Network Security M25

Jean-Marc THIRIET

jean-marc.thiriet@univ-grenoble-alpes.fr

http://www.gipsa-lab.grenoble-inp.fr/

~jean-marc.thiriet/lpro/cnms_en.html

………………./lpro/rims_fr.html

M25-1. Errors Detection and Correction

Jean-Marc THIRIET
jean-marc.thiriet@univ-grenoble-alpes.fr

Ojectives of this chapter

- Review the concepts of error coding and detection
- First level of integrity violation detection
- Revise binary and hexadecimal
- Revise the concept of modulo division (remainder of the integer division) widely used in cryptography…

1.1 Error detection and correction

- Frames Transmissions
  - To ensure the receiver has received correctly information
  - Control of the **integrity** of the received data
- Causes of transmission errors
  - Physical aspects
    - Thermal noise
    - Electromagnetic perturbations
    - Distance between two devices (wireless, mobility)
  - Characteristics of the protocols
    - CSMA/CD
    - Mobility (wireless, mobile telephony, mobile objects)

Type of methods

- BER (**Bit error rate**)
  - Single-bit errors
  - Groups of contiguous strings of bit errors (**burst errors**)
- BER : probability P of a single bit being corrupted in a defined time interval
- BER of $P=10^{-3}$ means that, on average, 1 bit in 10^{3} => corrupted
- For a string of N bits => $1-(1-P)^{N}$
  - With $P=10^{-3}$ and $N=10$ =>the probability of the string to be corrupted => 10^{-2}

Choice of the method

- Based on
  - the size of blocks (packets, frames)
  - Probability of errors
- Ex :
  - Block of 1250 bytes with $P=10^{-3}$
    - $1-(1-P)^{N} = 1-(1-10^{-3})^{1250*8} =$ 100 % errors !!
  - Block of 125 bytes with $P=10^{-3}$
    - $1-(1-P)^{N} = 1-(1-10^{-3})^{125*8} =$ 63 % errors ! (more than 1 block among 2 !)
- The length of the block (frame) is too long
- The maximal size of a packet to be transmitted depends on the quality of communications (error rates)

1.1 Hamming distance

- Number of positions having different bit values between two words (the words have the same length)
  - Ex :   1 0 1 1 1 0 1

                   1 **1 0** 1 1 **1** 1

  Here the Hamming distance is 3
  - If the distance between two words is d, d errors can transform one word into another one
- Generally, to detect "d" errors, a code with a "d+1" distance is necessary

1.2 Detector codes

- Vertical Parity (VRC)
- Longitudinal Parity (LRC)

<u>Example: parity control</u>

Even parity : 10110111

parity bit

Odd parity: 10110110

(Odd parity)

	0	1	0	0	0	1	1	**0**
	1	0	1	1	1	0	0	**1**
	1	1	0	1	0	1	0	**1**
LRC	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**

	0	1	0	0	0	1	1	**0**
	1	0	1	1	**0**	0	0	**1**
	1	1	0	1	0	1	0	**1**
LRC	**1**	**1**	**0**	**1**	**0**	**1**	**0**	**1**

Parity error in one line and one column

Homework 1, exercise 0

- Monday 3rd October
- 1.1
- 10110
- 11011
- 11001
- Provide the parity (even parity) bits both horizontically and vertically
- 1.2 A receiver receives (even parity) this message
- 111001
- 101110
- 001111
- 001000
- Are there transmission errors? If yes, is it possible to correct the transmission errors? What is the corrected message ? Make an extra comment.

