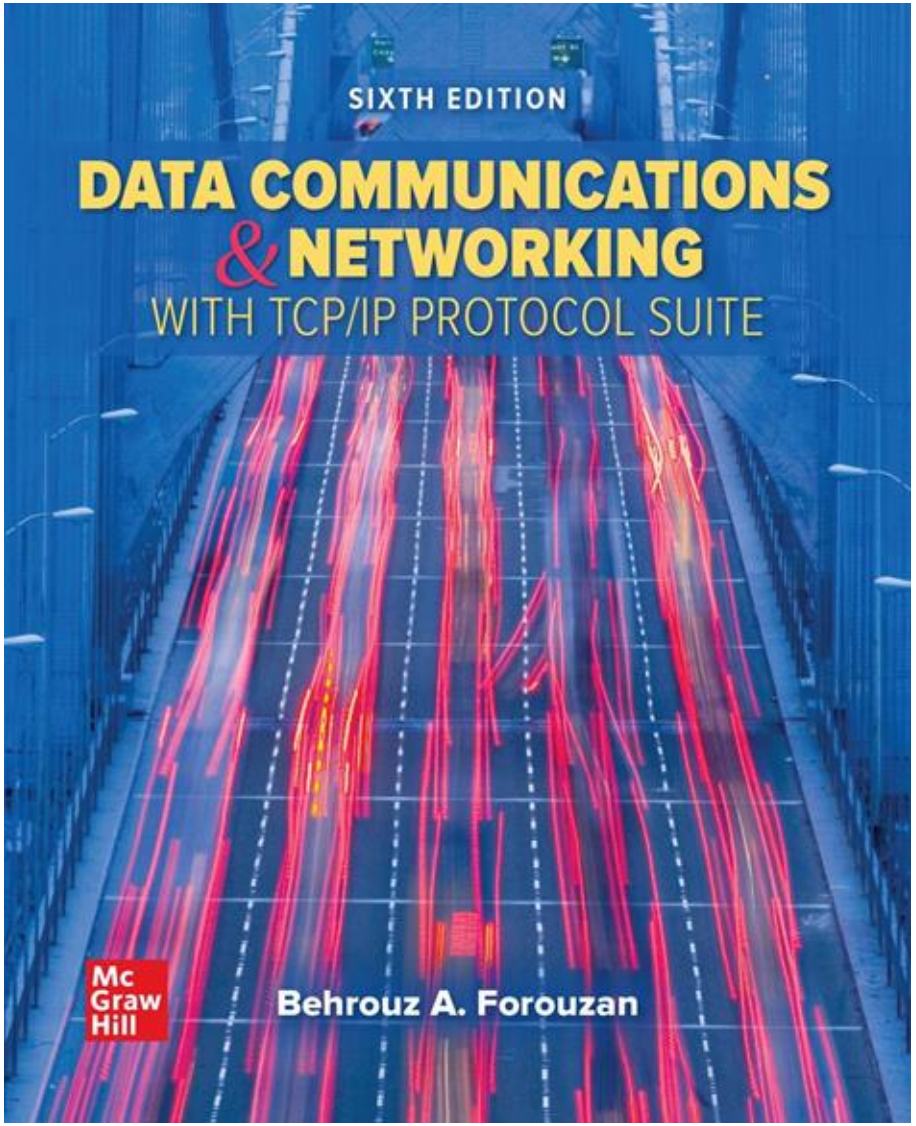
Tue 13 Jan

Chapter 03

Data-Link Layer

- Data Communications and Networking, With TCP/IP protocol suite Sixth Edition
- Behrouz A. Forouzan

Physical layer : done ✓



Chapter 3: Outline

Not included in quiz 2 1

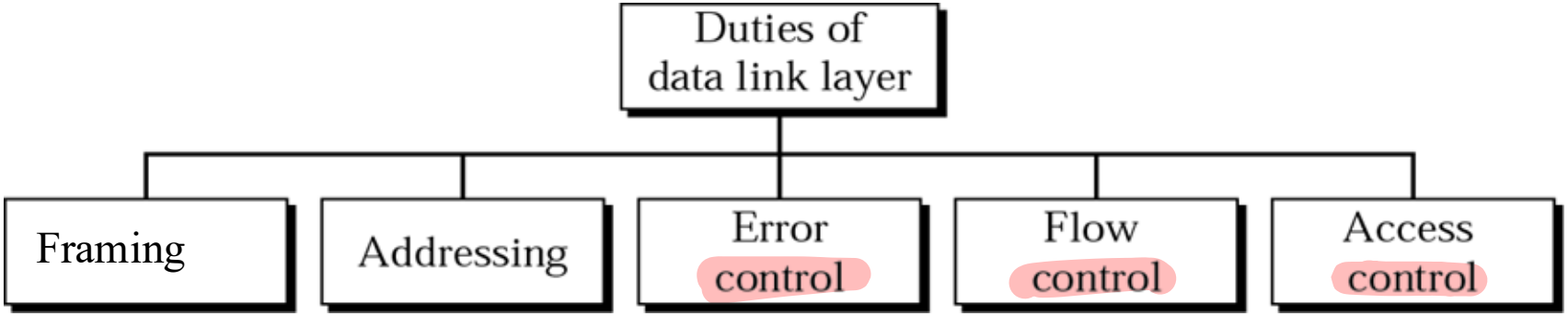
- *3.1 INTRODUCTION*
- *3.2 DATA LINK CONTROL*
- *3.3 MEDIA ACCESS CONTROL*
- *3.4 LINK-LAYER ADDRESSING*

Data link layer duties

moves (PDU) frame from one hop (node) to next

has header & tail

Use MAC address as destination

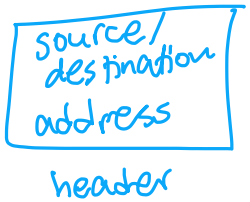


Chapter 3 DLL services

- **Framing/deframing**
  - Encapsulate datagram (packet) into frame, adding header, trailer
  - Different protocols have different format for the DLL frame
  - Deframing at the receiver

sender to reciever using address

- **Addressing**
  - In general it is called layer 2 address or data link layer address
  - In LAN, called physical address or MAC address
  - "physical addresses" used in frame headers to identify source, destination



Framing

write sender/reciever address?
if it couldn't reach reciever
can go back to sender

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.

Framing in the data-link layer separates a message from one source to a destination by adding a sender address and a destination address. The destination address defines where the packet is to go; the sender address helps the recipient acknowledge the receipt.

DLL services

data rate going and from being recieved by

1. **Error Control**
   - Error detection and correction techniques to provide reliable delivery
2. **Flow control**
   - The sender must NOT send frames at a rate faster then the receiver can process them
   - Based on set of procedures that tells that sender how much data it can transmit before it must **wait** for an acknowledgment from the receiver
3. **Access control**
   - Controlling access to a shared medium (shared wire, air)

Medium access control

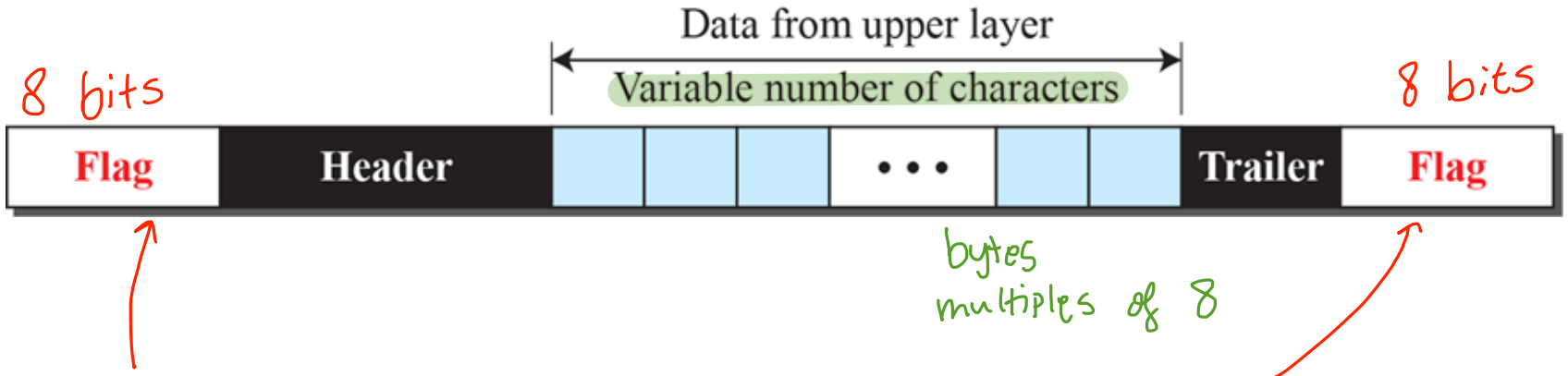
     - This is called Medium Access Control (MAC) sublayer

Framing

- **Two types of framing protocols:**
  - **Character oriented: length of frame is multiple of bytes (multiple of 8)**
  - **Bit oriented: length of frame is not multiple of bytes (NOT multiple of 8)**

characters form
1 byte
8 bits
go through my medium
number of bits in my frame

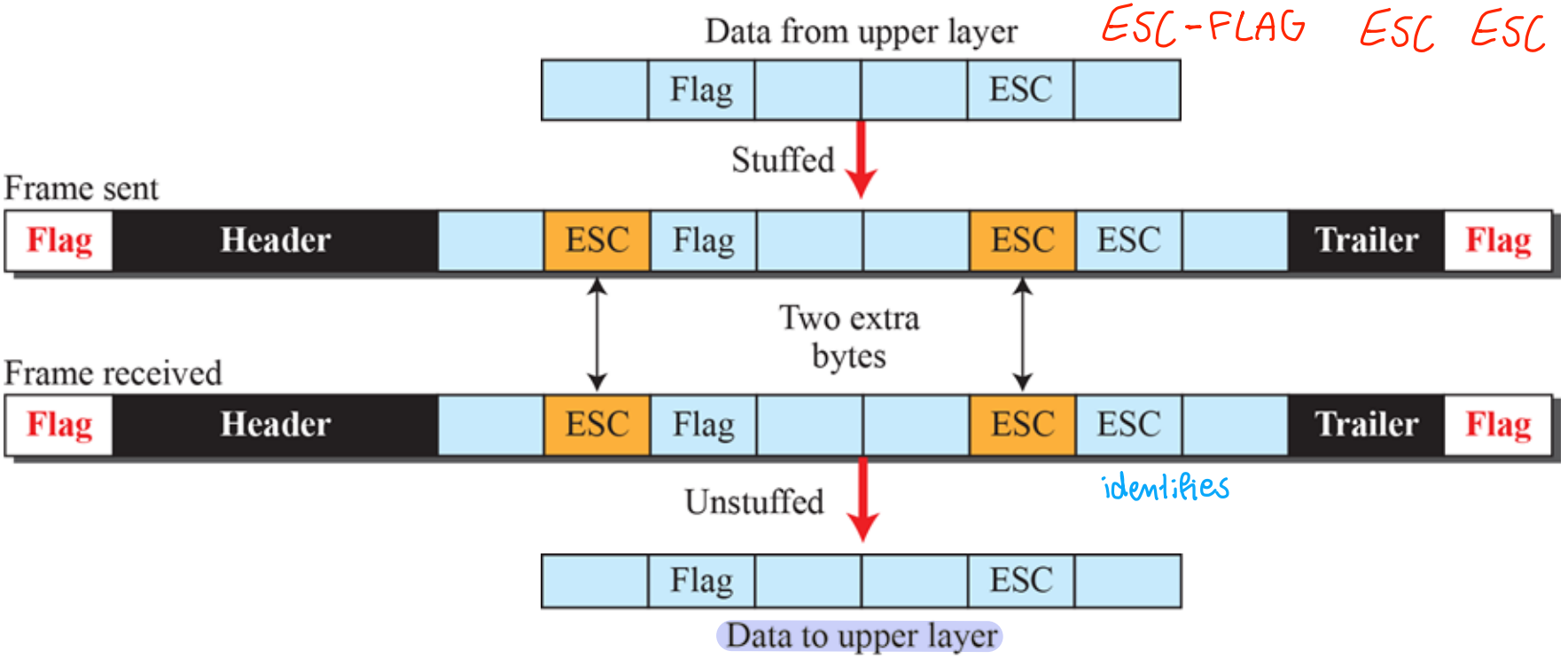
Figure: A frame in a character-oriented protocol



