CSE 311: Computer Networks

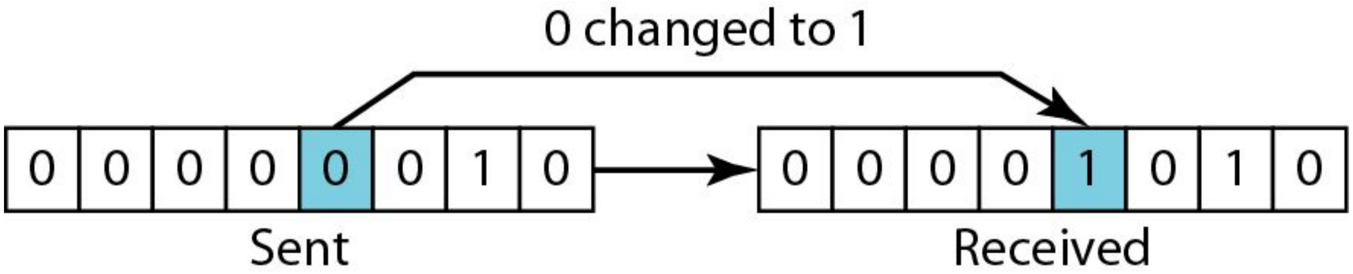
Lecture 5: Error Detection and Correction

Introduction

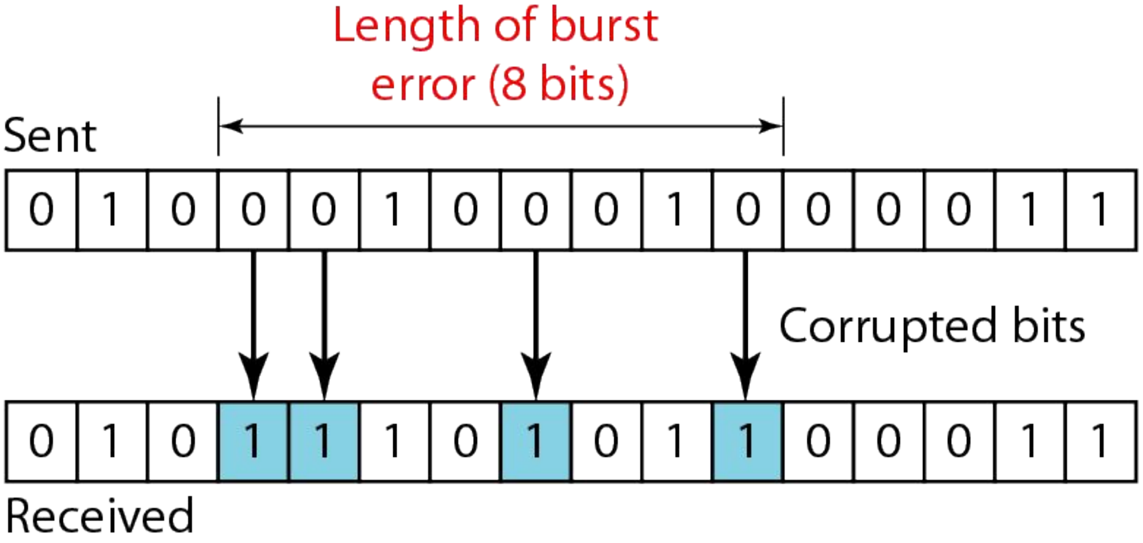
- Data can be corrupted during transmission.
- Some applications require that errors be detected and corrected.
- Types of errors
- Redundancy
- Detection Vs. Correction
- Coding

