Módulo 07

Detección y Corrección de Errores (Pt. 2)

Organización de Computadoras
Depto. Cs. e Ing. de la Comp.
Universidad Nacional del Sur

Copyright

Contenidos

- Concepto de error.
- Mínima distancia de un código.
- Mecanismos de detección de errores.
- Paridad aplicada en los códigos **VRC** y **LRC**.
- Generación y verificación de código **CRC**.
- Mecanismos de corrección de errores.
- Códigos correctores simples.
- Hamming mínima distancia 3 y 4.

Corrección de errores

- Recordemos que para poder implementar algún mecanismo de detección es necesario agregar redundancia al mensaje transmitido:
  - A la hora de enviar **m** bits de dato, se incorporan **r** bits adicionales (también denominados de código) .
  - En general habrá 2^m mensajes válidos, si bien no todos los 2^{m+r} patrones de bits representarán combinaciones válidas.
  - Es decir, el agregado de esta redundancia es lo que posibilita que un dado código evidencie una mínima distancia mayor.

Corrección de errores

- Hasta el momento nos hemos contentado con poder detectar que se produjo un error.
- Usualmente cuando se detecta un error se suele solicitar la retransmisión del mensaje recibido incorrectamente.
- Existen dominios de aplicación donde la retransmisión del mensaje original resulta costosa o incluso imposible.
- En esos casos tiene sentido intentar establecer dónde se produjo el error, a fin de corregirlo.

Ejemplo

- Consideremos el siguiente código, cuya mínima distancia es **3**:

A:	**000000**	C:	**101010**
B:	**010101**	D:	**111111**

- En este contexto, analizar qué se puede afirmar al recibir cada uno de los siguientes patrones:
  - **010100**
  - **110000**
  - **000111**

Análisis

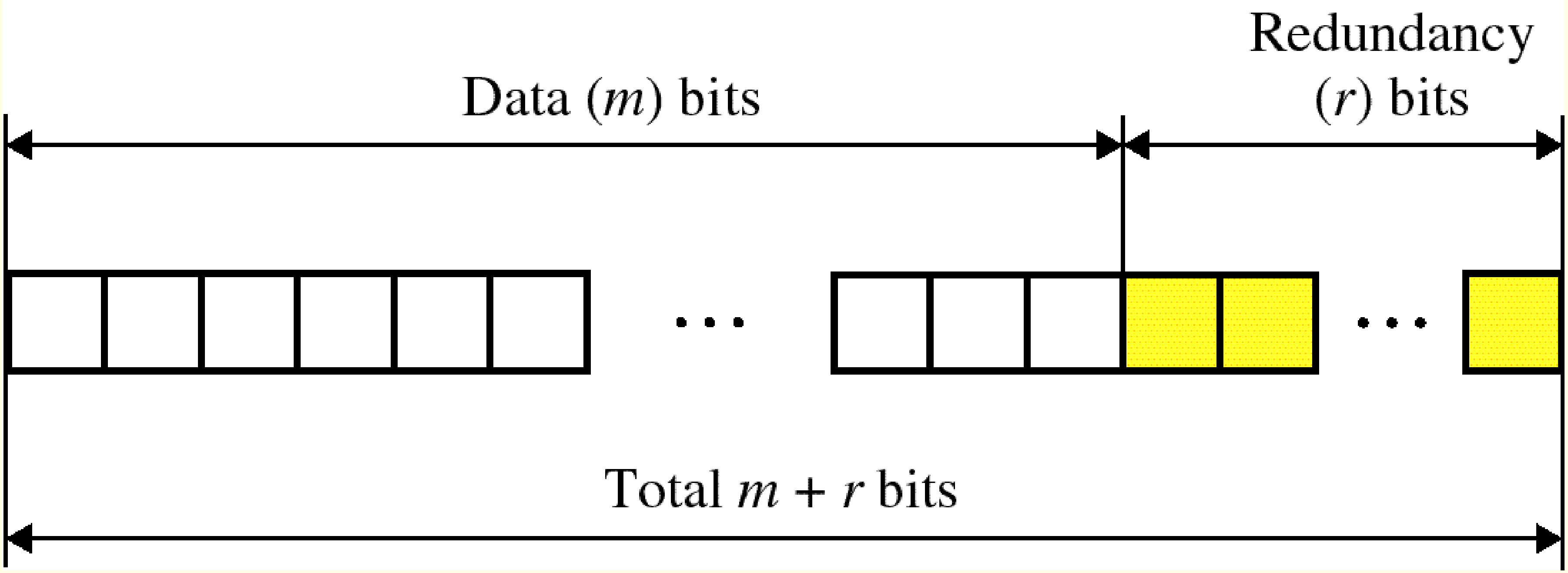
- Del análisis del ejemplo anterior se pueden sacar dos conclusiones interesantes:
  - Aparentemente ante los errores simples es posible reconocer con certeza en qué bit se produjo el error.
  - No obstante, el adoptar esta política de corrección de errores parece afectar la capacidad de detección.
- De alguna manera, al optar por corregir debemos asegurarnos que ningún otro patrón válido esté a la misma distancia que el que determinamos era el patrón original.