- To separate one frame from another an 8-bit flag is added at the beginning and end of the frame
- Flag is protocol dependent usually (01111110)

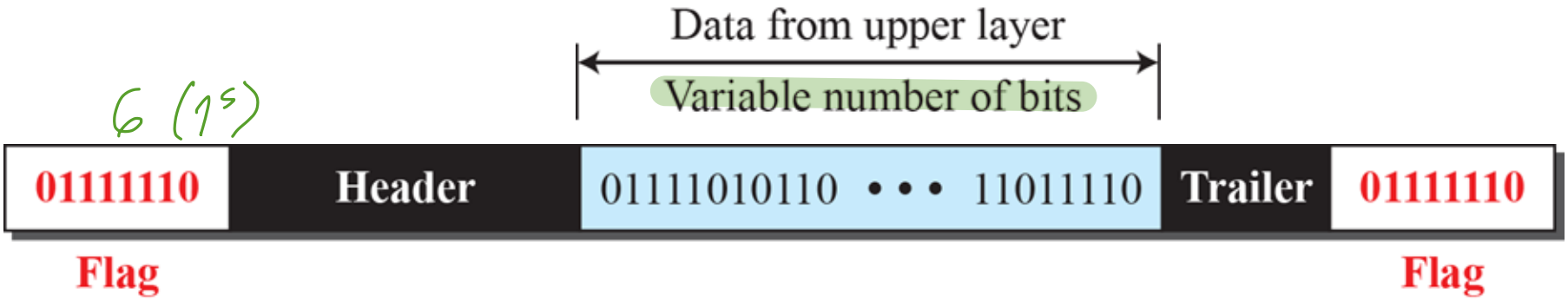
Quiz question (possible):
Screenshot
ESC-FLAG ESC ESC

Figure : Byte stuffing and unstuffing



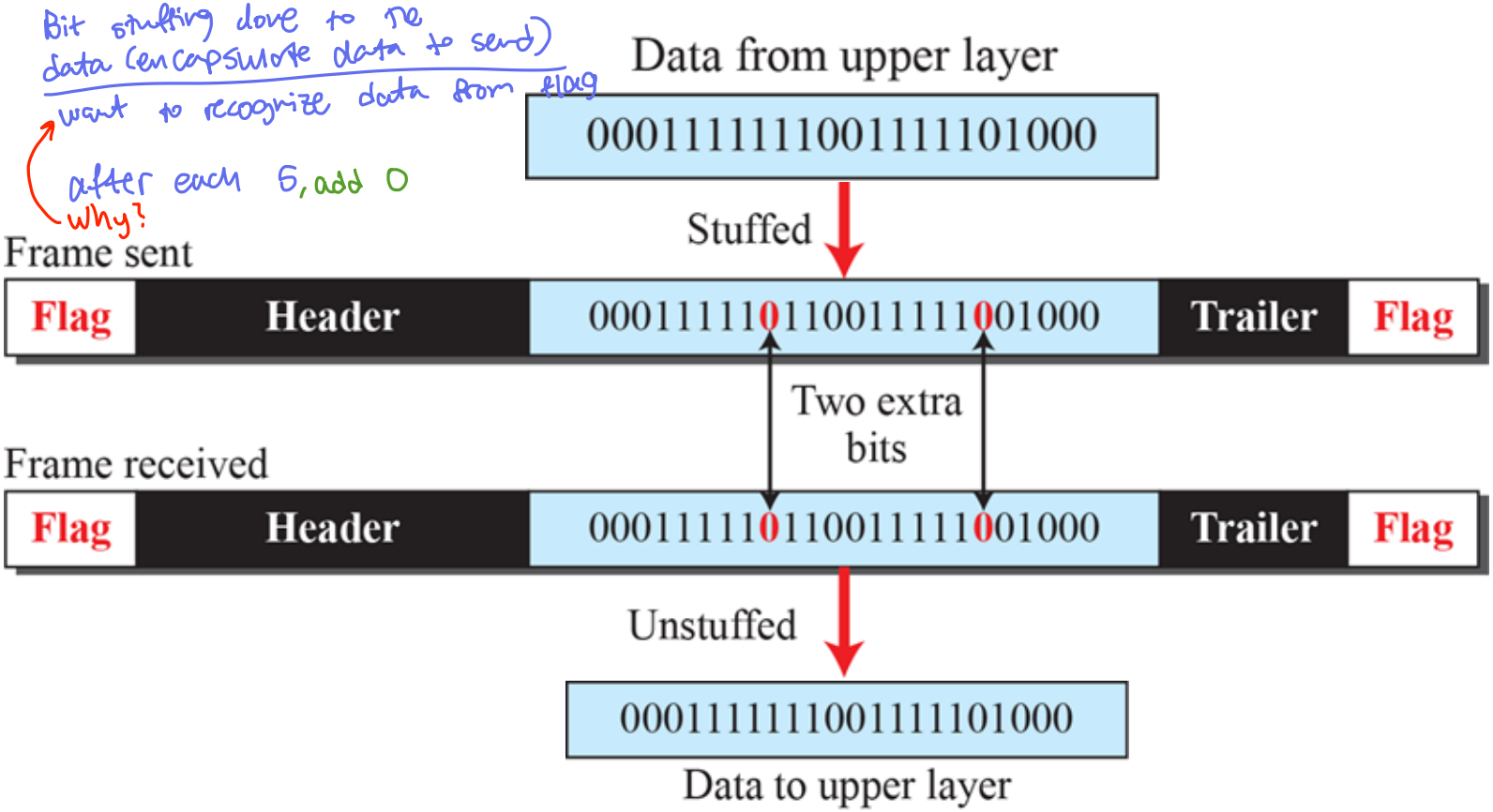
- If flag appears in the data to be carried by the frame, Byte stuffing is used.
- Byte stuffing is the process of adding one extra byte (ESC) whenever the is a flag character in the data. This is done at the Sender
- If (ESC) character also appears in the data, it is also preceded by ESC. This is done because ESC may be also sent by the application as part of the message.
- At the receiver, each time ESC appears in the data, it is removed and the following next character directly following it is treated as data not as flag.

Figure : A frame in a bit-oriented protocol



- To separate one frame from another an 8-bit flag is added at the beginning and end of the frame
- Flag is protocol dependent usually (01111110)

Figure : Bit stuffing and unstuffing



- If flag appears in the data to be carried by the frame, Bit stuffing is used.
- Bit stuffing is the process of adding (stuff) one extra 0 whenever 5 consecutive 1s. Appear in the data from upper layer This is done at the Sender.
- The receiver, on seeing a sequence of five 1s, checks the next bit. If it is 0, the bit is simply discarded (unstuff); if it is 1, then it notes that an end of data set has been flagged.

Note

Data can be corrupted during transmission.

Some applications require that errors be detected and corrected.

Types of Error

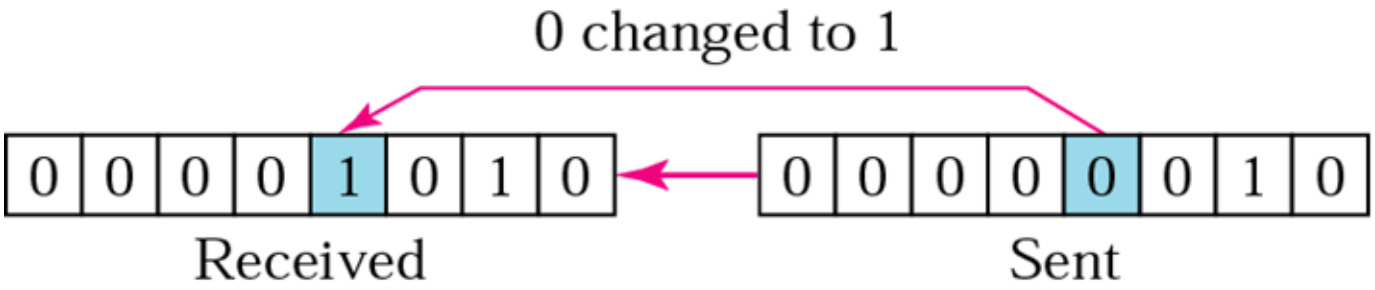
- **Single-Bit Error**
- **Burst Error**

