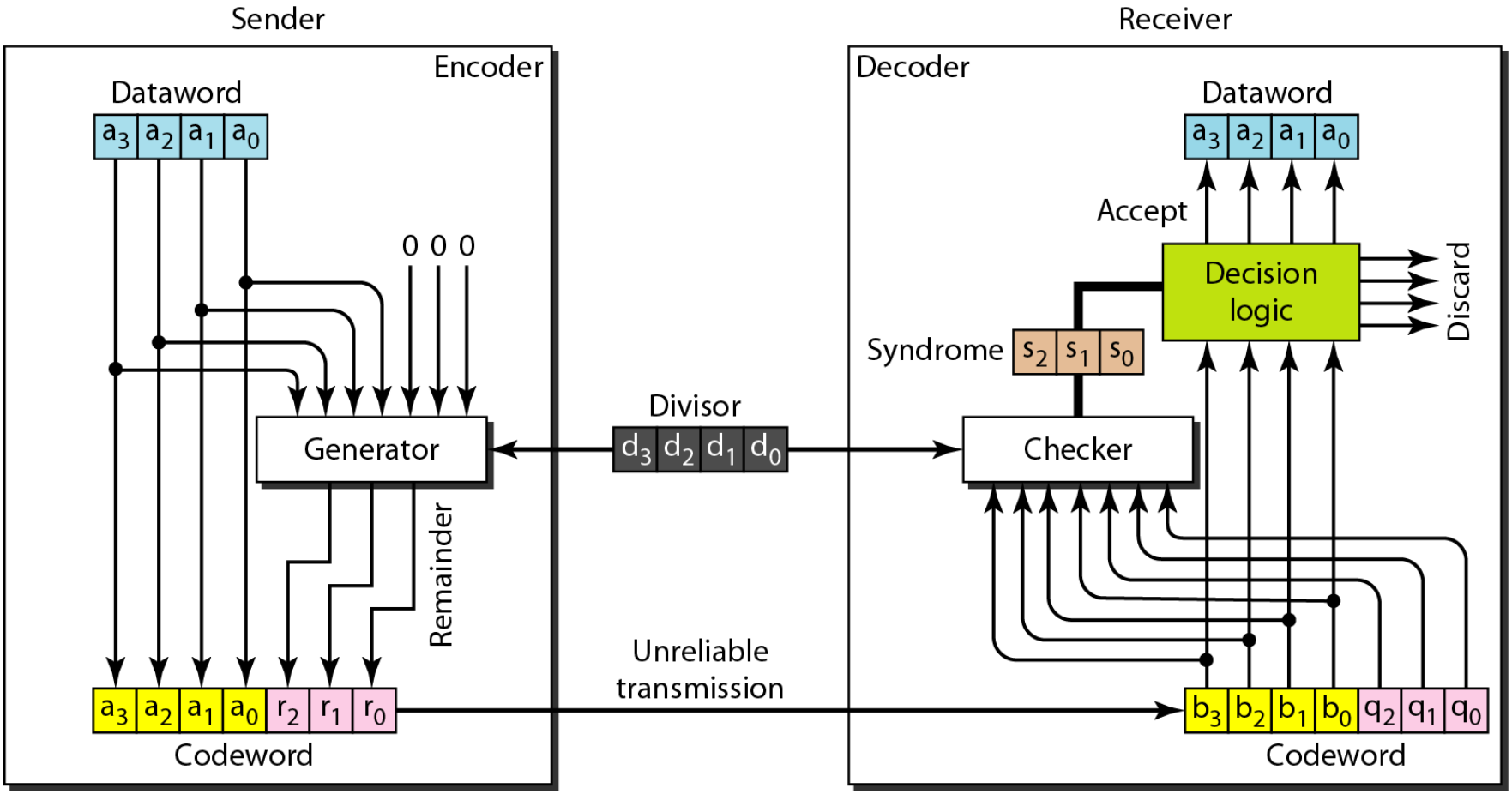
Cyclic codes are special linear block codes with one extra property. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.