Types of Errors

- Single-bit errors

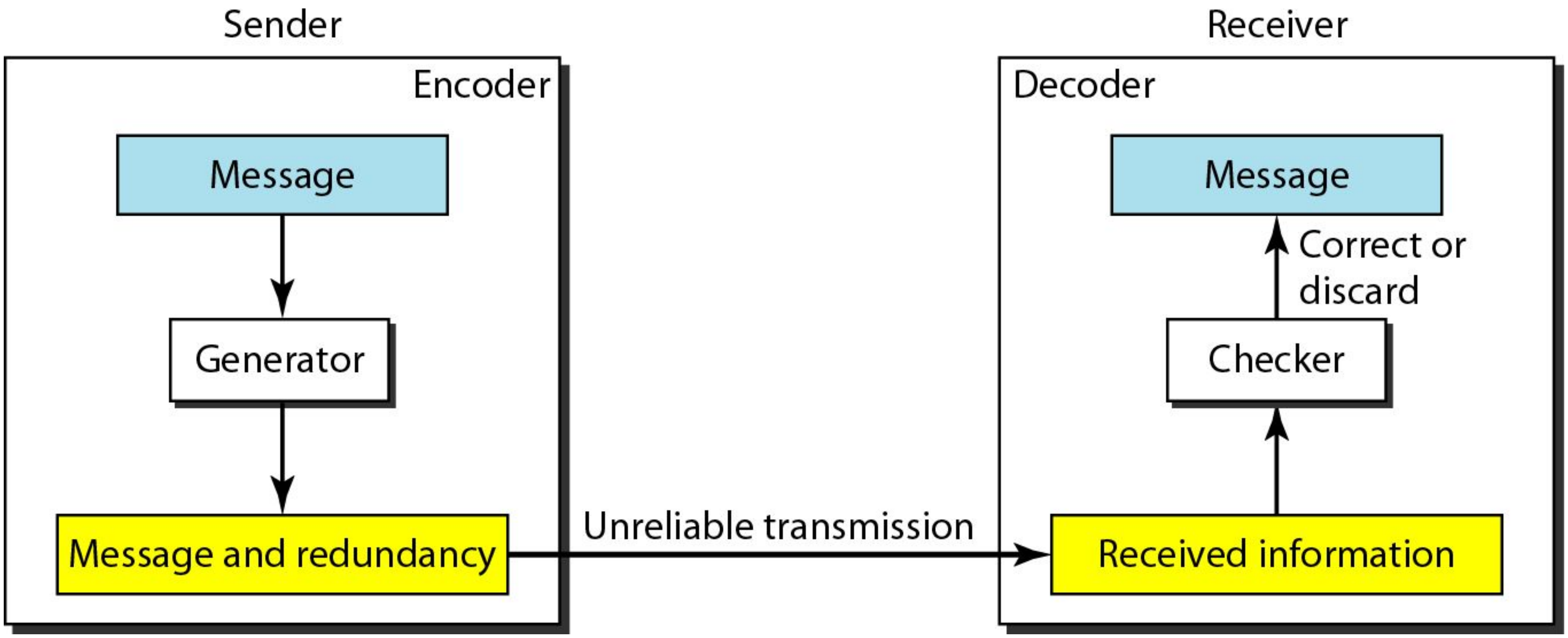


- Burst errors



Redundancy

- To detect or correct errors, redundant bits of data must be added

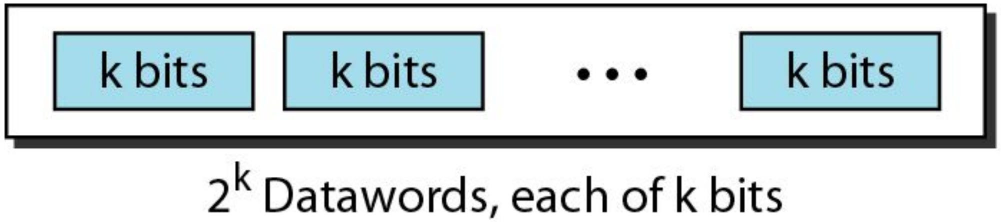


Block Codes

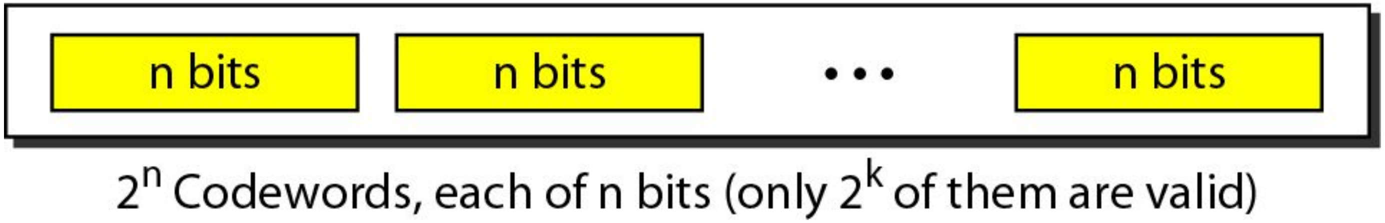
- Process of adding redundancy for error detection or correction
- Two types:
  - Block codes
    - Divides the data to be sent into a set of blocks
    - Extra information attached to each block
  - Convolutional codes
    - Treats data as a series of bits, and computes a code over a continuous series
    - The code computed for a set of bits depends on the current and previous input

Block Coding

- Message is divided into *k*-bit blocks
  - Known as *datawords*
- *r* redundant bits are added
  - Blocks become *n* = *k*+*r* bits
  - Known as *codewords*

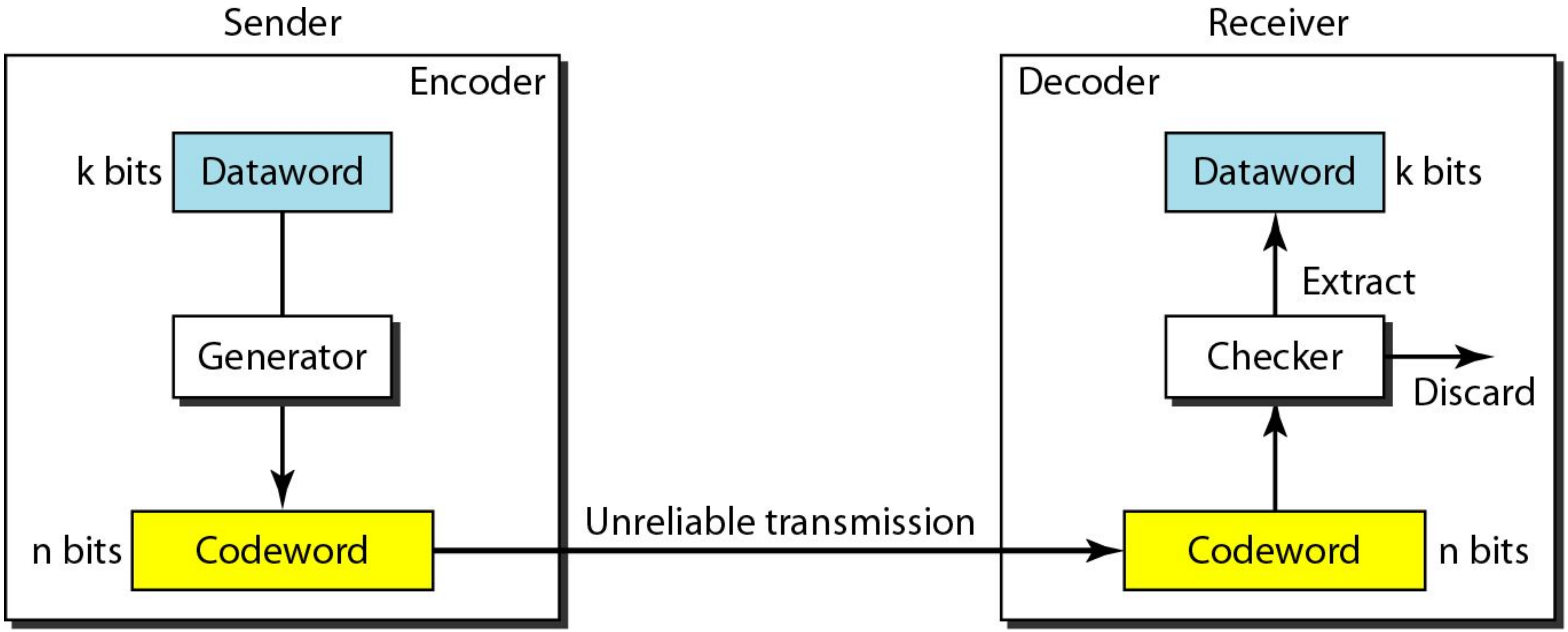


2^k Datawords, each of k bits



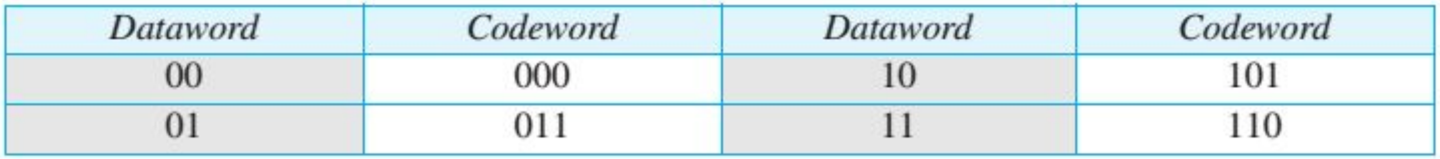
2^n Codewords, each of n bits (only 2^k of them are valid)

Error Detection in Block Coding



Example 10.1

- *Let us assume that k = 2 and n = 3. Table 10.1 shows the list of datawords and codewords.*



Dataword	Codeword	Dataword	Codeword
00	000	10	101
01	011	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 <u>00.Two corrupted bits have made the error undetectable.</u>*

Note

- An error-detecting code can detect only the types of errors for which it is designed; Other types of errors may remain undetected.
- There is no way to detect every possible error.

Hamming Distance

- One of the central concepts in coding for error control is the idea of the Hamming distance.

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

- The Hamming distance between two words *x and y is*

d(x, y).

- The Hamming distance can easily be found using XOR operation (⊕) on the two words and count the number of 1s in the result.

Example

- d(01, 00) = ?
- d(11, 00) = ?
- d(010, 100) = ?
- d(0011, 1000) = ?

Minimum Hamming Distance

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 following codebook

00000
01011
10101
11110

Minimum Hamming Distance

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: 10.5

The minimum Hamming distance for our first code scheme (Table 10.1) is 2.

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.