Note

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

Single-bit error

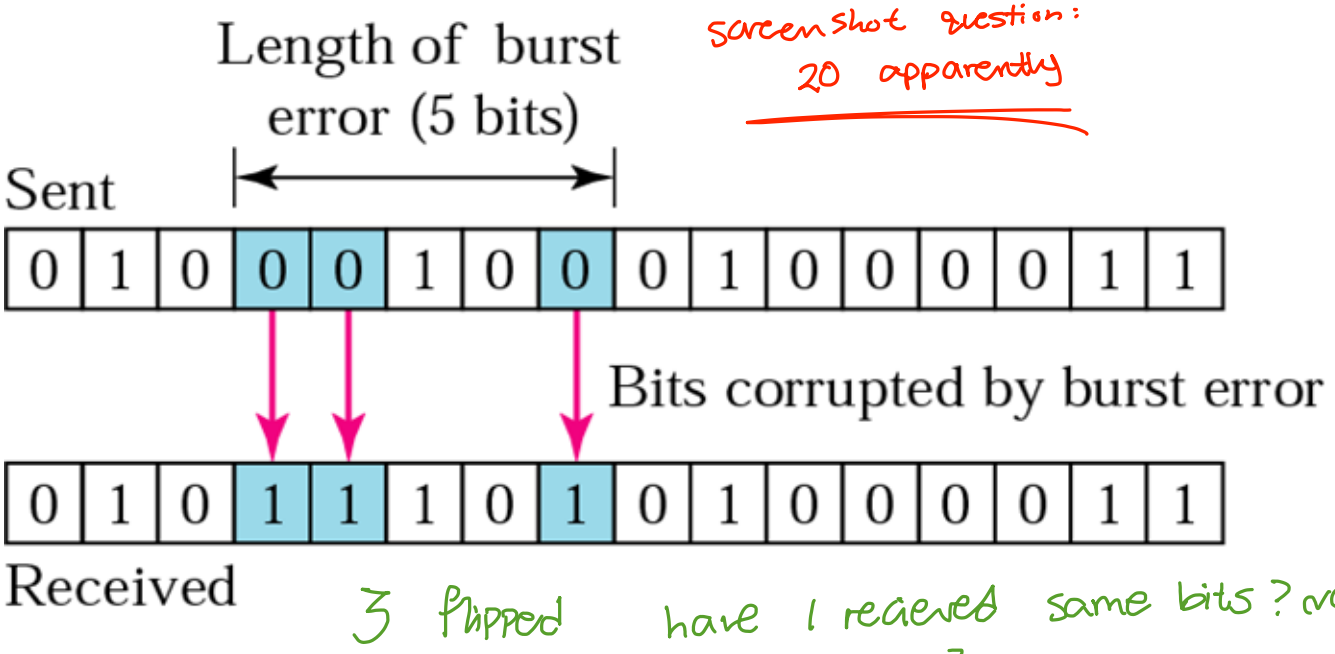
pretty straight forward



Note

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

Burst error of length 5



screenshot question: 20 apparently

3 flipped

have I received same bits? no (error)
1 or more? burst
how many bits before 1st error and between 1st & 2nd error?

- Burst error does NOT necessarily mean that the errors occur in **consecutive bits**
- **Length of the burst**: number of bits from the **first corrupted** bit to the **last corrupted** bit
- Effect of burst error is **higher at high data rates**

Length

Wed 14 Jan

Example

How many bits are affected by 2 ms burst error if:

milliseconds

a) The bandwidth is 1 Kbps

Kilobits per second

b) The bandwidth is 1 Gbps

bits = time of burst x bandwidth

$(s) \times \left(\frac{bits}{s}\right) = bits$

$(1 \times 10^9) \times (2 \times 10^{-3}) = 2000000$ bits

Solution

a) $1x\ 10^3\ x\ 2x10^{-3} = 2$ bits

b) $1x\ 10^9\ x\ 2x10^{-3} = 2000000$ bits

Note

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

Detection methods

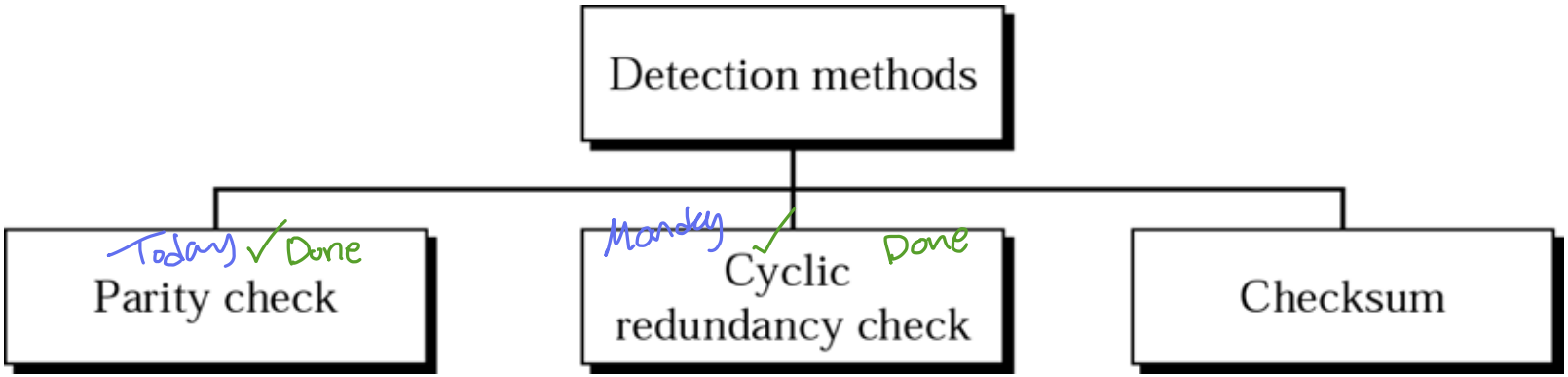


Figure *The structure of encoder and decoder*

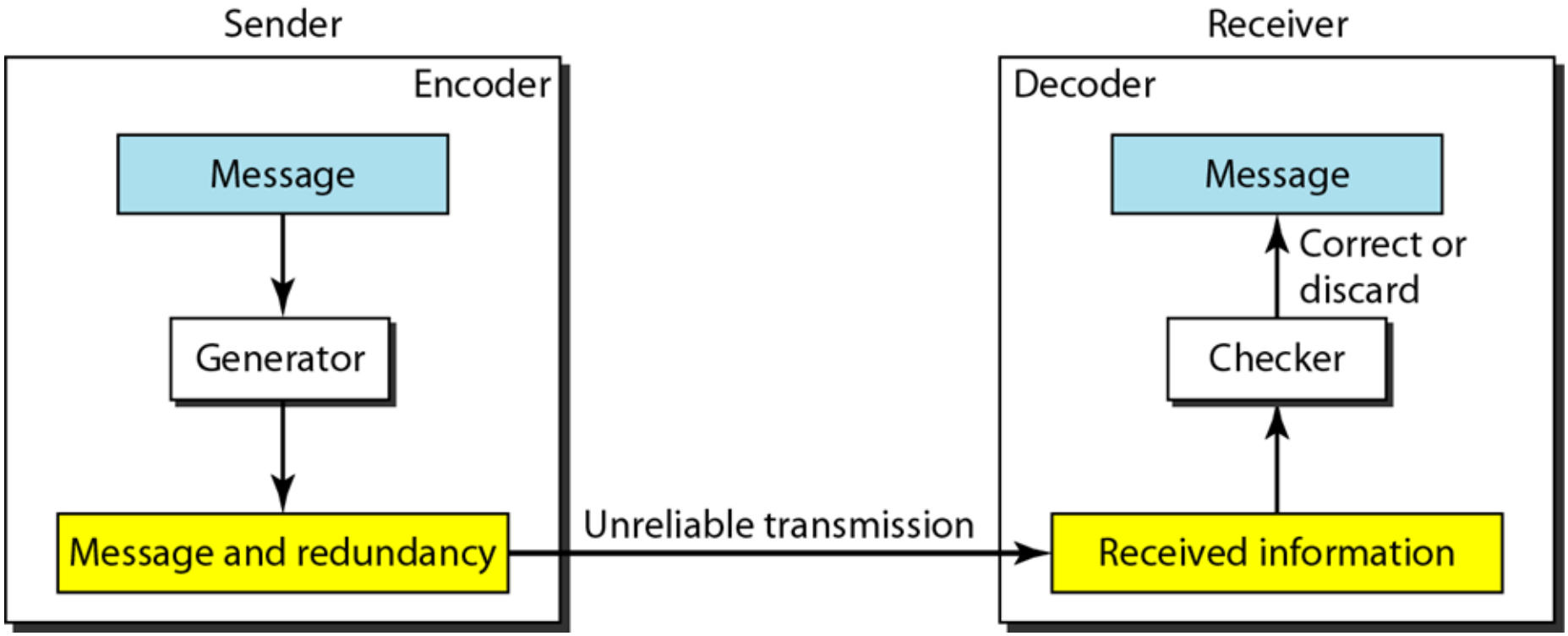
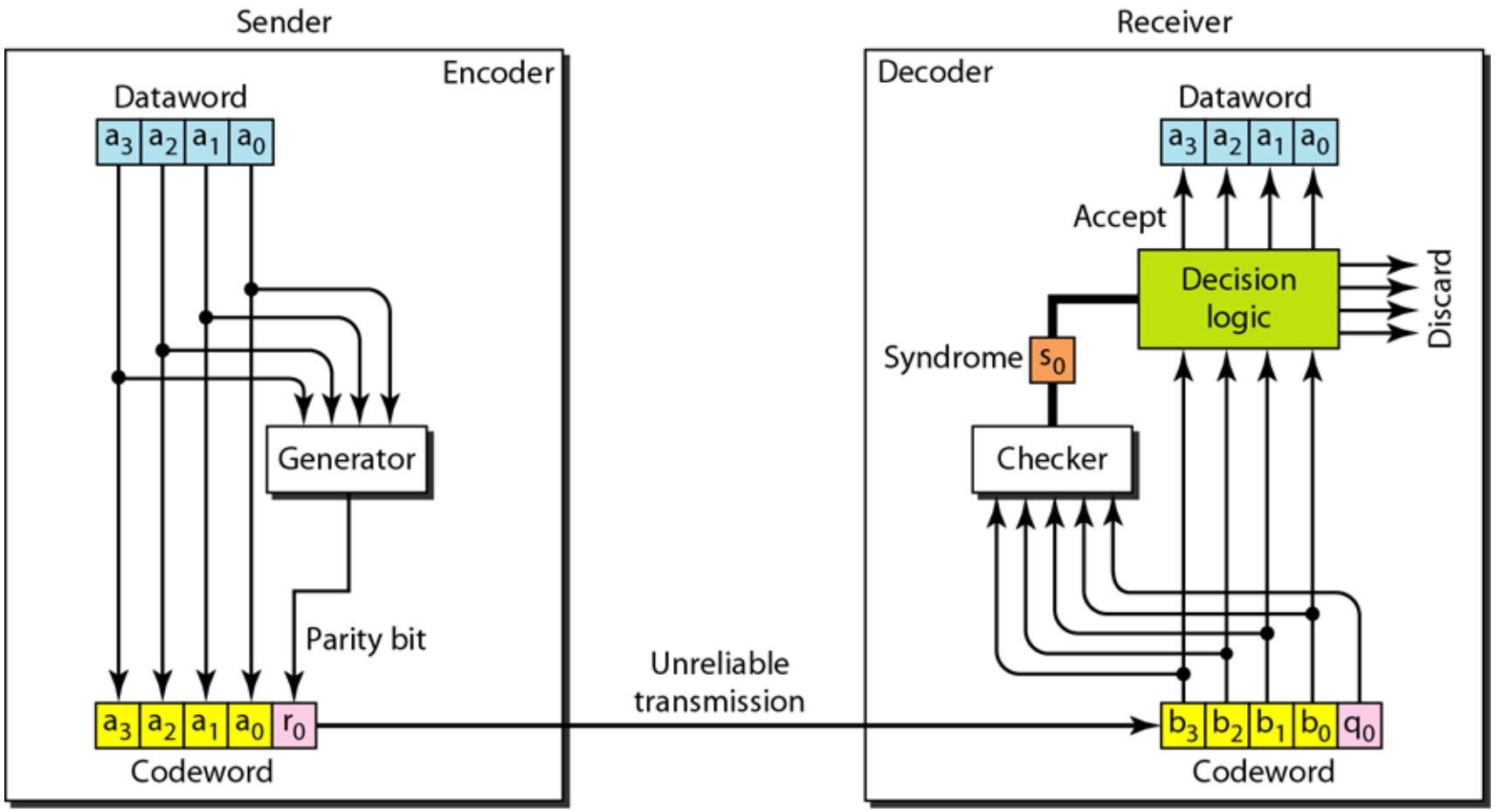


Figure *Encoder and decoder for simple parity-check code*



Note:

In parity check, a parity bit is added to every data unit so that the total number of 1s is even (or odd for odd-parity).

