Error Detection With Parity Bits - Encoding

Information Theory Lab 7

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 computes and appends the parity bit for every byte in a data file. The program shall be called as follows:

`Parity.exe input.dat output.dat`

- The arguments are:
  - `input.dat`: the input file
  - `output.dat`: the output file produced by the program (12.5% larger than the input)
- 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 8 bits from the input vector:
    - copy them to the output vector
    - compute the parity bit
    - append it to the output vector
  - write the output vector to the output data file

2. Run the program to produce the output file, for some input file. Use the `Frhed` program supplied to introduce 1 error into some locations in the output file (just 1 error every 9 bits).

Program design

1. The bit operation macro definitions are available in `bitmacros.h`.

   For quick access to a bit number i in a large array, use the following macros:

```
#define VECREAD_BIT(v,i) (READ_BIT((v[(i)/8]),(i)%8))
#define VECWRITE_BIT(v,i,val) (WRITE_BIT((v[(i)/8]),(i)%8, val))
```

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?