Common Detection Methods

- Parity check
- Cyclic Redundancy Check
- Checksum

Parity Check

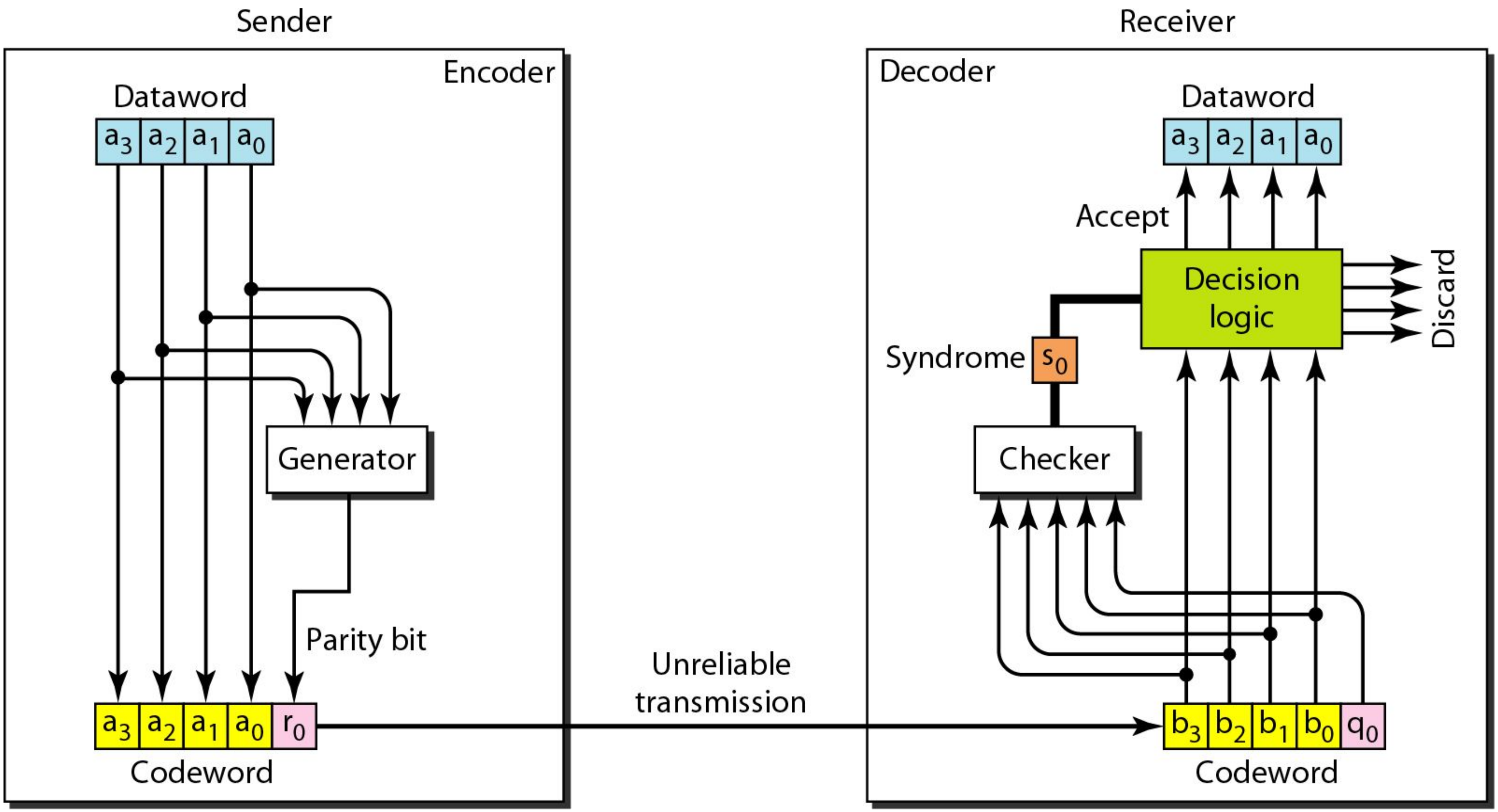
- Most common, least complex
- Single bit is added to a block
- Two schemes:
  - Even parity – Maintain even number of 1s
    - E.g., 1011 → 1011**1**
  - Odd parity – Maintain odd number of 1s
    - E.g., 1011 → 1011**0**

Even Parity Check

- 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.

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

Parity-Check: Encoding/Decoding



Example 10.12

- 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.

Note that although none of the dataword bits are corrupted, no dataword is created because the code is not sophisticated enough to show the position of the corrupted bit.

Example 10.7

- 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. The simple parity-check decoder cannot detect an even number of errors. The errors cancel each other out and give the syndrome a value of O.

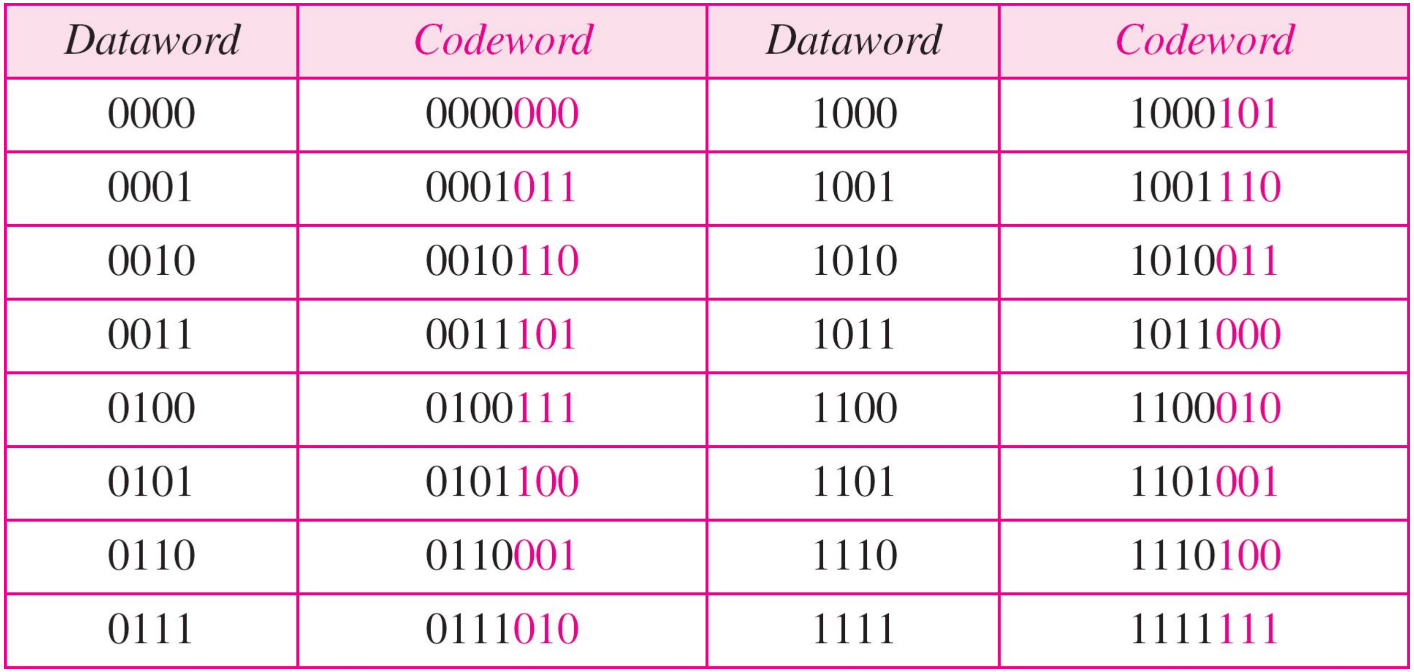
- 5. Three bits—a_3, a_2, and a_1—are changed by errors. The received codeword is 01011. The syndrome is 1. The dataword is not created. This shows that the simple parity check, guaranteed to detect one single error, can also find any odd number of errors.