0 is even number

5 bit for each 4 bits

Table ***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

C(n,k) = codeword that consists of n-bit for every k–bit data word

Example 1

Suppose the sender wants to send the word ***world***. In ASCII the five characters are coded as

1110111 1101111 1110010 1101100 1100100

The following shows the actual bits sent in case **Even parity** is used:

11101110 11011110 11100100 11011000 11001001

Example 2

Now suppose the word world in Example 1 is received by the receiver without being corrupted in transmission.

11101110 11011110 11100100 11011000 11001001

The receiver counts the 1s in each character and comes up with even numbers (6, 6, 4, 4, 4). The data are **accepted**.

Example 3

Now suppose the word world in Example 1 is corrupted during transmission.

11111110 11011110 11101100 11011000 11001001

The receiver counts the 1s in each character and comes up with even and odd numbers (7, 6, 5, 4, 4). The receiver knows that the data are **corrupted**, discards them, and asks for retransmission.

Challange

Note:

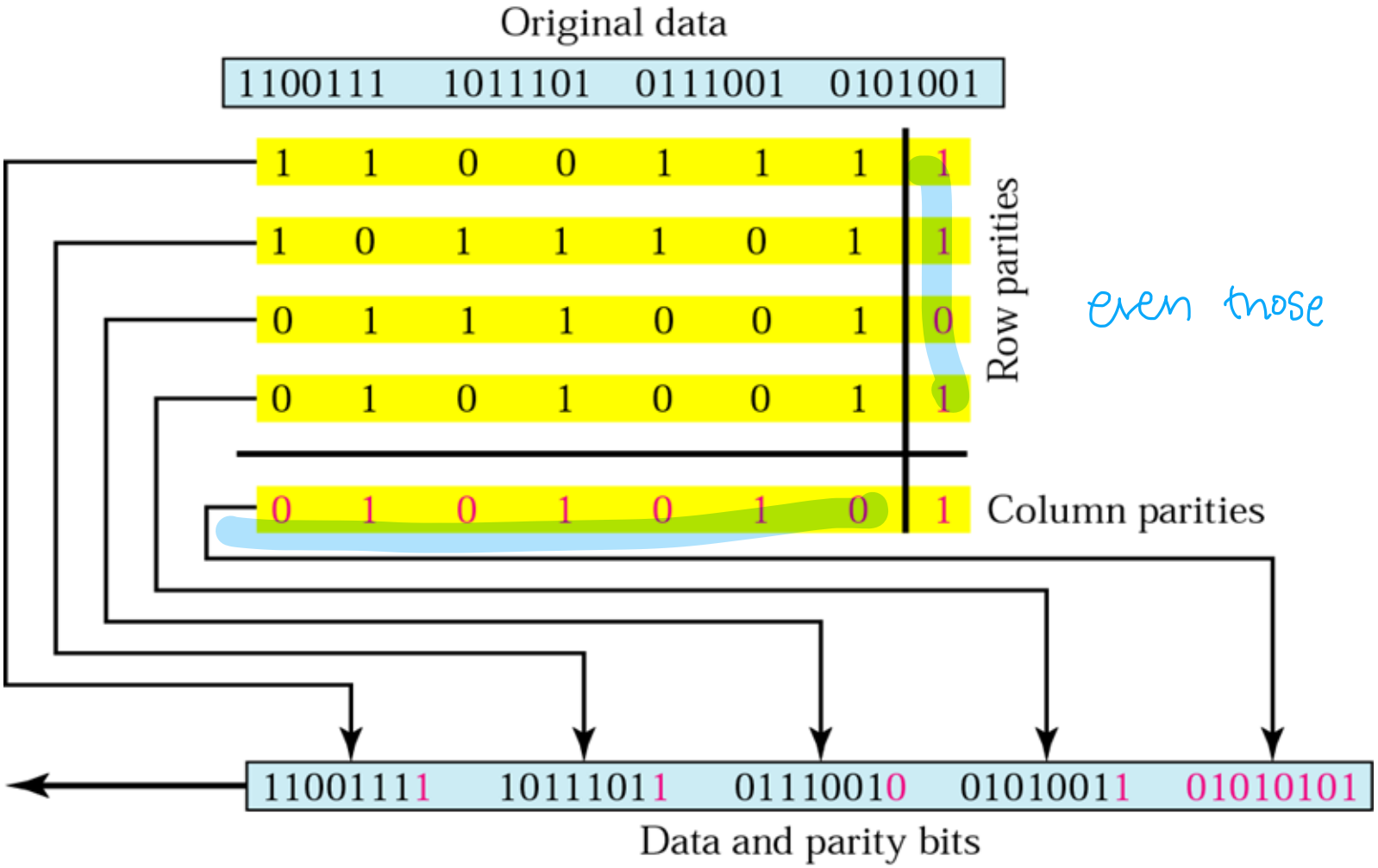
Simple parity check can detect all single-bit errors. It can detect burst errors only if the* total number *of bits changed in each data unit is odd.

Note:

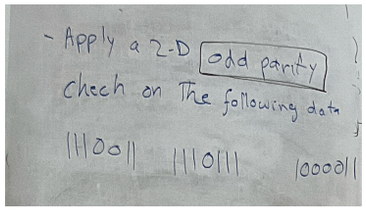
A solution

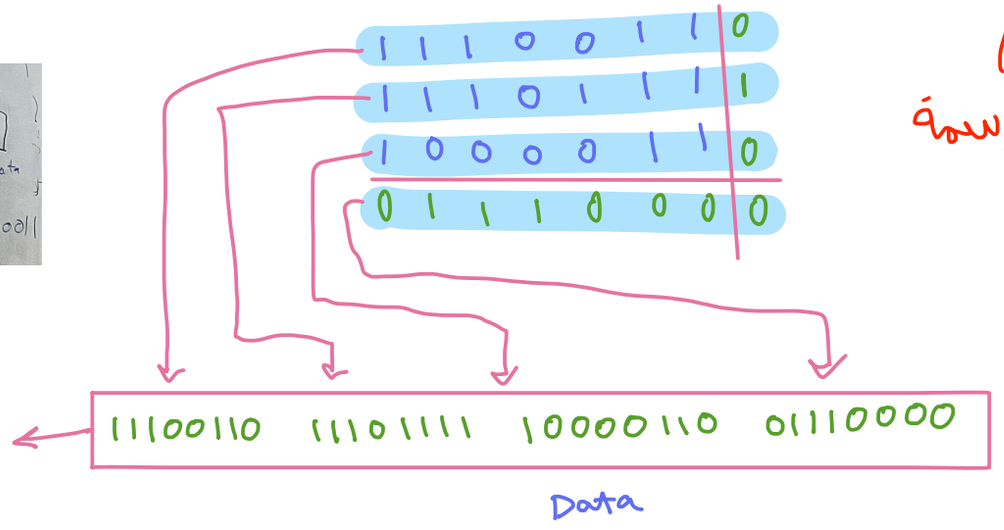
In two-dimensional parity check, a block of bits is divided into rows and a redundant row of bits is added to the whole block.

Two-dimensional parity



Question:



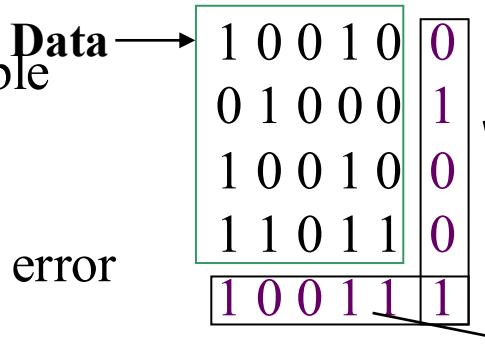


Major:

محاسبة على ان الرسمة
يكون الجواب كامل

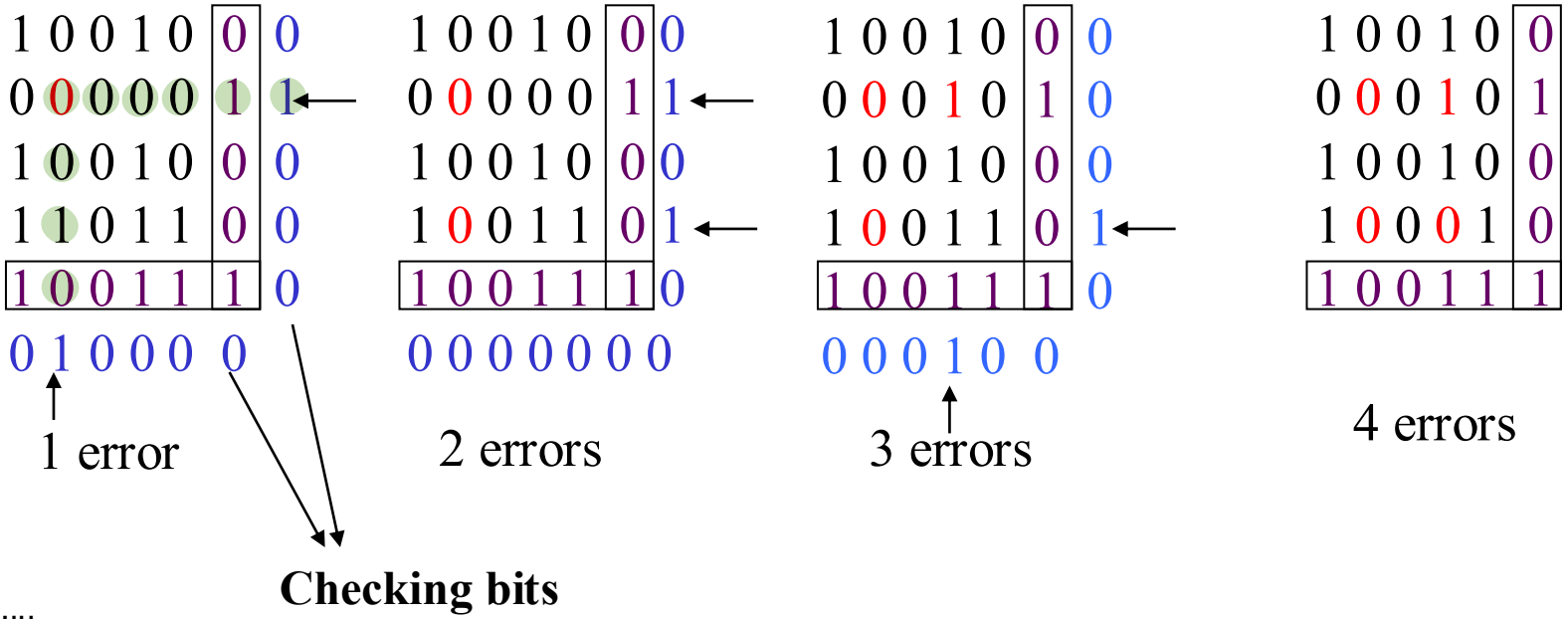
Two-dimension parity

1. Data blocks are organized into table
2. Last column: check bits for rows
3. Last row: check bits for columns
4. Can **detect** and **correct single** bit error