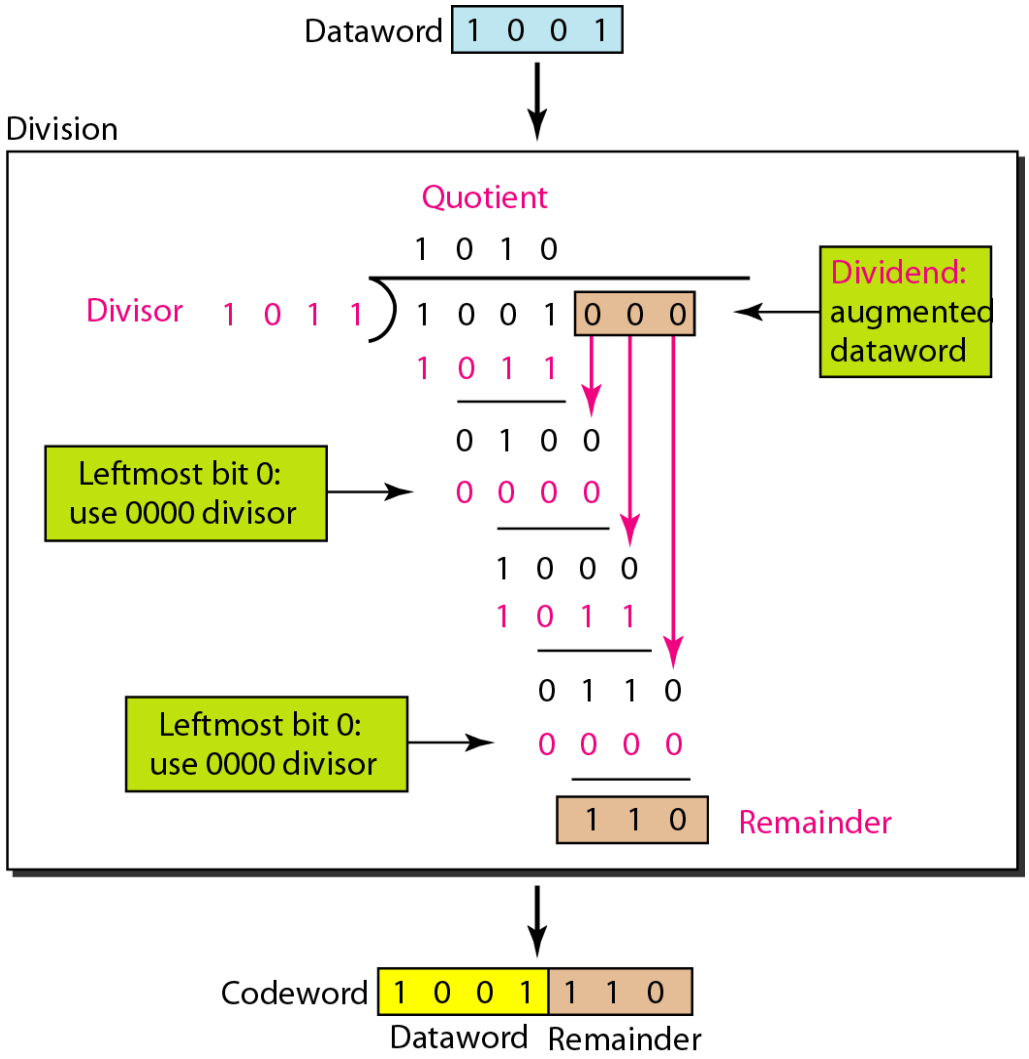
A *CRC code with C(7, 4)*

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

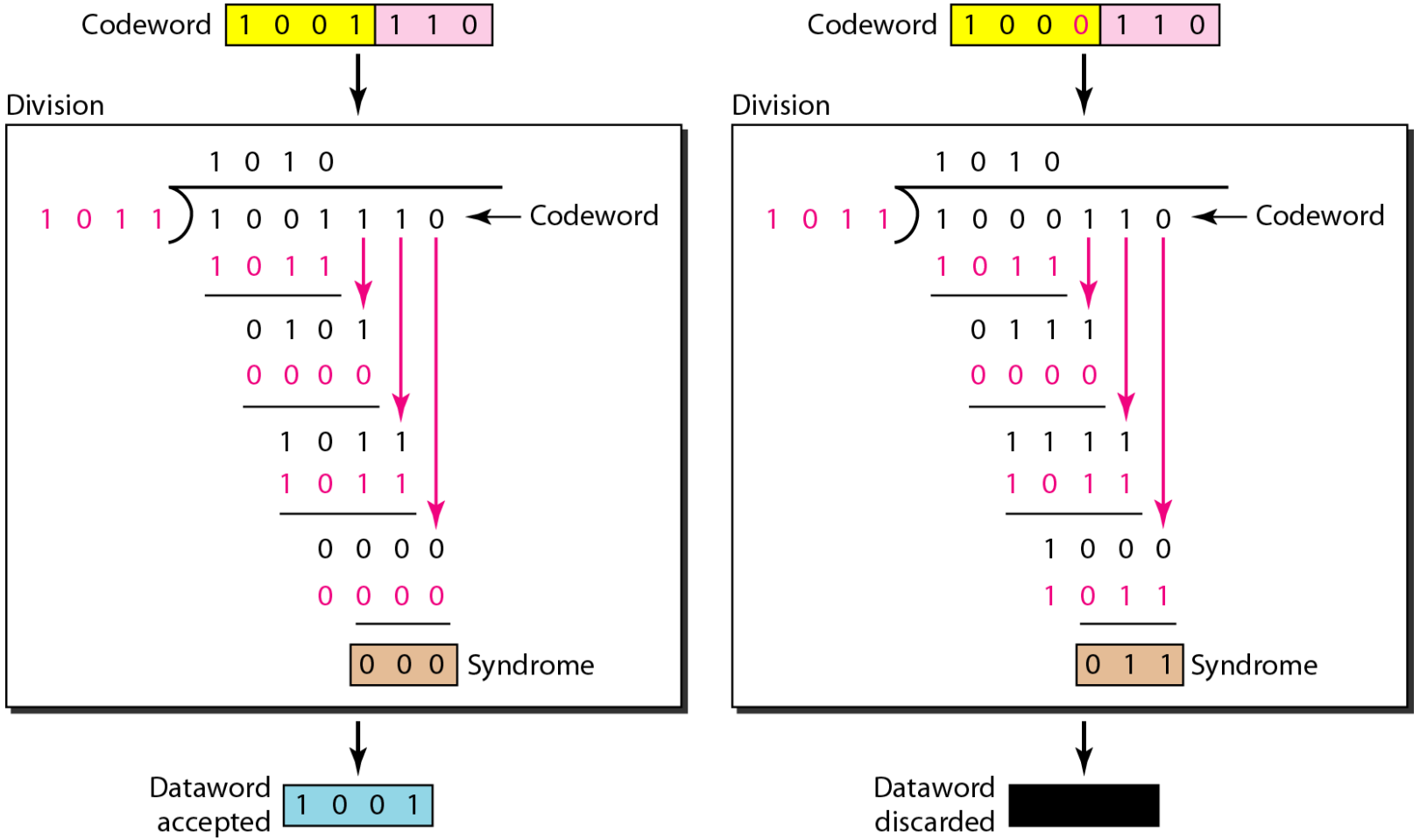
CRC encoder and decoder



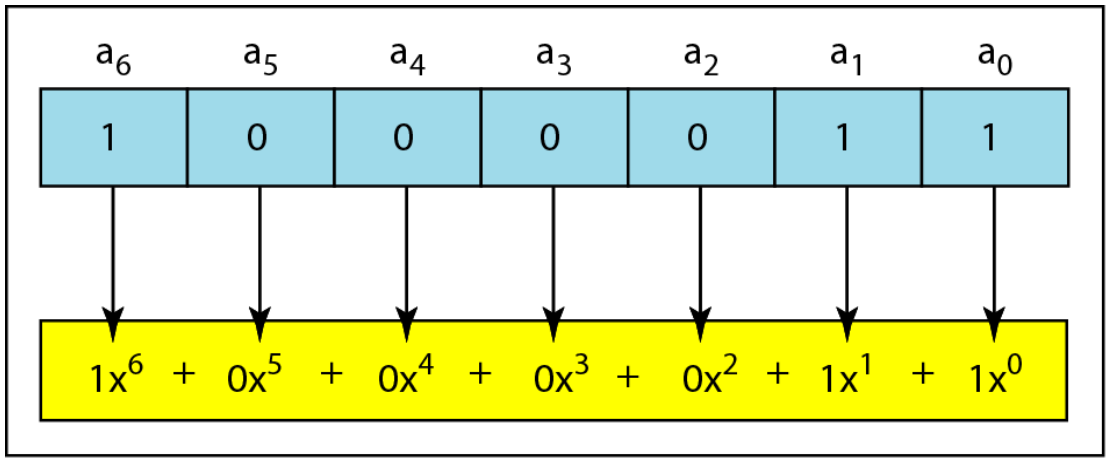
Division in CRC encoder



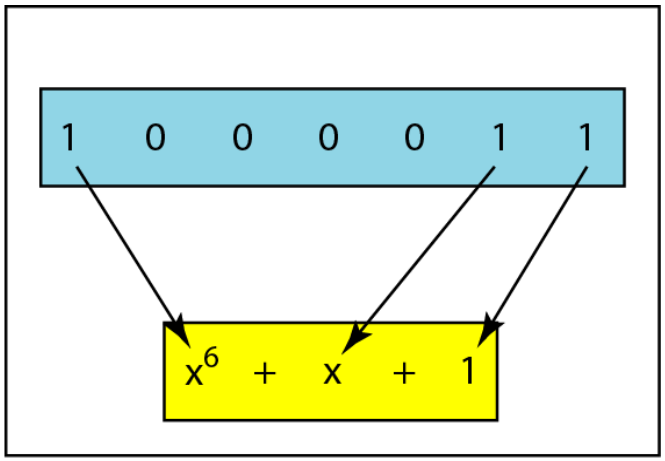
Division in the CRC decoder for two cases