Cyclic Redundancy Check

- 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 Redundancy Check - CRCs are so called because the check i.e. data verification code is a redundancy (it adds zero information) and the algorithm is based on cyclic codes.
- The CRC is a very powerful but easily implemented technique to obtain data reliability.
- It is a non-secure hash function designed to detect accidental changes to raw computer data.
- It is commonly used in digital networks and storage devices such as hard disk drives.

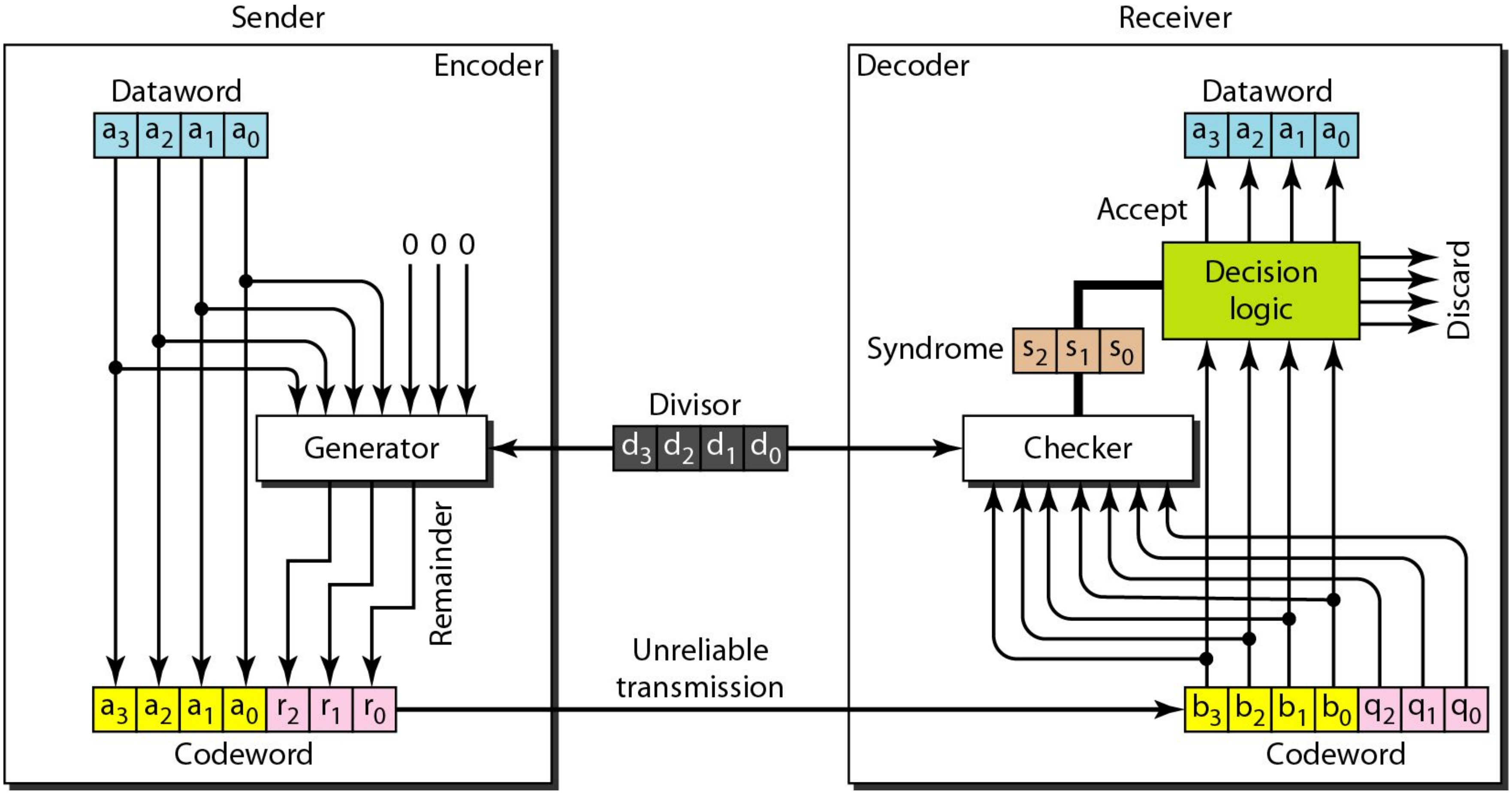
Table 10.3

- *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/Decoder



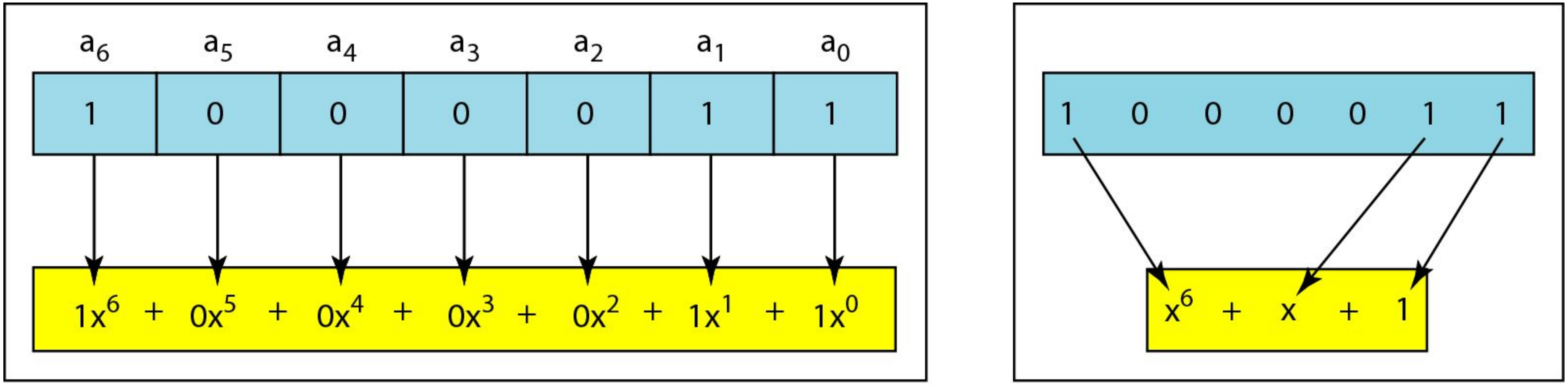
Advantages of CRC

- Extreme error detection capabilities: CRCs are particularly good at detecting common errors caused by noise in transmission channels.
- Little overhead.
- Ease of implementation: Simple to implement in binary hardware.
- Easy to analyze mathematically.