Análisis

- Continúa:
  - Por caso, si al recibir el patrón **010100** interpretaremos que el se produjo un error simple ya que el patrón original era **010101**, no detectaremos el error doble
    cuando el patrón original fuera **000000**.
  - Es decir, en el código del ejemplo si decidimos corregir errores simples, debemos dejar de detectar los errores dobles.
  - En general, para **M = 3** se observa que podremos detectar errores simples y dobles, o bien detectaremos y corregiremos sólo errores simples.

Diseño de un código

- Supongamos que se desea diseñar un código para corregir errores simples con **n = m + r**.



Diseño de un código

- En este escenario, para cada patrón válido debemos reservar cada uno de los $m + r$ patrones que se obtienen al adulterar un bit.
  - Es decir, para cada uno de los 2^m mensajes debemos disponer de $m + r + 1$ patrones, pero todos estos patrones tiene que codificarse con $m + r$ bits:

$$2^m \times (m + r + 1) \leq 2^{m+r} \rightarrow m + r + 1 \leq 2^r$$

  - En otras palabras, una vez elegido un cierto m, es posible derivar matemáticamente cuál será el límite inferior para r.

Diseño de un código

- Extendiendo este análisis para el caso de errores dobles se observa lo siguiente:
  - Para cada patrón válido se debe reservar por un lado $m + r$ patrones para corregir errores simples pero por otro lado hacen falta $[(m + r)(m + r - 1)] / 2$ patrones para corregir los errores dobles.
  - En síntesis, simplificando tenemos que:

$$m + r + [(m + r)(m + r - 1)] / 2 + 1 \leq 2^r$$

  - Nuevamente, fijado m se puede derivar matemáticamente un límite inferior para r.

El rol de la mínima distancia

- El concepto de mínima distancia de un código desempeña un rol central en la determinación de la capacidad de detección y de corrección:
  - Para una cierta distancia mínima **M** es posible detectar hasta **d = M – 1** bits en error.
  - No obstante, al incorporar la capacidad de corrección, por cada error corregido debemos disminuir de manera acorde la capacidad de detección.
  - En otras palabras, **d = M – 1 – c**, lo que equivale a afirmar que **M – 1 = c + d**.

El rol de la mínima distancia

- Continúa:
  - Por otra parte, como por cada bit corregido debemos asegurarnos que ningún otro patrón válido esté a la misma distancia aparte del elegido, también se debe verificar que $c \le (M - 1)/2$.
  - Reemplazando $M - 1$ por el valor antes obtenido se deriva que $2c \le c + d$, es decir, que $c \le d$.
- En resumidas cuentas:
  - $M - 1 = c + d$
  - $c \le d$

Ejemplo

- Resolvamos este conjunto de inecuaciones para valores concretos de **M**:
  - Para **M = 2** (por caso, paridad), se observa que la única solución posible es **c = 0** y **d = 1**.
  - Para **M = 3** (por caso, Hamming mínima distancia 3), existen dos solucione posibles: bien **c = 1** y **d = 1** o bien **c = 0** y **d = 2**.
  - Para **M = 4** (por caso, Hamming mínima distancia 4), también existen dos soluciones: bien **c = 1** y **d = 2** o bien **c = 0** y **d = 3**. Nótese que ni **c = 2** y **d = 1** ni tampoco **c = 3** y **d = 0** satisfacen que **c ≤ d**.

Código corrector naive

- Una manera no muy eficiente de introducir redundancia consiste en replicar los datos.
  - Por ejemplo, una posibilidad es repetir **k** veces cada bit del mensaje original.
  - ¿Qué mínima distancia manifestará este código?
  - Luego, ¿qué capacidad de detección y de corrección tendrá este código?
- El código del primer ejemplo que vimos hacía uso de esta propiedad:

A:	**000000**	**C:**	**101010**
B:	**010101**	**D:**	**111111**

Código Hamming

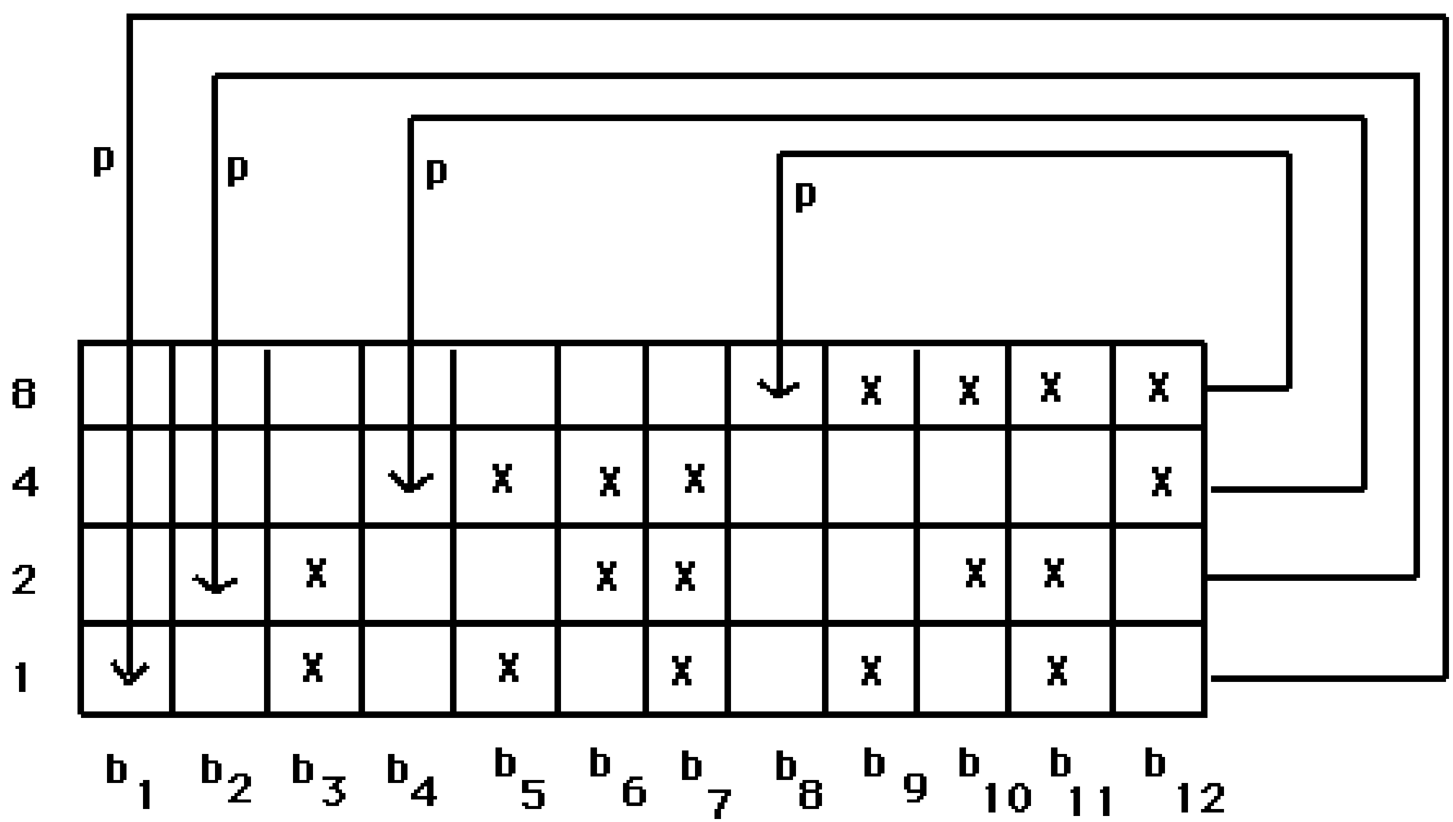
- El código Hamming fue inventado en 1950 por Richard W. Hamming, uno de los padres fundadores de la computación.

- Se basa en conceptos conocidos:
  - El mensaje se divide en dos partes, los datos a ser transmitidos y la redundancia que se le agregará.
  - La redundancia agregada al dato se compone esencialmente de bits de paridad.

Código Hamming

- Se trata de un código mínima distancia **3**, el cual intercala bits de código con bits de datos.
  - Para esto se reservan las posiciones potencias de **2** para alojar bits de código, usando las restantes para los bits de dato.
  - Por caso, para $m = 8$ se deben incorporar tantos bits de código cómo sea necesario para satisfacer la inecuación $8 + r + 1 \leq 2^r$.
  - Recién $r = 4$ satisface esta restricción, por lo que los bits de código ocuparan las posiciones **1**, **2**, **4** y **8**.

Código Hamming



relación entre los bits de código
y los bits de datos

Código Hamming

- Como se puede observar, cada bit de código cubre sólo algunos de los bits de datos:
  - El bit de código en la posición **1** sólo cubre los datos en las posiciones **3**, **5**, **7**, **9** y **11**.
  - El bit de código en la posición **2** sólo cubre los datos en las posiciones **3**, **6**, **7**, **10** y **11**.
  - El bit de código en la posición **4** sólo cubre los datos en las posiciones **5**, **6**, **7** y **12**.
  - El bit de código en la posición **8** sólo cubre los datos en las posiciones **8**, **9**, **10**, **11** y **12**.

