

IUG - Faculty of Engineering
Department of Computer Engineering
Data communications Lab
Summer 2022-2023



Data Communications

Lab 10

Chapter 10

Error Detection and Correction

Abdullah Idris Ouda

P10-1. What is the maximum effect of a 2-ms burst of noise on data transmitted at the following rates?

- a. 1500 bps b. 12 kbps c. 100 kbps d. 100 Mbps

Solution:

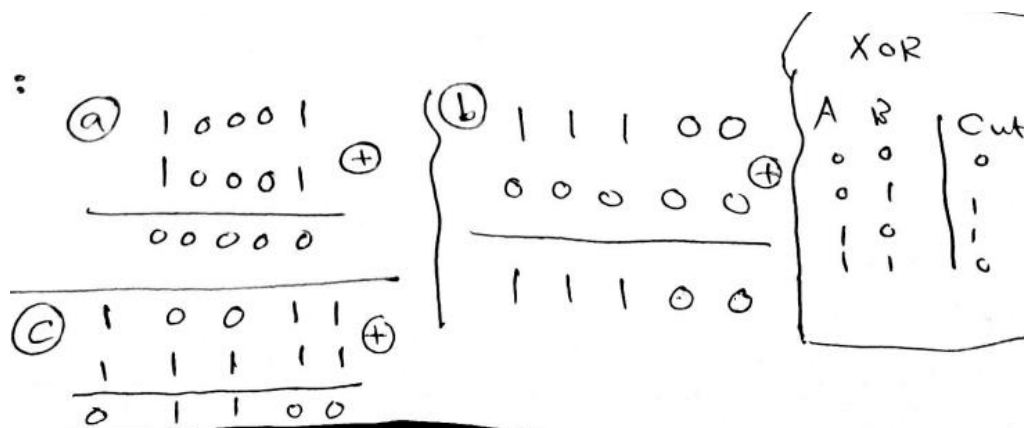
Effect Bits = data rate * burst length

- a. $1500(2 \times 10^{-3}) = 3$ bits effected
b. $12 \times 10^3(2 \times 10^{-3}) = 24$ bits effected
c. 200 bits effected
d. 200000 bits effected

P10-3. Exclusive-OR (XOR) is one of the most used operations in the calculation of codewords. Apply the exclusive-OR operation on the following pairs of patterns. Interpret the results.

- a. $(10001) \oplus (10001)$ b. $(11100) \oplus (00000)$ c. $(10011) \oplus (11111)$

Solution:



P10-4. In Table 10.1, the sender sends dataword 10. A 3-bit burst error corrupts the codeword. Can the receiver detect the error? Defend your answer.

Table 10.1 A code for error detection in Example 10.1

Dataword	Codeword	Dataword	Codeword
00	000	10	101
01	011	11	110

Solution:

The answer depends heavily on what the codeword is corrupted to. If the codeword is corrupted to 010 for example the error will be detected because that codeword is invalid. If the codeword is corrupted to 000 however, the error will not be detected.

P10-5. Using the code in Table 10.2, what is the dataword if each of the following codewords is received?

a. 01011

b. 11111

c. 00000

d. 11011

Table 10.2 Simple parity-check code $C(5, 4)$

Dataword	Codeword	Dataword	Codeword
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

Solution:

a. ERROR

b. ERROR

c. 0000

d. 1101

P10-6. Prove that the code represented by the following codewords is not linear. You need to find only one case that violates the linearity.

$\{(00000), (01011), (10111), (11111)\}$

Solution:

$\{(00000), (01011), (10111), (11111)\}$

Sub:

أخذ XOR بين أول اثنين

1	0	1	1	1
0	1	0	1	1
⊕				
1	1	1	0	0

This is not valid codeword (see table 10.2)

خطأ: linear block code
أي أخذ XOR بين
two codewords
النتيجة تكون
valid codeword

P10-7. What is the Hamming distance for each of the following codewords?

a. $d(10000, 00000)$

b. $d(10101, 10000)$

c. $d(00000, 11111)$

d. $d(00000, 00000)$

Solution:

The Hamming distance between two words (of the same size) is the number of differences between the corresponding bits

- a. 1
- b. 2
- c. 5
- d. 0

P10-8. Although it can be formally proved that the code in Table 10.3 is both linear and cyclic, use only two tests to partially prove the fact:

- a. Test the cyclic property on codeword 0101100.
- b. Test the linear property on codewords 0010110 and 1111111.

Solution:

- a. In a cyclic code, if a codeword is cyclically shifted (rotated), the result is another codeword.
0101100 left shifted = 1011000 which is a **valid codeword** in 10.3,
therefore **cyclic**
- b. 0010110 XOR 1111111 = 1101001 which is a **valid codeword** in 10.3,
therefore **linear**

P10-10. Assuming even parity, find the parity bit for each of the following data units.