Polynomials

- A better way to understand cyclic codes and how they can be analyzed is to represent them as polynomials
- We can use a polynomial to represent a binary word.
- 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.
- Each bit from right to left is mapped onto a power term.
- The rightmost bit represents the “0” power term. The bit next to it is the “1” power term, etc.
- If the bit is of value zero, the power term is deleted from the expression.

A polynomial to represent a binary



a_6 a_5 a_4 a_3 a_2 a_1 a_0
1 0 0 0 0 1 1
$1x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 1x^1 + 1x^0$
1 0 0 0 0 1 1
$x^6 + x + 1$

a. Binary pattern and polynomial

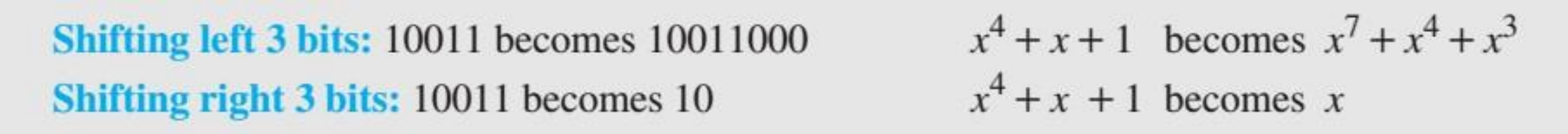
b. Short form

Adding, Subtracting and multiplying Polynomials

- First, addition and subtraction are the same. Second, adding or subtracting is done by combining terms and deleting pairs of identical terms.
- For example, adding $x^5+ x^4+ x^2$ and $x^6+ x^4+ x^2$ gives just $x^6+ x^5$. The terms x^4 and x^2 are deleted.
- Note that if we add three polynomials and we get x^2 three times, we delete a pair of them and keep the third.
- Multiplying is adding the powers; For Example: $x^3 \times x^4 = x^7$

Shifting

- Shifting to the Left means: adding extra 0s as rightmost bits
- Shifting to the Right means: deleting some rightmost bits

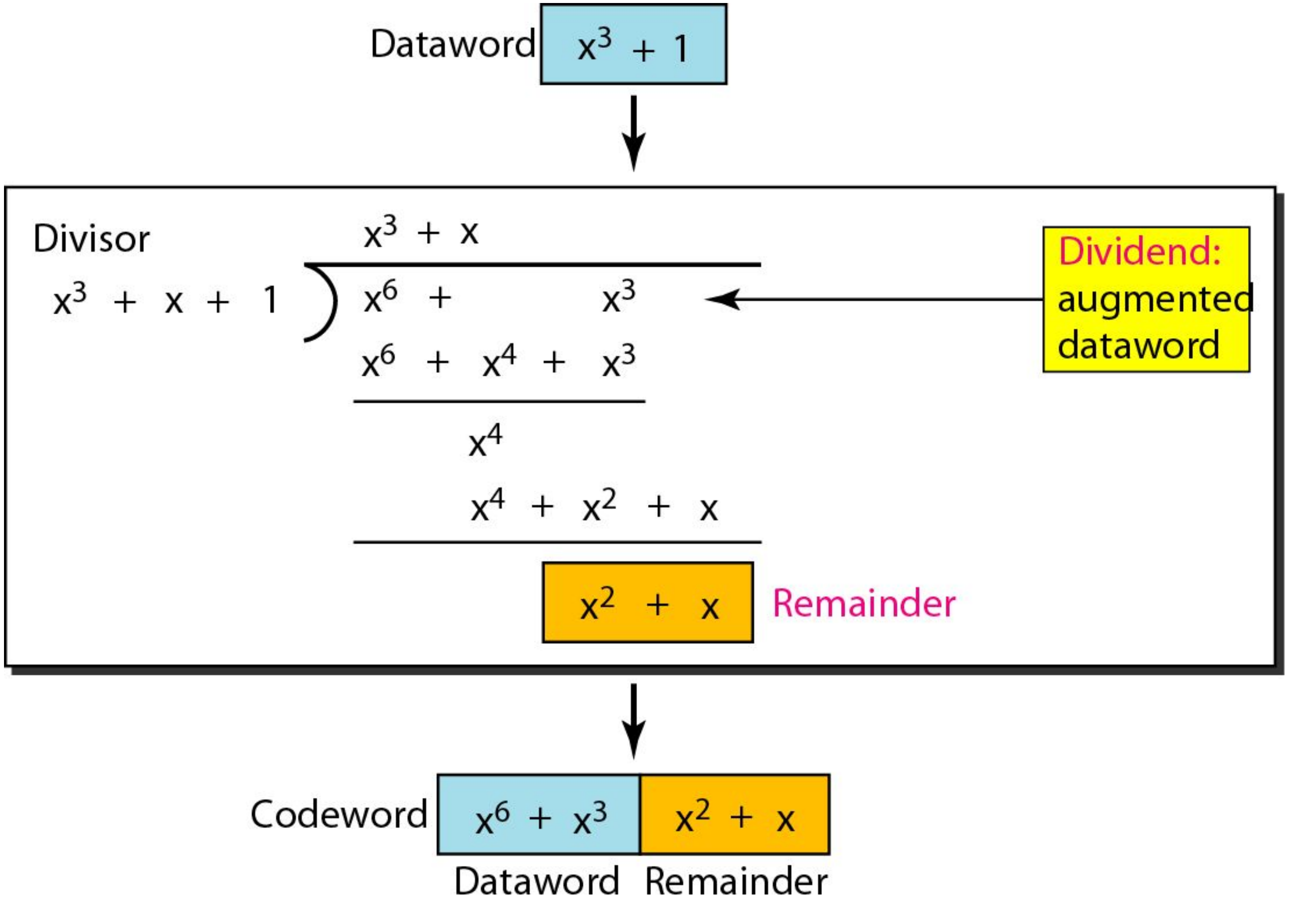


Shifting left 3 bits: 10011 becomes 10011000 $x^4 + x + 1$ becomes $x^7 + x^4 + x^3$