Código Hamming

- Nótese que cada bit de código sólo cubre los bits de datos en las posiciones binarias que involucren a ese bit:

0	0	0	0	0	0	0	1	1	1	1	1
0	0	0	1	1	1	1	0	0	0	0	1
0	1	1	0	0	1	1	0	0	1	1	0
1	0	1	0	1	0	1	0	1	0	1	0
c_1	c_2	d_8	c_3	d_7	d_6	d_5	c_4	d_4	d_3	d_2	d_1

Cálculo del síndrome

- Esta matriz de posiciones binarias se puede usar en conjunción al patrón de bits recibido para determinar si se produjeron o no errores.

matriz de posiciones

patrón recibido

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_2 \\ d_4 \\ c_3 \\ d_3 \\ d_2 \\ d_1 \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix}$$

síndrome calculado

Cálculo del síndrome

- El síndrome obtenido permite establecer si se produjo algún error en la transmisión:
  - Si se obtiene un síndrome nulo (esto es, $[0\ 0\ 0]^t$), no se produjeron errores.
  - Caso contrario, el síndrome marca la posición donde aparentemente se produjo el error.
- Este cálculo equivale a calcular paridad par sobre los bits del mensaje, puesto que:

$$S_1 = c_1 \oplus d_4 \oplus d_3 \oplus d_1$$
$$S_2 = c_2 \oplus d_4 \oplus d_2 \oplus d_1$$
$$S_3 = c_3 \oplus d_3 \oplus d_2 \oplus d_1$$

¿Paridad par o impar?

- El producto de matrices anterior permite calcular los bits de código en caso de hacer uso de paridad par.
- Naturalmente, también se puede hacer uso de paridad impar, tomando su complemento.
- La capacidad de detección y corrección de errores del código Hamming no depende del esquema de paridad elegido.

Ordenamiento de los bits

- Existen dos formas de numerar las posiciones en un cierto patrón de bits:
  - De derecha a izquierda, con la primera posición en el extremo derecho y la última en el extremo izquierdo.
  - De izquierda a derecha, con la primera posición en el extremo izquierdo y la última en el extremo derecho.
- El funcionamiento del código Hamming no se verá afectado por el ordenamiento de los bits, siempre y cuando los bits de código se sigan computando correctamente.

Cálculo de los bits de código

- Supongamos que se desea transmitir el patrón de bits **01010101**, usando Hamming, paridad par, ordenando los bits de izquierda a derecha.

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
?	?	0	?	1	0	1	?	0	1	0	1
c_1	c_2	d_8	c_3	d_7	d_6	d_5	c_4	d_4	d_3	d_2	d_1

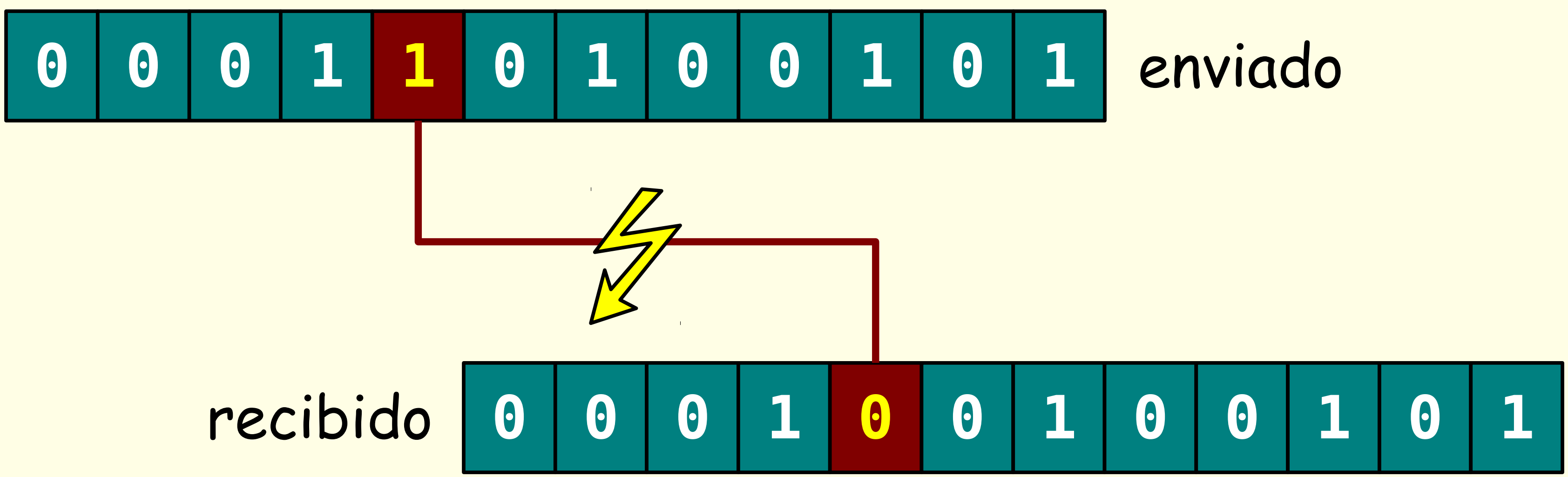
$$c_1 = p(d_8,d_7,d_5,d_4,d_2) = p(0,1,1,0,0) = 0$$
$$c_2 = p(d_8,d_6,d_5,d_3,d_2) = p(0,0,1,1,0) = 0$$
$$c_3 = p(d_7,d_6,d_5,d_1) = p(1,0,1,1) = 1$$
$$c_4 = p(d_4,d_3,d_2,d_1) = p(0,1,0,1) = 0$$

Política

- Recodemos que Hamming por tratarse de un código con **M = 3**, admite dos soluciones a las inecuaciones que vinculan **M**, **d** y **c**:
  - Una posibilidad es detectar y corregir errores simples.
  - La otra es sólo detectar errores simples y dobles.
- Cada una de estas alternativas constituye una política de detección y corrección de errores.
  - La política que se vaya a usar en una determinada transmisión tiene que estar acordada de antemano.

