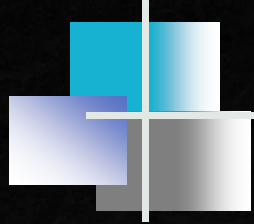




ERROR DETECTION

Double Tap to Add Subtitle



Note

Data can be corrupted
during transmission.

Some applications require that
errors be detected and corrected.

10-1 INTRODUCTION

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

Topics discussed in this section:

Types of Errors

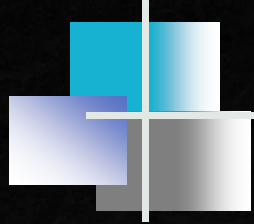
Redundancy

Detection Versus Correction

Forward Error Correction Versus Retransmission

Coding

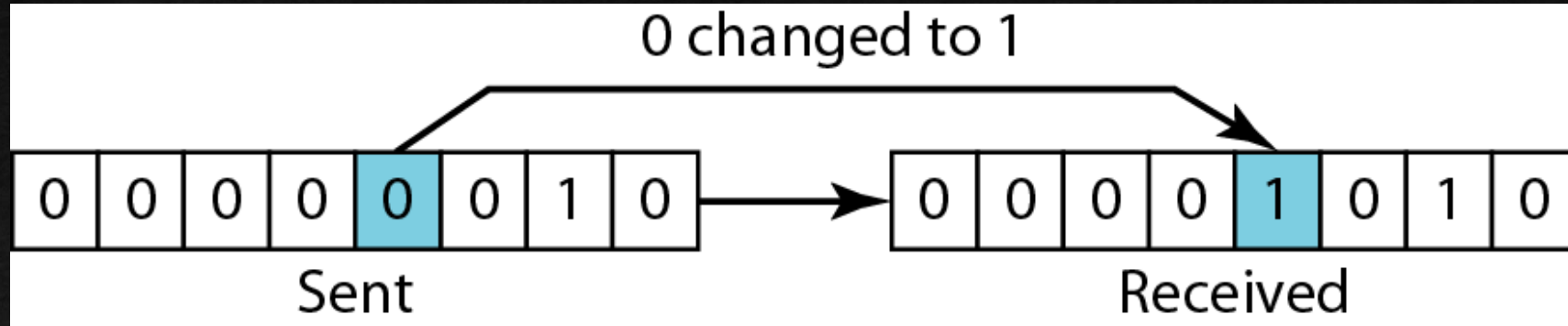
Modular Arithmetic

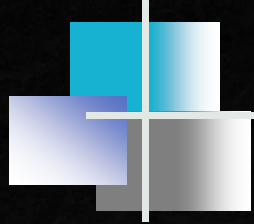


Note

