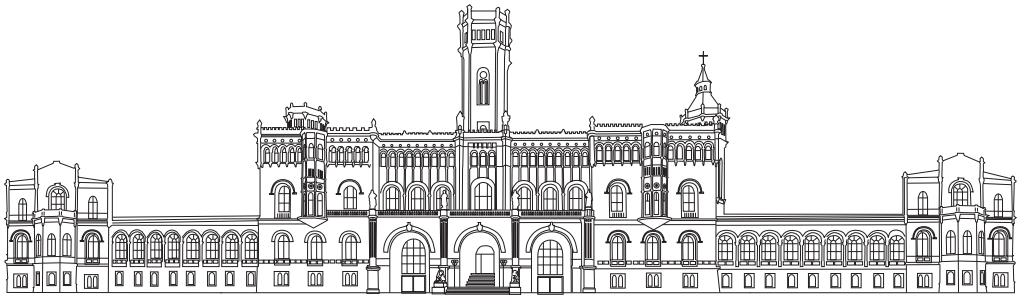
# Rechnernetze – Computer Networks

## Problem Set 3: Forward Error Correction & Error Detection

Markus Fidler, Mark Akselrod, Lukas Prause



Institute of Communications Technology
Leibniz Universität Hannover

April 22, 2024

# Outline



Channel Capacity

Forward Error Correction

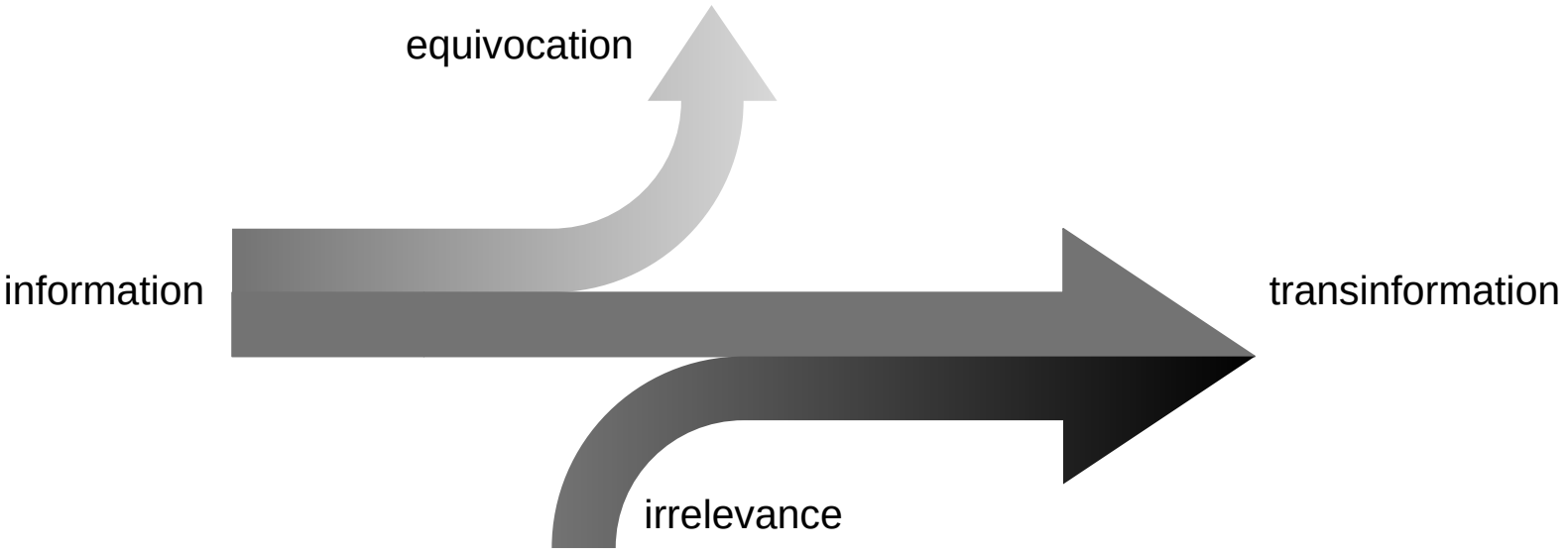
Error Detection

# Transinformation



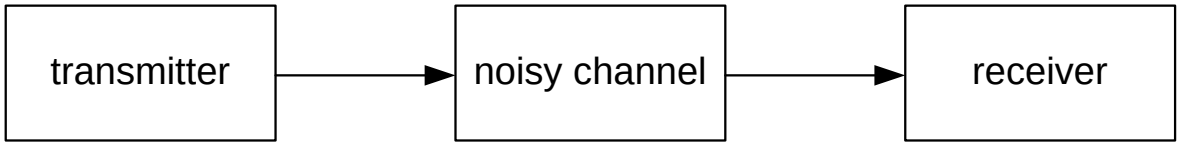
Without formal definition:

- equivocation: amount of information lost
- irrelevance: amount of (useless) information added
- transinformation: amount of the original information received



# AWGN channel



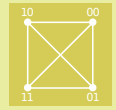



Transmissions over a channel are subject to noise. A useful model is the additive white Gaussian noise (AWGN) channel

Noise distorts transmitted symbols causing symbol resp. bit errors at the receiver resulting in a loss of information.

How can transmission errors be corrected?

- Triple modular redundancy
  - Send three independent copies of the data.
  - correction using majority decision
- Linear block codes
  - each data word is mapped to a unique (longer) codeword (e.g. 4 bit to 7 bit) thereby creating redundancy
  - the receiver compares the received codeword with the set of possible codewords and corrects/detects error if possible




A sender uses triple modular redundancy to send a packet. The receiver receives the following 3 versions of the packet:

1: 10011001
2: 10111001
3: 10011001

What can be said about the original packet?

- The original packet was most likely 10011001
  - can be corrected by majority decision
- With a smaller probability it could also be 10111001
  - can be detected
- With an even smaller probability it could anything else (e.g. 00011001)
  - cannot be detected