A polynomial to represent a binary word

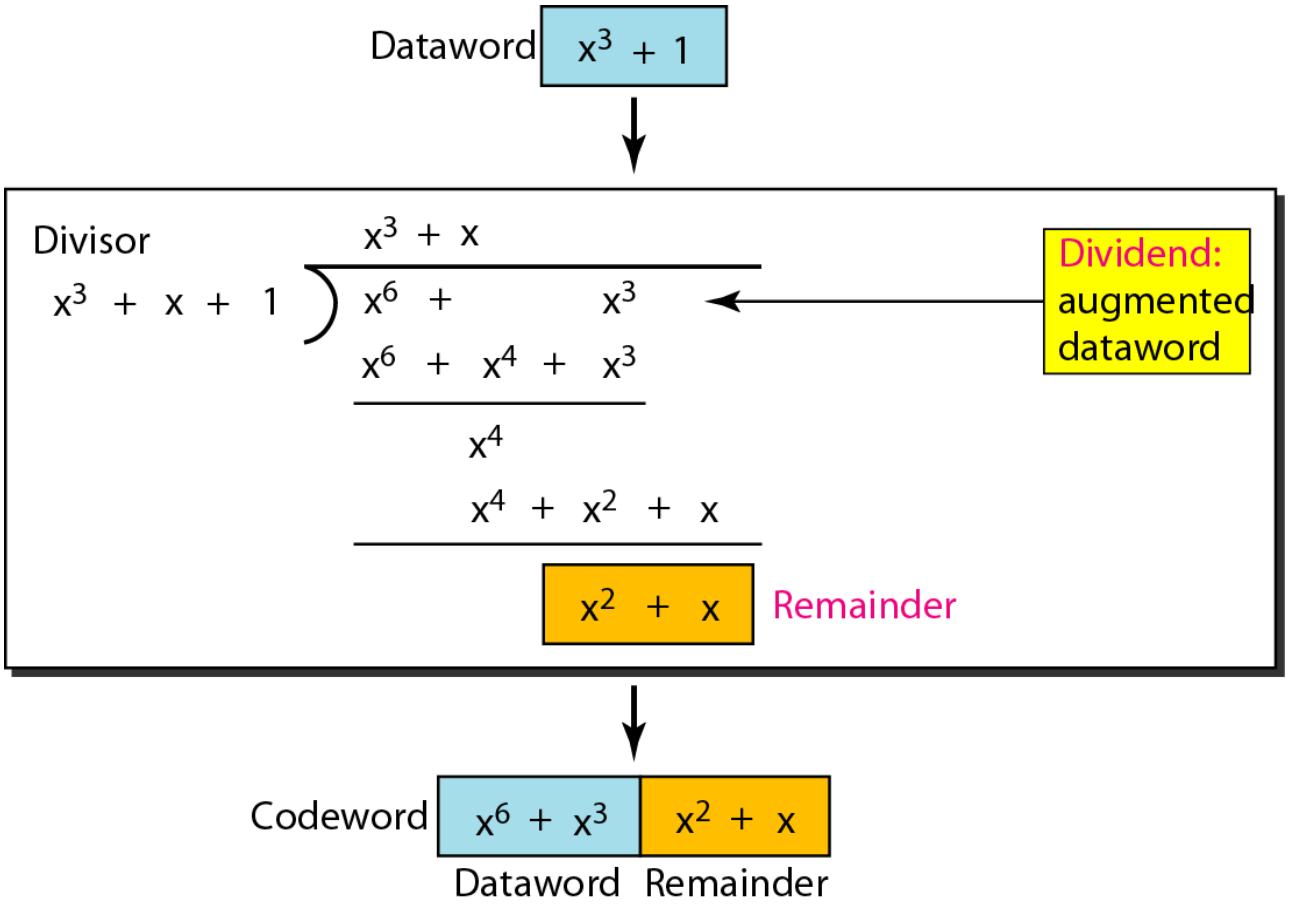


a. Binary pattern and polynomial



b. Short form

CRC division using polynomials



Note

The divisor in a cyclic code is normally called the generator polynomial or simply the generator.

Note

In a cyclic code,

If $s(x) \neq 0$, one or more bits is corrupted.

If $s(x) = 0$, either

a. No bit is corrupted. or

b. Some bits are corrupted, but the decoder failed to detect them.

Note

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

Note

If the generator has more than one term and the coefficient of x^0 is 1, all single errors can be caught.

Which of the following g(x) values guarantees that a single-bit error is caught? For each case, what is the error that cannot be caught?

a. $x + 1$ *b.* x^3 *c.* 1

Solution

a. No x^i can be divisible by $x + 1$. Any single-bit error can be caught.

b. If i is equal to or greater than 3, x^i is divisible by g(x). All single-bit errors in positions 1 to 3 are caught.

c. All values of i make x^i divisible by g(x). No single-bit error can be caught. This g(x) is useless.

Note

A generator that contains a factor of $x + 1$ can detect all odd-numbered errors.

Note

- All burst errors with $L \leq r$ will be detected.
- All burst errors with $L = r + 1$ will be detected with probability $1 - (1/2)^{r-1}$.
- All burst errors with $L > r + 1$ will be detected with probability $1 - (1/2)^{r}$.

Note

A good polynomial generator needs to have the following characteristics:

1. It should 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 $x + 1$.

Standard polynomials

Name	*Polynomial*	*Application*
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

CHECKSUM

The last error detection method we discuss here is called the checksum. The checksum is used in the Internet by several protocols although not at the data link layer.

Suppose our data is a list of five 4-bit numbers that we want to send to a destination. In addition to sending these numbers, we send the sum of the numbers. For example, if the set of numbers is (7, 11, 12, 0, 6), we send (7, 11, 12, 0, 6, 36), where 36 is the sum of the original numbers. The receiver adds the five numbers and compares the result with the sum. If the two are the same, the receiver assumes no error, accepts the five numbers, and discards the sum. Otherwise, there is an error somewhere and the data are not accepted.

We can make the job of the receiver easier if we send the negative (complement) of the sum, called the checksum. In this case, we send (7, 11, 12, 0, 6, −36). The receiver can add all the numbers received (including the checksum). If the result is 0, it assumes no error; otherwise, there is an error.

Note

Sender site:

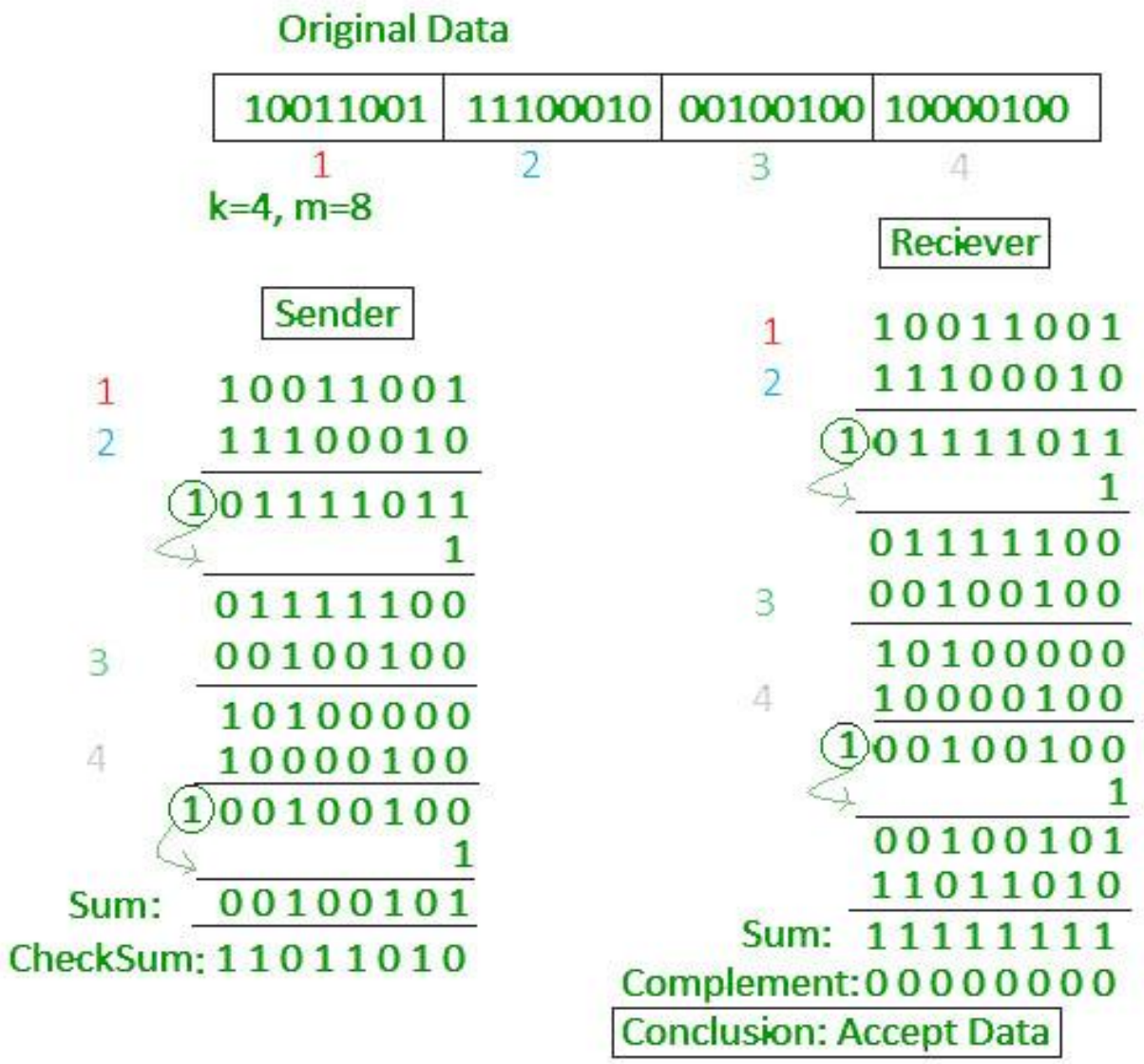
1. The message is divided into 16-bit words.
2. The value of the checksum word is set to 0.
3. All words including the checksum are added using one's complement addition.
4. The sum is complemented and becomes the checksum.
5. The checksum is sent with the data.