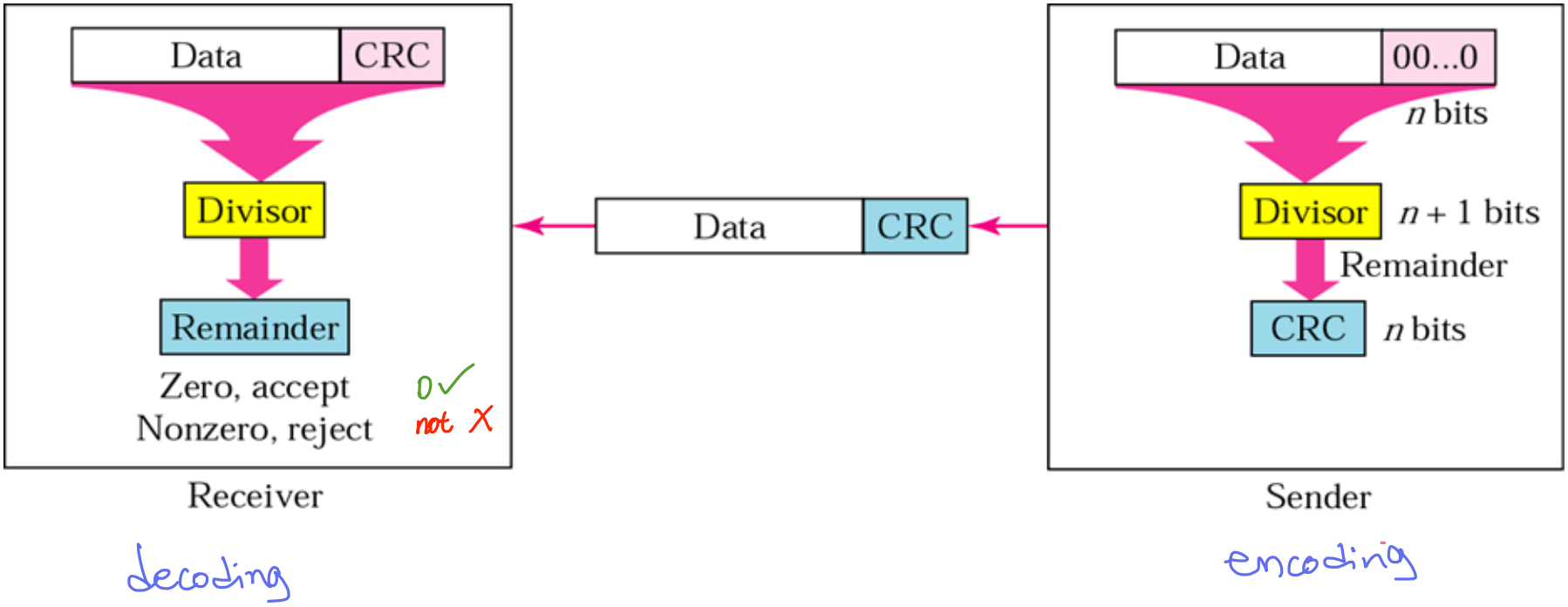
Can detect one, two, three errors, (Red bits are errors)

But NOT all four errors.



Cyclic Redundancy Check (CRC)

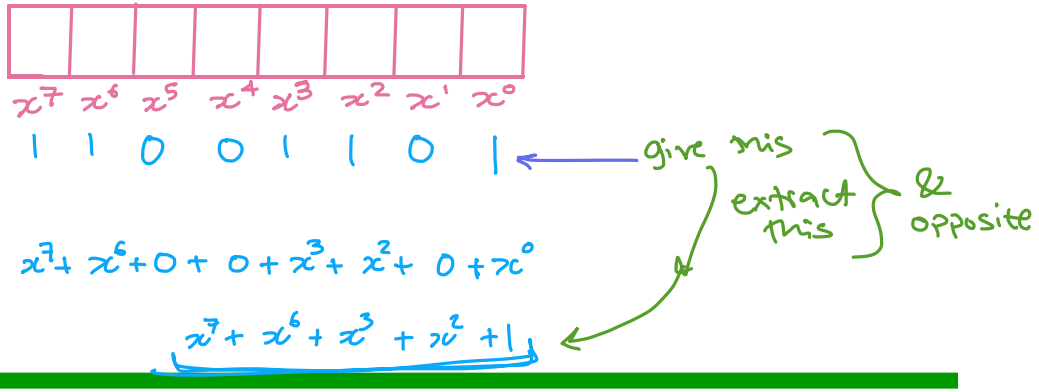
Added divisor can be referred to as polynomial