Corrección de errores simples

- Asumida una política de detección y corrección de errores simples, supongamos que se produjo el siguiente error simple sobre el dato antes calculado:



Corrección de errores simples

- Supongamos que se recibe el siguiente dato codificado usando Hamming y que la política adoptada es **c = 1** y **d = 1**.

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
0	0	0	1	0	0	1	0	0	1	0	1
c_1	c_2	d_8	c_3	d_7	d_6	d_5	c_4	d_4	d_3	d_2	d_1

$$c'_1 = p(d_8,d_7,d_5,d_4,d_2) = p(0,0,1,0,0) = 1$$
$$c'_2 = p(d_8,d_6,d_5,d_3,d_2) = p(0,0,1,1,0) = 0$$
$$c'_3 = p(d_7,d_6,d_5,d_1) = p(0,0,1,1) = 0$$
$$c'_4 = p(d_4,d_3,d_2,d_1) = p(0,1,0,1) = 0$$

Corrección de errores simples

- Al haber adoptado la política de detección y corrección de errores simples, el síndrome apuntará al bit en error, en caso de existir.
  - En primer lugar se recalculan los bits de código:

$$c'_4 = 0 \quad c'_3 = 0 \quad c'_2 = 0 \quad c'_1 = 1$$

  - Luego cotejamos estos valores con los recibidos, a fin de determinar el síndrome:

$$S_4 = 0 \oplus 0 \quad S_3 = 1 \oplus 0 \quad S_2 = 0 \oplus 0 \quad S_1 = 0 \oplus 1$$

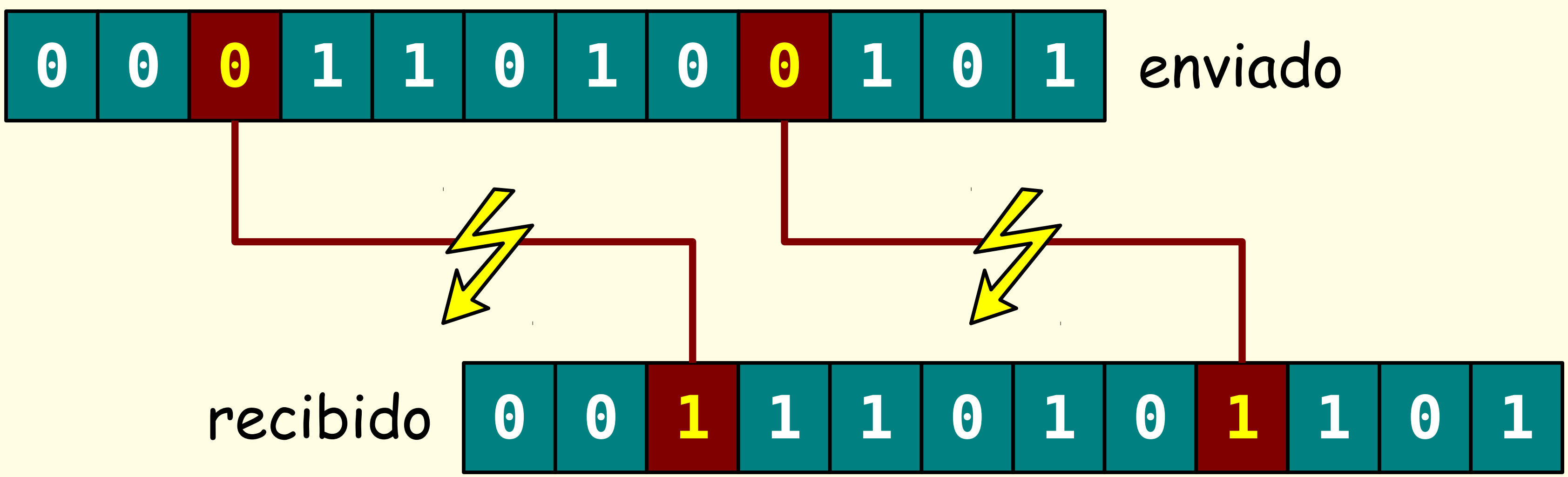
  - Como el síndrome apunta a la posición **5** (**0101**), <u>ese bit fue el afectado por el error, se lo corrige</u>.

