Error Detection and Correction

Data can be corrupted during transmission.

Some applications require that errors be detected and corrected.

- Some applications can tolerate small level of errors, eg.
- But, while transferring text, we except very high level of accuracy

10-1 INTRODUCTION

Let us first discuss some issues related, directly or indirectly, to error detection and correction.

Types of Errors (Single Bit error, Burst error)

Figure 10.1 ***Single-bit error (type of error)***

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

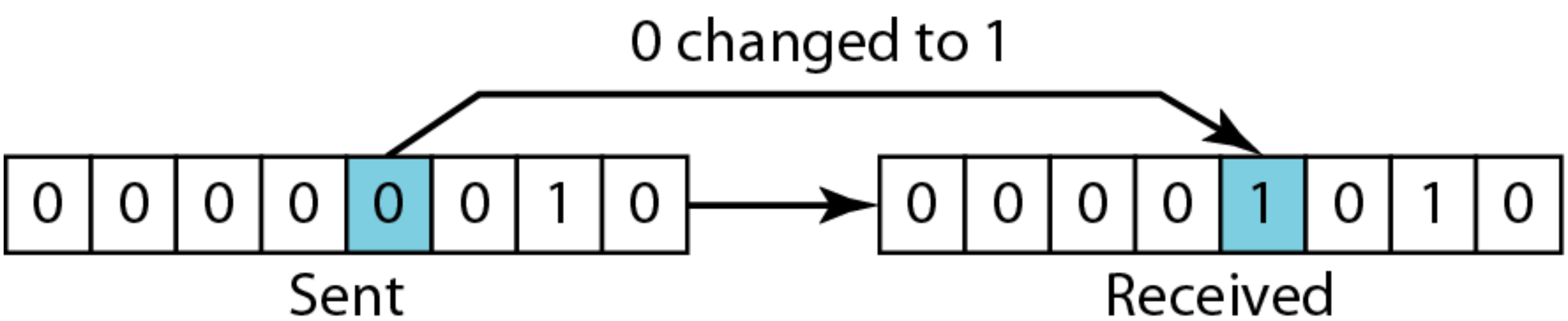


Figure 10.2 ***Burst error of length 8 (type of error)***

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

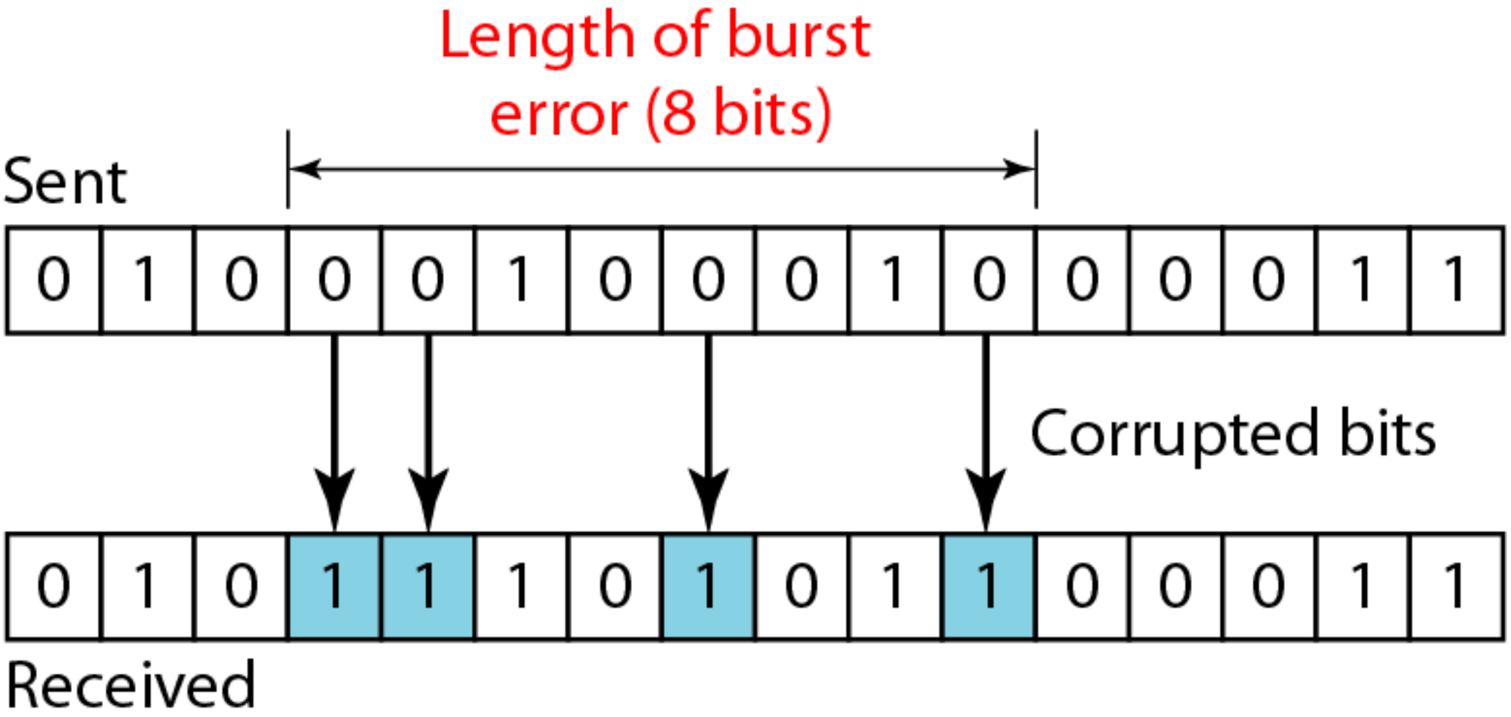
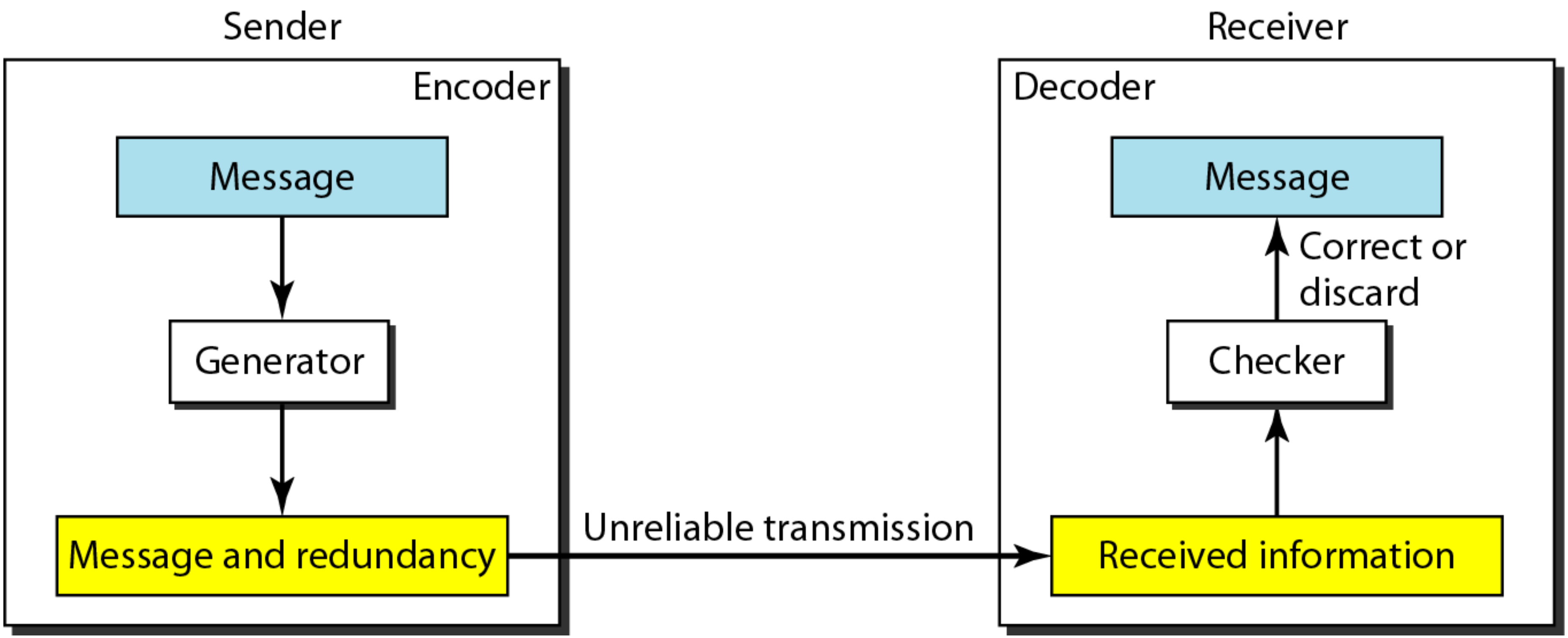