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

Figure 10.1 *Single-bit error*

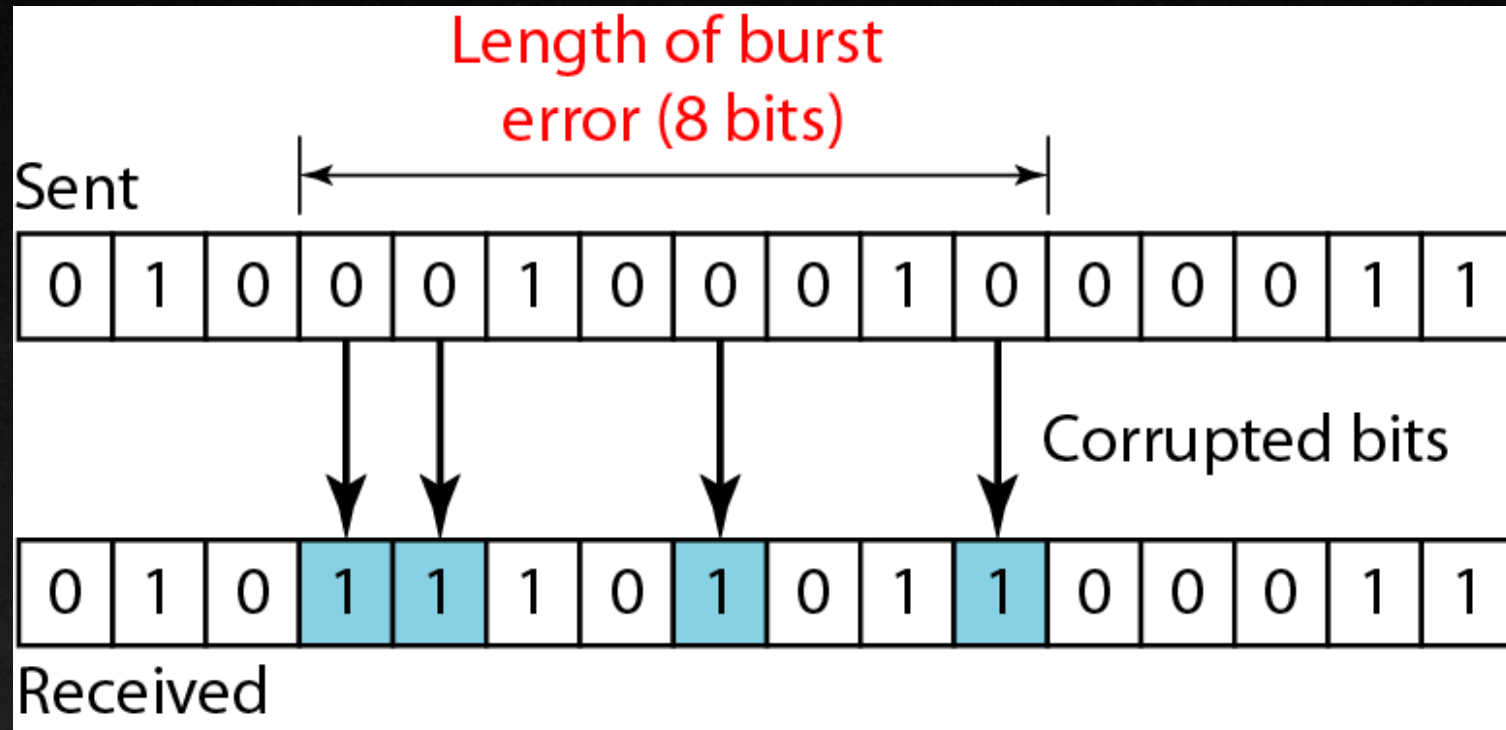


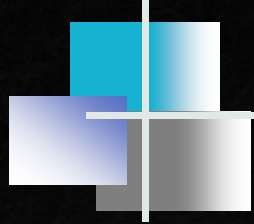


Note

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

Figure 10.2 *Burst error of length 8*

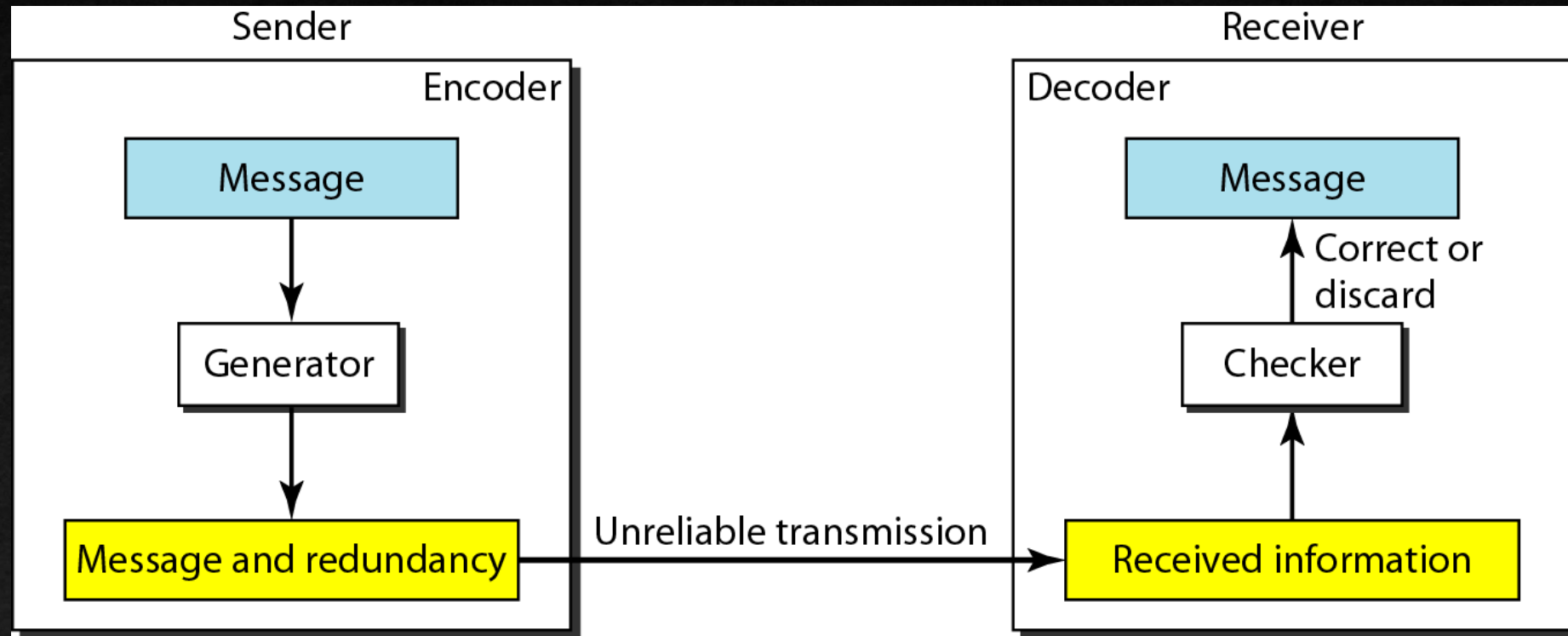


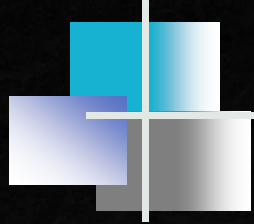


Note

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

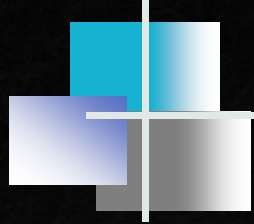
Figure 10.3 *The structure of encoder and decoder*





Note

In this book, we concentrate on block codes; we leave convolution codes to advanced texts.



Note

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

Figure 10.4 *XORing of two single bits or two words*

$$0 \oplus 0 = 0$$

$$1 \oplus 1 = 0$$

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

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

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

$$\begin{array}{rcccccc} & 1 & 0 & 1 & 1 & 0 \\ \oplus & 1 & 1 & 1 & 0 & 0 \\ \hline & 0 & 1 & 0 & 1 & 0 \end{array}$$

c. Result of XORing two patterns

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**.*

Topics discussed in this section:

