Error Detection

15th of September, 2015

Over more reliable channels there is no need to replicate data for error correction, instead it may be more efficient just to identify errors and resend the data.

Error Detection

Parity

The first error detection algorithm is the parity algorithm. Parity algorithms can split the data in to words and keep a parity bit for each position in the word. We could split our data into bytes and keep a parity byte. Unlike error correction, one parity byte is all that is needed for any length of data.

The data as transmitted.

c	0	1	1	0	0	0	1	1
m	0	1	1	0	1	1	0	1
s	0	1	1	1	0	0	1	1
c	0	1	1	0	0	0	1	1
3	0	0	1	1	0	0	1	1
6	0	0	1	1	0	1	1	0
0	0	0	1	1	0	0	0	0
p	0	0	1	0	1	0	1	1

The data as received.

c	0	1	1	0	0	0	1	1
m	0	1	1	0	1	1	0	1
s	0	1	1	1	0	0	1	1
c	0	1	1	0	0	0	1	1
0	0	0	1	1	0	0	0	0
F	0	1	0	0	0	1	1	0
0	0	0	1	1	0	0	0	0
p	0	0	1	0	1	0	1	1

The parity scheme does not identify errors which change two bits (only even/odd). Our parity scheme would fail if a column contained only one 1 but was changed to three 1s.

Checksums

Checksums are a natural extension of parity bits and divide the data into words. A simple checksum algorithm would store the number of 1s in each column. This algorithm would eliminate the even/odd situation. Other checksum algorithms count all of the 1s together using one's-complement addition. In one's-complement the high bits will wrap around to the low bits on a carry or borrow. When would checksums fail?

Cyclic Redundancy Checks

Cyclic Redundancy Checks (CRC) work by treating the data as a polynomial and dividing by a generator polynomial. The CRC is the remainder after dividing the data by the generator. If the generator is of n length we first add $n-1$ 0s to the data. Before sending the data the CRC is appended to the end, if the received data has a remainder of zero there are no errors.

For data `1101011111` and generator `10011`

CRC Calculation	CRC Check
11010111110000 10011	11010111110010 10011
1001111110000 10011	1001111110010 10011
000011110000 00000	000011110010 00000
00011110000 00000	00011110010 00000
0011110000 00000	0011110010 00000
011110000 00000	011110010 00000
11110000 10011	11110010 10011
1101000 10011	1101010 10011
100100 10011	100110 10011
00010 10011	00000
0010	

For data `101110111110` and generator `1011`

CRC Calculation	CRC Check
101110111110000 1011	101110111110100 1011
00010111110000 0000	00010111110100 0000
0010111110000 0000	0010111110100 0000
010111110000 0000	010111110100 0000
10111110000 1011	10111110100 1011
0001110000 0000	0001110100 0000
001110000 0000	001110100 0000
01110000 0000	01110100 0000
1110000 1011	1110100 1011
101000 1011	101100 1011
00100 0000	00000
0100 0000	
100	

A polynomial code with r check bits will detect all burst errors of length $\leq r$. There are many CRC generator, most famously the IEEE 802 CRC: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$. The IEEE CRC will detect all burst errors ≤ 32 bits and all burst errors affecting an odd number of bits. Remember that the generator polynomial has 33 terms so that the remainder can be 32 bits.

Generator Name	Hexadecimal
IEEE 802	0x04C11DB7
Castagnoli	0x1EDC6F41
Koopman	0x741B8CD7