Shifting right 3 bits: 10011 becomes 10 $x^4 + x + 1$ becomes x

- Note that, we do not have negative powers in the polynomial representation

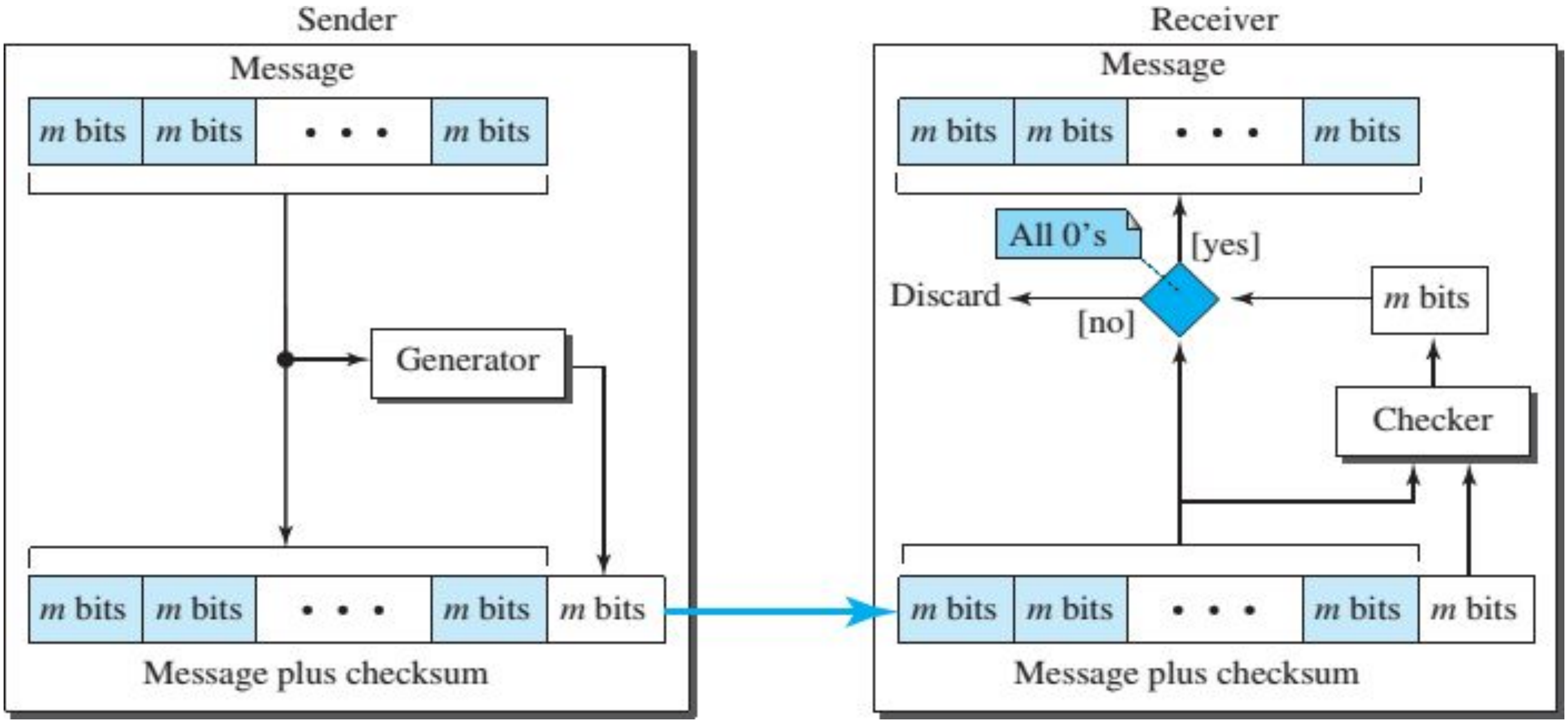
CRC division using polynomials



Checksum

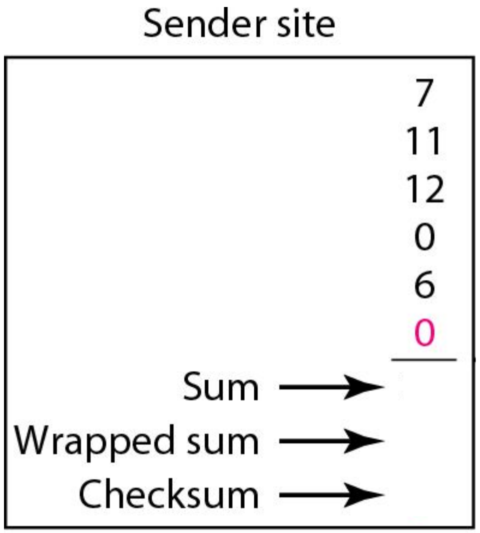
- Checksum is an error-detecting technique that can be applied to a message of any length.
- In the Internet, the checksum technique is mostly used at the
  network and transport layer rather than the data-link layer.
- At the source, the message is first divided into m-bit units.
- The generator then creates an extra m-bit unit called the checksum,
  which is sent with the message.
- At the destination, the checker creates a new checksum from the combination of the message and sent checksum.
- If the new checksum is all 0s, the message is accepted; otherwise,
  the message is discarded.

Checksum


Sender
Message
m bits
m bits
• • •
m bits
Generator
m bits
m bits
• • •
m bits
m bits
Message plus checksum
Receiver
Message
m bits
m bits
• • •
m bits
All 0's
[yes]
Discard
[no]
m bits
Checker
m bits
m bits
• • •
m bits
m bits
Message plus checksum