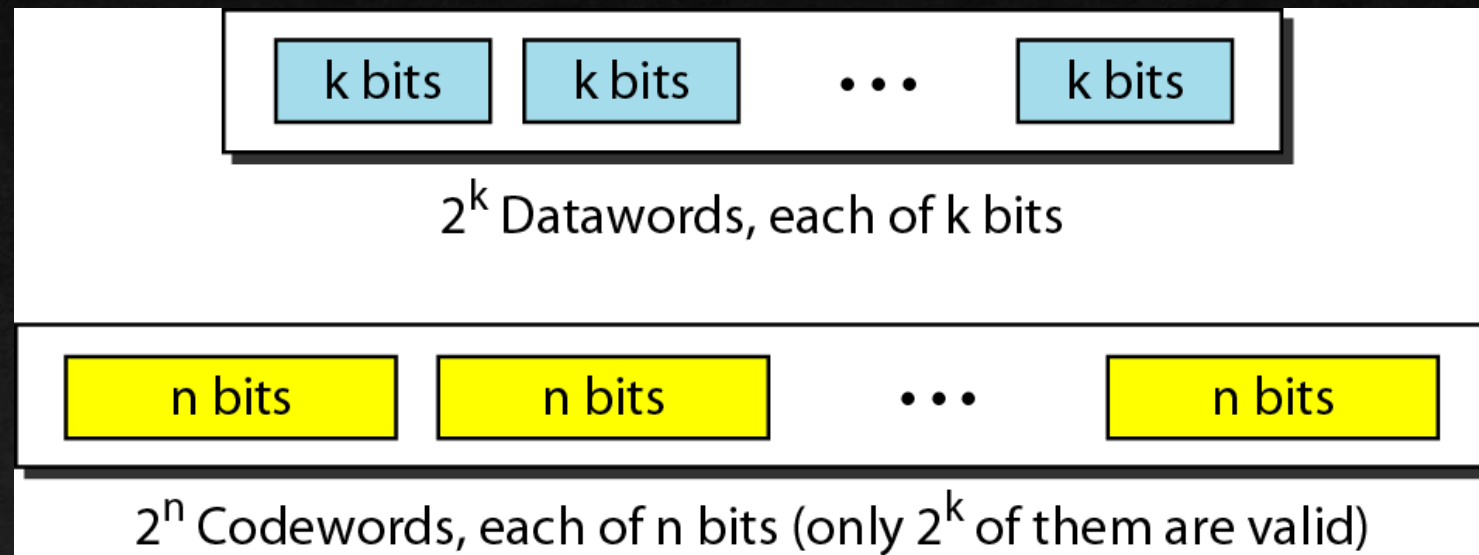
Error Detection

Error Correction

Hamming Distance

Minimum Hamming Distance

Figure 10.5 *Datawords and codewords in block coding*

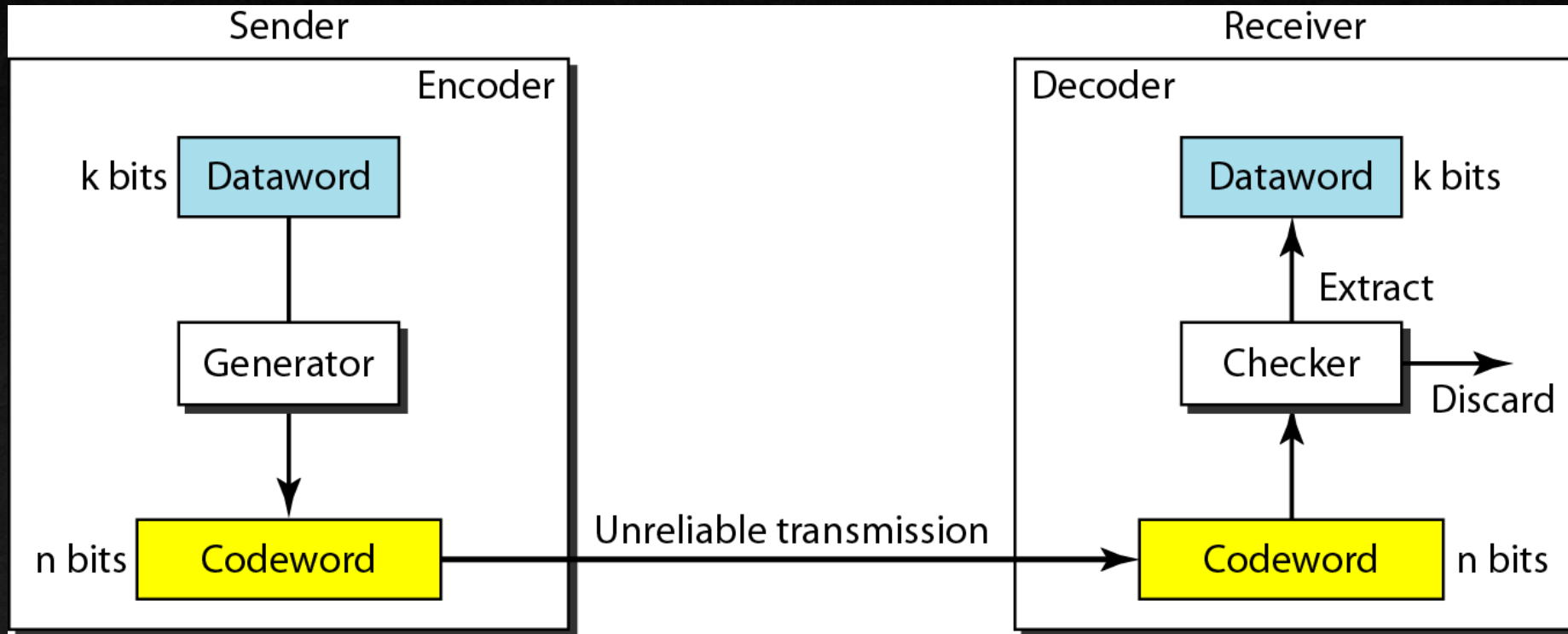


The 4B/5B block coding discussed in Chapter 4 is a good example of this type of coding. In this coding scheme, $k = 4$ and $n = 5$. As we saw, we have $2^k = 16$ datawords and $2^n = 32$ codewords. We saw that 16 out of 32 codewords are used for message transfer and the rest are either used for other purposes or unused.

