Codificación y decodificación de códigos usando códigos `Checksum` y `Berger`

Arquitecturas Tolerantes a Fallos

Curso 2012/2013

Pereira Guerra, Adrián <adrian.pereira@udc.es>
https://github.com/adrisons/ATF

Índice

1. Checksum de doble precisión

Este código añade poca redundancia comparado con otros. Para transmitir una matriz de $i * j$ elementos sólo añadimos $2 * j$ para el checksum. Controlando el número el número de filas a transmitir, puede ser un código muy eficiente, además es muy sencillo de implementar. El precio que hay que pagar por esta sencillez es que se necesita hardware adicional. Detecta errores simples y múltiples, por lo tanto tiene alta cobertura. Se ve su funcionamiento en la Figura 1

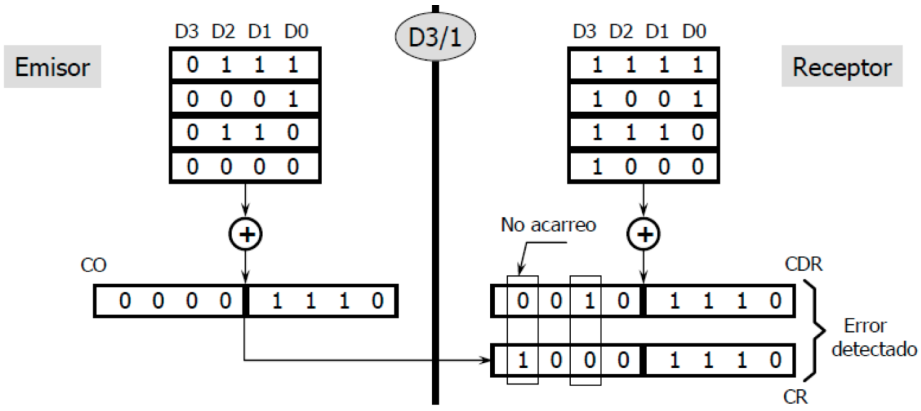

Figura 1: Ejemplo checksum

2. Berger

redundancia que introduce el código y cobertura del mismo ante diferente tipo de errores

La redundancia que añade Berger se vuelve insignificante cuando tratamos con cadenas de datos largas (véase Figura 2.1), sin embargo, para cadenas cortas inserta demasiada. Además es un código separable, lo que hace muy sencilla su decodificación. Lo que se hace para comprobar que la transmisión es correcta es realizar la suma y ver que está bien. Al tratarse de datos binarios, puede darse el caso en el que cambien dos datos y la suma siga dando lo mismo, por lo tanto no se detectaría el fallo. Se muestra un ejemplo en la Figura 2. Fallan dos bits

Código enviado	→	Código recibido
1001 101	→	1010 101

Figura 2: Ejemplo de fallo

pero, como la suma da lo mismo, no se detectaría el fallo.

2.1. *Una vez codificado, ¿cuántos bits tiene el código original?*

Sabemos que el número de bits que añade Berger es $k = [log_2(I+1)]$ siendo k la longitud de bits añadidos e I la longitud del dato original no codificado. Sin embargo, al decodificar, sabemos que el código codificado tiene longitud $I + k$, es decir, $I + log_2(I+1)$ y, de esta fórmula no se puede despejar I. Por lo tanto, estudio los datos de la Figura2.1 para encontrar una relación que me permita conseguir la longitud del dato original a partir del codificado.

Como se ve en la tabla, cuando el dato original tiene como longitud $2^i - 1$ se añade un nuevo bit a la codificación. Por lo tanto, se cumple que el $2^{Anhadido} <= Total < 2^{Anhadido+1}$, y sólo hay que calcular la potencia de dos a la que corresponde el Total para hallar el número de bits del dato original.

Dato original	Añadido	Total
1	1	2
2	1	3
3	2	5
4	2	6
...	2	...
7	3	10
8	3	11
9	3	12
...	3	...
15	4	19
16	4	20
...	4	
31	5	36
...	5	...
63	6	69
...	6	...
127	7	134
...	...	...

Figura 3: Tabla de código Berger

3. Comparación de los códigos

En resumen, los códigos Berger son muy cómodos de usar, por la facilidad de codificación y decodificación. Los que usan checksum, sin embargo, tienen que realizar la suma de todos los elementos y enviarla a mayores, por lo que puede resultar más costoso. Sin embargo, cuando se transmiten datos en grandes bloques, la técnica del checksum funciona bien, ya que suma por bloques y no individualmente.

Por último, y un factor muy importante en sistemas críticos, el código checksum otorga mayor cobertura que el Berger, por lo que dependiendo de lo que busque nuestro sistema, será más adecuado uno u otro. Si necesitamos alta cobertura usaremos checksum, a pesar de la latencia, sin embargo, si es importante la rapidez del sistema y no se necesita tanta cobertura, nos convendrá un código Berger.