decoding

encoding

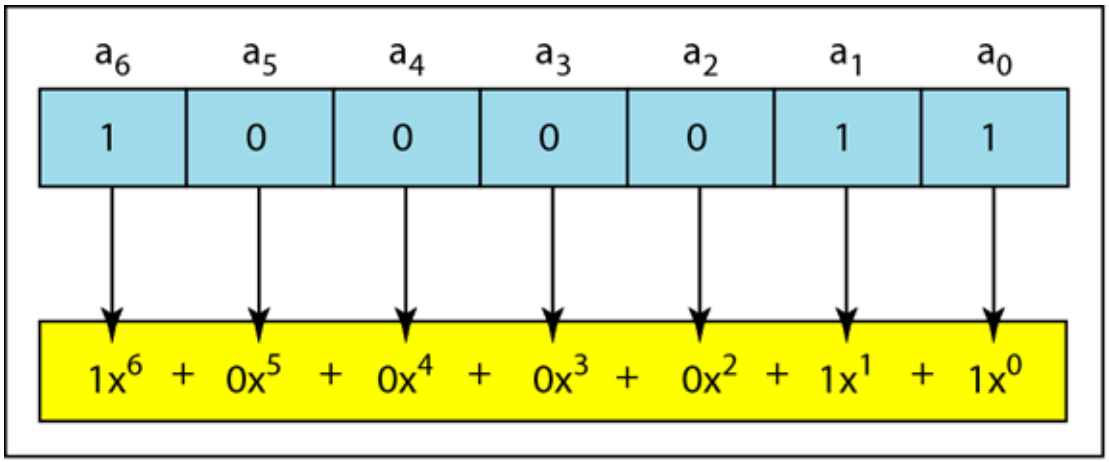
binary data
↓ extract
some information



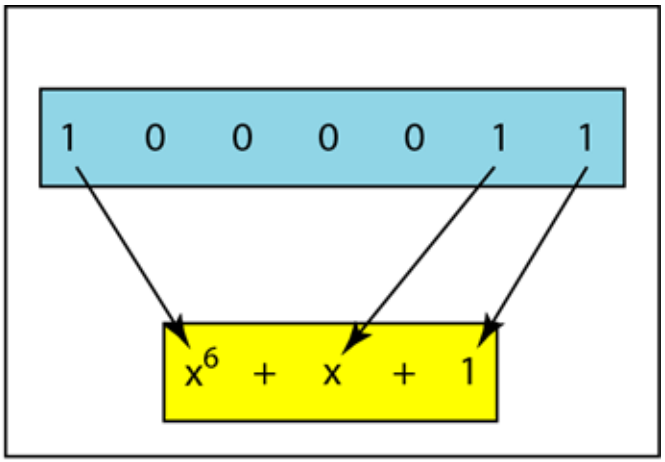
Note

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

Figure *A polynomial to represent a binary word*

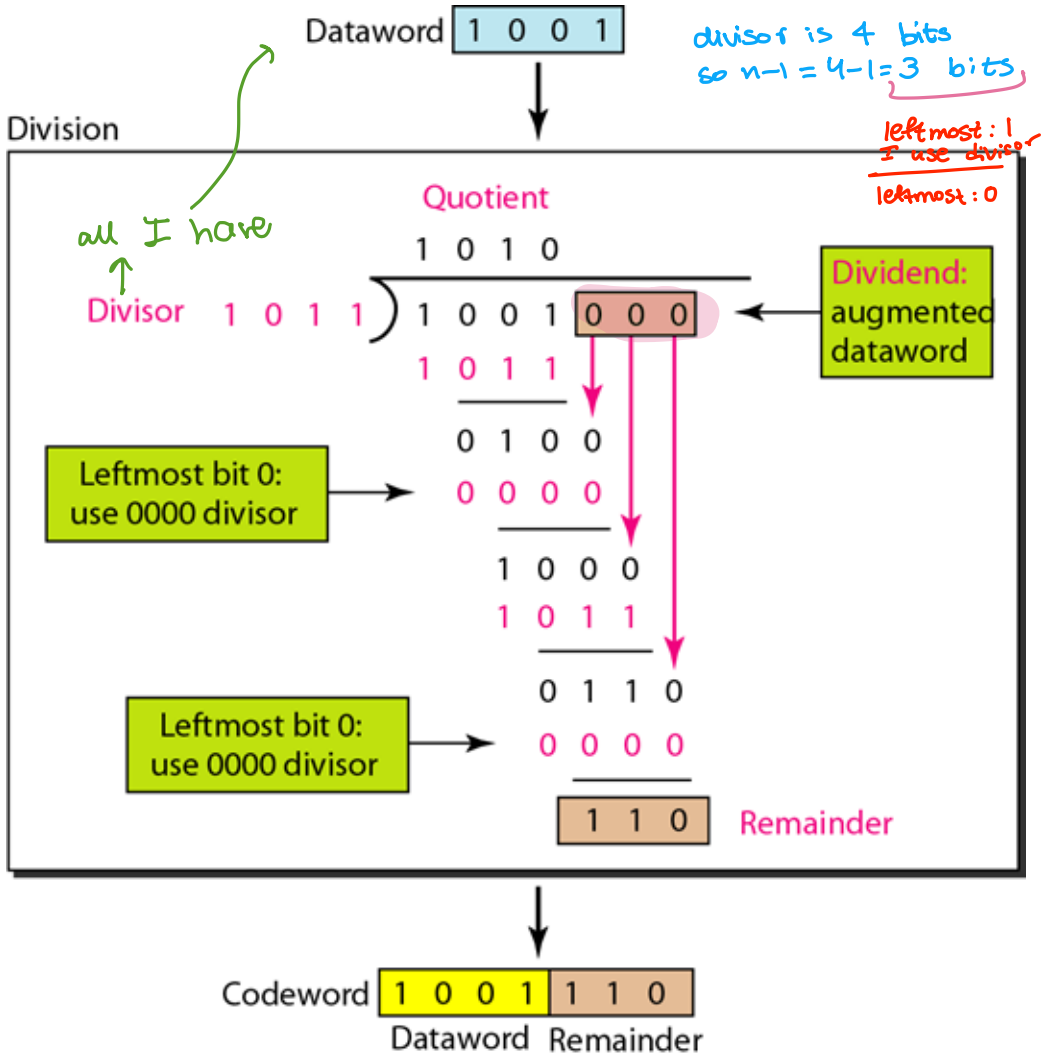


a. Binary pattern and polynomial



b. Short form

Figure *Division in CRC encoder*



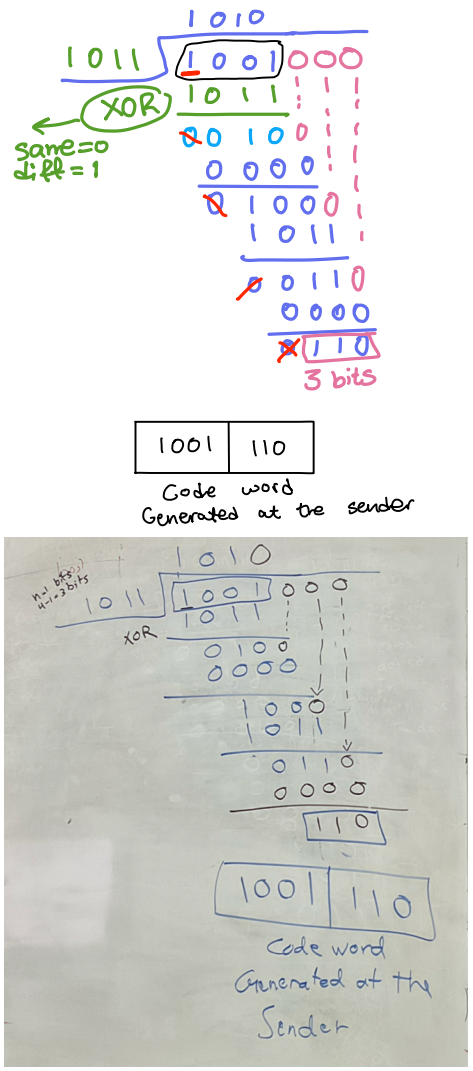
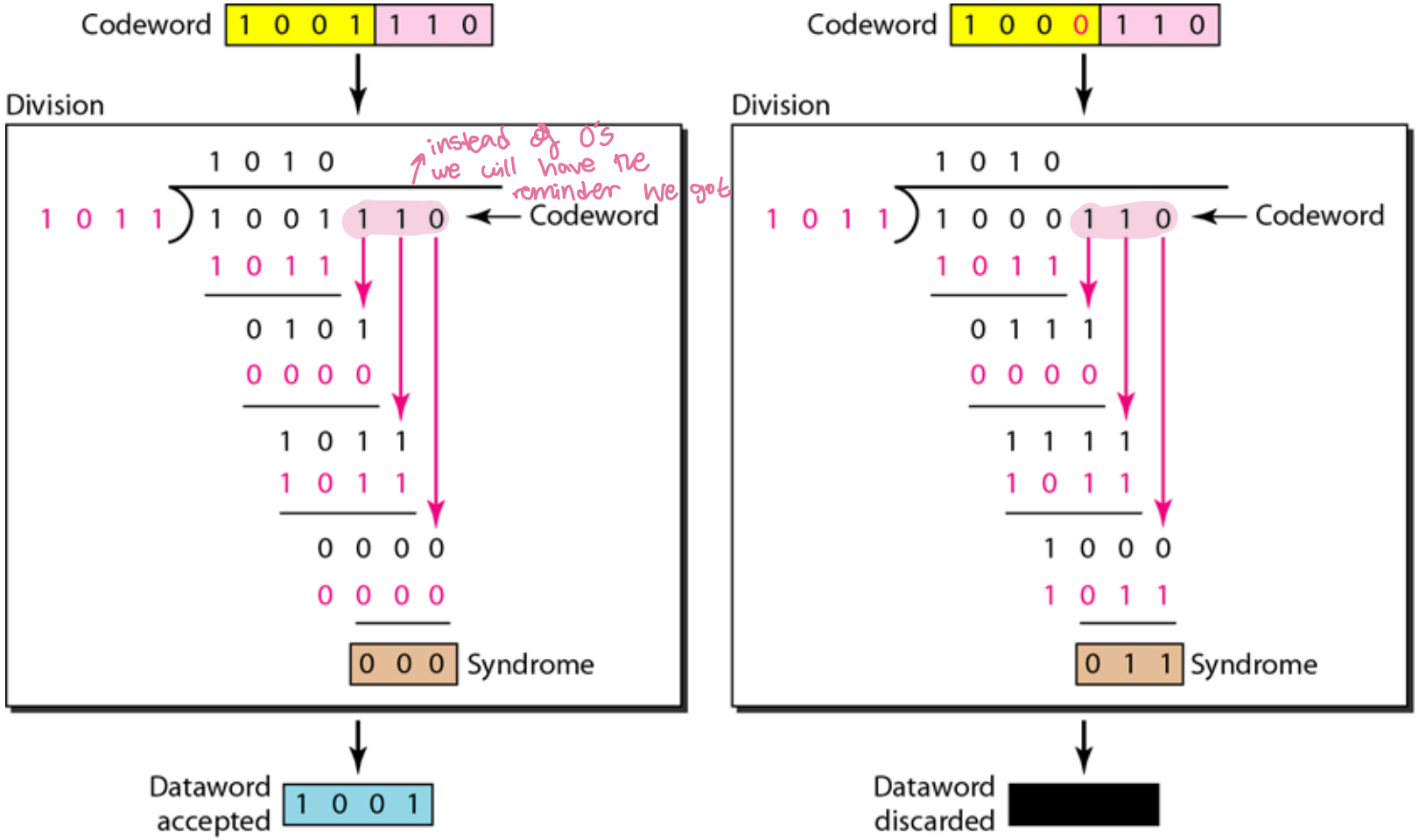
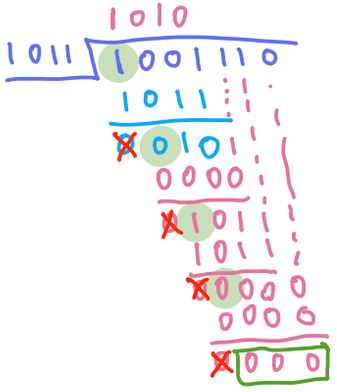
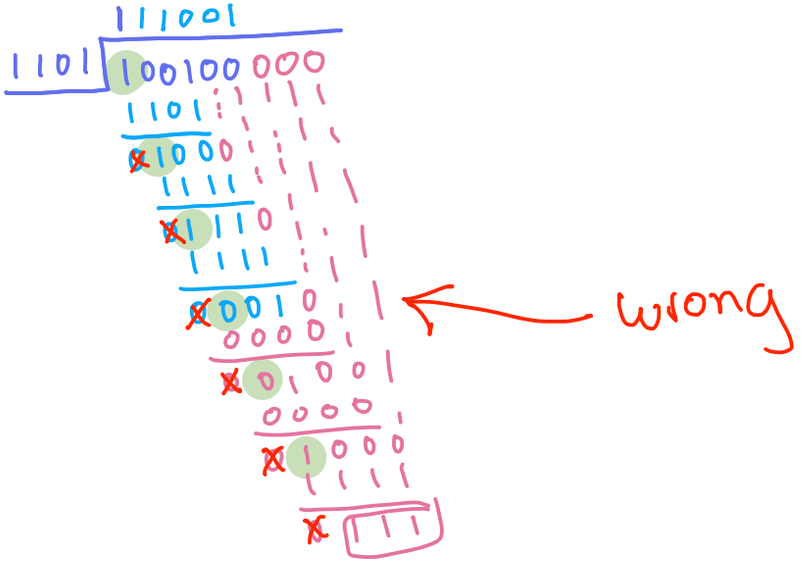


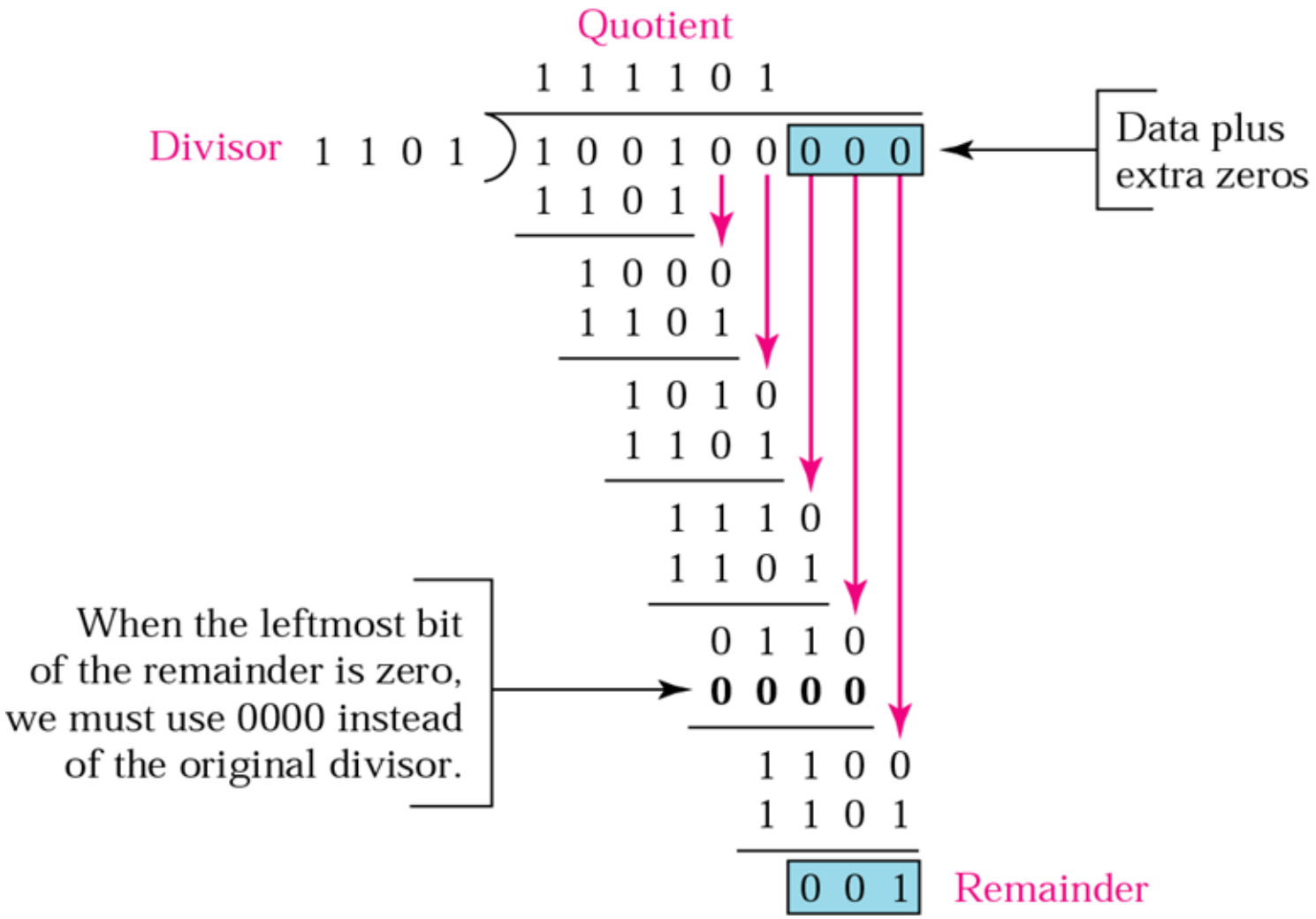
Figure *Division in the CRC decoder for two cases*