Figure 10.3 *The structure of encoder and decoder*

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

Block Coding vs Convolution coding



Detection Versus Correction

- Detection is more like a binary answer, either yes or not. We are not interested in number of errors
- In correction we should know exact number of errors and their location
- If we need to correct 2 errors in a 8-bit data, we need to consider 28 possibilities

Forward Error Correction Versus Retransmission

- Study *modulus N, what we use in binary numbers ?*

10-2 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.

Error Detection
Error Correction
Hamming Distance
Minimum Hamming Distance

Figure 10.5 *Datawords and codewords in block coding*

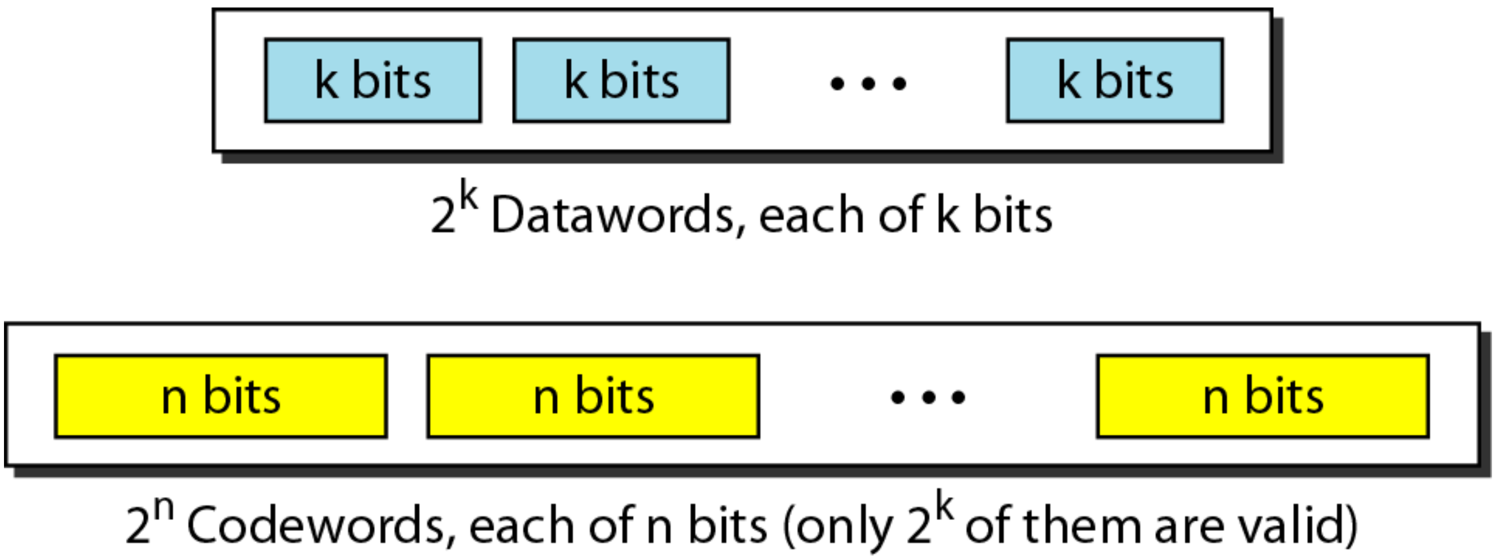
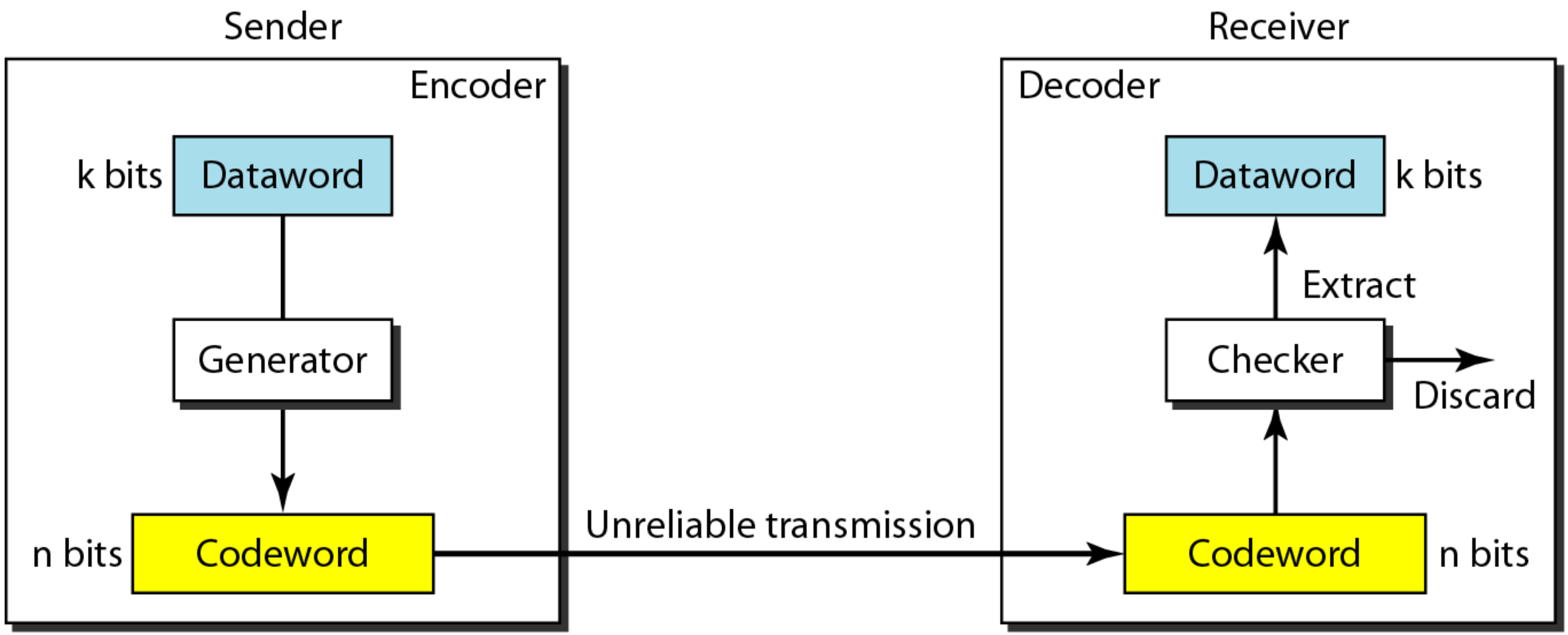


Figure 10.6 *Process of error detection in block coding*



Datawords	Codewords
00	000
01	011
10	101
11	110

Let us assume that k = 2 and n = 3. Table 10.1 shows the list of datawords and codewords. Later, we will see how to derive a codeword from a dataword.

