ELEC 207
Instrumentation and Control

14 – Errors in Digital Signals

Dr Roberto Ferrero

Email: Roberto.Ferrero@liverpool.ac.uk

Telephone: 0151 7946613

Office: Room 506, EEE A block

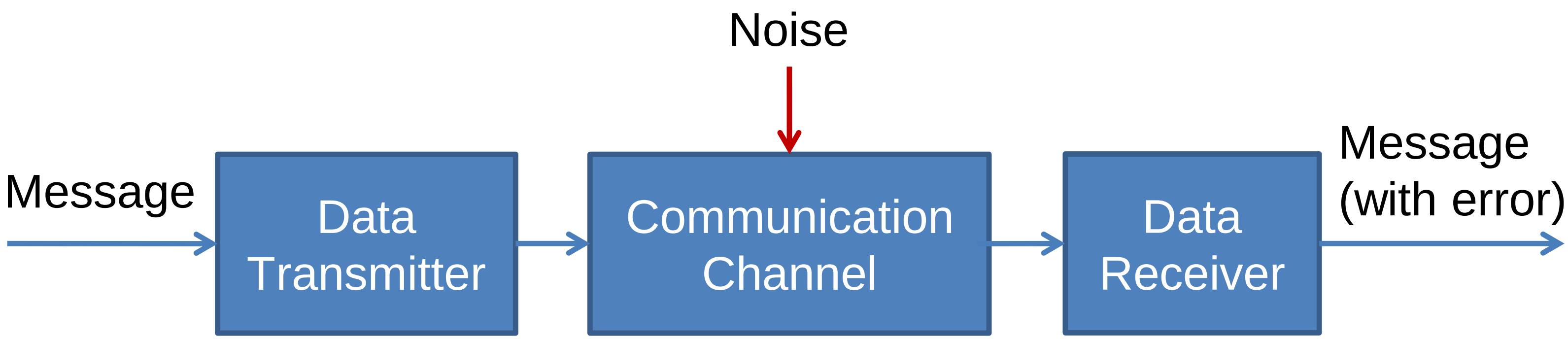
UNIVERSITY OF LIVERPOOL

Noise in digital transmission

Errors in digital signals

Noise and interference affect not only analog signals but also digital ones, particularly when digital signals are transmitted over long distances:

- In case of digital signals, noise may result in **data corruption**, i.e. errors in the transmitted data;



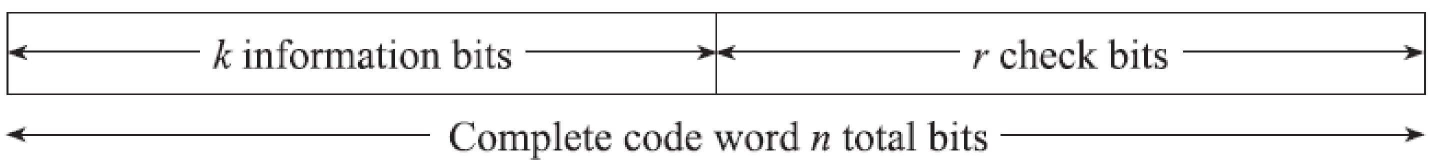
- **Error detection and correction methods** are therefore usually employed to correct errors and ensure reliable communication.

Error detection and correction

Transmission redundancy

Error detection methods are based on **redundancy**:

- The information to be transmitted is composed of *k* bits;
- Other *r* bits are added to the message and are used to check the presence of errors;
- The total transmitted message is composed of *n* = *k*+*r* bits, and its redundancy is *r*/*n*.



Two methods are used for **error correction**:

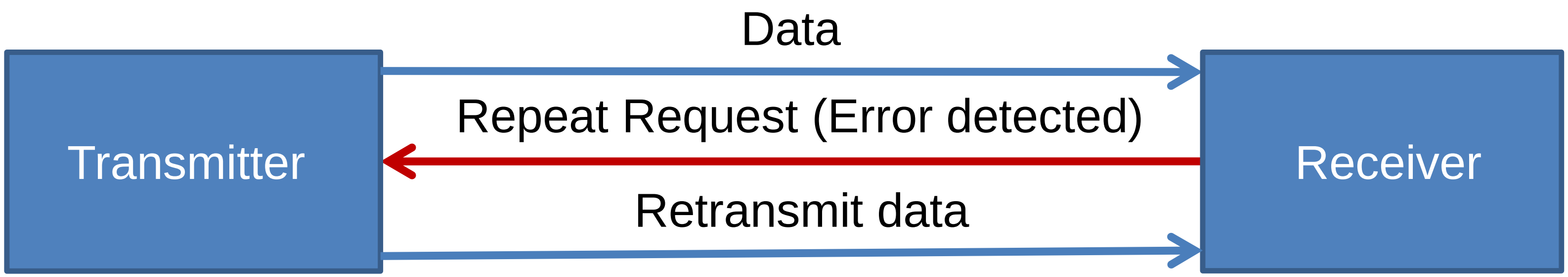
- **Automatic repeat request (ARQ)**;
- **Forward error correction (FEC)**.

Automatic repeat request

General principles

The transmitted (check) data only contains sufficient information to detect errors, not to correct them:

- The receiver must then request a **re-transmission** in order to get the correct information;



- This method uses a **low number of check bits**.

Automatic repeat request

Parity check (1)

The simplest **parity check** employs only one check bit ($r = 1$):

- If **even parity** code is used, the value of r is such that the total number of 1s in the code word is even;
- If **odd parity** code is used, the value of r is such that the total number of 1s in the code word is odd.

Information bits	Even parity code word	Odd parity code word
1011	10111	10110
1000	10001	10000
0101	01010	01011
1111	11110	11111

Automatic repeat request

Parity check (2)

This method is very simple but it has important **limitations**:

- It can only detect the presence of an **odd number of errors**, as an even number of errors produce the correct parity check bit:
  - This method is suitable when errors are unlikely to happen;

- When an error is detected, the corrupted bit cannot be located and therefore **the error cannot be corrected**:
  - The whole message has to be sent again;
  - This is convenient only if errors are unlikely to happen.

Forward error correction
General principles

In order to allow **error correction**, the additional (redundant) bits must be enough to locate the position of the corrupted bits:

- The message (information bits) is split into **groups** and a **parity check bit** is assigned to each group:

- The groups are overlapping, so that each information bit belongs to more than one group:

    - ➢ If the parity check of two or more groups fails, the error is in the information bit;

    - ➢ If the parity check of only one group fails, the error is in the parity check bit itself.

Forward error correction

Hamming code (1)

An example of forward error correction method is the **Hamming (*n*,*k*) code**:

- The total number of transmitted bits (*n*) includes *k* information bits and *n*-*k* check bits (*r*), therefore the *k* information bits are split into *r* = *n*-*k* groups;

- E.g., the following message uses the Hamming (7,4) code:

0	0	1	0	1	1	0
I_1	I_2	I_3	I_4	R_1	R_2	R_3

Group	Binary	Parity bit	Even parity
$I_2I_3I_4$	010	R_1	1
$I_1I_3I_4$	010	R_2	1
$I_1I_2I_4$	000	R_3	0

Forward error correction

Hamming code (2)

The following examples allow understanding how an error in the information bits can be detected and located (therefore corrected):

$I_2I_3I_4$ (R_1)	$I_1I_3I_4$ (R_2)	$I_1I_2I_4$ (R_3)	Faulty bit
OK	OK	OK	None
OK	OK	Error	R_3
OK	Error	OK	R_2
Error	OK	OK	R_1
OK	Error	Error	I_1
Error	OK	Error	I_2
Error	Error	OK	I_3
Error	Error	Error	I_4

<u>1 fail</u>: parity bit error

<u>2+ fails</u>: data bit error

Forward error correction
Hamming code (3)

The required number of check bits (r) to detect and correct **single bit errors** in a complete code word of n bits is calculated as follows:

- The check bits must be able to distinguish between n+1 possible situations:
  - n different positions of the corrupted bit;
  - 1 situation corresponding to no errors;
- Therefore r must be chosen so that:

$$2^r \geq n + 1 \quad \Longrightarrow \quad r \geq \log_2(n + 1)$$

  - E.g., for n = 7, the minimum value of r is 3.

References

Textbook: Principles of Measurement Systems, 4th ed.

For further explanation about the points covered in this lecture, please refer to the following chapters and sections in the **Bentley** textbook:

- Chapter 18, Sec. 18.5: **Error detection and correction**.

NOTE: Topics not covered in the lecture are not required for the exam.