Homework 1, exercise 1, CORRECTION part 1

- exercise 1.1
- 10110**1**
- 11011**0**
- 11001**1**
- **101000**
- Provide the parity (even parity) bits both horizontically and vertically

- A receiver receives (even parity) this message (of course the receiver has no information about what was the message sent by the transmitter)
- 111001 **correct**
- 101110 **correct**
- 001111 **correct**
- 001000 **Uncorrect**
- **There is an error in the last line (last packet of 6 bits), but at this stage we don’t know where is this error**

Homework 1, exercise 1, CORRECTION part 2

- **Checking of columns**
- **cucccc => There is an error second column (c=correct ; u=uncorrect)**
- Are there transmission errors? If yes, is it possible to correct the transmission errors?
- **If we combine both known information, vertically and horizontally, we know which bit is wrong.**
- 111001
- 101110
- 001111
- 0**0**1000
- **Because this bit is wrong, we will alternate it, which means change this 0 into 1. We just achieved a correction, without needing a retransmission from the transmitter**
- What is the corrected message ?
- **The correct message initially sent is so:**
- 111001
- 101110
- 001111
- 0**1**1000
- Make an extra comment.
- **What is interesting to notice here is that the wrong bit was not a bit from the data, but a parity bit. This is just to be aware that these security mechanisms are taking account of all the bits in the same way: data bits and control bits… We may also notice that there are actually no transmission errors in the data bits in this example…**

1.3 Modulo 2 polynomial arithmetics

- A polynomial represents a bit suite:

110001 -------- > x^5+x^4+1

- Addition and soustraction **without any carry**: EXCLUSIVE OR

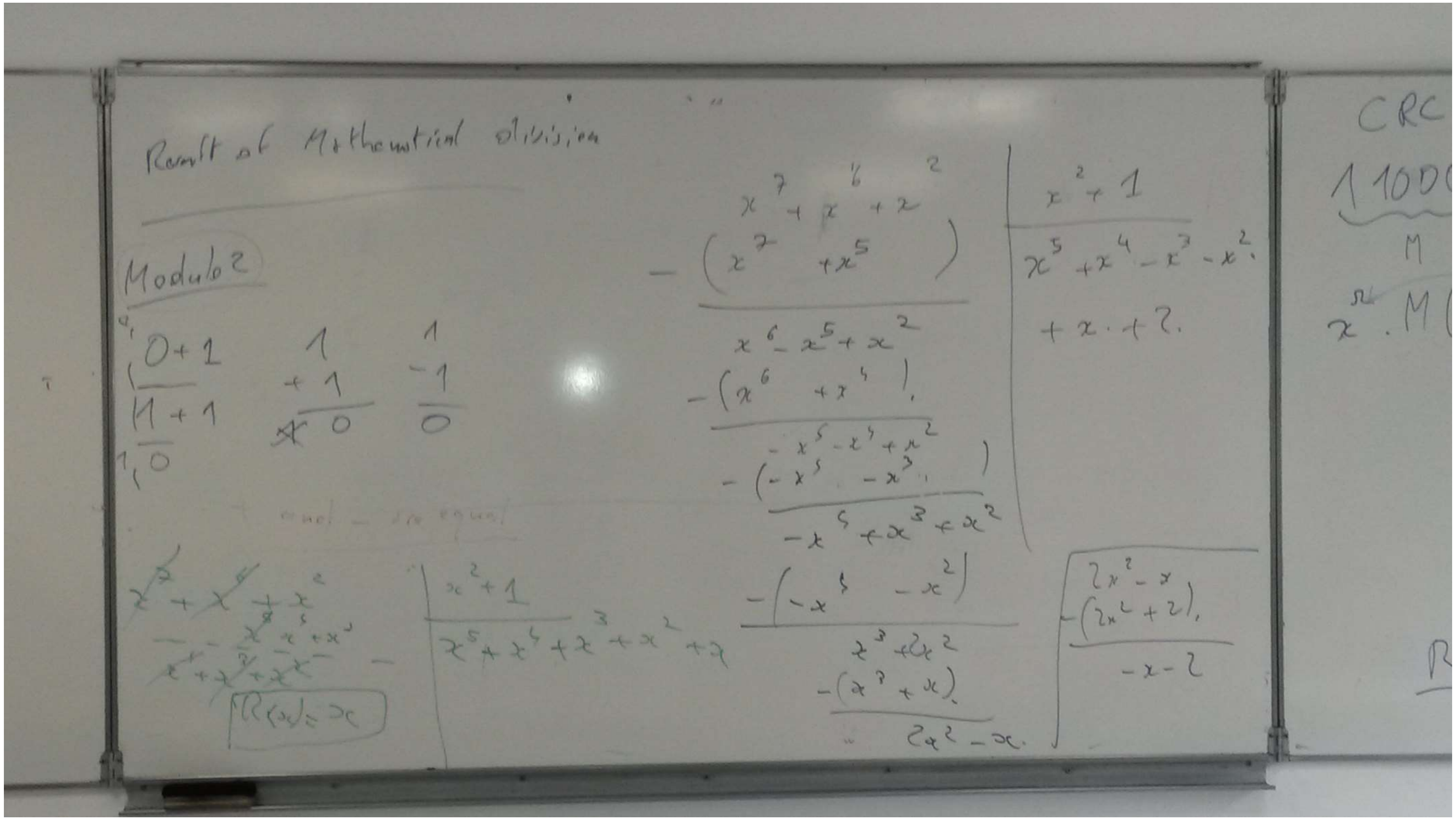
	1 0 0 1 1 0 1 1		$x^7 + x^4 + x^3 + x + 1$
+	1 1 0 0 1 0 1 0	+	$x^7 + x^6 + x^3 + x$
=	0 1 01 0 0 0 1	=	$x^6 + x^4 + \quad + 1$