Corrección de errores simples

- En caso de hacer uso de la política de detección y corrección de errores simples, el síndrome necesariamente apuntará a la posición en la cual se produjo el error.
- No obstante, en caso de hacer uso de la política de sólo detección de errores simples y dobles, el síndrome no va a representar una posición en concreto, por lo que simplemente se analiza si es o no nulo.

Detección simple y doble

- Asumida una política de detección de errores simples y dobles, supongamos que se producen los siguientes errores:



Detección simple y doble

- Supongamos que se recibe el siguiente dato codificado usando Hamming y que la política adoptada es **c = 0** y **d = 2**.

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
0	0	1	1	1	0	1	0	1	1	0	1
c_1	c_2	d_8	c_3	d_7	d_6	d_5	c_4	d_4	d_3	d_2	d_1

$$c'_1 = p(d_8,d_7,d_5,d_4,d_2) = p(1,1,1,1,0) = 0$$
$$c'_2 = p(d_8,d_6,d_5,d_3,d_2) = p(1,0,1,1,0) = 1$$
$$c'_3 = p(d_7,d_6,d_5,d_1) = p(1,0,1,1) = 1$$
$$c'_4 = p(d_4,d_3,d_2,d_1) = p(1,1,0,1) = 1$$

Detección simple y doble

- Al haber adoptado la política de detección de errores simples y dobles, el síndrome sólo se usa para detectar y no para corregir.
  - Los bits de código recalculados son:

$$c'_4 = 1 \quad c'_3 = 1 \quad c'_2 = 1 \quad c'_1 = 0$$

  - Cotejando estos valores con los recibidos da que:

$$S_4 = 0 \oplus 1 \quad S_3 = 1 \oplus 1 \quad S_2 = 0 \oplus 1 \quad S_1 = 0 \oplus 0$$

  - El síndrome apunta a la posición **10** (**1010**), (¡¡que no está en error!!), como es no nulo, se detecta el error. Obsérvese que se desconoce si fue simple o doble.

Ejemplo para pensar

- Supongamos que se recibe el siguiente dato codificado usando Hamming y que la política adoptada es **c = 0** y **d = 2**.

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100
0	0	1	1	0	0	1	0	1	1	0	1
c_1	c_2	d_8	c_3	d_7	d_6	d_5	c_4	d_4	d_3	d_2	d_1

$$c'_1 = p(d_8,d_7,d_5,d_4,d_2) = p(1,0,1,1,0) = 1$$
$$c'_2 = p(d_8,d_6,d_5,d_3,d_2) = p(1,0,1,1,0) = 1$$
$$c'_3 = p(d_7,d_6,d_5,d_1) = p(0,0,1,1) = 0$$
$$c'_4 = p(d_4,d_3,d_2,d_1) = p(1,1,0,1) = 1$$

Ejemplo para pensar

- El síndrome en este caso da fuera de rango, ¿qué se puede inferir de esto?
  - Los bits de código recalculados son:

    $$c'_4 = 1 \quad c'_3 = 0 \quad c'_2 = 1 \quad c'_1 = 1.$$

  - Cotejando estos valores con los recibidos da que::

    $$S_4 = 0 \oplus 1 \quad S_3 = 1 \oplus 0 \quad S_2 = 0 \oplus 1 \quad S_1 = 0 \oplus 1$$

  - El síndrome apunta a la posición **15** (**1111**), por lo que se tiene la certeza de que <u>no se produjo un error simple</u>; en consecuencia, se detecta un error doble.

Ejemplo para pensar

- ¿Qué sucederá si rehacemos el ejemplo anterior bajo la restante política, **`c = 1`** y **`d = 1`**?
  - El recálculo de los bits de código y del síndrome no cambia, es decir, el síndrome volverá a apuntar a la posición **15** (**1111**).
  - Nuevamente, se tiene la certeza de que no se produjo un error simple, por lo que no se corrige y se detecta el error de más de un bit.
  - Ahora bien, ¿no estamos acaso corrigiendo errores simples y detectando errores simples y dobles con un código cuya mínima distancia es **3**?

Detección de ráfagas

- El código Hamming mínima distancia **3** corrige a lo sumo errores simples.
  - Es decir, no es capaz de corregir errores en ráfaga.
- No obstante, es posible aplicar una estrategia análoga a la usada en el código **LRC**:
  - Supongamos que se desea transmitir **k** mensajes codificados con Hamming, cada una de longitud **n**.
  - La idea es ubicar estos patrones en una matriz y transmitir esos bits columna por columna en vez de hacerlo fila por fila.