Checksum

- Suppose our data is a list of five 4-bit numbers: 7, 11, 12, 0, 6



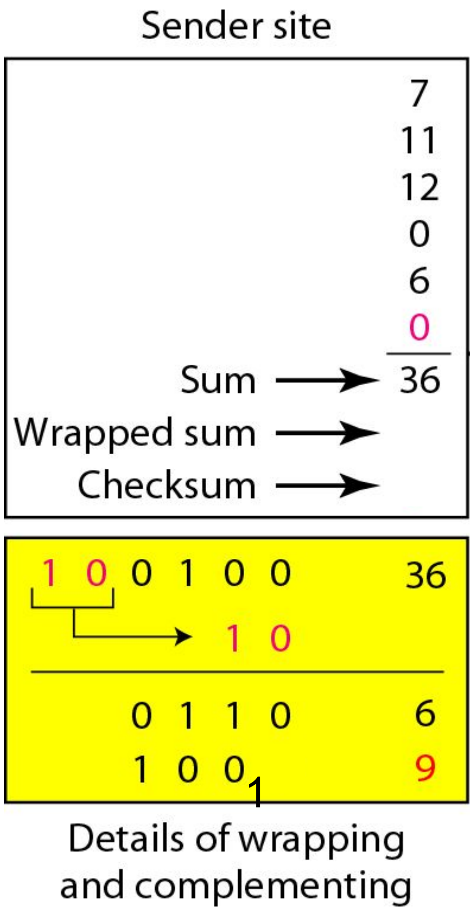
Sender site

7
11
12
0
6
0

Sum →
Wrapped sum →
Checksum →

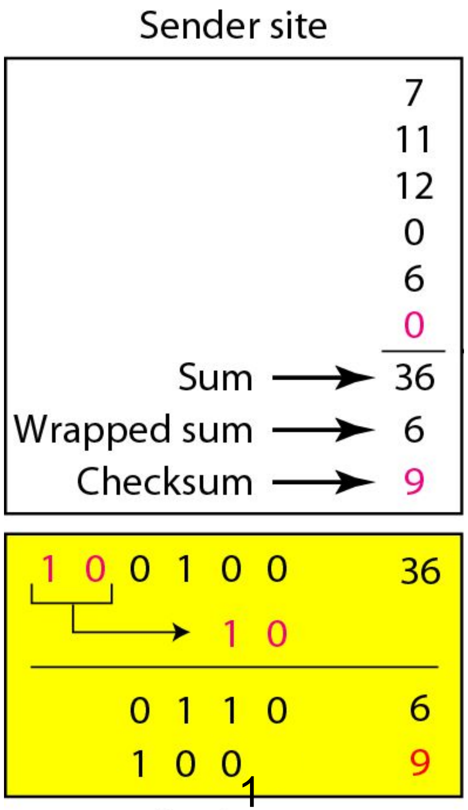
Checksum

- Suppose our data is a list of five 4-bit numbers: 7, 11, 12, 0, 6



Checksum

- Suppose our data is a list of five 4-bit numbers: 7, 11, 12, 0, 6



Sender site

7
11
12
0
6
0

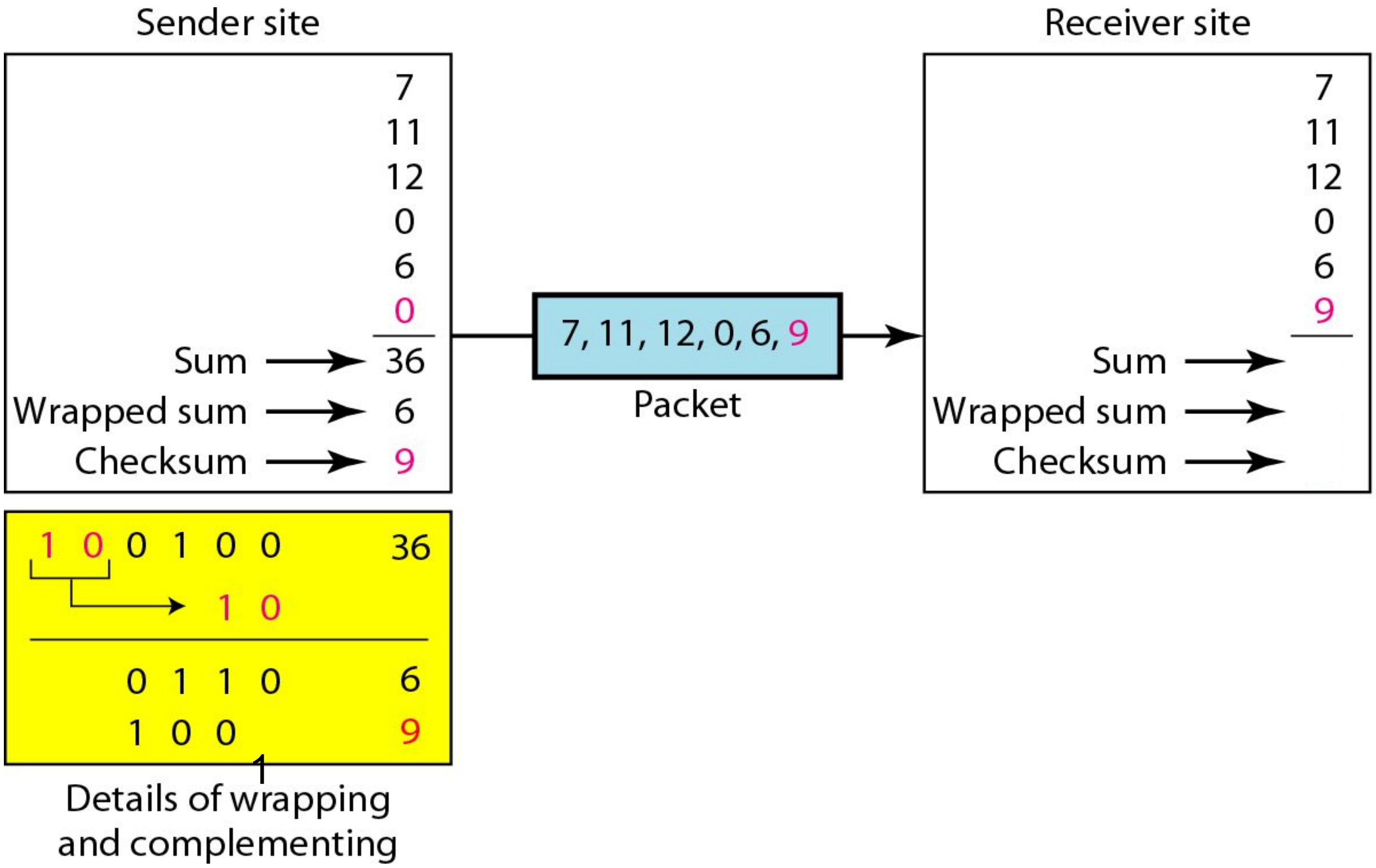
Sum → 36
Wrapped sum → 6
Checksum → 9

1 0 0 1 0 0 36
1 0
0 1 1 0 6
1 0 0 1 9

Details of wrapping and complementing

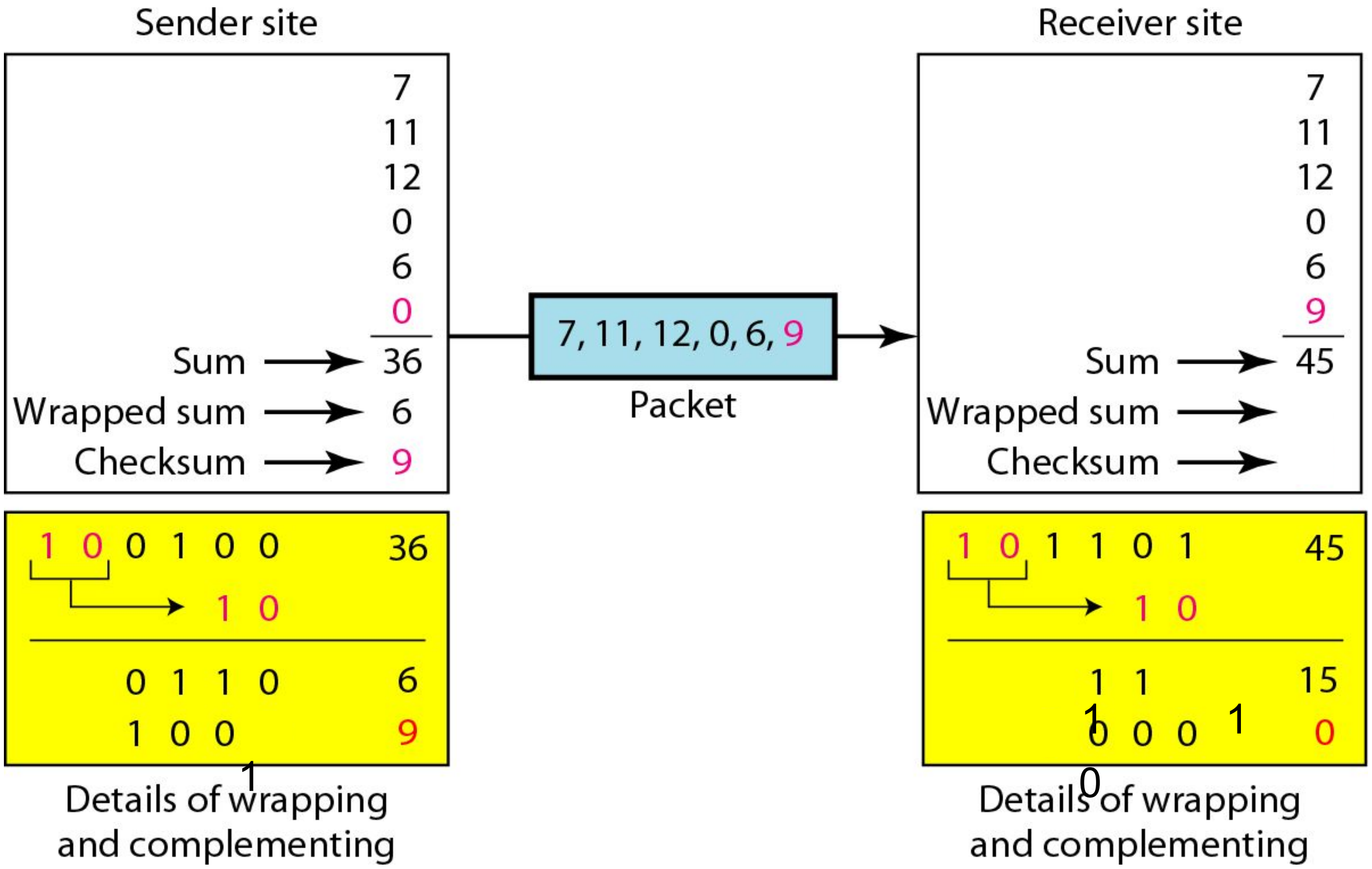
Checksum

- Suppose our data is a list of five 4-bit numbers: 7, 11, 12, 0, 6



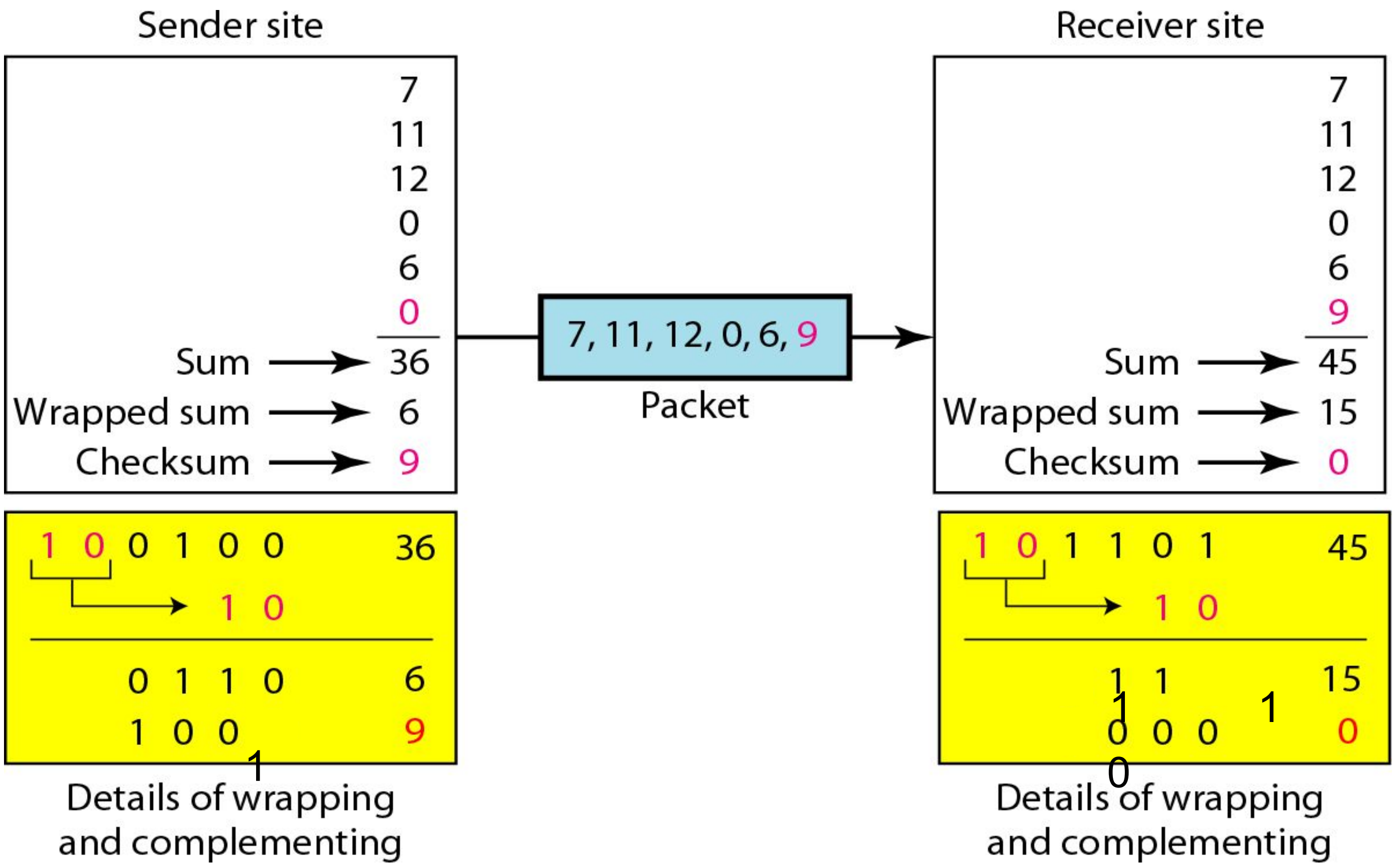
Checksum

- Suppose our data is a list of five 4-bit numbers: 7, 11, 12, 0, 6



Checksum

- Suppose our data is a list of five 4-bit numbers: 7, 11, 12, 0, 6



Performance

- The traditional checksum uses a small number of bits (16) to detect errors in a message of any size (sometimes thousands of bits).
- However, it is not as strong as the CRC in error-checking capability.
- For example, if the value of one word is incremented and the value of another word is decremented by the same amount, the two errors cannot be detected because the sum and checksum remain the same.