# (7,4)-Hamming code



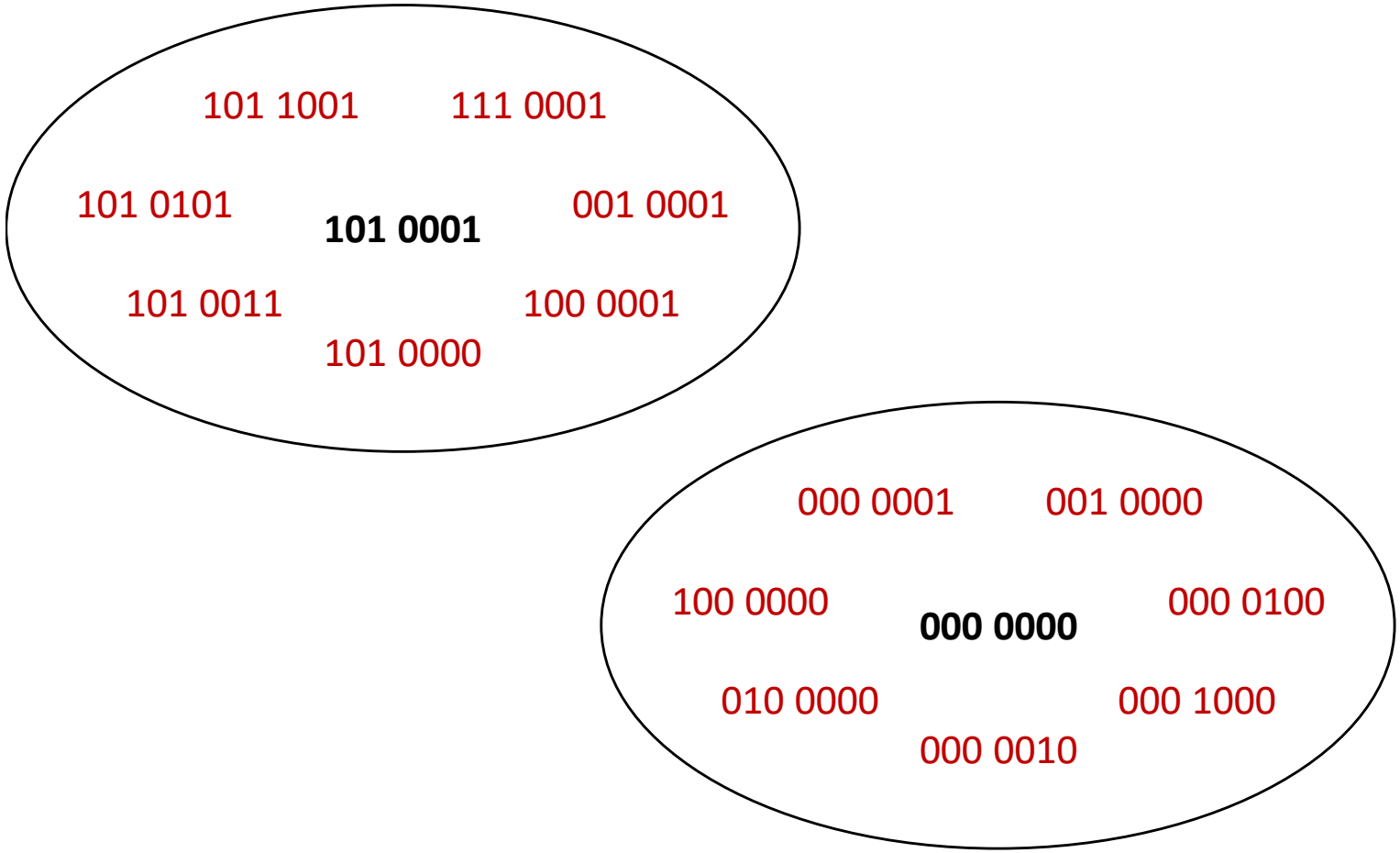
Tabelle: (7,4)-Hamming code

| data word | codeword | data word | codeword |
|---|---|---|---|
| 0000 | 000 0000 | 0001 | 101 0001 |
| 0010 | 111 0010 | 0011 | 010 0011 |
| 0100 | 011 0100 | 0101 | 110 0101 |
| 0110 | 100 0110 | 0111 | 001 0111 |
| 1000 | 110 1000 | 1001 | 011 1001 |
| 1010 | 001 1010 | 1011 | 100 1011 |
| 1100 | 101 1100 | 1101 | 000 1101 |
| 1110 | 010 1110 | 1111 | 111 1111 |

Note that the codewords differ in at least three bit positions.

# Intuition






Each valid codeword can be surrounded by a protecting shell of invalid codewords that each differ in one single bit position.

# Hamming distance



The Hamming distance specifies the number of bit positions at which two codewords differ, e.g. the two codewords $0000000$ and $1010001$ have a Hamming distance of $d = 3$.

The minimal Hamming distance of a code is the minimum of the distance between any two codewords. In case of the (7,4)-Hamming code it is $d_{\min} = 3$.

- the code can detect $d_{\min} - 1 = 2$ bit errors
- the code can correct $\lfloor (d_{\min} - 1)/2 \rfloor = 1$ bit errors