Detección de ráfagas

Char.	ASCII	Check bits
H	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	01111001111
	0100000	10011000000
c	1100011	11111000011
o	1101111	10101011111
d	1100100	11111001100
e	1100101	00111000101

Order of bit transmission

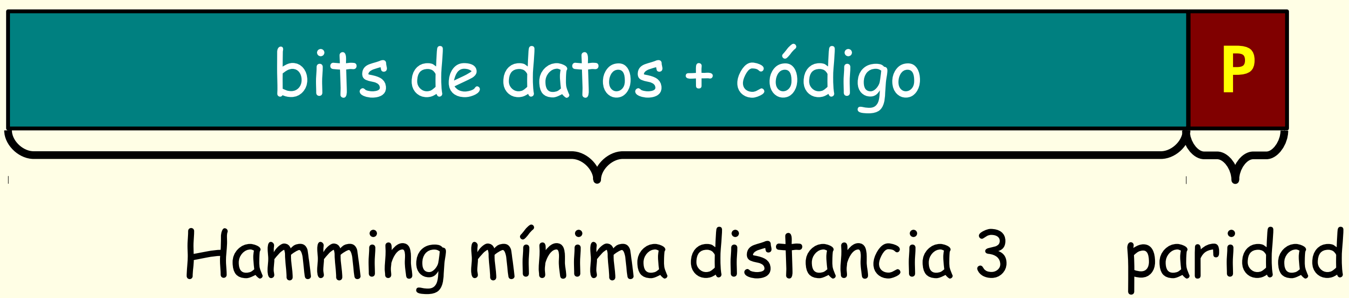
los bits amarillo denotan el error en ráfaga que se produjo

Análisis

- Si ocurre una ráfaga de longitud **k** en el bloque de **k x n** (y ningún otro error), se verá afectado a lo sumo un bit de cada patrón.
  - Por ende, el código Hamming será capaz de reconstruir cada patrón correctamente.
  - Es decir, esta codificación está en condiciones de reconstruir el bloque por completo.
- En síntesis, utilizar **kr** bits de control permite hacer que **km** bloques de bits de datos resulten inmunes a ráfagas hasta de longitud **k**.

Hamming mínima distancia 4

- La mínima distancia del código Hamming puede ser incrementada de **3** a **4** incorporando un bit de paridad que cubra la totalidad del mensaje:



- En ocasiones denominaremos a la variante sin paridad Hamming básico y a la variante con paridad Hamming extendido.

Hamming mínima distancia 4

- Qué efecto tiene la incorporación del bit adicional del paridad?
  - Al incrementar la mínima distancia a 4, aparecen nuevas soluciones a las inecuaciones que relacionan a **`M`**, **`c`** y **`d`**: ahora es posible hacer uso de las políticas **`c = 0`** y **`d = 3`** ó **`c = 1`** y **`d = 2`**.
  - Nótese que Hamming extendido ahora permite distinguir los errores simples de los errores dobles.
  - Esto posibilita corregir al estar en presencia de un error simple y no hacerlo ante un error doble.

Detección y corrección

- Supongamos que se recibe el siguiente dato codificado usando Hamming y que la política adoptada es **c = 1** y **d = 2**.

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	----
1	0	1	0	1	1	0	0	1	0	1	0	1
c_1	c_2	d_8	c_3	d_7	d_6	d_5	c_4	d_4	d_3	d_2	d_1	P

$$c'_1 = p(d_8,d_7,d_5,d_4,d_2) = p(1,1,0,1,1) = 0$$
$$c'_2 = p(d_8,d_6,d_5,d_3,d_2) = p(1,1,0,0,1) = 1$$
$$c'_3 = p(d_7,d_6,d_5,d_1) = p(1,1,0,0) = 0$$
$$c'_4 = p(d_4,d_3,d_2,d_1) = p(1,0,1,0) = 0$$

Detección y corrección

- Al haber adoptado la política **c = 1** y **d = 2**, el síndrome apuntará al bit en error sólo en caso de que se produzca un error simple:
  - La paridad y los bits de código recalculados dan:

    $$c'_4 = 0 \quad c'_3 = 0 \quad c'_2 = 1 \quad c'_1 = 0 \quad P' = 0$$

  - La construcción del síndrome da que:

    $$S_4 = 0 \oplus 0 \quad S_3 = 0 \oplus 0 \quad S_2 = 0 \oplus 1 \quad S_1 = 1 \oplus 0$$

  - Como $P \oplus P' = 1$, se detecta una cantidad impar de errores, por lo que se corrige el bit señalado por el síndrome (esto es, el bit en la posición **0011**)

Detección y corrección

- Supongamos que se recibe el siguiente dato codificado usando Hamming y que la política adoptada es **c = 1** y **d = 2**.