Note

Receiver site:

1. The message (including checksum) is divided into 16-bit words.
2. All words are added using one's complement addition.
3. The sum is complemented and becomes the new checksum.
4. If the value of checksum is 0, the message is accepted; otherwise, it is rejected.

Checksum


Original Data
10011001
11100010
00100100
10000100
1
2
3
4
k=4, m=8
Sender
1 10011001
2 11100010
01111011
1
01111100
3 00100100
10100000
4 10000100
00100100
1
Sum: 00100101
CheckSum: 11011010
Reciever
1 10011001
2 11100010
01111011
1
01111100
3 00100100
10100000
4 10000100
00100100
1
00100101
11011010
Sum: 11111111
Complement: 00000000
Conclusion: Accept Data

Exercise

1 . Checksums use _____ arithmetic.

A) one's complement arithmetic

B) two's complement arithmetic

C) either (a) or (b)

D) none of the above

Exercise

1 . Checksums use _____ arithmetic.

A) one's complement arithmetic

B) two's complement arithmetic

C) either (a) or (b)

D) none of the above

Exercise

2. In modulo-11 arithmetic, we use only the integers in the range _____, inclusive.

A) 1 to 10
B) 0 to 10
C) 1 to 11
D) none of the above

Exercise

2. In modulo-11 arithmetic, we use only the integers in the range _____, inclusive.

A) 1 to 10

B) 0 to 10

C) 1 to 11

D) none of the above

Exercise

3. In cyclic redundancy checking, the divisor is _____ the CRC.

A) one bit less than

B) one bit more than

C) The same size as

D) none of the above

Exercise

3. In cyclic redundancy checking, the divisor is _____ the CRC.

A) one bit less than

B) one bit more than

C) The same size as

D) none of the above

Exercise

4. In modulo-2 arithmetic, _____ give the same results.

A) addition and subtraction

B) addition and multiplication

C) addition and division

D) none of the above

Exercise

4. In modulo-2 arithmetic, _____ give the same results.

A) addition and subtraction

B) addition and multiplication

C) addition and division

D) none of the above

Exercise

5. In cyclic redundancy checking, what is the CRC?

A) The quotient

B) The dividend

C) The divisor

D) The remainder

Exercise

5. In cyclic redundancy checking, what is the CRC?

A) The quotient

B) The dividend

C) The divisor

D) The remainder

Exercise

6. Which error detection method consists of just one redundant bit per data unit?

A) CRC

B) Checksum

C) Simple parity check

D) Two-dimensional parity check

Exercise

6. Which error detection method consists of just one redundant bit per data unit?

A) CRC

B) Checksum

C) Simple parity check

D) Two-dimensional parity check

Exercise

7. To guarantee correction of up to 5 errors in all cases, the minimum Hamming distance in a block code must be _____.

A) 11
B) 6
C) 5
D) none of the above

Exercise

7. To guarantee correction of up to 5 errors in all cases, the minimum Hamming distance in a block code must be _____.

A) 11
B) 6
C) 5
D) none of the above

Exercise

8. In block coding, if k = 2 and n = 3, we have _____ invalid codewords.

A) 4

B) 8

C) 2

D) none of the above

Exercise

8. In block coding, if k = 2 and n = 3, we have _____ invalid codewords.

A) 4

B) 8

C) 2

D) none of the above

Exercise

9. Let G(x) be the generator polynomial used for CRC checking. What is the condition that should be satisfied by G(x) to detect odd number of bits in error?

(a) G(x) contains more than two terms

(b) G(x) does not divide $1+x^k$, for any k not exceeding the frame length

(c) 1+x is a factor of G(x)

(d) G(x) has an odd number of terms

Exercise

9. Let G(x) be the generator polynomial used for CRC checking. What is the condition that should be satisfied by G(x) to detect odd number of bits in error?

(a) G(x) contains more than two terms

(b) G(x) does not divide $1+x^k$, for any k not exceeding the frame length

(c) 1+x is a factor of G(x)

(d) G(x) has an odd number of terms

Exercise

10. The message 11001001 is to be transmitted using the CRC polynomial x^3 + 1 to protect it from errors. The message that should be transmitted is:

(a) 11001001000

(b) 11001001011

(c) 11001010

(d) 110010010011

Exercise

10. The message 11001001 is to be transmitted using the CRC polynomial x^3 + 1 to protect it from errors. The message that should be transmitted is:

(a) 11001001000

(b) 11001001011

(c) 11001010

(d) 110010010011

Sol : The polynomial x^3+1 corresponds to divisor is 1001.

```
11001001 000 <--- input right padded by 3 bits
1001 <--- divisor
01011001 000 <---- XOR of the above 2
  1001              <--- divisor
00010001 000
     1001
00000011 000
         10 01
00000001 010
           1 001
00000000 011 <------- remainder (3 bits)
```

Exercise

11. A computer network uses polynomials for error checking with 8 bits as information bits and uses $x^3 + x + 1$ as the generator polynomial to generate the check bits. In this network, the message 01011011 is transmitted as