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

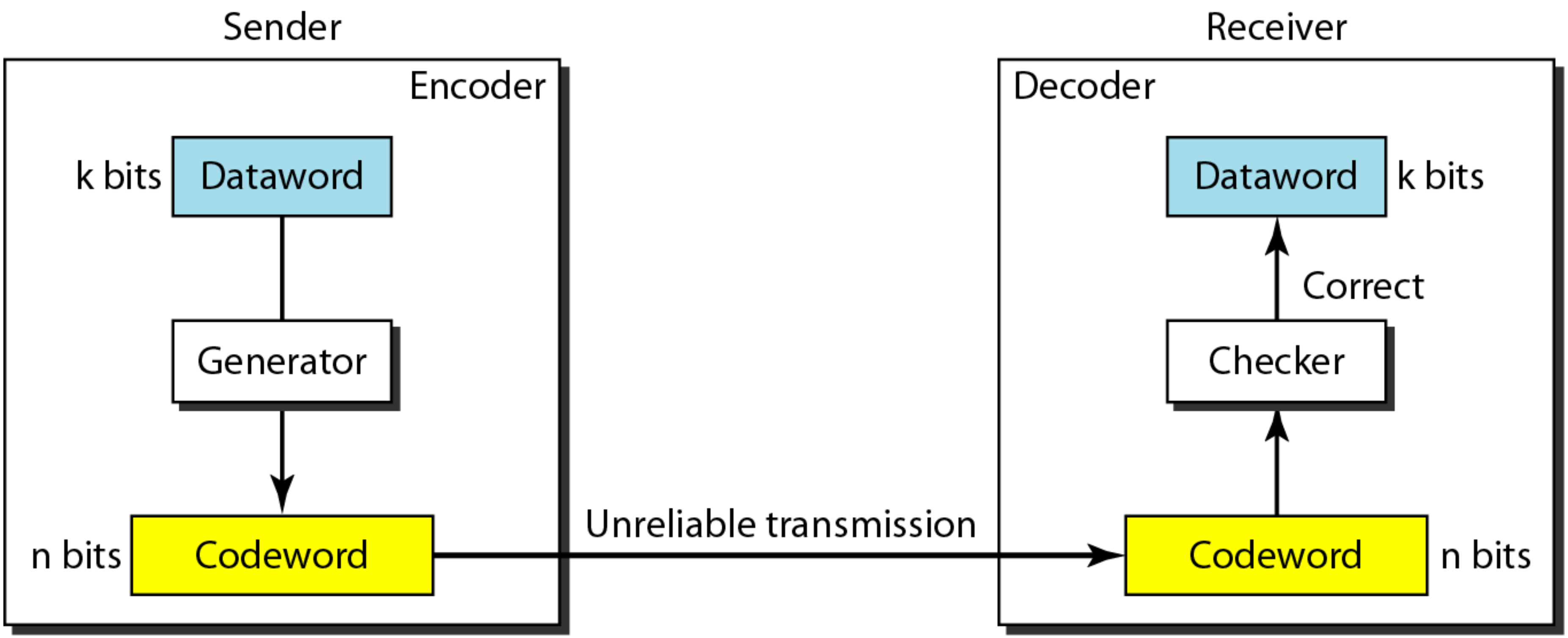
1. The receiver receives 011. It is a valid codeword. The receiver extracts the dataword 01 from it.

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.***

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

Figure 10.7 *Structure of encoder and decoder in error correction*



Usually, we need more redundant bits in error correction than in error detection

Dataword	Codeword
00	00000
01	01011
10	10101
11	11110

We add 3 redundant bits to the 2-bit dataword to make 5-bit codewords.

Assume the dataword is 01. The sender creates the codeword 01011. 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.***

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)

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 in Table 10.1.

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.

Example 10.6

Find the minimum Hamming distance of the coding scheme in Table 10.2.

Solution

We first find all the Hamming distances.

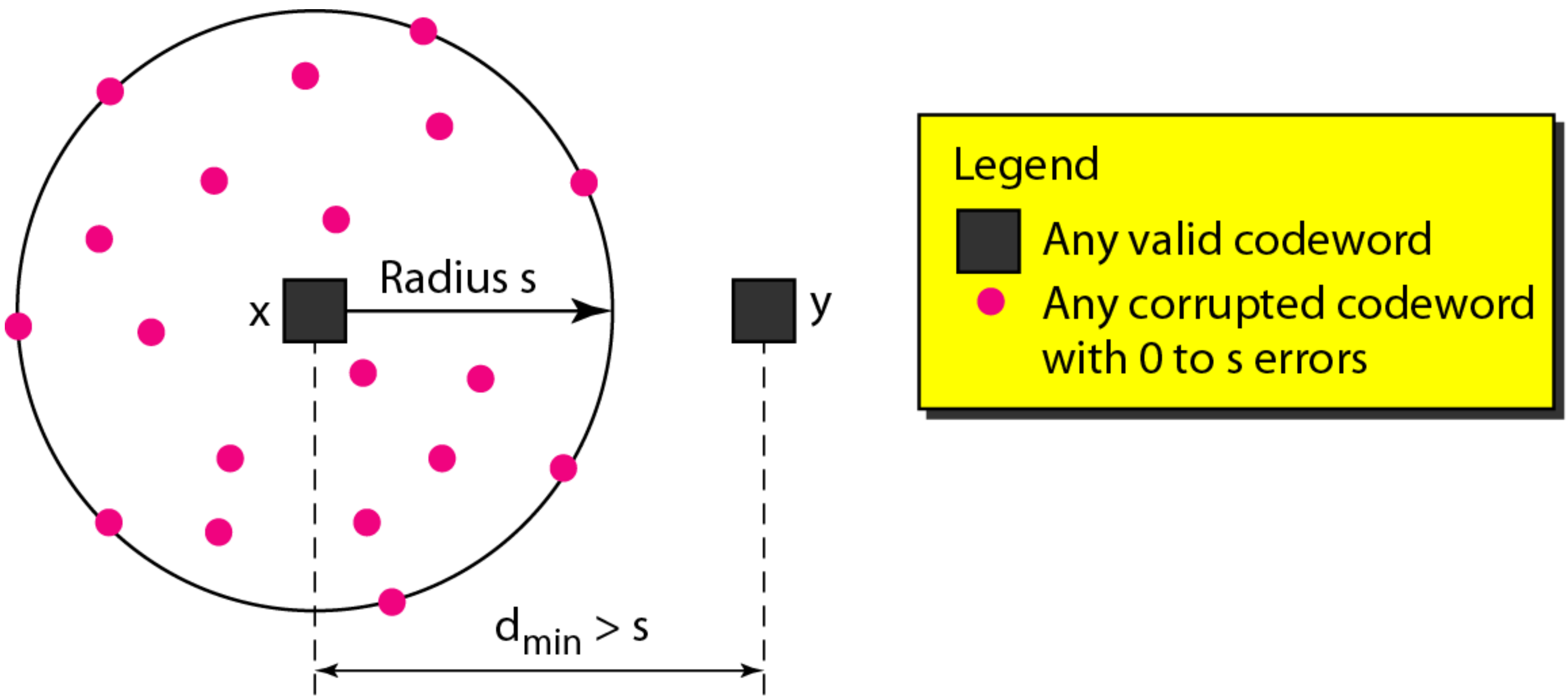
$d(00000, 01011) = 3$	$d(00000, 10101) = 3$	$d(00000, 11110) = 4$
$d(01011, 10101) = 4$	$d(01011, 11110) = 3$	$d(10101, 11110) = 3$

The d_{min} in this case is 3.