- **Cyclic codes: CRC (Cyclic Redundandy Check)**

- **Transmitter**
  - **Data (message): bits suite represented by M(x)**
  - **The transmitter divides x^r.M(x) by G(x) with G(x) (degree r, r+1 bits), generator polynomial**
  - **x^r.M(x) = G(x).Q(x) + R(x), the size (number of bits) of R(x) is exactly r bits**
  - **Let's transmit the frame T(x) = x^r.M(x) + R(x)**
- **Receiver**
  - **The receiver divides T(x) by G(x) and the remainder should be 0**

1.3 Normalised generator polynomials

- **CRC12 generator polynomial = $x^{12} + x^{11} + x^3 + x^2 + x + 1$**
- **CRC16 generator polynomial = $x^{16} + x^{15} + x^2 + 1$**
- **CRC-CCITT generator polynomial = $x^{16} + x^{12} + x^5 + 1$ (V41 recommendation used in HDLC protocol)**
- **CRC-32 (Ethernet) generator polynomial : = $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + X + 1$**



Result of Mathematical division
Modulo 2
0 + 1
1 + 1
1 0
1 + 1 0
1 − 1 0
x^7 + x^6 + x^2
− (x^7 + x^5)
x^6 − x^5 + x^2
− (x^6 + x^4)
− x^5 − x^4 + x^2
− (− x^5 − x^3)
− x^4 + x^3 + x^2
− (− x^4 − x^2)
x^3 + 2x^2
− (x^3 + x)
2x^2 − x
x^2 + 1
x^5 + x^4 − x^3 − x^2 + x + 2
2x^2 − x
− (2x^2 + 2)
− x − 2
x^7 + x^6 + x^2
x^5 + x^4 + x^3 + x^2 + x
R(x) = x
CRC
1 100
M
x^n . M

1.3 Example of a CRC calculation by the transmitter

- M : 1101011011
- Generator polynomial G : 10011,
- What is the size of the remainder?
- Demonstrate, by using the polynomial division, that the remainder is ????
- The transmitted message is so:
  - 1101011011????