(a) 01011011010

(b) 01011011011

(c) 01011011101

(d) 01011011100

Exercise

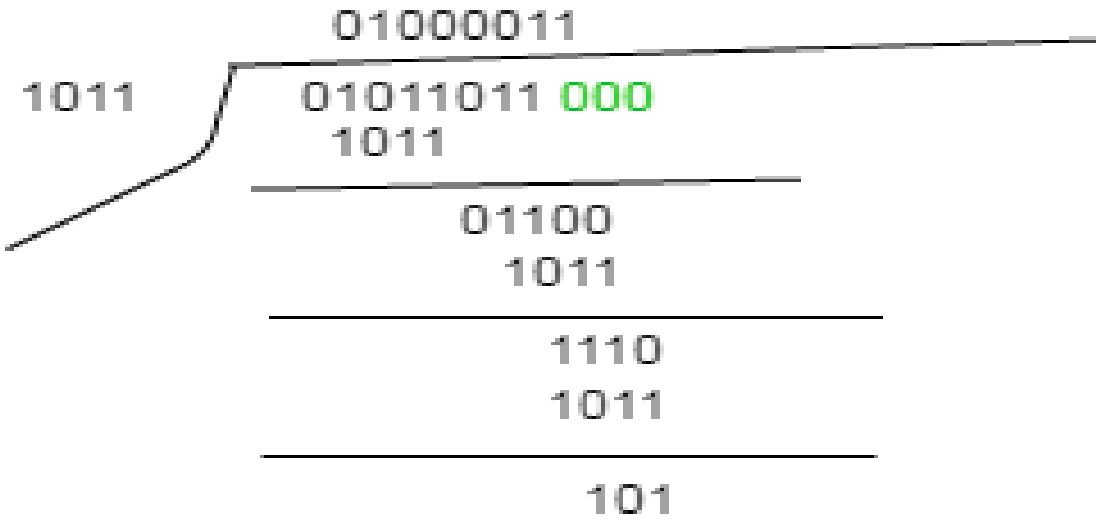
11. A computer network uses polynomials for error checking with 8 bits as information bits and uses $x^3 + x + 1$ as the generator polynomial to generate the check bits. In this network, the message 01011011 is transmitted as

(a) 01011011010

(b) 01011011011

(c) 01011011101

(d) 01011011100



Exercise

12. A code scheme has a Hamming distance d_{min} = 4. What is the error detection and correction capability of this scheme?

(a) 3,1

(b) 4,2

(c) 4,1

(d) 3,2

Exercise

12. A code scheme has a Hamming distance d_{min} = 4. What is the error detection and correction capability of this scheme?

(a) 3,1

(b) 4,2

(c) 4,1

(d) 3,2

Exercise

13. When 2 or more bits in a data unit has been changed during the transmission, the error is called _____

a) random error

b) burst error

c) inverted error

d) double error

Exercise

13. When 2 or more bits in a data unit has been changed during the transmission, the error is called _____

a) random error

b) burst error

c) inverted error

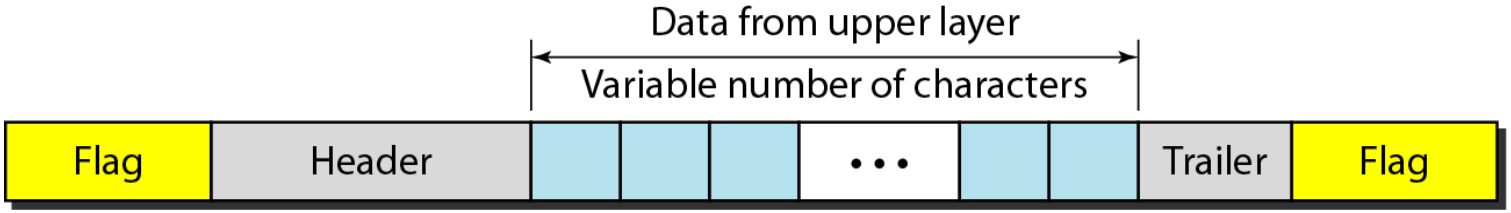
d) double error

Data Link Control: FRAMING

The data link layer needs to pack bits into frames, so that each frame is distinguishable from another.
Our postal system practices a type of framing. The simple act of inserting a letter into an envelope separates one piece of information from another; the envelope serves as the delimiter.

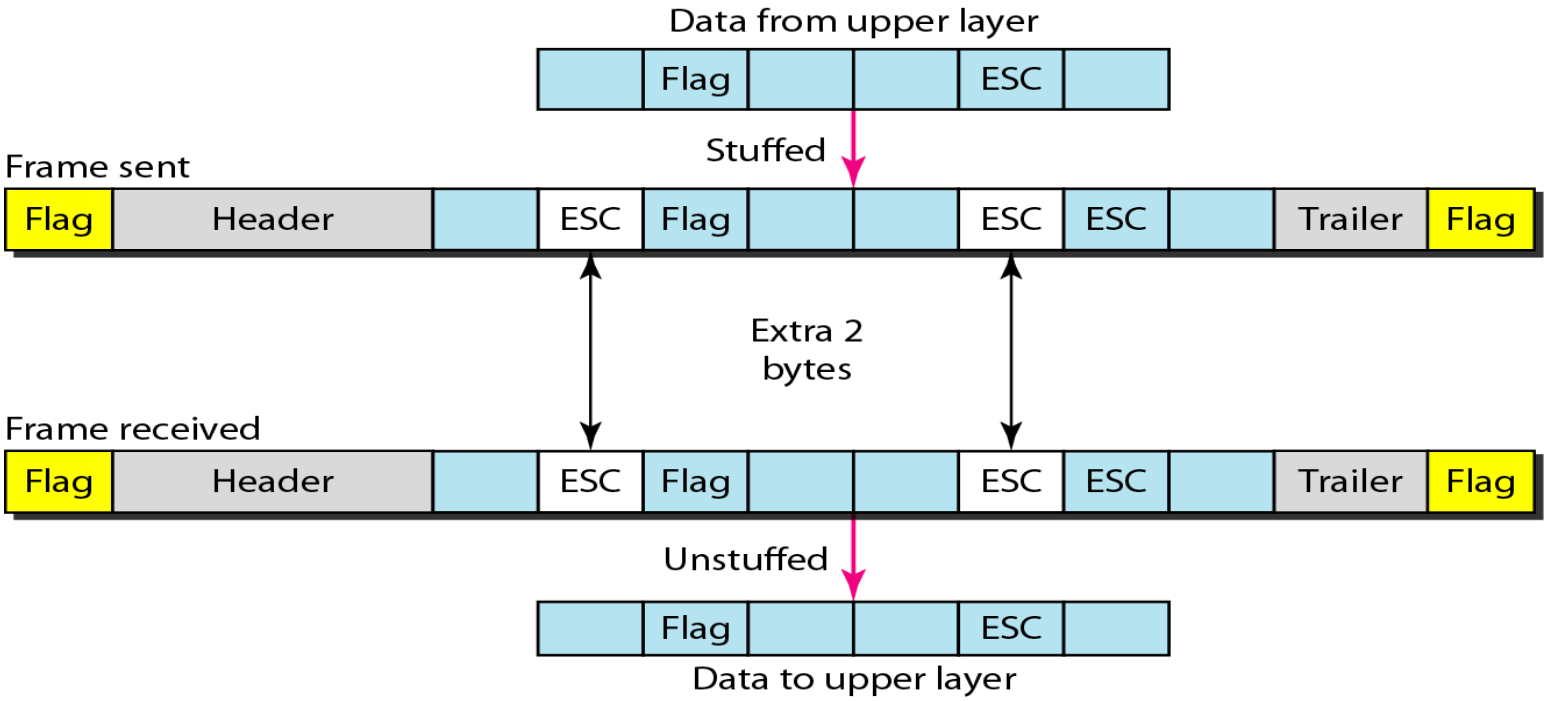
- Fixed-Size Framing
- Variable-Size Framing