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

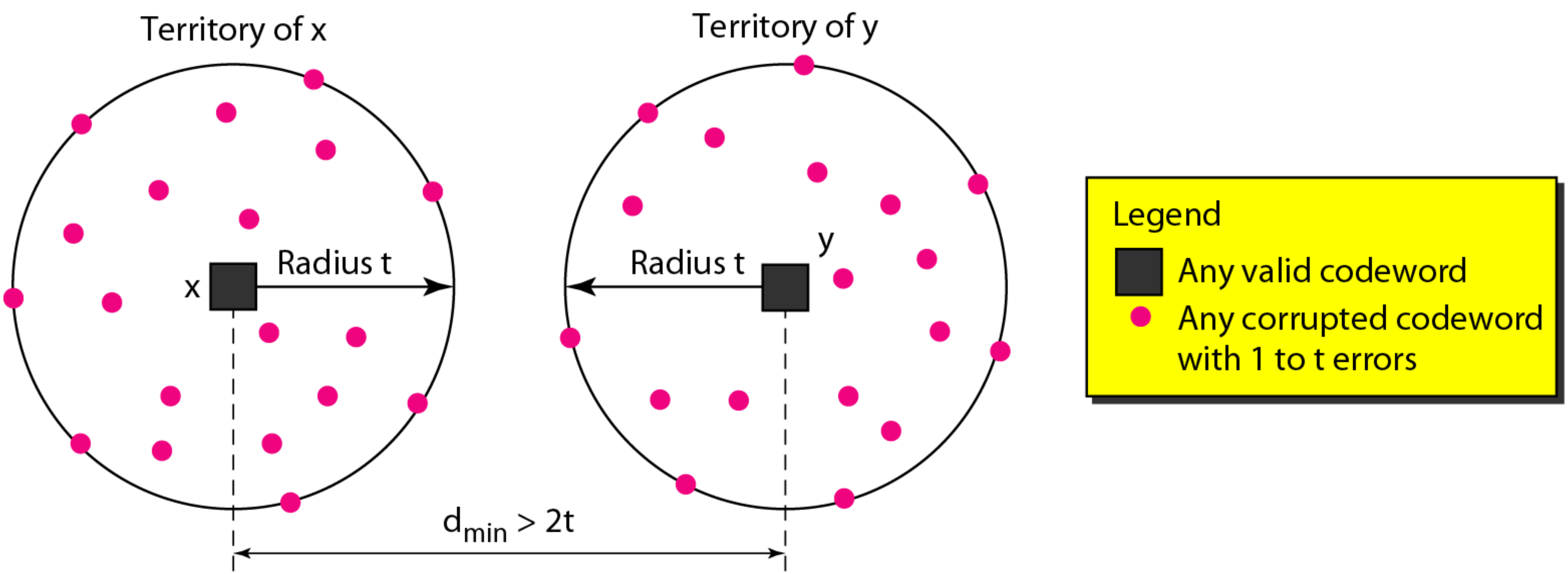
- Explain minimum hamming distance (2) in first example
- Explain minimum hamming distance (3) in second example

Figure 10.8 *Geometric concept for finding d_{min} in error detection*



- As d must be an integer, thus d_{min}=s+1 should hold

Figure 10.9 *Geometric concept for finding d_{min} in error correction*



- To gurantee error correction of upto t errors, minimum hamming distance in the block code must be d_{min}=2t+1

Example 10.9

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 ($s = 3$) as $4=s+1$,

but it can correct up to one error as $4=2t+1$.

10-3 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.

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

The XORing of the second and third codewords creates the fourth one.

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

We can create all four codewords by XORing two other codewords.

- If you have a liner block code. The minimum hamming distance is number of 1s in the non zero valid codeword

- ***Minimum Hamming distance is d_{min} = 2 (example 1).***

- ***In example 2, the numbers of 1s in the nonzero codewords are 3, 3, and 4. So in this code we have d_{min} = 3.***

Simple Parity Check Code (error detection)

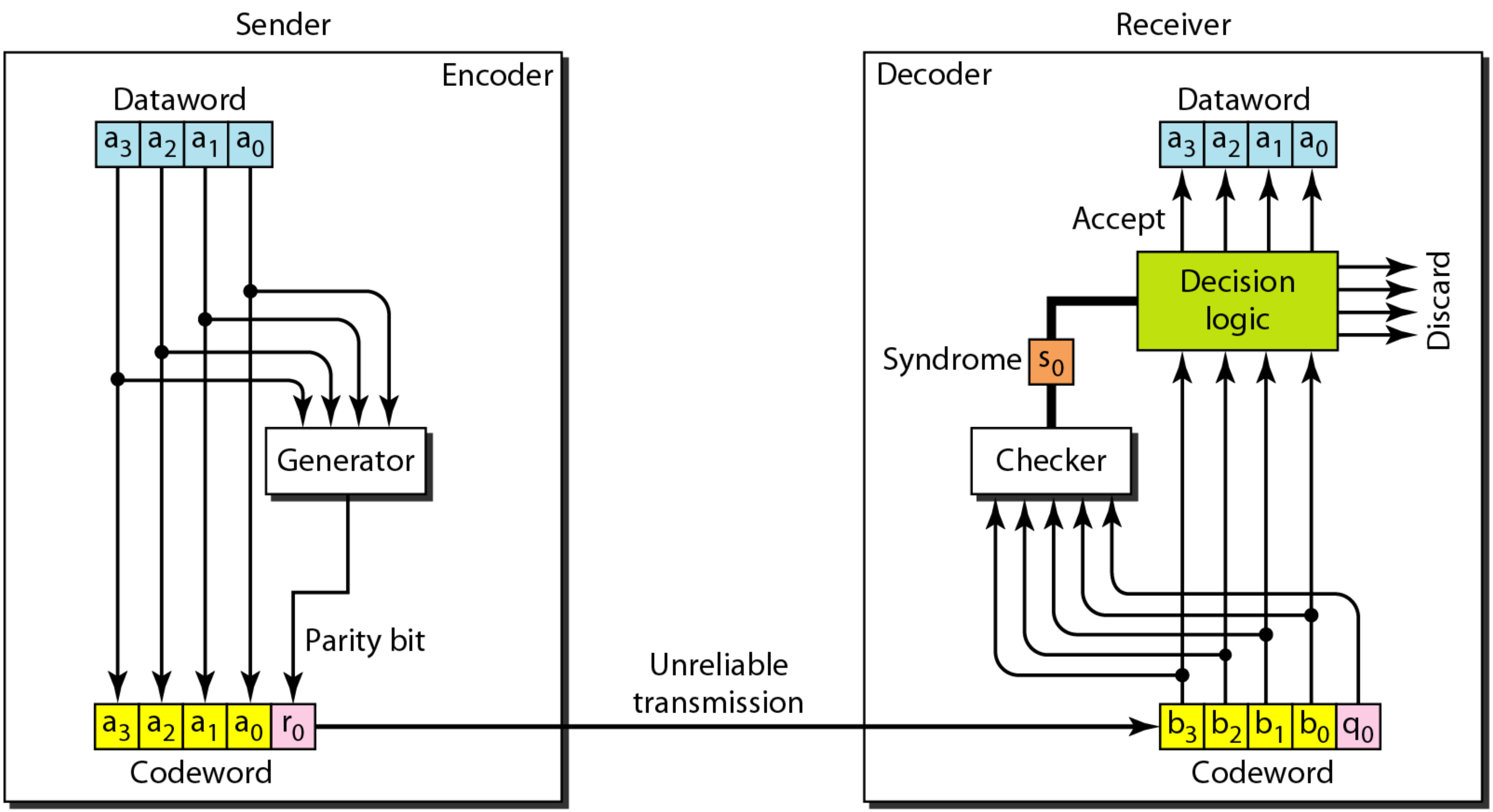
- A k-bit dataword is changed to an n-bit codeword where n=k+1.
- The extra bit is called parity bit
- The parity bit make the total number of 1s in the codeword even.
- The minimum hamming distance for this category is 2