1.3 Example of a CRC calculation by the transmitter

- M : 1101011011
- Generator polynomial G : 10011, so the degree is 4=> the degree of the remainder will be 3 (4-1) so the size will be 4 bits
- Demonstrate, by using the polynomial division, that the remainder is 1110
- The transmitted message is so:
  - 11010110111110

1.3 Role of the receiver ?

- It receives 11010110111110 and divides by the generator polynomial 10011
  - ⇒The remainder should be 0,
  - ⇒If it is not the case, each bit which is at 1 in the remainder corresponds to an erroneous bit
    - ! Under the condition to respect the rule ! If the remainder contains n bits, it is possible at best to detect and correct n errors, here 4 !
- Ex : I receive 11010010111110 with G(x)=10011 => Is there an error?

Homework 1 (2/2), due 3rd October 2022

- Exercises 1 and 2 and 6, page 5 in the textbook

- **Ex M25-1.4** We want to transmit the word "AB". Each character is encoded with 7 bits (ASCII) and an 8th bit is added on the right as parity bit (even parity). Define the content to be transmitted as a binary form.
- Then we will calculate the CRC for the defined binary suite. Calculate the corresponding CRC using $x^8 + x^3 + 1$ as a generator polynomial.

Homework 1 (2/2), CORRECTION Ex.1 p. 4 Part 1

- In the ASCII code, the chain “INT” is encoded with the 3 following 7-bits characters (the ASCII code is given as hexadecimal):
- I → 49 **=> 1001001**
- N → 4E **=> 1001110**
- T → 54 **=> 1010100**
- Provide the transmitted message, by adding an even parity both for VRC and LRC. We consider that the parity bit is on the right.
- **10010011**
- **10011100**
- **10101001**
- **10100110**

Homework 1 (2/2), CORRECTION Ex.1 p. 4 Part 2

- Same question with an odd parity
- **10010010**
- **10011101**
- **10101000**
- **01011000**

- **! When we compare even and odd parities, all the bits are reverse except the bit at the bottom on the right (intersection between LRC and VRC) !**

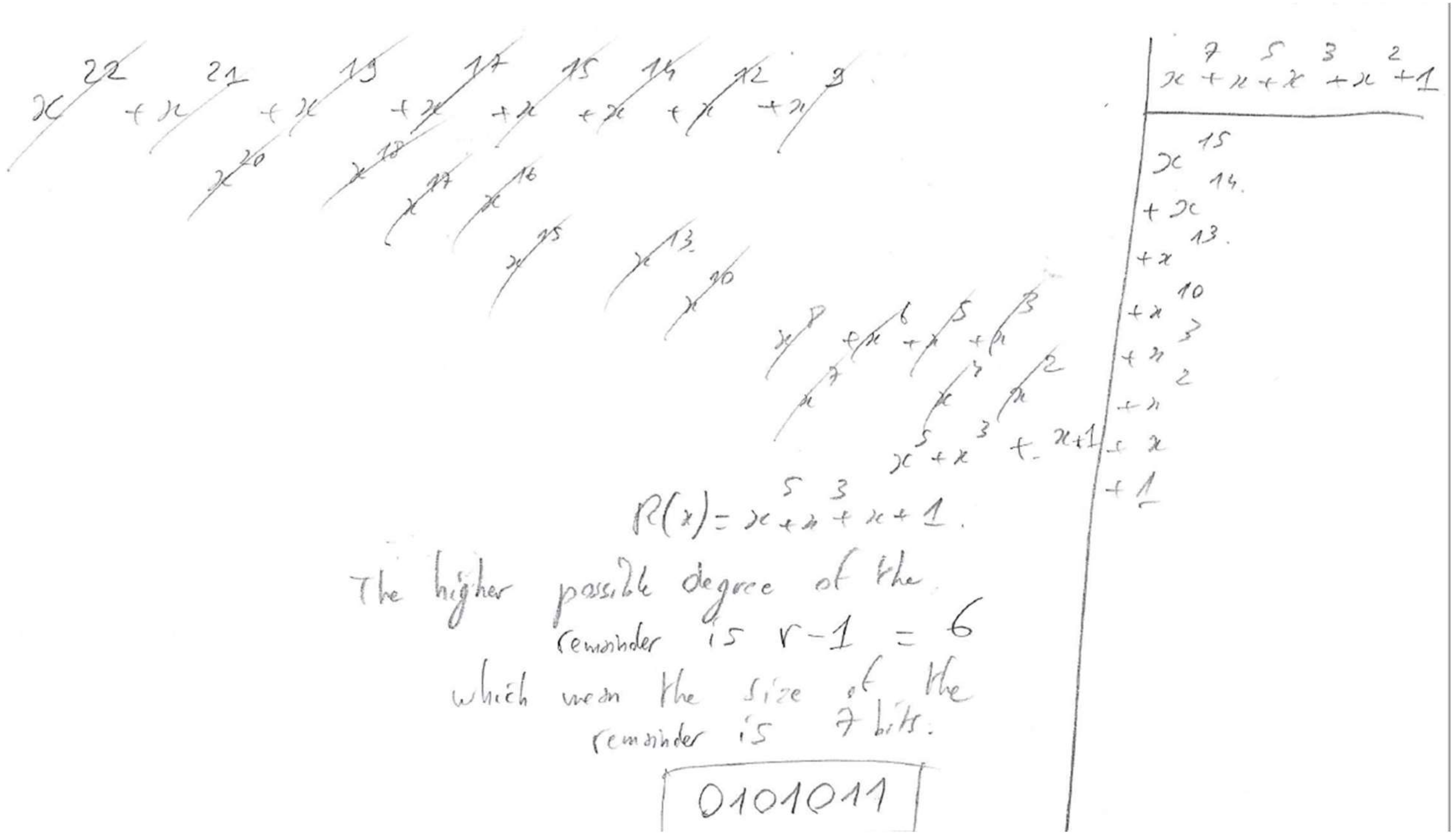
Homework 1 (2/2), CORRECTION Ex.2 p. 4 Part 1