- a. 1001011 b. 0001100 c. 1000000 d. 1110111

Solution:

- a. 0
- b. 0
- c. 1
- d. 0

P10-11. A simple parity-check bit, which is normally added at the end of the word (changing a 7-bit ASCII character to a byte), cannot detect even numbers of errors. For example, two, four, six, or eight errors cannot be detected in this way. A better solution is to organize the characters in a table and create row and column parities. The bit in the row parity is sent with the byte, the column parity is sent as an extra byte (Figure 10.23).

Figure 10.23 P10-11

	C1	C2	C3	C4	C5	C6	C7	
R1	1	1	0	0	1	1	1	1
R2	1	0	1	1	1	0	1	1
R3	0	1	1	1	0	0	1	0
R4	0	1	0	1	0	0	1	1
	0	1	0	1	0	1	0	1

a. Detected and corrected

	C1	C2	C3	C4	C5	C6	C7	
R1	1	1	0	0	1	1	1	1
R2	1	0	1	1	1	0	1	1
R3	0	1	1	0	0	1	1	0
R4	0	1	0	1	0	0	1	1
	0	1	0	1	0	1	0	1

b. Detected

	C1	C2	C3	C4	C5	C6	C7	
R1	1	1	0	0	1	1	1	1
R2	1	0	1	0	0	0	1	1
R3	0	1	1	0	0	0	1	0
R4	0	1	0	1	0	0	1	1
	0	1	0	1	0	1	0	1

c. Detected

	C1	C2	C3	C4	C5	C6	C7	
R1	1	0	0	0	1	0	1	1
R2	1	0	1	1	1	0	1	1
R3	0	0	1	1	0	1	1	0
R4	0	1	0	1	0	0	1	1
	0	1	0	1	0	1	0	1

d. Not detected

Show how the following errors can be detected:

- An error at (R3, C3).
- Two errors at (R3, C4) and (R3, C6).
- Three errors at (R2, C4), (R2, C5), and (R3, C4).
- Four errors at (R1, C2), (R1, C6), (R3, C2), and (R3, C6).

Solution:

- In the case of one error, it can be detected and corrected because the two affected parity bits can define where the error is.
- Two errors can definitely be detected because they affect two bits of the column parity. The receiver knows that the message is somewhat corrupted (although not where). It discards the whole message.
- Three errors are detected because they affect two parity bits, one of the column parity and one of the row parity. The receiver knows that the message is somewhat corrupted (although not where). It discards the whole message.
- The last case cannot be detected because none of the parity bits are affected.

Solution:

⇒ dataword ⇒ with (k) bits, [1]
codeword with (n) bits.

• [dataword $\frac{\quad}{\quad}$]
 (k bits) $\uparrow$
 النص
 الزيادة

3) تم إضافة باقي لقسمه (remihder) للـ dataword (k بت) فيحصل
الكود الـ Codeword .

at receiver:

الـ receiver يقسم codeword بنفس الـ divisor

الـ قسم على الـ sender .

لو باقي القسمة 0 -> بلا خطأ -> error

أما لو باقي القسمة 1 -> خطأ -> error

حل سؤال [12] : CRC codeword ?