- Also, if the values of several words are incremented but the sum and the checksum do not change, the errors are not detected.
- Fletcher and Adler have proposed some weighted checksums that eliminate the first problem.
- Now checksum is being replaced with a CRC.

Error Correction

- Retransmission of corrupted and lost packets is not useful for real-time multimedia transmission because it creates an unacceptable delay in reproducing: we need to wait until the lost or corrupted packet is resent.
- Several schemes have been designed and used in this case that are collectively referred to as Forward Error Correction (FEC) techniques.

Using Hamming Distance

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

Using XOR

$$R = P_1 \oplus P_2 \oplus \ldots \oplus P_i \oplus \ldots \oplus P_N \quad \rightarrow \quad P_i = P_1 \oplus P_2 \oplus \ldots \oplus R \oplus \ldots \oplus P_N$$

- If we apply the exclusive OR operation on N data items (P_1 to P_N), we can recreate any of the data items by exclusive-ORing all of the items, replacing the one to be created by the result of the previous operation (R).

$$011 \oplus 100 \oplus 010 = 101$$

$$011 \oplus 010 \oplus 101 = 100$$

- We can divide a packet into N chunks, create the exclusive OR of all the chunks and send N +1 chunks.
- If any chunk is lost or corrupted, it can be created at the receiver site.