Error Detection With Parity Bits - Checking (Decoding)

Information Theory Lab 8

Objective

Understand the use of parity bits for error detection, in particular the detection of 1 error based on 1 parity bit.

Theoretical notions

Given a set of N bits $b_1, \ldots\ b_N$, a parity bit p is defined as the **modulo-2 sum** of their values:

$$p = b_1 \oplus b_2 \oplus ... \oplus b_N.$$

The parity bit can be used for detection of a single error:

- when transmitting a group of bits, compute and send their parity bit as well
- when receiving a group of bits and their parity bit, compute the parity bit of the first bits again and compare with the parity bit received:
  a. If they are the same, decide that no error has happened;
  b. If they differ, an error has been detected.

This scheme represents an 1-error detection code.

In the C language, modulo-2 sum is done by the bitwise `XOR` operation (`^`).

Exercises

1. Write a C program that checks the parity bit for every byte in a data file.

The input file is the file produced by the previous program, which has a parity bit inserted after every 8 bits of data.

The output file should contain only the data, with the parity bits removed.

The program checks the parity bits and, whenever it detects a mismatch, it prints a message in the console: "Error detected ad byte number X", where X is the byte position in the file.

The program shall be called as follows:

```
‘ParityCheck.exe input.dat output.dat‘

* The arguments are:
    * ‘input.dat‘: the input file produced by the program in the previous lab
    * ‘output.dat‘: the output file, containing only the data (parity bits removed)

* The program should consist of the following steps:
    * declare two large vectors of ‘unsigned char‘, for input
    and output bits
    * open the input file and read everything into the input vector
    * for every group of 9 bits from the input vector:
        * copy the first 8 bits to the output vector
        * compute the parity bit of these 8 bits
        * check if the parity bit is the same as the 9th bit (parity bit) from the in
        * if not, print a message
    * write the output vector to the output data file
```

3. Take an output file produced by the last week's program. Use the `Frhed` program supplied in the zip file to introduce 1 error into some locations in the output file (just 1 error every 9 bits). Call the second program and observe if the errors are detected.

Final questions

1. How could we make the code detect 1 error in every 4 bits? How about 1 error in every 2 bits? What is the downside?