codeword dataword

$$\Rightarrow \text{divisor} = n - k + 1$$

$$5 = n - 9 + 1$$

$$5 = n - 8$$

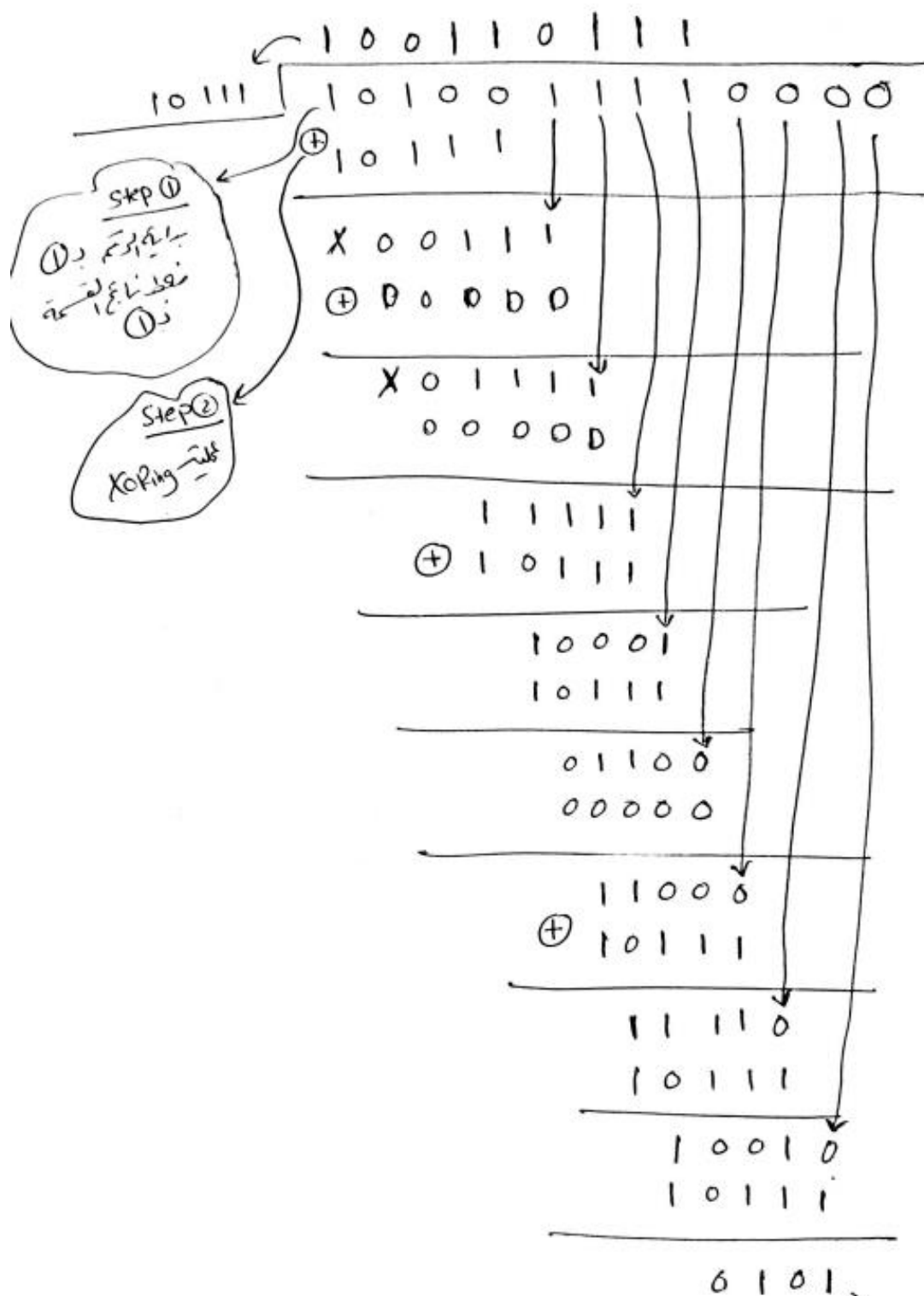
$$n = 13$$

عدد البايتات الـ مطمئن : dataword

$$n - k = 13 - 9 = 4$$

$$\therefore \text{dataword} = 1010011110000$$

↓
الـ زيرو دهن



The code word will be 10100111 0101

↓ ↓
dataword remainder

P10-18. Assume a packet is made only of four 16-bit words $(A7A2)_{16}$, $(CABF)_{16}$, $(903A)_{16}$, and $(A123)_{16}$. Manually simulate the algorithm in Figure 10.17 to find the checksum.

Solution:

$$\begin{array}{r}
 \text{①} \quad \text{①} \quad \text{①} \\
 A \quad 7 \quad A \quad 2 \\
 C \quad A \quad B \quad F \\
 + \quad 9 \quad 0 \quad 3 \quad A \\
 A \quad 1 \quad 2 \quad 3 \\
 \hline
 \end{array}$$

(Note: The handwritten solution shows a carry of 1 from the first column, which is added to the second column. The result of the first addition is 10, where 0 is written and 1 is carried over.)

$$\begin{array}{r}
 \text{②} \quad A \quad 3 \quad B \quad E \\
 \quad \quad \quad \quad \quad \quad 2 + \\
 \hline
 \end{array}$$

(Note: The handwritten solution shows a carry of 1 from the second column, which is added to the third column. The result of the second addition is 12, where 2 is written and 1 is carried over.)

$$\begin{array}{r}
 \boxed{A \quad 3 \quad C \quad 0} \quad \# \\
 \boxed{5 \quad C \quad 3 \quad F} \Rightarrow \text{checksum}
 \end{array}$$

(Note: The handwritten solution shows the final checksum as 5C3F, which is the one's complement of the sum A3C0.)

P10-20. This problem shows a special case in checksum handling. A sender has two data items to send: $(4567)_{16}$ and $(BA98)_{16}$. What is the value of the checksum?