# Modulo-2 arithmetic



Coding frequently uses modulo-2 arithmetic (Galois field 2)

- addition $\oplus$ coincides with the XOR operation
- multiplication $\odot$ coincides with the AND operation

| $\oplus$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| $\odot$ | 0 | 1 |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |

The (7,4)-Hamming code (and other block codes as well) is determined by its generator matrix $\mathbf{G}$.

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Codewords can be generated by modulo-2 matrix multiplication

$$\mathbf{v} = \mathbf{u} \odot \mathbf{G}$$

where $\mathbf{u}$ is the data word and $\mathbf{v}$ the corresponding codeword.

What is the (7,4)-Hamming code for the data word (1 0 1 0)?

$$\begin{pmatrix}1 & 0 & 1 & 0\end{pmatrix} \odot \begin{pmatrix}1 & 1 & 0 & 1 & 0 & 0 & 0\\ 0 & 1 & 1 & 0 & 1 & 0 & 0\\ 1 & 1 & 1 & 0 & 0 & 1 & 0\\ 1 & 0 & 1 & 0 & 0 & 0 & 1\end{pmatrix} = \begin{pmatrix}a & b & c & 1 & 0 & 1 & 0\end{pmatrix}$$

The data part can generally be read from the last four bits of the codeword, since the last four columns of the generator matrix are the identity matrix. Codes with this property are called systematic.