wrong

Binary division in a CRC generator



Binary division in CRC checker

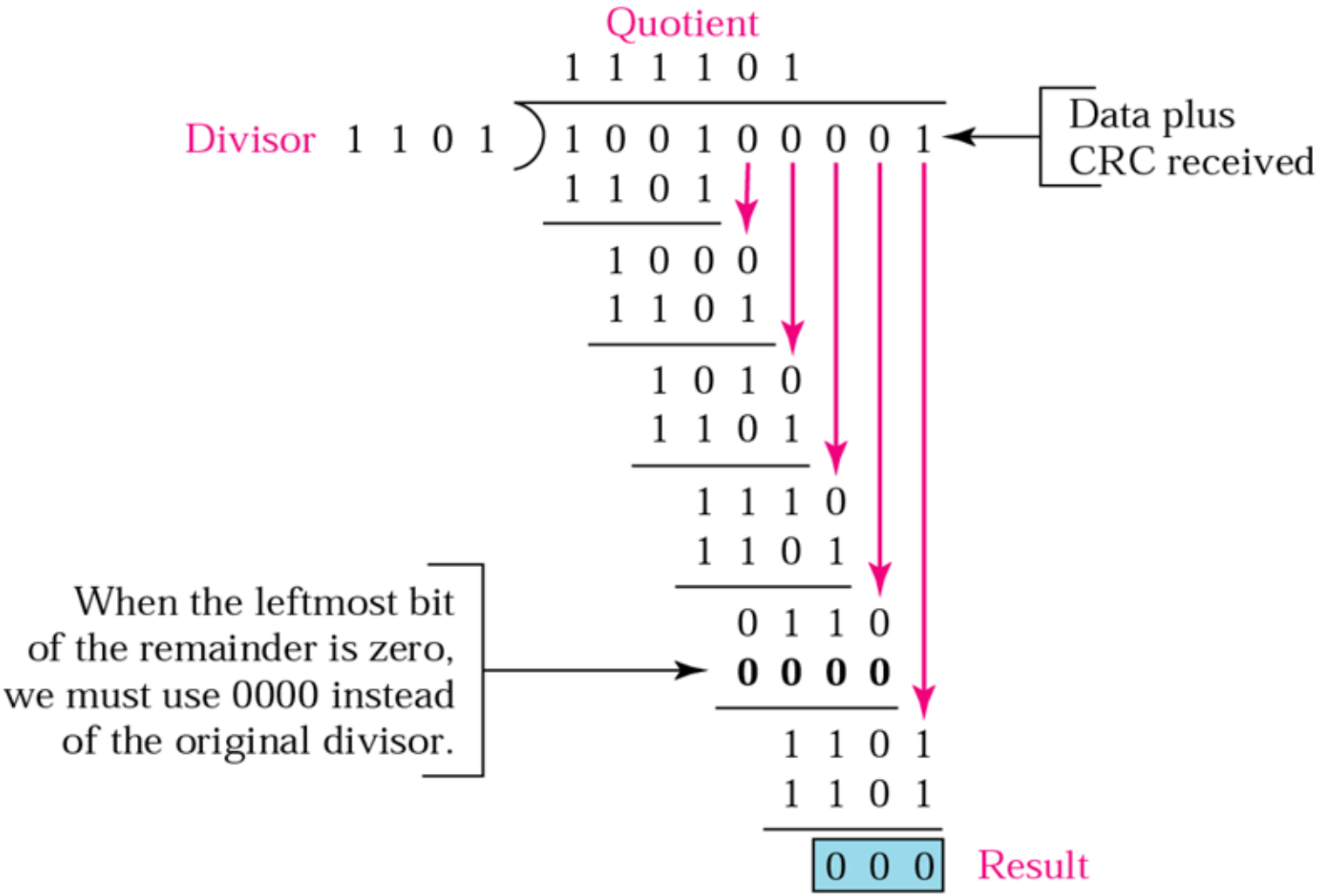


Figure *Division in CRC encoder*

Modulo 2 Division:

The process of modulo-2 binary division is the same as the familiar division process we use for decimal numbers. Just that instead of subtraction, we use XOR here.

- In each step, a copy of the divisor (or data) is XORed with the k bits of the dividend (or key).
- The result of the XOR operation (remainder) is (n-1) bits, which is used for the next step after 1 extra bit is pulled down to make it n bits long.
- When there are no bits left to pull down, we have a result. The (n-1)-bit remainder which is appended at the sender side.

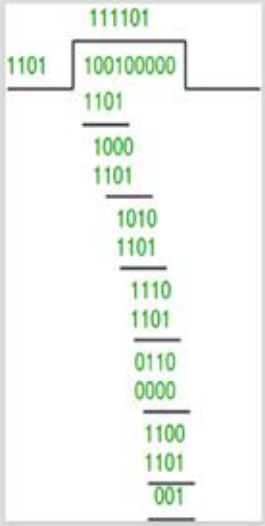
Illustration:

Example 1 (No error in transmission):

Data word to be sent - 100100

Key - 1101 [Or generator polynomial $x^3 + x^2 + 1$]

Sender Side:



Therefore, the remainder is 001 and hence the encoded data sent is 100100001.

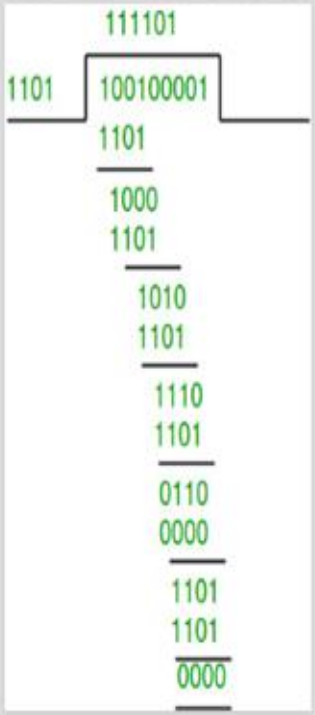
CRC uses **Generator Polynomial** which is available on both sender and receiver side. An example generator polynomial is of the form like $x^3 + x + 1$. This generator polynomial represents key 1011. Another example is $x^2 + 1$ that represents key 101. This will be CRC divisor.

Modulo-2 binary division is used

Figure : Division in the CRC decoder for two cases

Receiver Side:

Code word received at the receiver side 100100001

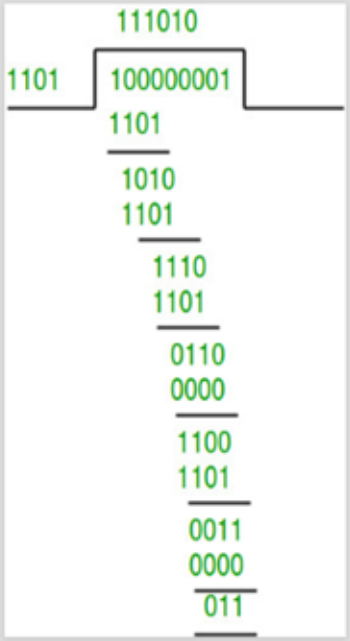


Therefore, the remainder is all zeros. Hence, the data received has no error.

Receiver Side

Let there be error in transmission media

Code word received at the receiver side - 100000001



Since the remainder is not all zeroes, the error is detected at the receiver side.

Standard polynomials

Major maybe
Quizzes no

Name	*Polynomial*	*Used in*
CRC-8	$x^8 + x^2 + x + 1$ **100000111**	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$ **11000110101**	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$ **1000100000100001**	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ **10000010011000001000111011011011 1**	LANs

well-known

Advantages of Cyclic Codes

Tue 20 Jan

We have seen that cyclic codes have a very good performance in detecting single-bit errors, double errors, an odd number of errors, and burst errors. They can easily be implemented in hardware and software. They are especially fast when implemented in hardware. This has made cyclic codes a good candidate for many networks.

Many network devices like it

Example

Find the suitability of the following generators in relation to burst errors of different lengths.

a. $x^6 + 1$ ***b.*** $x^{18} + x^7 + x + 1$ ***c.*** $x^{32} + x^{23} + x^7 + 1$