Solution:

$$\begin{array}{r}
 4 \quad 5 \quad 6 \quad 7 \\
 B \quad A \quad 9 \quad 8 \quad + \\
 \hline
 F \quad F \quad F \quad F
 \end{array}$$

(Note: The handwritten solution shows a carry of 1 from the first column, which is added to the second column. The result of the first addition is 11, where 1 is written and 1 is carried over. The result of the second addition is 12, where 2 is written and 1 is carried over. The result of the third addition is 11, where 1 is written and 1 is carried over. The result of the fourth addition is 12, where 2 is written and 1 is carried over. The final result is FFFF.)

$$\boxed{0 \quad 0 \quad 0 \quad 0} \quad \#$$

(Note: The handwritten solution shows the final checksum as 0000, which is the one's complement of the sum FFFF.)

P10-25. In the interleaving approach to FEC, assume each packet contains 10 samples from a sampled piece of music. Instead of loading the first packet with the first 10 samples, the second packet with the second 10 samples, and so on, the sender loads the first packet with the odd-numbered samples of the first 20 samples, the second packet with the even-numbered samples of the first 20 samples, and so on. The receiver reorders the samples and plays them. Now assume that the third packet is lost in transmission. What will be missed at the receiver site?

Solution:

1st Packet: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19 .
 2nd Packet: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20 .
 3rd Packet: 21, 23, 25, 27, 29, 31, 33, 35, 37, 39 .
 (تعبئة جزئية الساتر الرابع) (تعبئة جزئية الساتر الثالث)
 هلفت بجلي انه Receiver بي يتلقاهم برجع رتبهم ، وياتي :
 ←

at receiver

1st: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 .
 2nd: 11, 12, 13, 14, 15, 16, 17, 18, 19, 20 .
 3rd: 21, 22, 23, 24, 25, 26, 27, 28, 29, 30 .
 4rd: 31, 32, 33, 34, 35, 36, 37, 38, 39, 40 .

↓
 كما قلت : جزء من الساتر نفسه وجزء من الساتر الرابع

P10-26. Assume we want to send a dataword of two bits using FEC based on the Hamming distance. Show how the following list of datawords/codewords can automatically correct up to a one-bit error in transmission.

00 → 00000 01 → 01011 10 → 10101 11 → 11110

Sloution:

We earlier discussed the **Hamming** distance for error detection. We said that to detect s errors, the minimum **Hamming** distance should be $d_{\min} = s + 1$. For error detection, we definitely need more distance. It can be shown that to detect t errors, we need to have $d_{\min} = 2t + 1$. In other words, if we want to correct 10 bits in a packet, we need to make the minimum **hamming** distance 21 bits, which means a lot of redundant bits need to be sent with the data. To give an example, consider the famous BCH code. In

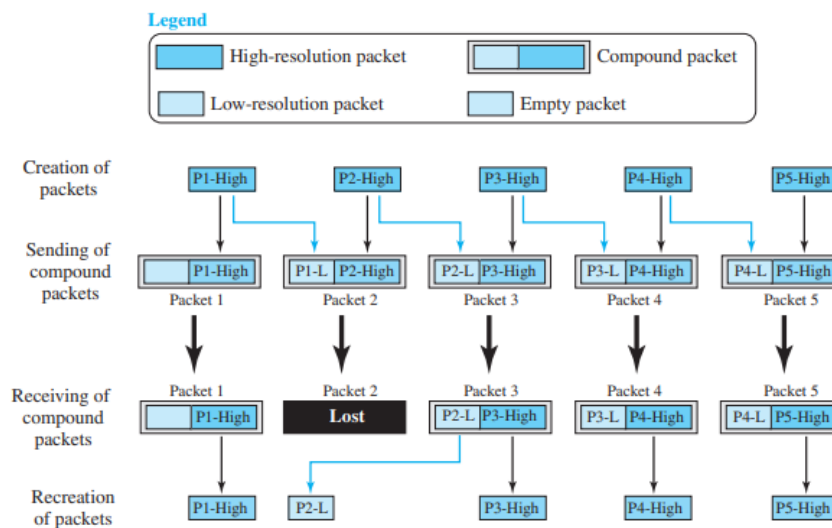
$$d_{\min} = 2t + 1$$

t is the number of errors

$$d_{\min} = 2 \cdot 1 + 1 = 3$$

P10-30. In Figure 10.22, assume we have 100 packets. We have created two sets of packets with high and low resolutions. Each high-resolution packet carries on average 700 bits. Each low-resolution packet carries on average 400 bits. How many extra bits are we sending in this scheme for the sake of FEC? What is the percentage of overhead?

Figure 10.22 *Compounding high- and low-resolution packets*



Solution:

$$N = 100, H=700, L=400$$

$$\# \text{ of extra bits} = (N-1) \cdot L = (100-1) \cdot 400 = 39,600 \text{ bits.}$$

$$\text{Percentage of overhead} = 39600 / (100 \cdot 700 + 99 \cdot 400) \cdot 100\% \approx 36.19\%$$