- the first three bits of the codeword are parity bits
- the last four bits of the codeword are the data bits

$$a = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix} \odot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = 1 \oplus 0 \oplus 1 \oplus 0 = 0$$

$$b = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix} \odot \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix} = 1 \oplus 0 \oplus 1 \oplus 0 = 0$$

$$c = \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix} \odot \begin{pmatrix} 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = 0 \oplus 0 \oplus 1 \oplus 0 = 1$$

The (7,4)-Hamming code for the data word (1 0 1 0) is

$$\begin{pmatrix}1 & 0 & 1 & 0\end{pmatrix} \odot \begin{pmatrix}1 & 1 & 0 & 1 & 0 & 0 & 0\\ 0 & 1 & 1 & 0 & 1 & 0 & 0\\ 1 & 1 & 1 & 0 & 0 & 1 & 0\\ 1 & 0 & 1 & 0 & 0 & 0 & 1\end{pmatrix} = \begin{pmatrix}0 & 0 & 1 & 1 & 0 & 1 & 0\end{pmatrix}$$

# Syndrome decoding

The parity check can be formulated in matrix notation as

$$\mathbf{s} = \mathbf{v} \odot \mathbf{H}$$

where the parity check matrix is

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}$$

In case of error free transmission the syndrome is $\mathbf{s} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$.

What is the syndrome of the codeword $\begin{pmatrix}0 & 0 & 1 & 0 & 1 & 0 & 1\end{pmatrix}$?

$$\begin{pmatrix}0 & 0 & 1 & 0 & 1 & 0 & 1\end{pmatrix} \odot \begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 1 & 0\\ 0 & 1 & 1\\ 1 & 1 & 1\\ 1 & 0 & 1\end{pmatrix} = \begin{pmatrix}1 & 1 & 1\end{pmatrix}$$

$\Rightarrow$ a transmission error occured!

# Syndrome table

The (7,4)-Hamming code has the nice property that any single bit error results in a unique syndrome such that single bit errors can be corrected.

Denote $\mathbf{r}$ the codeword at the receiver. The syndrome table provides the position of single bit errors

| bit error position | $r_0$ | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ |
|---|---|---|---|---|---|---|---|
| syndrome $\mathbf{s}$ | 100 | 010 | 001 | 110 | 011 | 111 | 101 |

The attempt to correct a bit error fails, however, if more than one bit is erroneous. In this case:

- the syndrome may but need not be unequal zero
- if nonzero the syndrome cannot indicate whether there are one or more bit errors

Assuming only 1 bit error occured in $\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$, how can it be corrected?

$\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$ has the syndrome $\begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$

Using the syndrome table

| bit error position | $r_0$ | $r_1$ | $r_2$ | $r_3$ | $r_4$ | $r_5$ | $r_6$ |
|---|---|---|---|---|---|---|---|
| syndrome s | 100 | 010 | 001 | 110 | 011 | 111 | 101 |

we see that the error occured in bit $r_5$

$\Rightarrow$ The corrected codeword is $\begin{pmatrix} 0 & 0 & 1 & 0 & 1 & \color{red}{1} & 1 \end{pmatrix}$

If we decode a codeword and get the syndrome $\begin{pmatrix}0 & 0 & 0\end{pmatrix}$, does it mean that there is no error in the received codeword?

No, it is possible that multiple bit errors have occured and we received the wrong codeword.

$$\begin{pmatrix}0 & 0 & 1 & 0 & 1 & 1 & 1\end{pmatrix}$$
$$\begin{pmatrix}1 & 0 & 1 & 1 & 1 & 0 & 0\end{pmatrix}$$

# Frame check sequence

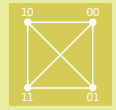


- Hamming-(7,4) code can correct 1 bit error in any 4 bit data word
- bit errors occur, however, typically in bursts, i.e. several consecutive bits are erroneous
- data frames have a trailer that contains parity bits, aka frame check sequence (FCS), for error detection