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

Table 10.3 ***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

Figure 10.10 *Encoder and decoder for simple parity-check code*



Generator uses modulo-2

If syndrome is 0, no error

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.***

Hamming Code (Error Correcting Code)

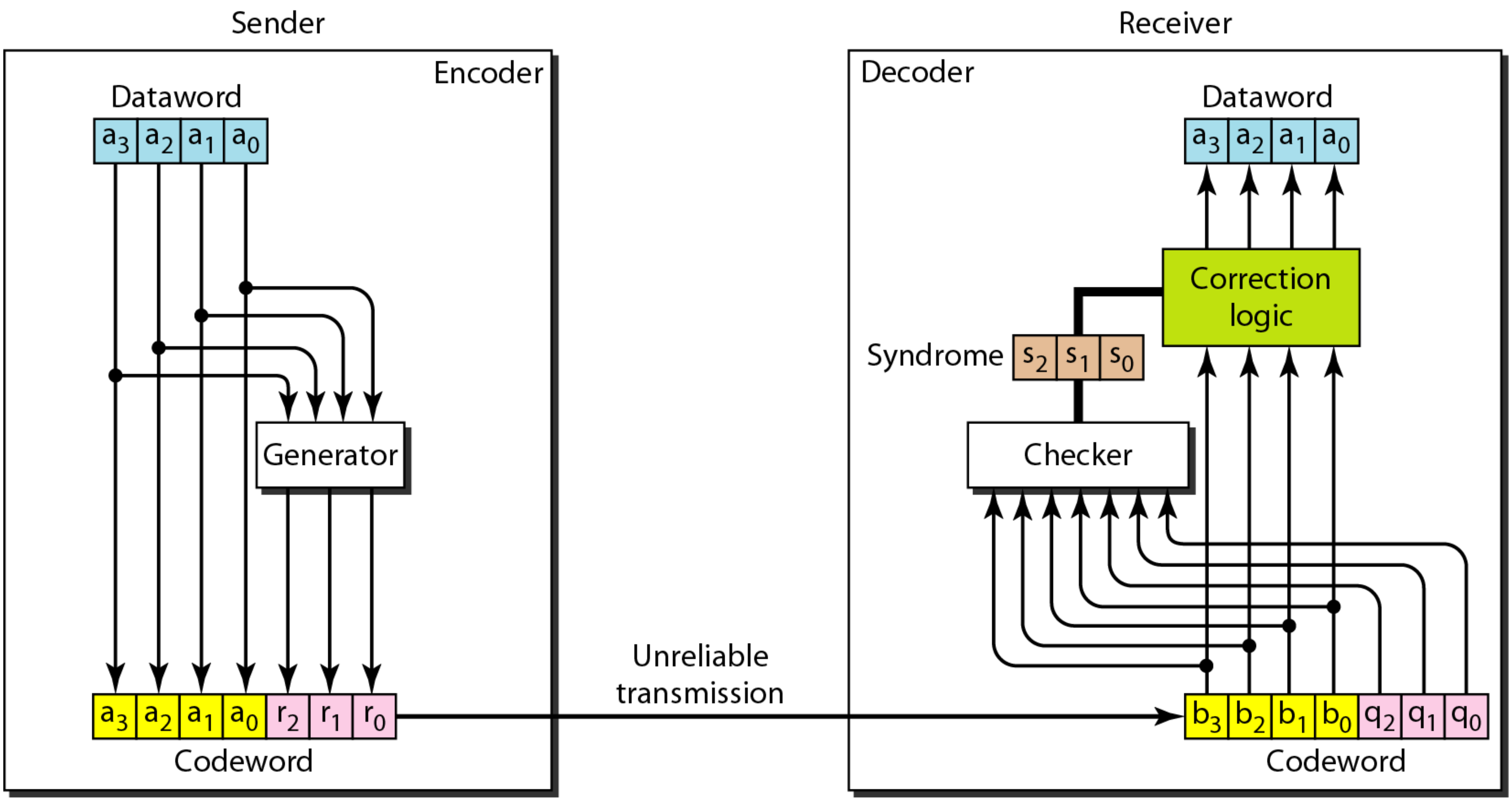
- If m is number of parity bits, n is number of codeword bits and k are dataword bits.

- $n=2^m-1$ and $k=n-m$
- eg., if $m=3$, then $n=7$ and $k=4$ and equivalent Hamming Code is C(7,4).

Table 10.4 ***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

Figure 10.12 *The structure of the encoder and decoder for a Hamming code*



- $r_0=a_2+a_1+a_0$ modulo-2

$r_1=a_3+a_2+a_1$ modulo-2

$r_2=a_3+a_1+a_0$ modulo-2

- $s_0=b_2+b_1+b_0+r_0$ modulo-2,

$s_1=b_3+b_2+b_1+r_1$ modulo-2

$s_2=b_3+b_1+b_0+r_2$ modulo-2

Logical decision made by the correction logic analyzer

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 received is 0011001, 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 codeword received is 0001000, syndrome is 101. After flipping b_0, we get 0000, the wrong dataword. This shows that our code cannot correct two errors.*

Example 10.14

We need a dataword of at least 7 bits. Calculate values of k and n that satisfy this requirement.

- ***If m is number of parity bits, n is number of codeword bits and k are dataword bits.***

Solution

We need to make $k = n - m$ greater than or equal to 7.

1. ***If we set $m = 3$, the result is $n = 2^3 - 1$ and $k = 7 - 3$, or 4, which is not acceptable.***
2. ***If we set $m = 4$, then $n = 2^4 - 1 = 15$ and $k = 15 - 4 = 11$, which satisfies the condition. So the code is***