Error Detection

- Enough redundancy is added to detect an error.
- The receiver knows an error occurred but does not know which bit(s) is(are) in error.
- Has less overhead than error correction.

Figure 10.6 *Process of error detection in block coding*



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.

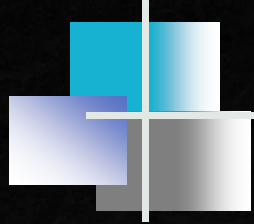
Assume the sender encodes the dataword 01 as 011 and sends it to the receiver. Consider the following cases:

- 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.*

Table 10.1 *A code for error detection (Example 10.2)*

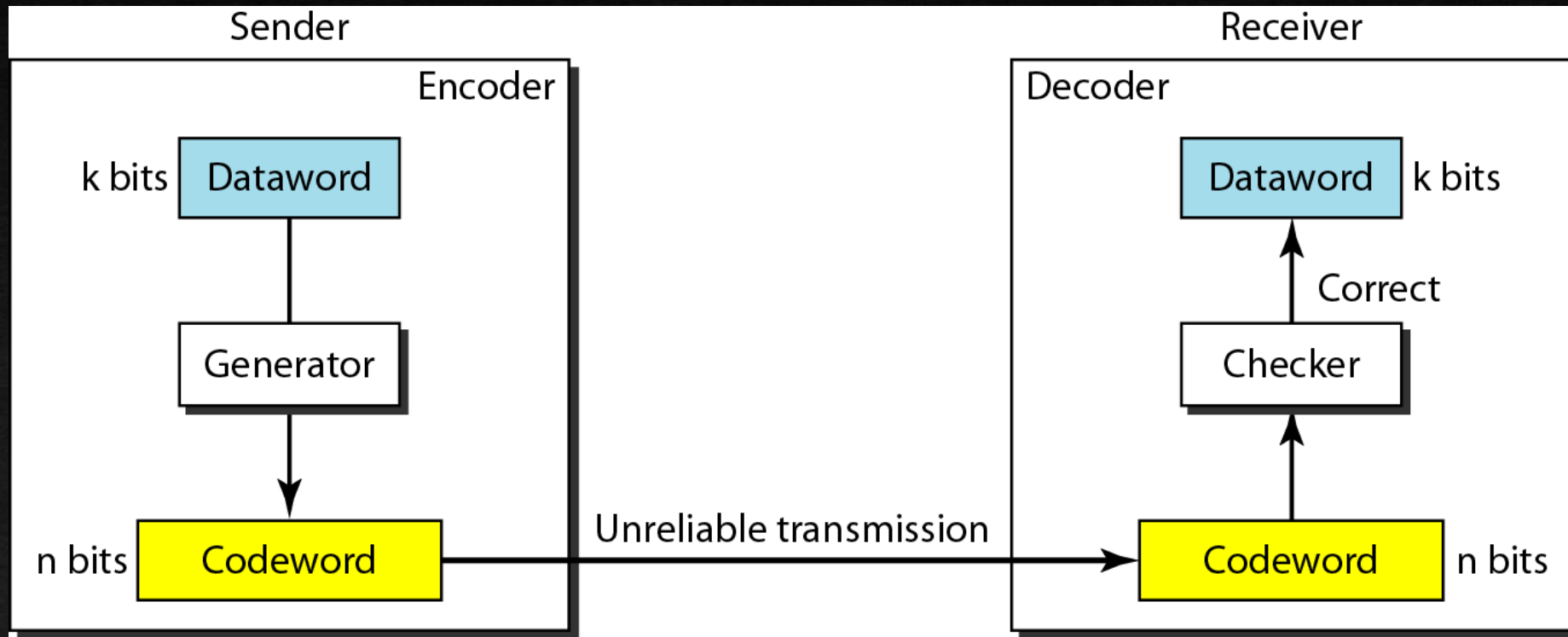
<i>Datawords</i>	<i>Codewords</i>
00	000
01	011
10	101
11	110



Note

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*



Let us add more redundant bits to Example 10.2 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 10.2 shows the datawords and 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.

Table 10.2 *A code for error correction (Example 10.3)*

<i>Dataword</i>	<i>Codeword</i>
00	00000
01	01011
10	10101
11	11110