# Internet checksum

Internet protocols (IP, UDP, TCP) use a simple checksum to detect bit errors:

- at the sender
  - the frame is divided into 16 bit words
  - the 16 bit checksum field is initialized using all 0s
  - the one's complement sum of all 16 bit words of the frame including the checksum field is computed
  - the one's complement of the result is stored as the checksum
- at the receiver
  - the one's complement sum of all 16 bit words of the frame including the checksum field is computed
  - the one's complement of the result is
    - all 0s if no bit error occurred (also possible if multiple bit errors occurred)
    - otherwise an error has occurred




What is the checksum of the frame 010000111101?

1. Divide the frame into 4 bit words (to keep the example simple - normally 16 bit words) $\Rightarrow$ 0100 0011 1101

2. Calculate the sum of the resulting words and the initial checksum

$\Rightarrow 0100 + 0011 + 1101 + 0000$

| | | | | | | |
|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | | word one |
| | 0 | 0 | 1 | 1 | + | word two |
| | 0 | 1 | 1 | 1 | | sum |
| | 1 | 1 | 0 | 1 | + | word three |
| 1 | 0 | 1 | 0 | 0 | | sum |
| | 0 | 1 | 0 | 1 | | carry around |
| | 0 | 0 | 0 | 0 | + | initial checksum |
| | 0 | 1 | 0 | 1 | | sum |
| | 1 | 0 | 1 | 0 | | one's complement |

$\Rightarrow$ We send the frame 0100 0011 1101 1010

How would the receiver check whether there were any bit errors in the received frame 0100001111011010?

1. Divide the frame into 4 bit words ⇒ 0100 0011 1101 1010

2. Calculate the sum of the resulting words 0100 + 0011 + 1101 + 1010

| | | | | | | |
|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | | word one |
| | 0 | 0 | 1 | 1 | + | word two |
| | 0 | 1 | 1 | 1 | | sum |
| | 1 | 1 | 0 | 1 | + | word three |
| 1 | 0 | 1 | 0 | 0 | | sum |
| | 0 | 1 | 0 | 1 | | carry around |
| | 1 | 0 | 1 | 0 | + | checksum |
| | 1 | 1 | 1 | 1 | | sum |
| | 0 | 0 | 0 | 0 | | one's complement |

What kind of error can be detected using the Internet checksum?

⇒ Single bit errors, e.g.:

0100 0011 1101 and
0101 0011 1101

have different checksums

⇒ Burst errors, e.g.:

0100 0011 1101 and
0111 1011 1101

have different checksums

What kind of error cannot be detected using the Internet checksum?

$\Rightarrow$ An even number of bit flips in the same position, e.g.:

0100 0011 1101 and
0101 0011 1100

both have the checksum 1010

# Polynomial codes

Polynomial codes are more efficient than the Internet checksum.

- bit strings are viewed as polynomials with coefficients 0 and 1
- e.g. the string 100101 has polynomial $x^5 + x^2 + x^0$

Arithmetic is done modulo-2

- no carries in case of addition: $1 \oplus 1 = 0$
- no borrows in case of subtraction: $0 \ominus 1 = 1$
- hence both addition and subtraction are identical: XOR

Generation of the checksum

- sender and receiver agree on a generator polynomial $G(x)$ of degree $r$, e.g. the CRC-4 polynomial with $r = 4$ is $x^4 + x^1 + x^0$
- the data frame of $m$ bits is viewed as polynomial $M(x)$
  - the sender appends $r$ zeros at the end of $M(x)$
  - the resulting polynomial $x^r M(x)$ has $m + r$ bits
- the sender divides $x^r M(x)$ modulo-2 by $G(x)$
- the sender subtracts the division remainder from $x^r M(x)$ resulting in the polynomial $T(x)$
- from the construction of $T(x)$ it follows that $T(x)$ is divisible by $G(x)$ (with zero remainder)

Given the CRC-4 generator polynomial $G(x) = x^4 + x^1 + x^0$ and the data frame represented by the polynomial
$M(x) = x^7 + x^6 + x^5 + x^3 + x^0$

What does the transmitted polynomial $T(x)$ look like?