- We want to transmit the following message: 11010101 10100100 using the following generator suite: 10101101.
- Give the result of the CRC, explain what is the size of the CRC and why.

Homework 1 (2/2), CORRECTION Ex.2 p. 4 Part 2

- $M(x) = x^{15} + x^{14} + x^{12} + x^{10} + x^8 + x^7 + x^5 + x^2$
- $G(x) = x^7 + x^5 + x^3 + x^2 + 1$, degree 7
- So $x^r.M(x) = x^{22} + x^{21} + x^{19} + x^{17} + x^{15} + x^{14} + x^{12} + x^9$
- We have now to make the division $x^r.M(x)$ divided by $G(x)$

Homework 1 (2/2), CORRECTION Ex.2 p. 4 Part 3



$\cancel{x^{22}} + \cancel{x^{21}} + \cancel{x^{19}} + \cancel{x^{17}} + \cancel{x^{15}} + \cancel{x^{14}} + \cancel{x^{12}} + \cancel{x^{3}}$ | $x^7 + x^5 + x^3 + x^2 + 1$

$\cancel{x^{20}}$ $\cancel{x^{18}}$ $\cancel{x^{17}}$ $\cancel{x^{16}}$ $\cancel{x^{15}}$ $\cancel{x^{13}}$ $\cancel{x^{10}}$