A frame in a character-oriented protocol



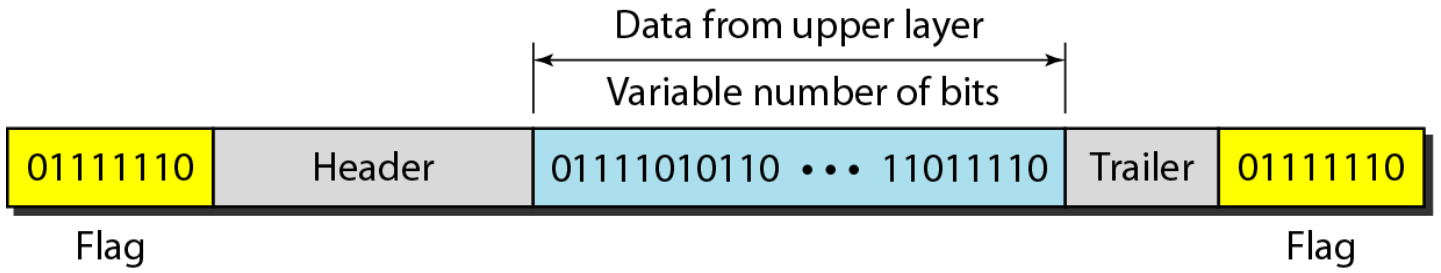
Byte stuffing and unstuffing

(Point-to-Point Protocol (PPP) is a byte-oriented protocol).

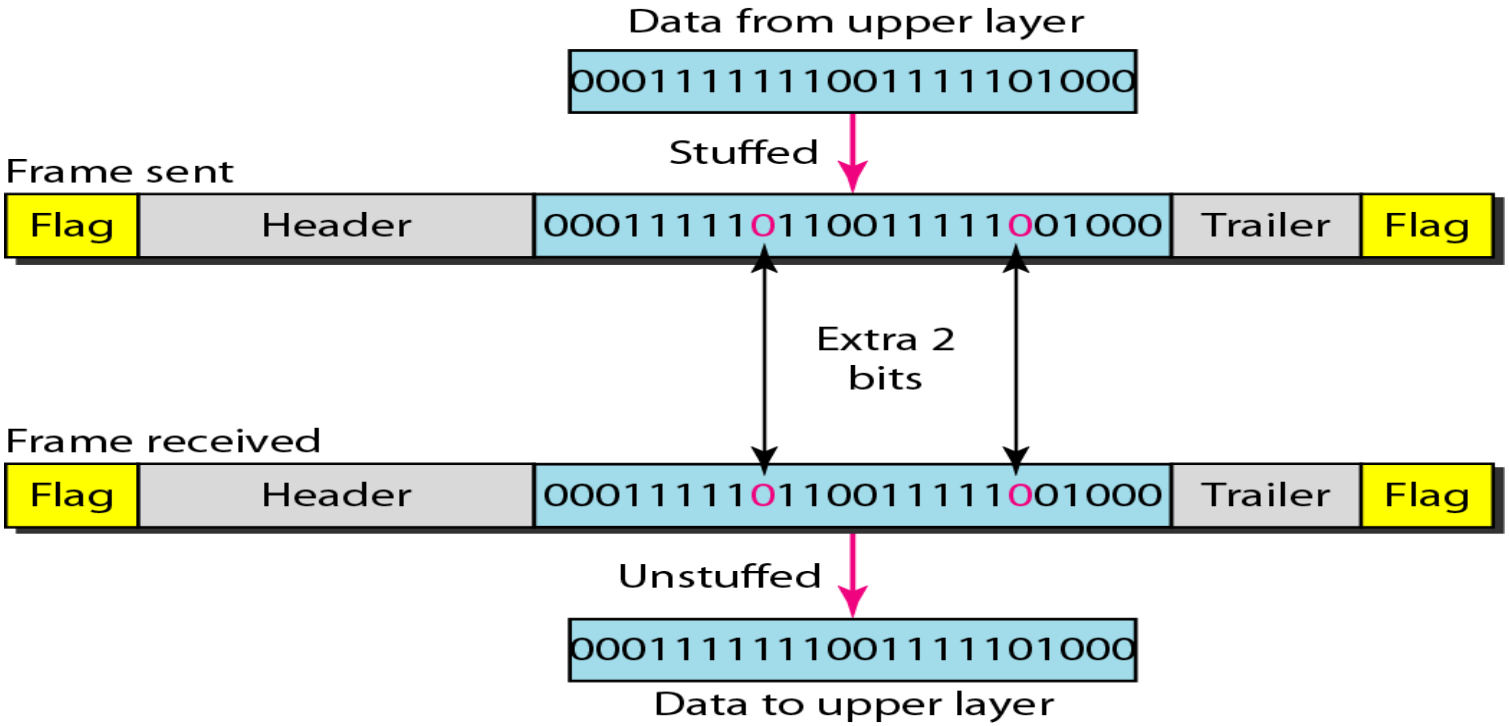


Byte stuffing is the process of adding 1 extra byte whenever there is a flag or escape character in the text.

A frame in a bit-oriented protocol (HDLC)



Bit stuffing and unstuffing



Bit stuffing is the process of adding one extra 0 whenever five consecutive 1s follow a 0 in the data, so that the receiver does not mistake the pattern 0111110 for a flag.