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	----
1	0	1	0	0	1	0	0	1	0	1	0	1
c_1	c_2	d_8	c_3	d_7	d_6	d_5	c_4	d_4	d_3	d_2	d_1	P

$$c'_1 = p(d_8,d_7,d_5,d_4,d_2) = p(1,0,0,1,1) = 1$$

$$c'_2 = p(d_8,d_6,d_5,d_3,d_2) = p(1,1,0,0,1) = 1$$

$$c'_3 = p(d_7,d_6,d_5,d_1) = p(0,1,0,0) = 1$$

$$c'_4 = p(d_4,d_3,d_2,d_1) = p(1,0,1,0) = 0$$

Detección y corrección

- Recordemos que producto de la política en uso, el síndrome apuntará al bit en error sólo en caso de que se produzca un error simple:
  - La paridad y los bits de código recalculados dan:

    $$c'_4 = 0 \quad c'_3 = 1 \quad c'_2 = 1 \quad c'_1 = 1 \quad P' = 1$$

  - La construcción del síndrome da que:

    $$S_4 = 0 \oplus 0 \quad S_3 = 0 \oplus 1 \quad S_2 = 0 \oplus 1 \quad S_1 = 1 \oplus 1$$

  - Como $P \oplus P' = 0$, se detecta una cantidad par de errores, por lo que el síndrome al ser no nulo permite detectar que se produjo un error doble.

Detección sin corrección

- Supongamos que se recibe el siguiente dato codificado usando Hamming y que la política adoptada es **c = 0** y **d = 3**.

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	----
1	0	1	0	0	1	1	0	1	0	1	0	1
c_1	c_2	d_8	c_3	d_7	d_6	d_5	c_4	d_4	d_3	d_2	d_1	P

$$c'_1 = p(d_8,d_7,d_5,d_4,d_2) = p(1,0,1,1,1) = 0$$
$$c'_2 = p(d_8,d_6,d_5,d_3,d_2) = p(1,1,1,0,1) = 0$$
$$c'_3 = p(d_7,d_6,d_5,d_1) = p(0,1,1,0) = 0$$
$$c'_4 = p(d_4,d_3,d_2,d_1) = p(1,0,1,0) = 0$$

Detección sin corrección

- Al haber adoptado la política $\mathbf{c = 0}$ y $\mathbf{d = 3}$, con respecto al síndrome sólo importa determinar si es nulo:
  - La paridad y los bits de código recalculados dan:

$$c'_4 = 0 \quad c'_3 = 0 \quad c'_2 = 0 \quad c'_1 = 0 \quad P' = 0$$

  - La construcción del síndrome da que:

$$S_4 = 0 \oplus 0 \quad S_3 = 0 \oplus 0 \quad S_2 = 0 \oplus 0 \quad S_1 = 1 \oplus 0$$

  - Como $\mathbf{P \oplus P' = 1}$, se detecta una cantidad impar de errores; a su vez, como el síndrome es no nulo, se desconoce si se produjo un error simple o uno triple.

Ejemplo para pensar

- Supongamos que se recibe el siguiente dato codificado usando Hamming y que la política adoptada es **c = 0** y **d = 3**.

0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	----
1	0	1	0	0	0	0	0	1	0	1	0	1
c_1	c_2	d_8	c_3	d_7	d_6	d_5	c_4	d_4	d_3	d_2	d_1	P

$$c'_1 = p(d_8,d_7,d_5,d_4,d_2) = p(1,0,0,1,1) = 1$$
$$c'_2 = p(d_8,d_6,d_5,d_3,d_2) = p(1,0,0,0,1) = 0$$
$$c'_3 = p(d_7,d_6,d_5,d_1) = p(0,0,0,0) = 0$$
$$c'_4 = p(d_4,d_3,d_2,d_1) = p(1,0,1,0) = 0$$

Ejemplo para pensar

- Al haber adoptado la política $c = 0$ y $d = 3$, sólo importa determinar si es nulo el síndrome:
  - La paridad y los bits de código recalculados dan:

$$c'_4 = 0 \quad c'_3 = 0 \quad c'_2 = 0 \quad c'_1 = 1 \quad P' = 0$$

  - La construcción del síndrome da que:

$$S_4 = 0 \oplus 0 \quad S_3 = 0 \oplus 0 \quad S_2 = 0 \oplus 0 \quad S_1 = 1 \oplus 1$$

  - Como $P \oplus P' = 1$, <u>se detecta una cantidad impar de errores</u>; a su vez, como el síndrome es nulo, se detecta que se produjo un error en el bit de paridad o bien un error triple.

Ejemplo para pensar

- ¿Qué sucederá si rehacemos el ejemplo anterior bajo la otra política, **c = 1** y **d = 2**?
  - El recálculo de los bits de código, de la paridad y del síndrome no cambia, es decir, el bit de paridad no da y el síndrome vuelve a ser nulo.
  - En este caso se tiene la certeza de que no se produjo ni un error simple ni uno doble sobre los bits que corresponden al código Hamming mínima distancia **3**.
  - En consecuencia, la única posibilidad es que el bit de paridad haya sido afectado por un error simple, el cual se detecta y es corregido.

Tarea para el hogar

- Analizar qué determinación debemos tomar en cada uno de los siguientes escenarios bajo las dos políticas que admite Hamming mínima distancia 4:
  - Valida paridad, síndrome nulo.
  - Valida paridad, síndrome en rango.
  - Valida paridad, síndrome fuera de rango.
  - No valida paridad, síndrome nulo.
  - No valida paridad, síndrome en rango.
  - No valida paridad, síndrome fuera de rango.

¿Preguntas?