highest degree: 6 indicates all burst errors with a length less or equal to 6

18 highest degree generator can detect burst errors

32 highest degree

explained below

Solution

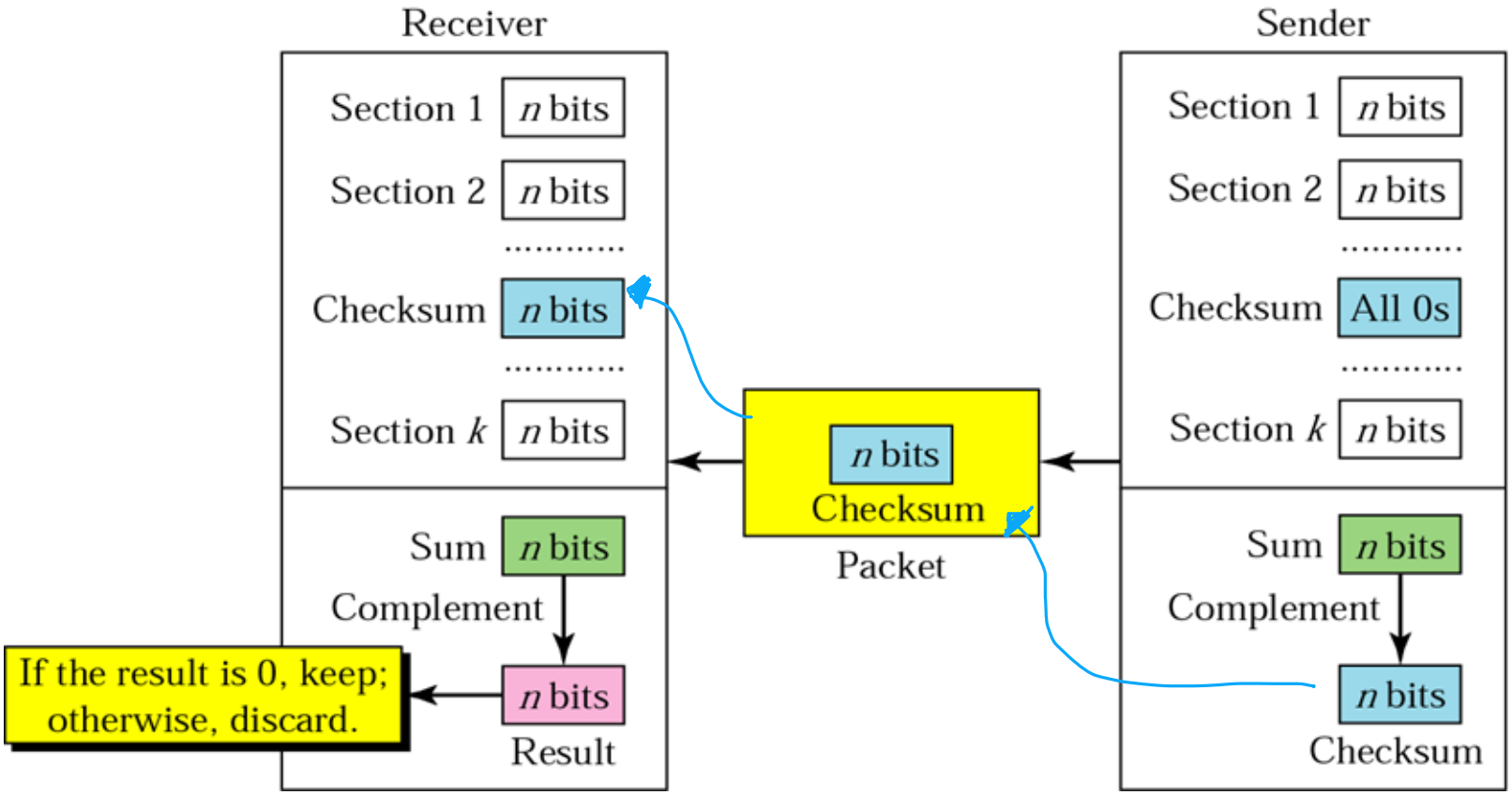
a. ***This generator can detect all burst errors with a length less than or equal to 6 bits; ($1/2^5 = 0.03$) 3 out of 100 burst errors with length 7 will not be detected; ($1/2^6 = 0.016$) 16 out of 1000 burst errors of length 8 or more will not be detected.***

Example (continued)

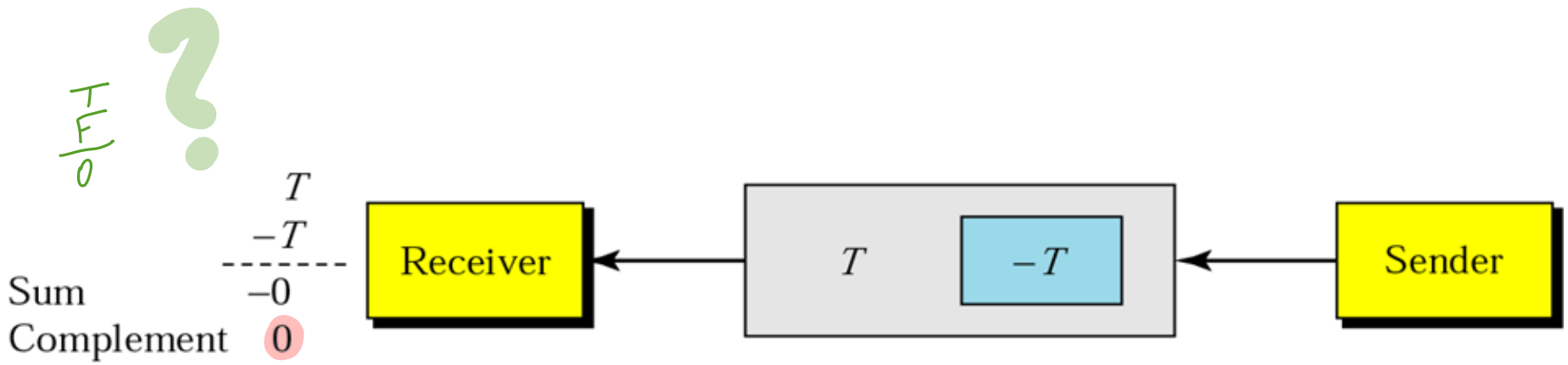
b. *This generator can detect all burst errors with a length less than or equal to 18 bits; 8 out of 1 million burst errors with length 19 will slip by; 4 out of 1 million burst errors of length 20 or more will slip by.*

c. *This generator can detect all burst errors with a length less than or equal to 32 bits; 5 out of 10 billion burst errors with length 33 will slip by; 3 out of 10 billion burst errors of length 34 or more will slip by.*

3 type of detection



Remember binary logic



Previous XOR
This is sum

Note

Sender site: 2 bytes

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.**

Example 7

Suppose the following block of 16 bits is to be sent using a checksum of 8 bits.

10101001 00111001

The numbers are added using one's complement

```
                1 1 1 1
                10101001        binary addition
                00111001
                ----------
Sum             11100010
Checksum        00011101        complement of each bit
```

The pattern sent is 10101001 00111001 **00011101**

Example 8

Now suppose the receiver receives the pattern sent in Example 7 and there is no error.

10101001 00111001 00011101

When the receiver adds the three sections, it will get all 1s, which, after complementing, is all 0s and shows that there is no error.

	10101001
	00111001
	00011101
Sum	11111111
Complement	**00000000** means that the pattern is OK.

Example 9

Now suppose there is a burst error of length 5 that affects 4 bits.

10101**111 11**111001 00011101

When the receiver adds the three sections, it gets

	10101111
	11111001
	00011101
Partial Sum	**1** 11000101
Carry	**1**
Sum	11000110
Complement	**00111001**

the pattern is corrupted.

once I detected my errors, using the last strategies

Error Correction

Retransmission

Forward Error Correction

Burst Error Correction

Error Correction- 1-Retransmission

in real-life example:

- For Wireless applications error detection and retransmission is *inefficient*:
  - The bit error rate on a wireless link can be very high which would result in large number of retransmissions.
- For real-time Multimedia applications, retransmissions create unacceptable delay. The application has to wait until the corrupted packet is retransmitted.

Error Correction-
2-Forward Error Correction (FEC)

Forward error correction (FEC) is an error correction technique to detect and correct a limited number of errors in transmitted data without the need for retransmission.

- Two-dimensional Parity 1st approach we
- Using XOR property
  - After adding the error detection bits to the datawords and create the codewords blocks
    - XOR all codewords together. This will create block (R)
    - Transmit the codewords with the block (R)
  - To recover any of the corrupted blocks, XOR all other blocks with the (R)
- Burst error correction using chunk interleaving

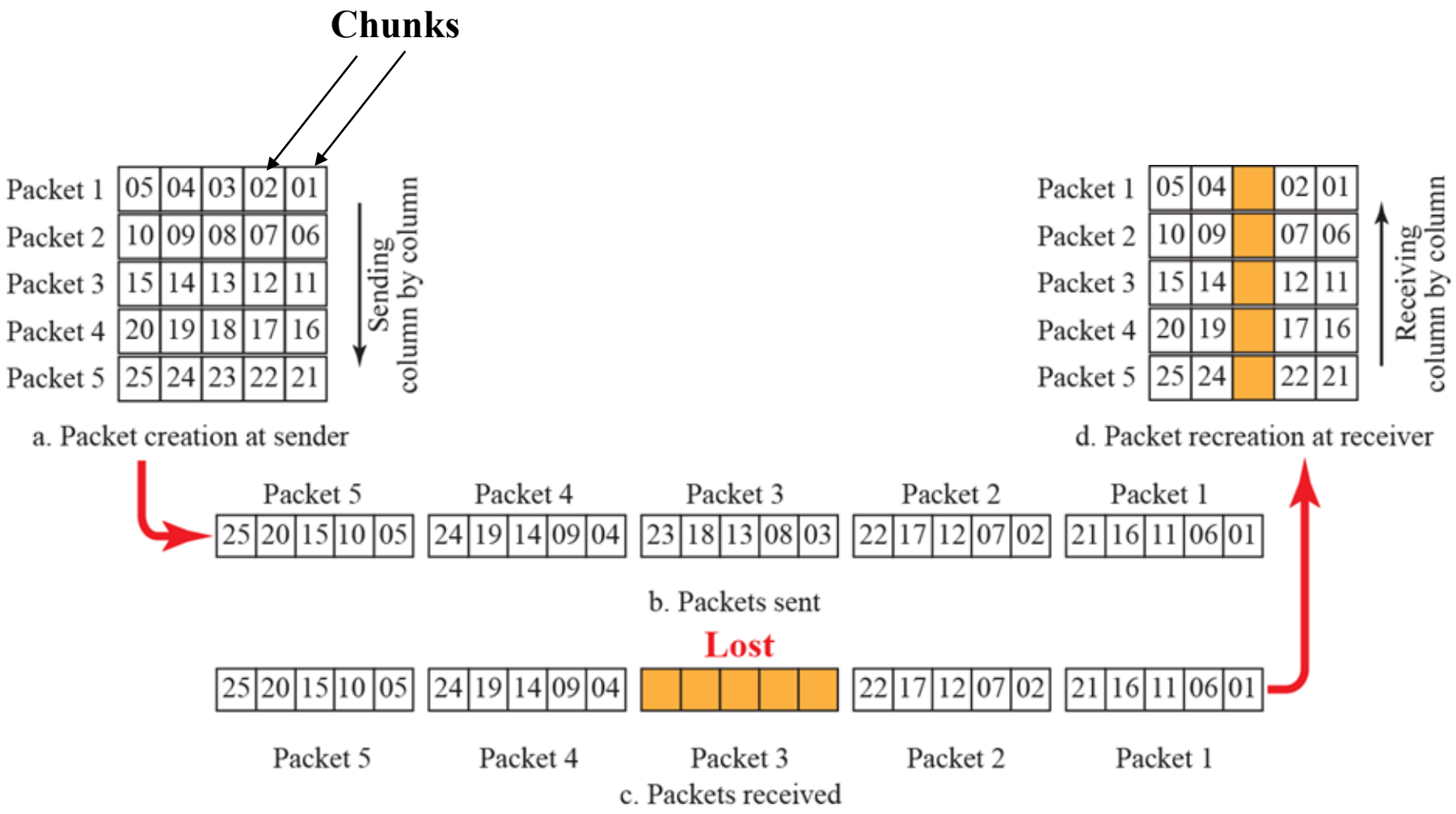
Burst Error Correction

Another way to achieve FEC in multimedia is to allow some small chunks to be missing at the receiver. We cannot afford to let all the chunks belonging to the same packet be missing; however, we can afford to let one chunk be missing in each packet.

One chunk is missing from each packet is acceptable in multimedia applications

Allowing missing chunks to not burden to retransmit

Figure : Interleaving



Done ✓

included

major