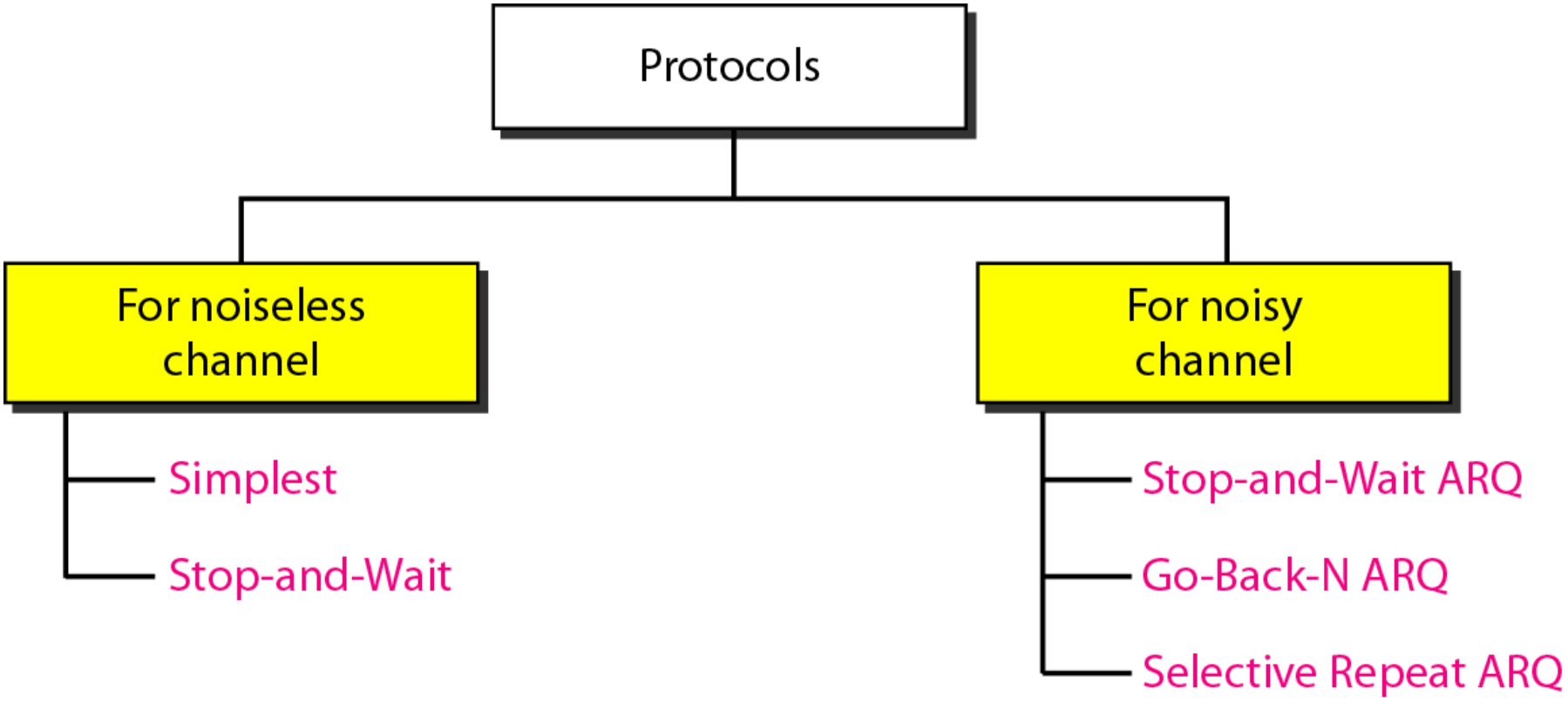
FLOW AND ERROR CONTROL

The most important responsibilities of the data link layer are flow control and error control. Collectively, these functions are known as data link control.

- ✓ Flow Control: set of procedures used to restrict the amount of data that the sender can send before waiting for acknowledgment.
- ✓ Error Control: data link layer is based on automatic repeat request, which is the retransmission of data.

Data Link Layer Protocols

combine framing, flow control, and error control to achieve the delivery of data from one node to another.

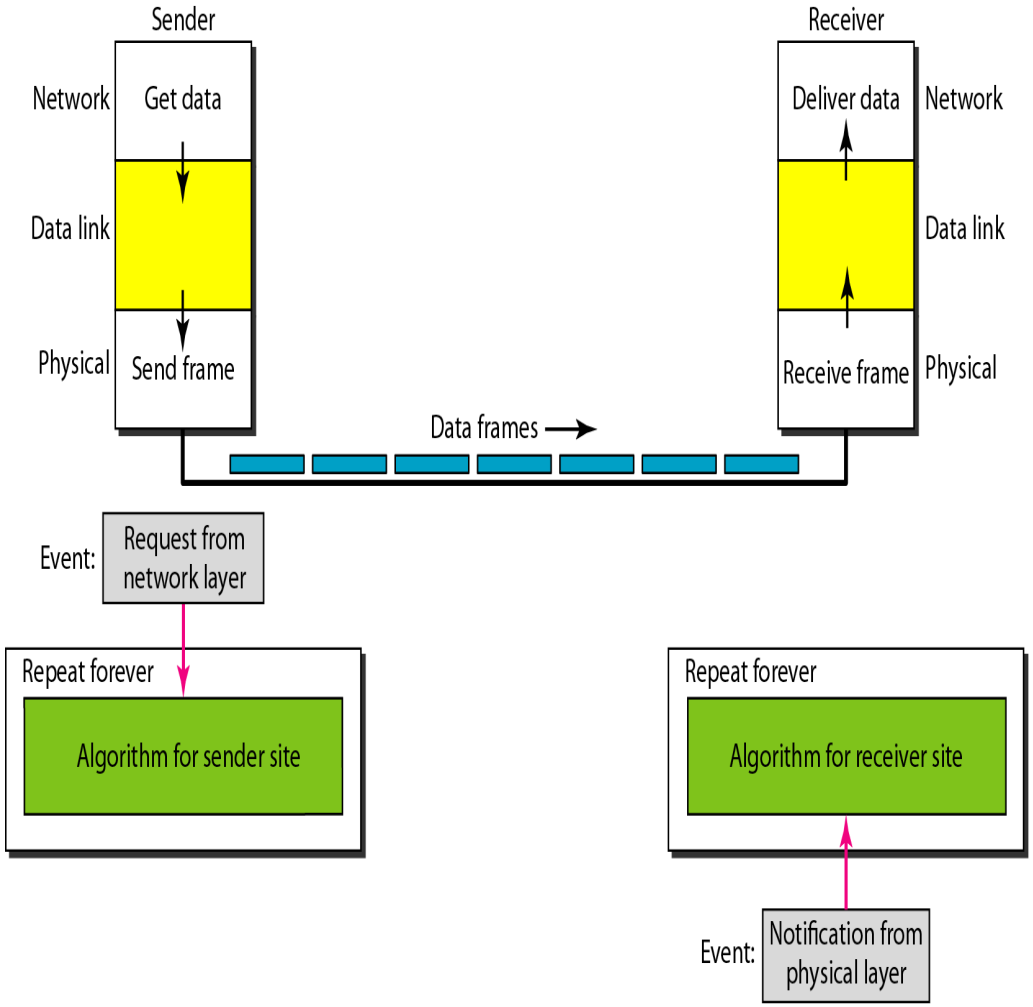


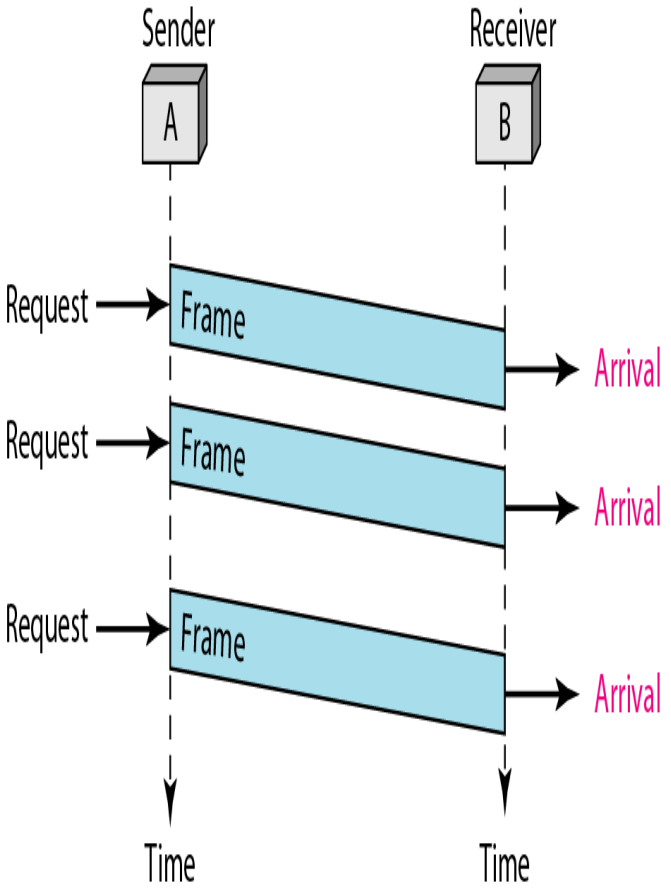
NOISELESS CHANNELS

Let us first assume we have an ideal channel in which no frames are lost, duplicated, or corrupted.
Two protocols for this type of channel.

