ARQUITECTURA DE COMPUTADORAS
(IN-5050; 4 créditos)

Prof.: Ing. Alberto Espinoza Zamora Email: alberto.espinoza@ucem.ac.cr

Semana # 3 DETECCION Y CORRECCION DE ERRORES

Como hemos venido conversando, en una sesión de trabajo con el computador, la información fluye, va y viene y el almacenamiento de esta información sucede en diferentes componentes, lo siguientes tres son de los más comunes:

_____
_____
_____

Ahora bien, resulta que los actores de arriba son medios de almacenamiento, pero la información como dije anteriormente viaja y son muchos los medios que se encargan de llevar esta información de un lado para otro. A ver cuáles medios de transmisión de datos podemos encontrar en una computadora o en un servidor:

_____
_____
_____

Debido a lo anterior y a lo crítico de la información, es que se han desarrollado técnicas para la detección y corrección de errores y con esto garantizar la integridad de la información. Y es lo que nos compete en este momento.

En la transmisión de datos puede suceder algo que se llama **ruido** y que proviene de fuentes externas y que afecta considerablemente la integridad de la información. En redes existen técnicas para garantizar que la información llegue como tiene que ser (**acknowledge-sync-acknowledge**), pero ese es otro curso, cuando lo llevemos hablamos de eso. Veamos que sucede con la información que se maneja a nivel local en una PC.

Una de las técnicas para detectar y corregir errores es la implementación de **bits de paridad** para garantizar **la redundancia** de datos.

La información en la computadora viaja en lenguaje binario y se procesa en bloques de datos, entonces una técnica puede consistir en incluir suficiente información redundante en cada bloque y con esto detectar y corregir (**códigos de corrección de errores**). La otra puede ser reducir el número de bits de parida, de manera tal que sólo se detecten los errores (**códigos de detección de errores**).

Recuerden que en todo flujo de comunicación siempre habrá un emisor y un receptor y ambos tienen que ponerse de acuerdo. En nuestra lamentable red de telefonía celular sucede que a veces tenemos que decirle a la persona que se le corta y que nos repita, en las computadoras sucede igual.

Prof.: Ing. Alberto Espinoza Zamora Email: alberto.espinoza@ucem.ac.cr

Códigos detectores

A. Paridad Simple / Paridad horizontal

Se agrega un único bit de paridad que indicará si la cantidad de unos es par o impar. Si es par entonces se agrega un 0 al final de la cadena y un 1 si es impar. Por ejemplo, en la cadena **11010101**, la cantidad de 1s es impar, entonces la cadena quedaría **110101011**, cuando el receptor recibe la información, procederá de la misma, a contar los unos sin incluir el último bit y luego compara el resultado. Si detecta un error lo notificará al emisor.

Este método puede fallar en algún momento por las razones obvias, se pueden cambiar dos valores y el resultado sería el mismo.

Por ejemplo, la cadena original es **11010101** y la que recibe el receptor es **101101011**, no habría manera para el recetor de darse cuenta del error y puede haber corrupción de datos.

B. Suma de comprobación

Este método se las trae, pero sólo funciona en cadenas pequeñas. Se recomiendan bloques no mayores a 16 bits.

Funciona de la siguiente manera:

1. El CPU o el sistema define la longitud de cada cadena (recuerden, nada mayor a 16 bits).
2. Luego se acuerda un sistema de numeración basado en la fórmula 2x - 1
3. Se crean los bloques de bits de acuerdo a lo que se decidió en el paso 1.
4. Los bloques se convierten a su respectivo valor decimal y se asocia ese número con el bloque.
5. Se suman todos los valores y luego se agrega a la cadena el valor pero en negativo.
6. Se envía la cadena al receptor.

Veamos lo anterior con números. La siguiente es la cadena que se desea transmitir:

1010110001011110

Ahora ejecutemos los pasos, la cadena posee un total de 16 bits entonces:

1. 4 bits
2. 24 – 1 = 15
3. 1010 1100 0101 1110
4. 10 12 5 14
5. 10+12+5+14 = 41 **-41**

Cuando el receptor recibe la cadena, sumará también todos los valores incluyendo el valor negativo y si el resultado es **cero** entonces procesa el mensaje, de lo contrario envía un mensaje de error.

Prof.: Ing. Alberto Espinoza Zamora Email: alberto.espinoza@ucem.ac.cr

C. Hamming (distancia basada en comprobación)

Este método está bien loco, fue desarrollado por **Richard W. Hamming** y voy a tratar de explicarlo de una forma sencilla y poco dolorosa. Este método al igual que los anteriores, emplea bits de paridad que en este caso son distribuidos a lo largo de la cadena y que tienen como objetivo detectar errores.

Esta técnica puede representarse en forma de una matriz o de una tabla. La nomenclatura es la siguiente:

(# bits totales, # bits información)

Por ejemplo, en la notación (10, 8) sabemos que son 10 los bits totales de la cadena, en donde 8 son aquellos que contienen la información. Cuando se hace la resta, entonces se puede sacar el número de bits de paridad, en nuestro ejemplo serían **2 bits**.

Hagamos un ejemplo. Para este caso, el código de Hamming que queremos implementar sería **(11, 7)** y en la tabla que se muestra a continuación **p: bit de paridad** y **d: bit de datos.**

Datos = 0101001

	p1	**p2**	**d1**	**p3**	**d2**	**d3**	**d4**	**p4**	**d5**	**d6**	**d7**
Posición	0001 (1)	0010 (2)	0011 (3)	0100 (4)	0101 (5)	0110 (6)	0111 (7)	1000 (8)	1001 (9)	1010 (10)	1011 (11)
Cadena			**0**		**1**	**0**	**1**		**0**	**0**	**1**
P_1	1		0		1		1		0		1
P_2		0	0			0	1				1
P_3				0	1	0	1			0	
P_4								1	0	0	1

Tener en cuenta:

1. Los bits de paridad en la tabla siempre irán en las posiciones de potencia de 2 (1,2,4,8,16, 32…).
En este ejemplo, los bits de paridad son 4 (11-7=4), entonces estos irían en las columnas 1-2-4 y8.
2. Los bits de datos se acomodarían en el resto de las columnas.
3. Luego se procede a acomodar los bits de la cadena, en los espacios que representan los bits de datos.
4. Seguidamente, vamos a comenzar a calcular los bits de paridad. En la fila p1, lo que se hizo fue bajar aquellos bits que al final tienen el bit menos significativo que es 1. Por eso es que en la fila de posición, hemos anotado en binario los números del 1

Prof.: Ing. Alberto Espinoza Zamora Email: alberto.espinoza@ucem.ac.cr

al 11 que es la longitud de la cadena. Ahora bien, en este sistema se va a trabajar con bits de paridad que sean pares. En p1 tenemos entonces un total de 3 1s y por eso es que hemos agregado un 1 (en rojo) como bit de paridad, para tener 4.

5. Para p2 hacemos lo mismo que hicimos con p1 pero bajando aquellos que en la segunda posición de derecha a izquierda, tengan el bit menos significativo, o sea 1. El bit de paridad queda en cero, porque como resultado nos quedaron dos 1s que ya es un número par.
6. Se procede a hacer lo mismo con p3 y con p4. Una vez llena la tabla, ya podremos formar la palabra código.
7. Si el receptor recibe la siguiente cadena 10001011000, notará el error porque el receptor haría el mismo proceso que ejecutó el emisor a la hora de calcular los bits de paridad y vería que ambas cadenas son diferentes

Palabra almacenada = 10001011001
Palabra con un error en un bit = 10001011000