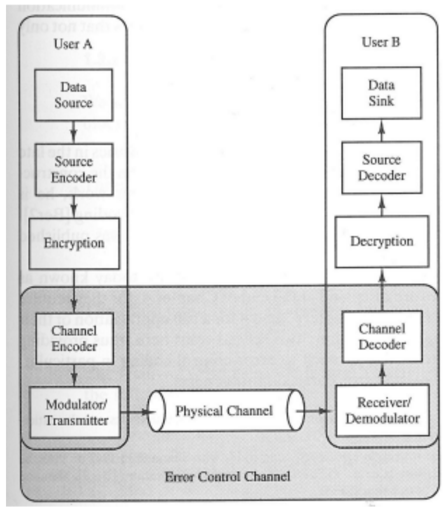
ECC Lecture 1

1 Error control

1.1 Why error control?

Messages are subject to errors when transmitted through a channel: spacial (data transmission) or temporal (storage)



1.2 The tenets of error control

Advantage of digital data v analog: we can perform error correction

Main assumption:

- Simplify the channel
- Suppose errors occur infrequently

Main idea: add redundancy to the data

1.3 The codebook idea

Everyday language uses a form of error control:
If you can't hear your friend in a crowded room, what do you do?

Basic error control code: repetition

The language itself is an error correcting code:
"I am going to the con**k**ert tonight"
Easy to detect and correct the error:
"I am going to the concert tonight"

Some sequences of letters are correct, others are incorrect
The correct ones form a **code**

2 Basic examples

2.1 Basic error detection

Parity-check code: add one more bit to the sequence of bits so that the overall number of 1's is even

Codebook: all sequences of a given length with an even number of bits equal to one

E.g. $1100011 \xrightarrow{\text{encoding}} 1100011\mathbf{0}$

Used in first version of ASCII (7 bits for the symbol +1 parity check bit)

2.2 Parity check code

Can **detect one error** buy simply checking the number of ones

$$1100011 \xrightarrow{\text{encoding}} 11000110 \xrightarrow{\text{channel}} 110\mathbf{1}0110$$

Cannot detect two errors

$$1100011 \xrightarrow{\text{encoding}} 11000110 \xrightarrow{\text{channel}} 110\mathbf{11}110$$

Cannot correct any error: e.g. if we receive 11010110, what happened?

$$1100011 \xrightarrow{\text{encoding}} 11000110 \xrightarrow{\text{channel}} 110\mathbf{1}0110 \quad \textit{or}$$

$$1100011 \xrightarrow{\text{encoding}} 11000110 \xrightarrow{\text{channel}} 1101011\mathbf{0}$$

2.3 Basic error correction

Repetition code: send the same but multiple times

$$0 \rightarrow 000$$

$$1 \rightarrow 111$$

Codebook $\{000, 111\}$

Can **correct one error** e.g.

$$0 \xrightarrow{\text{encoder}} 000 \xrightarrow{\text{channel}} 0\mathbf{1}0 \xrightarrow{\text{decoder}} 0$$
$$1 \xrightarrow{\text{encoder}} 111 \xrightarrow{\text{channel}} 11\mathbf{0} \xrightarrow{\text{decoder}} 1$$

Can detect two errors e.g. $0 \rightarrow 00 \rightarrow \mathbf{11}0$
Very low rate 1/3, so we need more rate efficient techniques

For n bits:

- We can detect n-1 errors
- We can correct $\lfloor \frac{n-1}{2} \rfloor$ errors

$$t < m - t$$

$$2t < m$$

$$2t \leqslant m - 1$$

As t is an integer

$$t \leqslant \left\lfloor \frac{m-1}{2} \right\rfloor$$

2.4 Objectives

We want to design a "good" error correcting code, i.e.

1. Detects and corrects many errors
2. High rate
3. Easy to encode and decode

The first two are conflicting: compromise depending on the channel quality

3 Minimum distance

3.1 Hamming distance

The **Hamming distance** between two sequences is the number of times they disagree. E.g. $d_H(100, 101) = 1$
It is a metric

1. $d_H(x, y) \geqslant 0$
2. $d_H(x, y) = d_H(y, x)$
3. $d_H(x, y) = 0$ iff $x = y$
4. $d_H(x, y) \leqslant d_H(x, z) + d_H(z, y)$ (triangular inequality)

In other words, it has a geometric meaning

The Hamming weight of a sequence is simply the number of 1s in it

$$w_H(x) = d_H(x, 0, ..., 0)$$

3.2 Hamming distance and error correction

Decoding: if it receives the sequence v, the decoder returns the unique nearest (in terms of Hamming distance) codeword to v if it exists.

If the codeword is not unique we need a sufficient condition to make sure the decoding is non-ambiguous

3.3 Minimum distance

Definition: $d_{min}(C)$ is the minimum distance between two distinct codewords in C
Theorem: a code can correct t errors iff it has minimum distance $d_{min} \geqslant 2t + 1$