$\cancel{x^{8}} + \cancel{x^{6}} + \cancel{x^{5}} + \cancel{x^{3}}$

$\cancel{x^{7}}$ $\cancel{x^{3}}$ $\cancel{x^{2}}$

$x^5 + x^3 + x + 1$

Quotient: $x^{15} + x^{14} + x^{13} + x^{10} + x^3 + x^2 + x + 1$

$R(x) = x^5 + x^3 + x + 1.$

The higher possible degree of the remainder is $r-1 = 6$

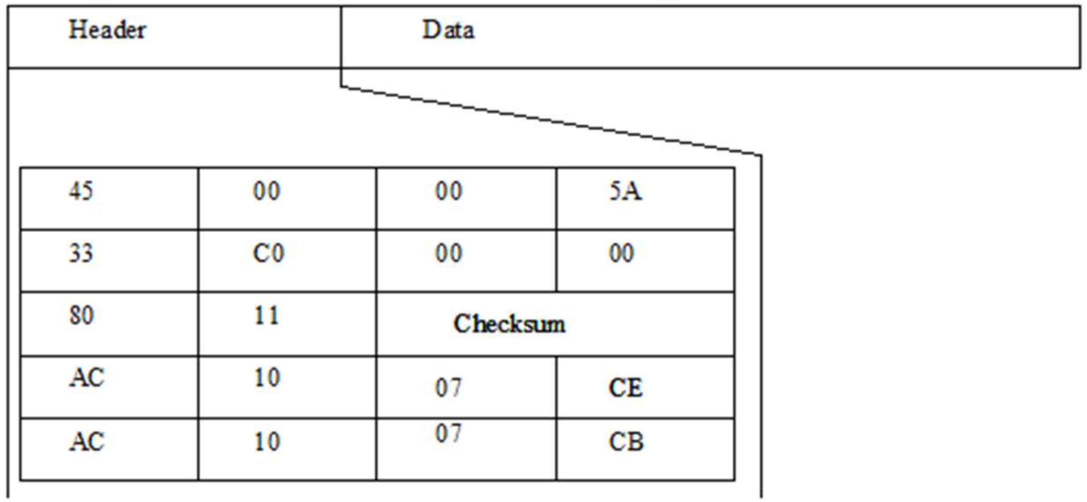
which mean the size of the remainder is 7 bits.

$\boxed{0101011}$

Homework 1 BIS, CORRECTION Ex.6 p. 4 Part 1

- Exercise 6, page 4 in the textbook

A packet is constituted of a 20-byte header and some data following the structure below:



45	00	00	5A
33	C0	00	00
80	11	Checksum	
AC	10	07	CE
AC	10	07	CB

Bytes 11 and 12 correspond to the checksum and the header.
Bytes 13 to 16 represent the hexadecimal address of the transmitter.
Bytes 17 to 20 represent the hexadecimal address of the receiver.

a) Calculate the checksum,
b) Determine IP source and destination addresses in a decimal notation

Homework 1 BIS, CORRECTION Ex.6 p. 4 Part 2

- Calculate the checksum
- 4500 + 005A + 33C0 + 0000 + 8011 + AC10 + 07CE + AC10 + 07CB
- 455A + 33C0 + 8011 + B3DE + B3DB
- 791A+133EF + B3DB
- 1AD09 + B3DB = 260E4

- Then we sum the carry: 60E4 + 2 = 60E6

- Then we should reverse, we can do it directly in hexadecimal (0 is the reverse of F, 1 from E, 2 from D …, 7 from 8) => you can see the penultimate slide of this présentation
- Or we can use the binary form
- 60E6 = 0110 0000 1110 0110
- The reverse is 1001 1111 0001 1001 = 9F19
- So the Cheksum is 9F19

Homework 1 BIS, CORRECTION Ex.6 p. 4 Part 3

- Determine IP source and destination addresses in a decimal notation
- IP source AC1007CE =>
  - ACh corresponds to 172d
  - 10h corresponds to 16d
  - 07h corresponds to 7d
  - CEh corresponds to 206d
- So the IP source is 172.16.7.206
- With the same strategy the IP destination AC1007CB will corresponds to 172.16.7.203

1.4 Checksums

- RFC 791:
  - *The checksum field is the 16-bit one's complement of the one's complement sum of all 16-bit words in the header.*
- IP and TCP headers used in Internet networks uses a simpler method (easier to implement): CHECKSUM (*somme de contrôle*).
- From RFC 1071 (RFC (« standards ») which defines calculation methods for checksum in IP environment)
  - Take a 16-bits word
  - Calculate the *one's complement sum* (! sum to which the carry is directly added to the result !)
  - At the end, determine the *one's complement* of the result (! Inversion of all the bits of the result !)
  - Then this result should be placed in the checksum field