- $G(x) = x^4 + x^1 + x^0$ that is $10011$
- $M(x) = x^7 + x^6 + x^5 + x^3 + x^0$ that is data frame $11101001$ with appended zeros $111010010000$

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | | | | | | | |
| | 1 | 1 | 1 | 0 | 0 | | | | | | |
| | 1 | 0 | 0 | 1 | 1 | ↓ | | | | | |
| | | 1 | 1 | 1 | 1 | 0 | | | | | |
| | | 1 | 0 | 0 | 1 | 1 | ↓ | | | | |
| | | | 1 | 1 | 0 | 1 | 1 | | | | |
| | | | 1 | 0 | 0 | 1 | 1 | ↓ | | | |
| | | | | 1 | 0 | 0 | 0 | 0 | | | |
| | | | | 1 | 0 | 0 | 1 | 1 | ↓ | ↓ | ↓ |
| | | | | | | | 1 | 1 | 0 | 0 | 0 |
| | | | | | | | 1 | 0 | 0 | 1 | 1 |
| | | | | | | | | 1 | 0 | 1 | 1 |

- remainder $1011$
- substract division remainder from $x^r M(x)$ resulting in $T(x)$
- transmitted polynomial $T(x) = 111010011011$

Error detection from the checksum

- the sender transmits the polynomial $T(x)$ (divisible by $G(x)$)
- during transmission errors occur written as polynomial $E(x)$
- the receiver receives polynomial $T(x) \oplus E(x)$
- the receiver divides $T(x) \oplus E(x)$ modulo-2 by $G(x)$
  - $(T \oplus E)/G = T/G \oplus E/G = E/G$ since $T/G = 0$
  - a non-zero remainder of $E/G$ indicates transmission errors

Assume you received the polynomial

$$T(x) \oplus E(x) = x^{11} + x^{10} + x^9 + x^7 + x^4 + x^1.$$

Given the CRC-4 generator polynomial $G(x) = x^4 + x^1 + x^0$, is $E(x) = 0$?

- $G(x) = x^4 + x^1 + x^0$ that is $10011$
- $T(x) \oplus E(x) = x^{11} + x^{10} + x^9 + x^7 + x^4 + x^1$ that is received frame $111010010010$

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 1 | ↓ | | | | | | |
| | 1 | 1 | 1 | 0 | 0 | | | | | | |
| | 1 | 0 | 0 | 1 | 1 | ↓ | | | | | |
| | | 1 | 1 | 1 | 1 | 0 | | | | | |
| | | 1 | 0 | 0 | 1 | 1 | ↓ | | | | |
| | | | 1 | 1 | 0 | 1 | 1 | | | | |
| | | | 1 | 0 | 0 | 1 | 1 | ↓ | | | |
| | | | | 1 | 0 | 0 | 0 | 0 | | | |
| | | | | 1 | 0 | 0 | 1 | 1 | ↓ | ↓ | ↓ |
| | | | | | | | 1 | 1 | 0 | 1 | 0 |
| | | | | | | | 1 | 0 | 0 | 1 | 1 |
| | | | | | | | | 1 | 0 | 0 | 1 |

$\rightarrow E(x) \neq 0$, error!

If the remainder obtained after dividing the received frame by the generator polynominal is zero, does it mean that there is no error in the received frame?

# Error detection capability



No, it does not mean that there is no error in the received frame as CRC can detect only that error pattern $E(x)$ that is not divisible by $G(x)$

- CRC-4 generator polynomial $G(x) = x^4 + x^1 + x^0$ that is $10011$
- transmitted frame $11010\mathbf{00111}11$
- burst error $00000\mathbf{10011}00$ of length 5
- received frame $11010\mathbf{10100}11$

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | ↓ | | | | | | |
| | 1 | 0 | 0 | 1 | 1 | | | | | | |
| | 1 | 0 | 0 | 1 | 1 | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| | | | | | | | 1 | 0 | 0 | 1 | 1 |
| | | | | | | | 1 | 0 | 0 | 1 | 1 |
| | | | | | | | | 0 | 0 | 0 | 0 |

- remainder is $0000 \Rightarrow$ error not detected