- Simplest Protocol
- Stop-and-Wait Protocol

The design of the simplest protocol with no flow or error control





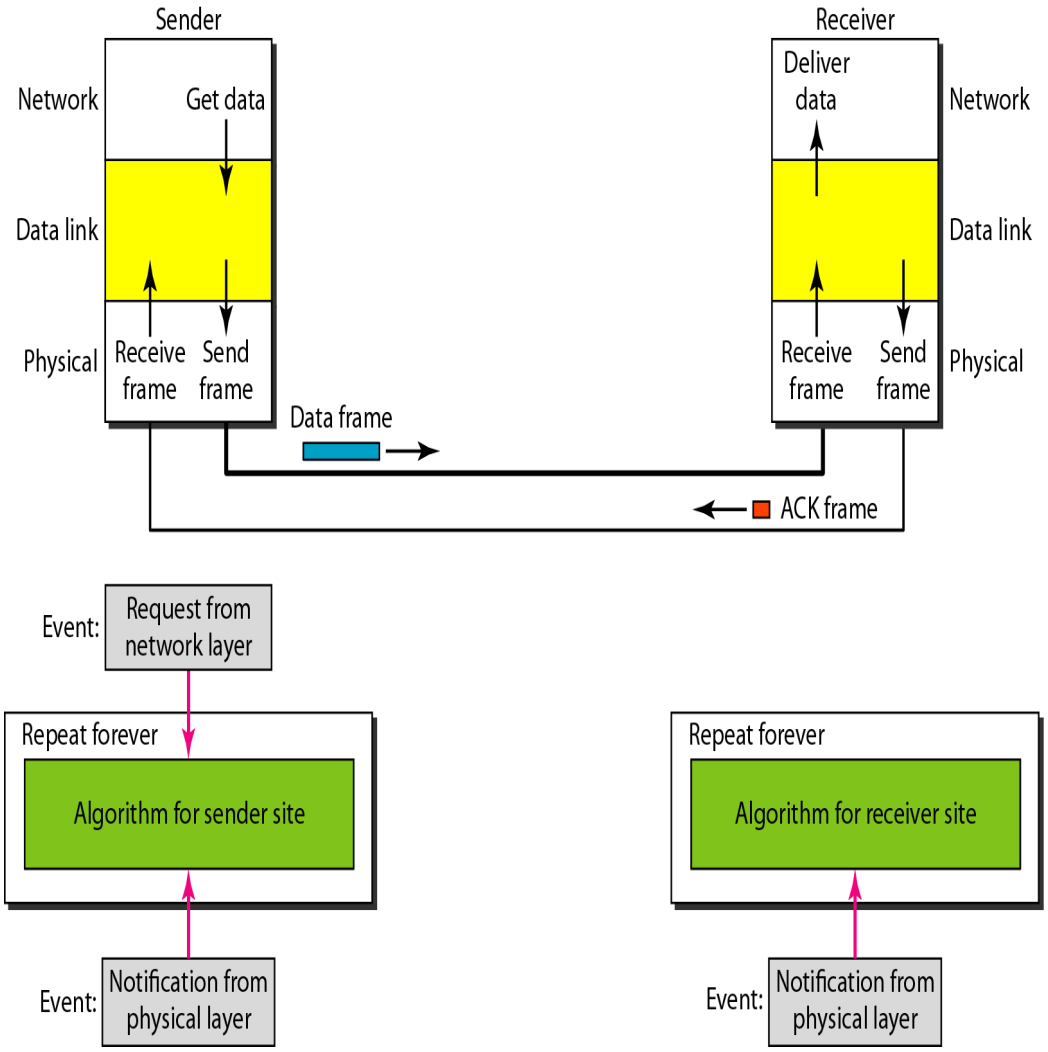
Sender-side algorithm for the simplest protocol

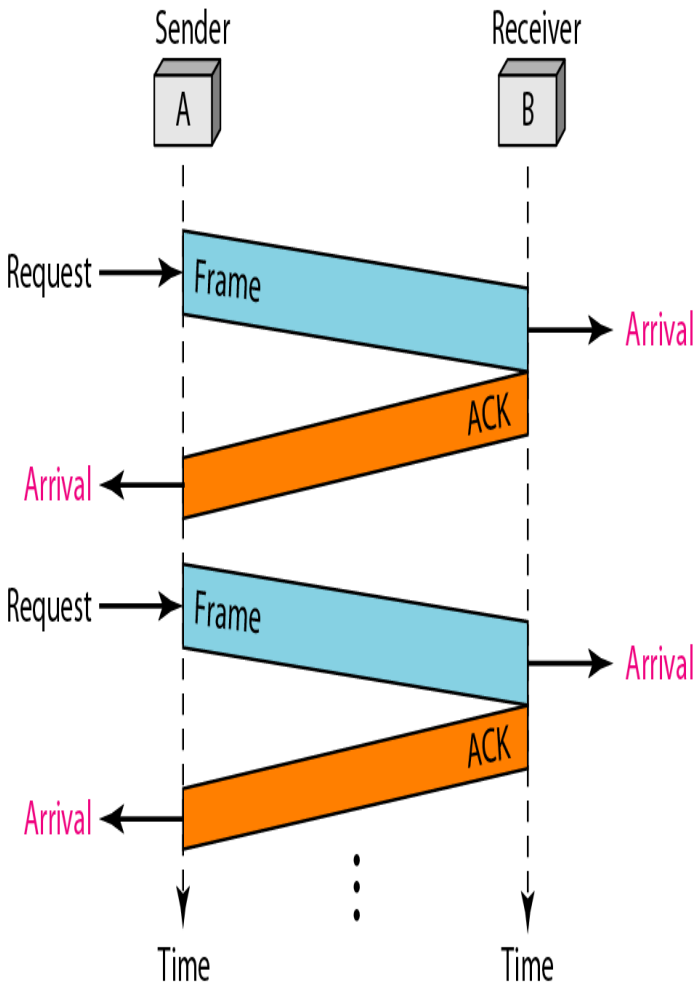
```
1  while(true)                          // Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(RequestToSend))           //There is a packet to send
5    {
6       GetData();
7       MakeFrame();
8       SendFrame();                    //Send the frame
9    }
10 }
```

Receiver-side algorithm for the simplest protocol

```
1  while(true)                          // Repeat forever
2  {
3    WaitForEvent();                    // Sleep until an event occurs
4    if(Event(ArrivalNotification)) //Data frame arrived
5    {
6       ReceiveFrame();
7       ExtractData();
8       DeliverData();                  //Deliver data to network layer
9    }
10 }
```

Design of Stop-and-Wait Protocol





Sender-site algorithm for Stop-and-Wait Protocol

```
1  while(true)                              //Repeat forever
2  canSend = true                           //Allow the first frame to go
3  {
4    WaitForEvent();                        // Sleep until an event occurs
5    if(Event(RequestToSend) AND canSend)
6    {
7        GetData();
8        MakeFrame();
9        SendFrame();                       //Send the data frame
10       canSend = false;                   //Cannot send until ACK arrives
11   }
12   WaitForEvent();                        // Sleep until an event occurs
13   if(Event(ArrivalNotification)          // An ACK has arrived
14     {
15       ReceiveFrame();                    //Receive the ACK frame
16       canSend = true;
17     }
18 }
```

Receiver-site algorithm for Stop-and-Wait Protocol

```
1  while(true)                              //Repeat forever
2  {
3    WaitForEvent();                        // Sleep until an event occurs
4    if(Event(ArrivalNotification))         //Data frame arrives
5    {
6        ReceiveFrame();
7        ExtractData();
8        Deliver(data);                     //Deliver data to network layer
9        SendFrame();                       //Send an ACK frame
10   }
11 }
```