1.4 Checksums

- The receiver achieves the same operation (checksum received included) => the result should be 0000 after the final inversion, which means No Errors!
- Ex : Calculation of the 4-bit checksum of 1110 0011

- 0xE + 0x3
  - Normal calculation 0xE + 0x3 = 0x11 (0b10001)
  - Calculate the 4-bits *one's complement sum* of 0xE + 0x3 = 0x2 = $(0010)_2$
- Ex : The *one's complement* of this result is $(1101)_2$, 0xD

1.4 Checksums

- <u>Example :</u>
- Calculate the checksum for this packet :   01 00 F2 03 F4 F5 F6 F7 00 00
  (00 00 is the checksum field)
- Let’s organise the packet as 16 bits-words:
- 0100 F203   F4F5   F6F7
- Calculate the sum:
- 0100 + F203 + F4F5 + F6F7

1.4 Checksums

- Example :
- Calculate the checksum for this packet : 01 00 F2 03 F4 F5 F6 F7 00 00 (00 00 is the checksum field)
- Let’s organise the packet as 16 bits-words:
- 0100 F203 F4F5 F6F7
- Calculate the sum:
- 0100 + F203 + F4F5 + F6F7 = 0002 DEEF (This sum is stored in a 32-bit word)
- Add the carry in order to get the 1-complemented sum:
- DEEF + 002 = DEF1

1.4 Checksums

- Example :
- Calculate the checksum for this packet : 01 00 F2 03 F4 F5 F6 F7 00 00 (00 00 is the checksum field)
- Let's organise the packet as 16 bits-words:
- 0100 F203 F4F5 F6F7
- Calculate the sum:
- 0100 + F203 + F4F5 + F6F7 = 0002 DEEF (This sum is stored in a 32-bit word)
- Add the carry in order to get the 1-complemented sum:
- DEEF + 002 = DEF1
- The checksum is the 1-complement of the result:
- ~DEF1 = 210E
- The sent packet includes the checksum:
- 01 00 F2 03 F4 F5 F6 F7 21 0E
- For the receiver:
- 0100+F203+ F4F5+ F6F7 + 210E =

1.4 Checksums

- Example :
- Calculate the checksum for this packet : 01 00 F2 03 F4 F5 F6 F7 00 00 (00 00 is the checksum field)
- Let's organise the packet as 16 bits-words:
- 0100 F203 F4F5 F6F7
- Calculate the sum:
- 0100 + F203 + F4F5 + F6F7 = 0002 DEEF (This sum is stored in a 32-bit word)
- Add the carry in order to get the 1-complemented sum:
- DEEF + 002 = DEF1
- The checksum is the 1-complement of the result:
- ~DEF1 = 210E
- The sent packet includes the checksum:
- 01 00 F2 03 F4 F5 F6 F7 21 0E
- For the receiver:
- 0100+F203+ F4F5+ F6F7 + 210E = 0002 FFFD
- FFFD + 0002 = FFFF
- After inversion => 0000
- It is correct !

1.5 Conclusion

- BER
- Size of blocks (frames)
- Parity better for random distribution of errors
- CRC better for bursts of errors
- Estimation that 999 errors out of 1000 are corrected thanks to CRC
  - If the BER on the physical medium is 10^{-6}, it becomes 10^{-9} thanks to the CRC algorithm

Fast Conversion decimal/binary/hexadecimal

- Decimal => binary
  - 125

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	1	1	1	0	1

Fast Conversion decimal/ binary/ hexadecimal

Base 2	Base 16	Base 10
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

Half a byte

Références

- Transmissions et réseaux, S. Lohier & D. Présent, Dunod, Paris, 2003.