$$\boxed{C(15, 11)}$$

10-4 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.***

Table 10.6 ***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

Figure 10.14 *CRC encoder and decoder*

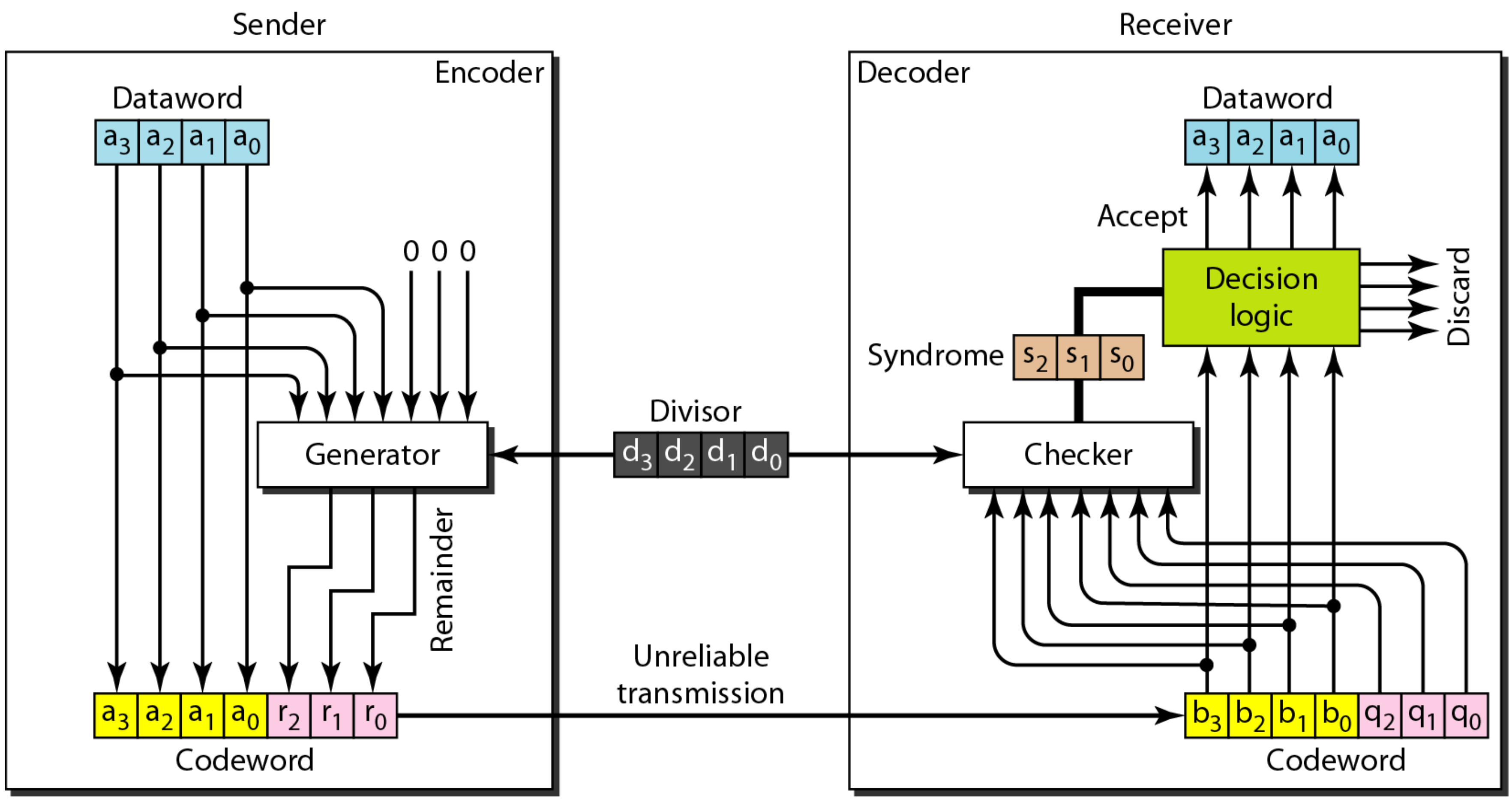


