Error Correction with Hamming Codes

Forward Error Correction (FEC), the ability of receiving station to correct a transmission error, can increase the throughput of a data link operating in a noisy environment. The transmitting station must append information to the data in the form of error correction bits, but the increase in frame length may be modest relative to the cost of re transmission. (sometimes the correction takes too much time and we prefer to re transmit). Hamming codes provide for FEC using a "block parity" mechanism that can be inexpensively implemented. In general, their use allows the correction of single bit errors and detection of two bit errors per unit data, called a code word.

The fundamental principal embraced by Hamming codes is parity. Hamming codes, as mentioned before, are capable of correcting one error or detecting two errors but not capable of doing both simultaneously. You may choose to use Hamming codes as an error detection mechanism to catch both single and double bit errors or to correct single bit error. This is accomplished by using more than one parity bit, each computed on different combination of bits in the data.

The number of parity or error check bits required is given by the Hamming rule, and is a function of the number of bits of information transmitted. The Hamming rule is expressed by the following inequality:

$$d + p + 1 <= 2^{p} \qquad (1)$$

Where d is the number of data bits and p is the number of parity bits. The result of appending the computed parity bits to the data bits is called the Hamming code word. The size of the code word c is obviously d+p, and a Hamming code word is described by the ordered set (c,d).

Codes with values of p< =2 are hardly worthwhile because of the overhead involved. The case of p=3 is used in the following discussion to develop a (7,4) code using even parity, but larger code words are typically used in applications. A code where the equality case of Equation 1 holds is called a perfect code of which a (7,4) code is an example.

A Hamming code word is generated by multiplying the data bits by a generator matrix G using modulo-2 arithmetic. This multiplication's result is called the code word vector (c1,c2.c3,.....cn), consisting of the original data bits and the calculated parity bits.

The generator matrix G used in constructing Hamming codes consists of I (the identity matrix) and a parity generation matrix A:

G = [I : A]

An example of Hamming code generator matrix:

```
        1  0  0  0  |  1  1  1
G  =    0  1  0  0  |  0  1  1
        0  0  1  0  |  1  0  1
        0  0  0  1  |  1  1  0
```

The multiplication of a 4-bit vector (d1,d2,d3,d4) by G results in a 7-bit code word vector of the form (d1,d2,d3,d4,p1,p2,p3). It is clear that the A partition of G is responsible for the generation of the actual parity bits. Each column in A represents one parity calculation computed on a subset of d. The Hamming rule requires that p=3 for a (7,4) code, therefore A must contain three columns to produce three parity bits.

If the columns of A are selected so each column is unique, it follows that (p1,p2,p3) represents parity calculations of three distinct subset of d. As shown in the figure below, validating the received code word r, involves multiplying it by a parity check to form s, the syndrome or parity check vector.

T

```
H = ⌊A   | I⌋
                                 |1|
                                 |0|
| 1  0  1  1  | 1  0  0  |       |0|                |0|
| 1  1  0  1  | 0  1  0  |  *    |1|      =         |0|
| 1  1  1  0  | 0  0  1  |       |0|                |0|
                                 |0|
                                 |1|

        H*r  =  s
```

If all elements of s are zero, the code word was received correctly. If s contains non-zero elements, the bit in error can be determined by analyzing which parity checks have failed, as long as the error involves only a single bit.

For instance if r=[10*1*1001], s computes to [101], that syndrome ([101]) matches to the third column in H that corresponds to the third bit of r - the bit in error.

OPTIMAL CODING

From the practical standpoint of communications, a (7,4) code is not a good choice, because it involves non-standard character lengths. Designing a suitable code requires that the ratio of parity to data bits and the processing time involved to encode and decode the data stream be minimized, a code that efficiently handles 8-bit data items is desirable. The Hamming rule shows that four parity bits can provide error correction for five to eleven data bits, with the latter being a perfect code. Analysis shows that overhead introduced to the data stream is modest for the range of data bits available (11 bits 36% overhead, 8 bits 50% overhead, 5 bits 80% overhead).

A (12,8) code then offers a reasonable compromise in the bit stream . The code enables data link packets to be constructed easily by permitting one parity byte to serve two data bytes.