

Introducción a la Arquitectura de Sistemas – Apunte Errores

Corrección de errores:

El método de Hamming consiste en incluir información redundante en cada bloque de datos transmitido para que el receptor pueda deducir lo que debió ser el carácter transmitido. Este método es eficaz a la hora de transmitir en canales que causan muchos errores, como los canales inalámbricos.

Básicamente, si deseamos poder corregir d bits, se necesita un código de distancia $2d+1$, pues así las palabras codificadas, legales, están tan separadas que, aun con d cambios, la palabra codificada original sigue estando mas cerca que cualquier otra palabra codificada, por lo que puede determinarse de manera única.

Ejemplo: si tenemos un hamming de 5 bits, para enviar un bit, debemos agregar 4 bits de redundancia, con lo que nuestro posible margen de error es de 2 bits:

0 = 0 0000 Mensaje Redundancia

Así, si recibimos 00011, sabremos que quisimos transmitir el 0, pues es mayoría, en cambio, si recibimos 10011, la mayoría sería 1 y la corrección estaría mal.

Detección de errores:

Realizar una transmisión y que el receptor obtenga un mensaje erróneo, no es grave si me doy cuenta. La estrategia del **CRC** consiste en agrupar un conjunto de bits a enviar, e incluir solo suficiente redundancia para permitir que el receptor sepa que ha ocurrido un error (pero no qué error) y entonces solicita una retransmisión. Sea una trama (paquete a enviar) compuesta por $M(x)$ (el mensaje a enviar con n bits) y la suma de verificación (redundancia) y $G(x)$ un polinomio generador generalmente primo que el emisor y el receptor deben acordar por adelantado (para que no haya otro polinomio que haga cero el resto), el algoritmo para calcular la suma de verificación es el siguiente:

- 1- Agregar al final de $M(x)$ tantos ceros como grado tenga el polinomio generador, recordando que el grado es la cantidad de bits de $G(x)$ menos uno, ya que la variable del coeficiente de menor grado tiene exponente 0. **(1)**
- 2- Dividir **(1)** por $G(x)$. (aritmética de modulo 2 divido aplicando XOR)
- 3- Sumarle a **(1)** el resto obtenido. (aplicando XOR)

De esta manera obtenemos la trama que enviaremos. Luego el receptor divide la trama que le llega por el polinomio acordado y si el resto es cero, el mensaje ha sido enviado correctamente.

Ejemplo:

$$M(x) = 1010010100000101001101$$

$$G(x) = x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x^0 = 11111101$$

Paso (1): $M(x) = 1010010100000101001101$ 0000000

Paso(2):

```

1010010100001010011010000000 | 11111101
01011000
010011010
011001110
0011001101
0011000001
0011110000
000011011101
0010000000
011111010
000001110000
0001110 Resto
    
```

Paso(3): 101001010000101001101 0001110

Luego el receptor divide lo que le llega por G(x):

```

1010010100001010011010001110 | 11111101
01011000
010011010
011001110
0011001101
0011000001
0011110000
000011011101
0010000000
011111010
000001111110
0000000 Resto
    
```

El mensaje llegó bien, por eso el resto dio cero.

Otro de los métodos es el de **Paridad**. Podemos analizarla según la cantidad de ceros o unos (Paridad de ceros o Paridad de unos). Básicamente consiste en agregar al final un bit (0 o 1) que indique si la cantidad de bits 0's o 1's es par o no. Este método solo reconoce errores de cantidad impar, es decir, si durante la transmisión se modifica una cantidad par de bits, pasará inadvertido.

Por ejemplo, si enviamos 101100 1, analizando paridad de unos y recibimos:

- 111100 1 Se modifico cantidad impar de bits, entonces se reconoce el error.
- 100110 1 Se modifico cantidad par de bits, el mensaje es erróneo pero no es detectado.

Estos métodos, suelen utilizarse en canales seguros, como la fibra óptica, ya que son más económicos que anexar tanta redundancia, como en un método de corrección.