Figure 10.15 *Division in CRC encoder*

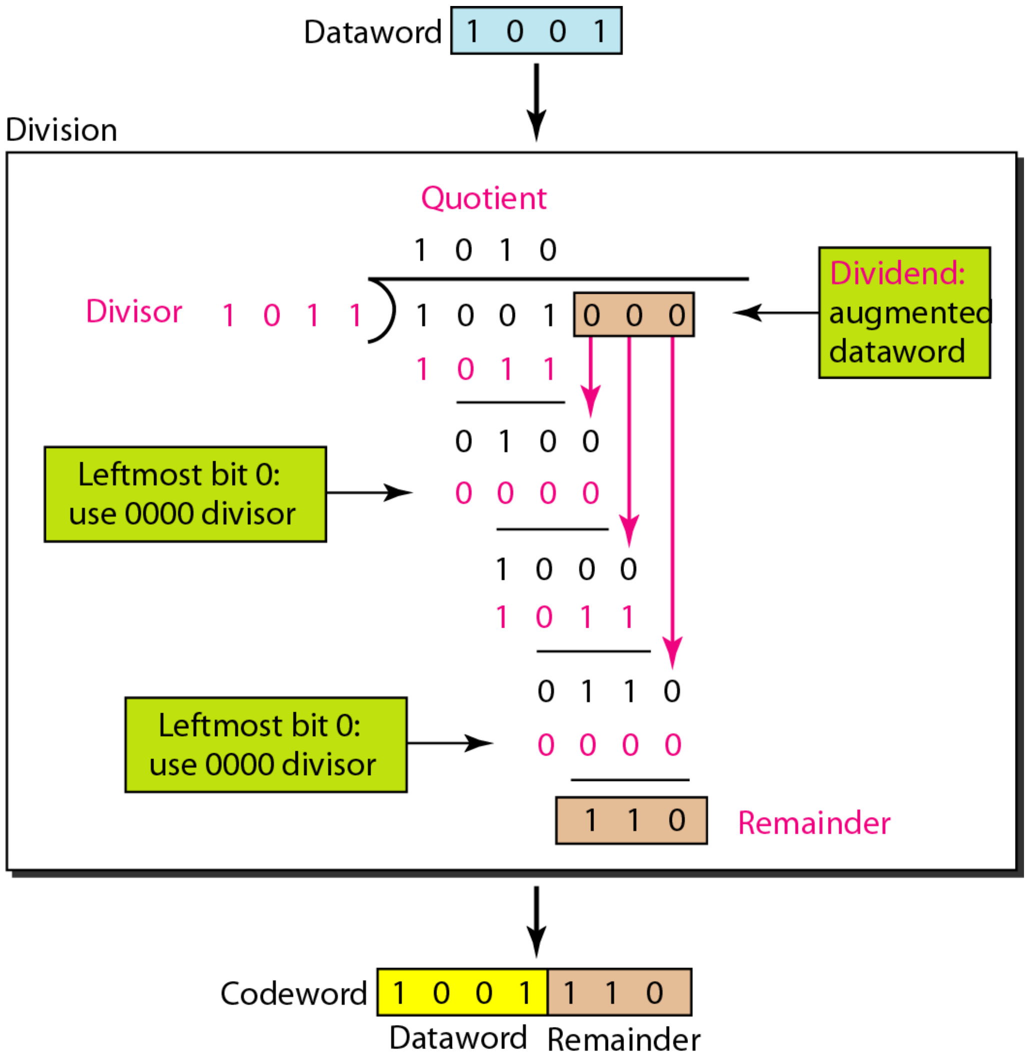


Figure 10.16 *Division in the CRC decoder for two cases*

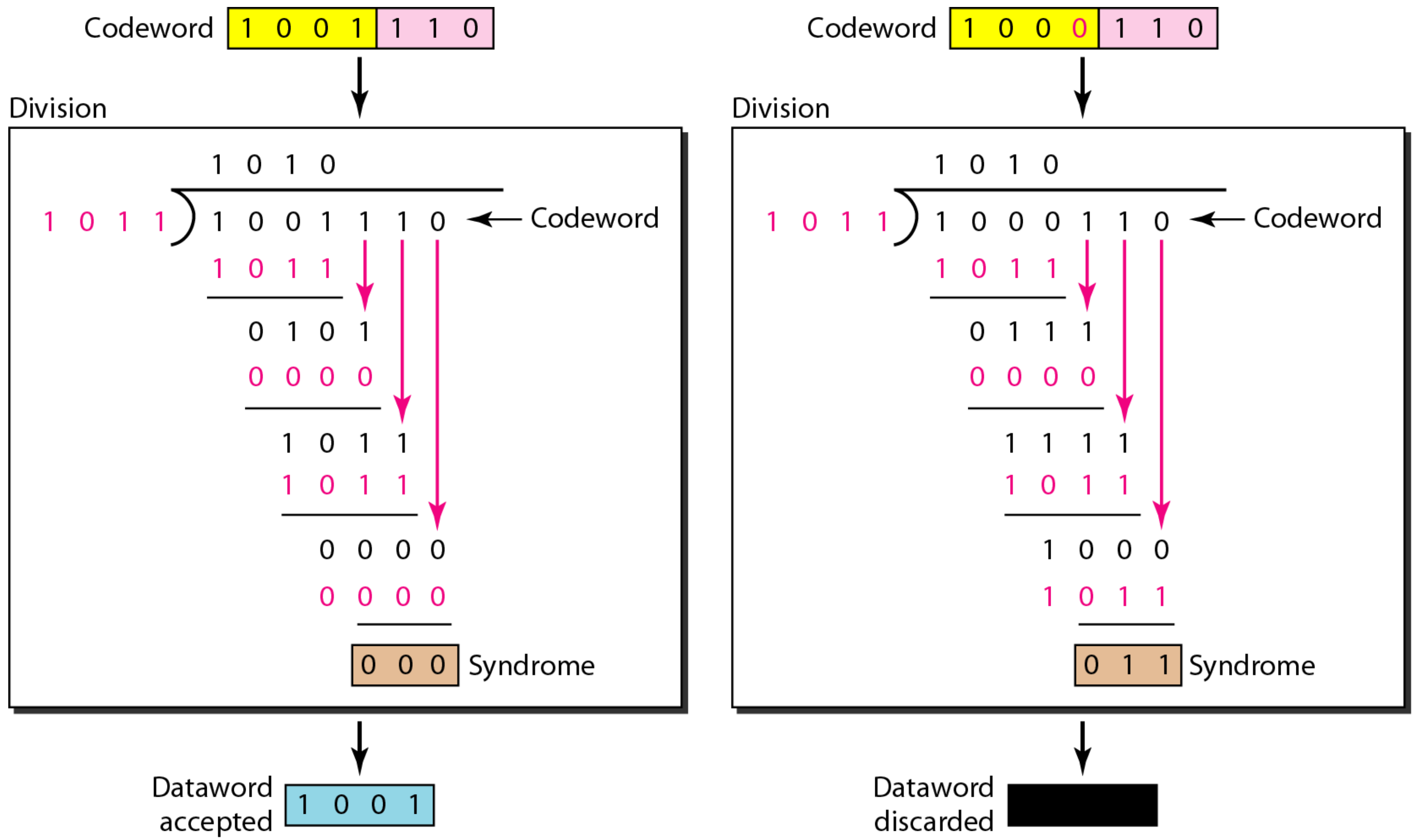
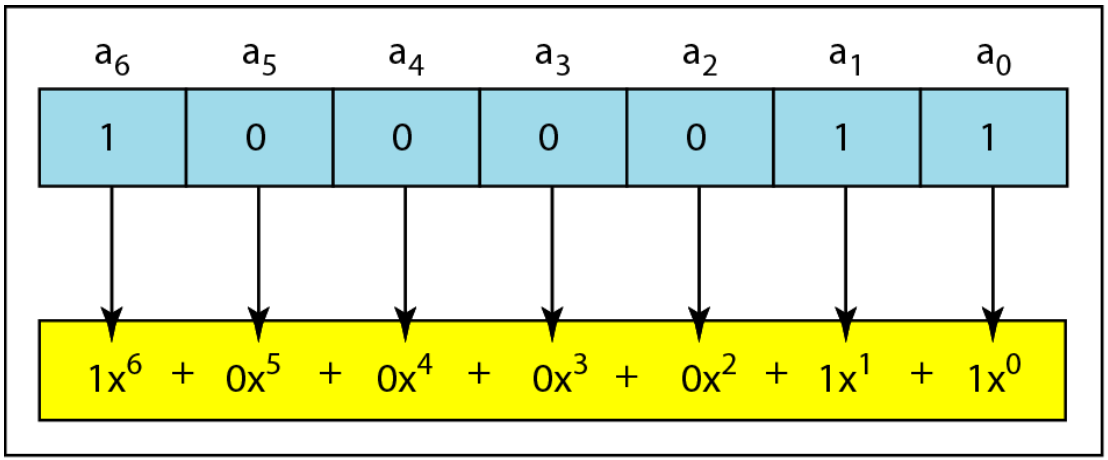
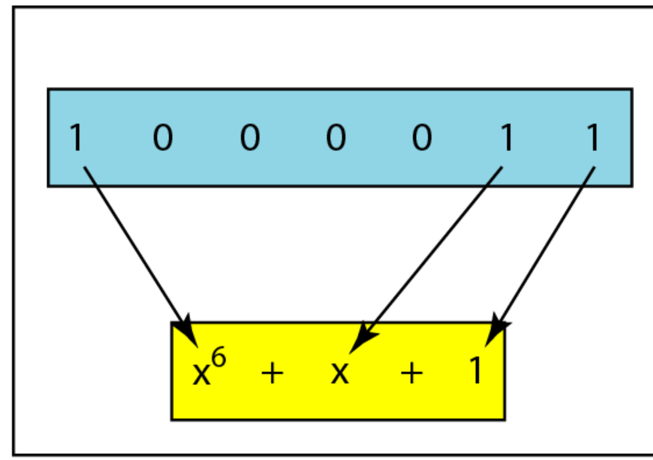


Figure 10.21 *A polynomial to represent a binary word*



a. Binary pattern and polynomial



b. Short form

Figure 10.22 *CRC division using polynomials*

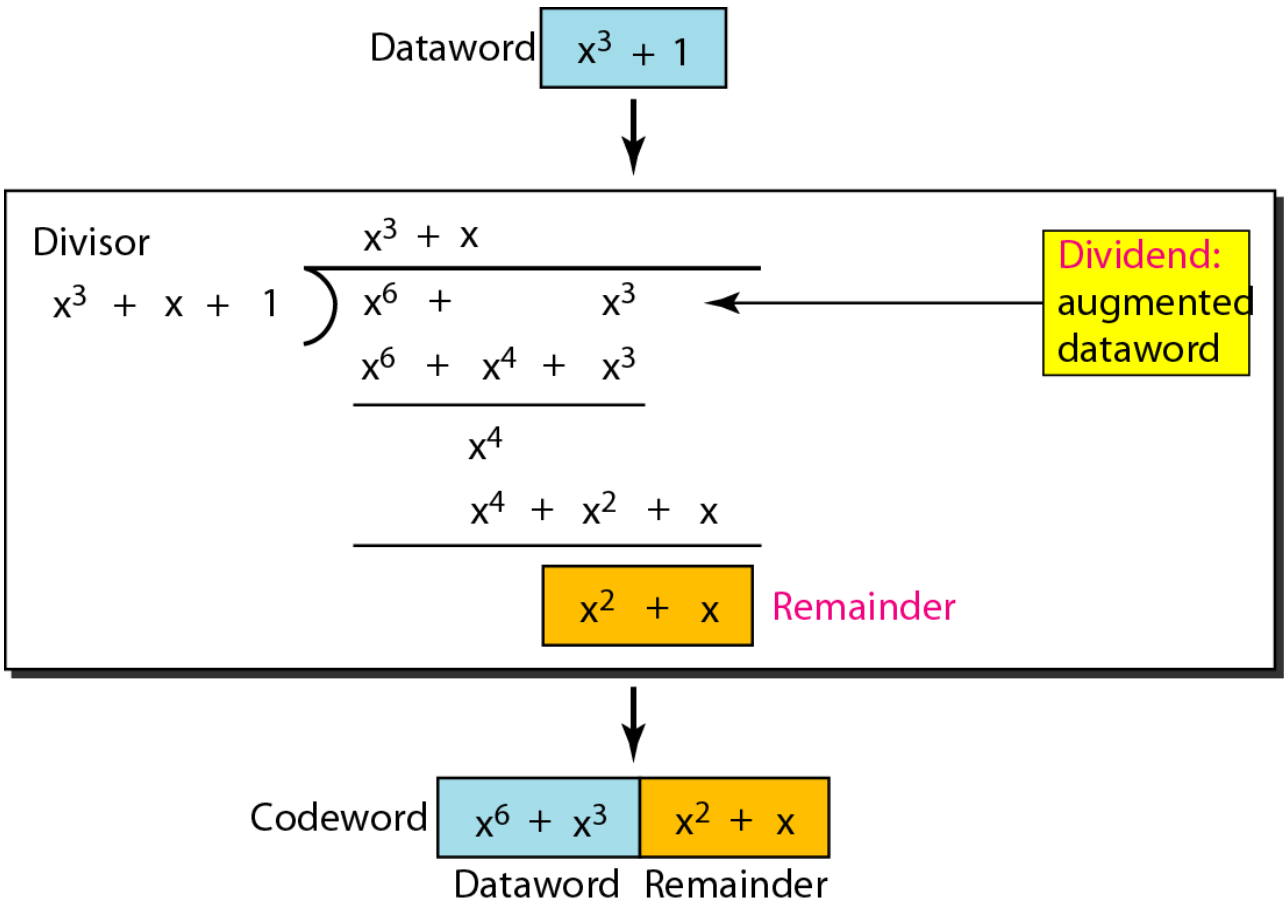


Table 10.7 ***Standard polynomials***

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

10-5 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. However, we briefly discuss it here to complete our discussion on error checking

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.

How can we represent the number 21 in one's complement arithmetic using only four bits?

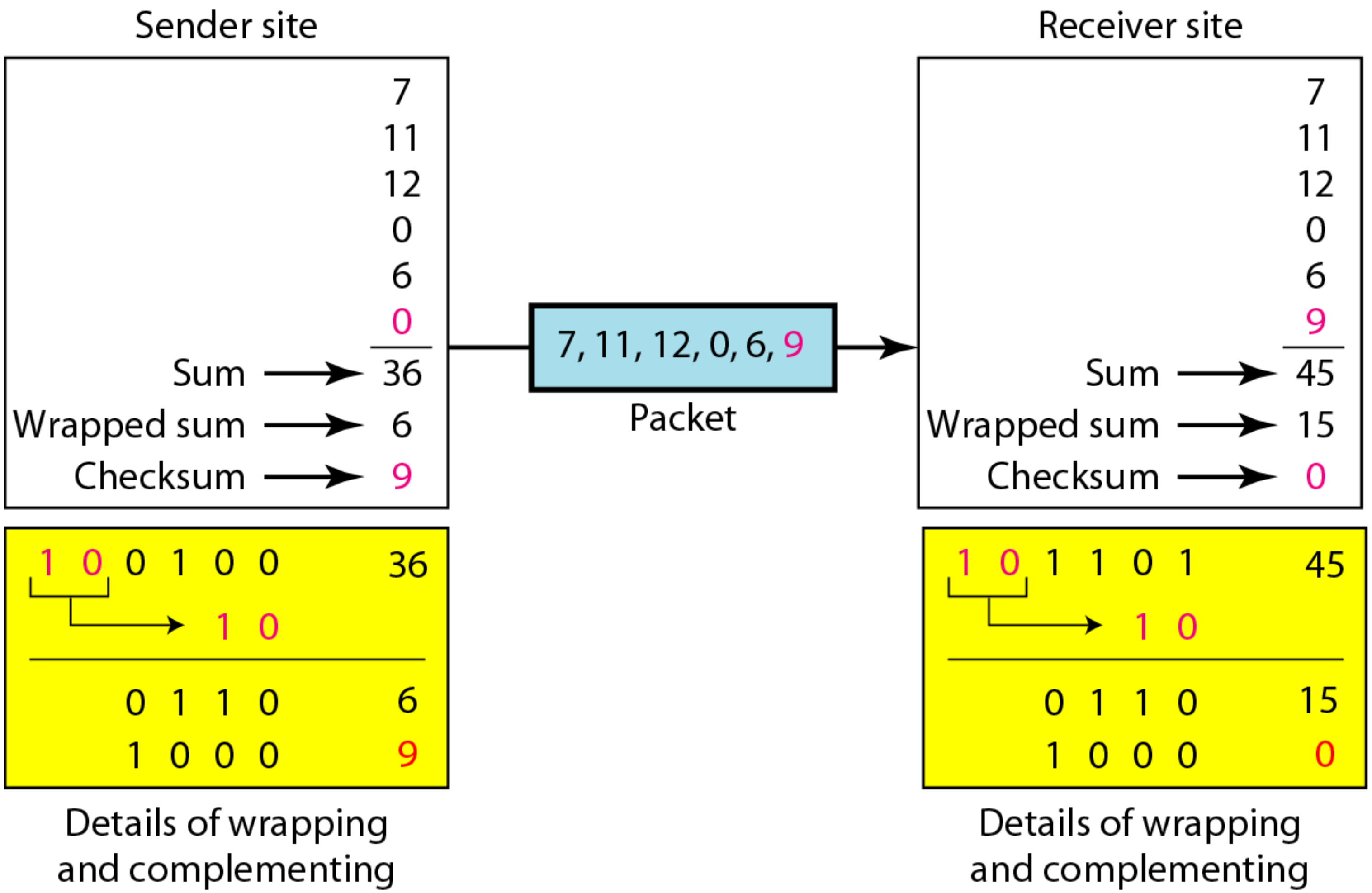
Solution

The number 21 in binary is 10101 (it needs five bits). We can wrap the leftmost bit and add it to the four rightmost bits. We have (0101 + 1) = 0110 or 6.

How can we represent the number -6 in one's complement arithmetic using only four bits?

Solution

In one's complement arithmetic, the negative or complement of a number is found by inverting all bits. Positive 6 is 0110; negative 6 is 1001. If we consider only unsigned numbers, this is 9. In other words, the complement of 6 is 9. Another way to find the complement of a number in one's complement arithmetic is to subtract the number from $2^n - 1$ (16 − 1 in this case).


Sender site
7
11
12
0
6
0
Sum 36
Wrapped sum 6
Checksum 9
1 0 0 1 0 0 36
1 0
0 1 1 0 6
1 0 0 0 9
Details of wrapping and complementing
7, 11, 12, 0, 6, 9
Packet
Receiver site
7
11
12
0
6
9
Sum 45
Wrapped sum 15
Checksum 0
1 0 1 1 0 1 45
1 0
0 1 1 0 15
1 0 0 0 0
Details of wrapping and complementing

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.

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.**

1	0	1	3	Carries
4	6	6	F	(Fo)
7	2	6	7	(ro)
7	5	7	A	(uz)
6	1	6	E	(an)
0	0	0	0	Checksum (initial)
8	F	C	6	Sum (partial)
			1	
8	F	C	7	Sum
7	0	3	8	Checksum (to send)

a. Checksum at the sender site

1	0	1	3	Carries
4	6	6	F	(Fo)
7	2	6	7	(ro)
7	5	7	A	(uz)
6	1	6	E	(an)
7	0	3	8	Checksum (received)
F	F	F	E	Sum (partial)
			1	
8	F	C	7	Sum
0	0	0	0	Checksum (new)

a. Checksum at the receiver site