Códigos IRA

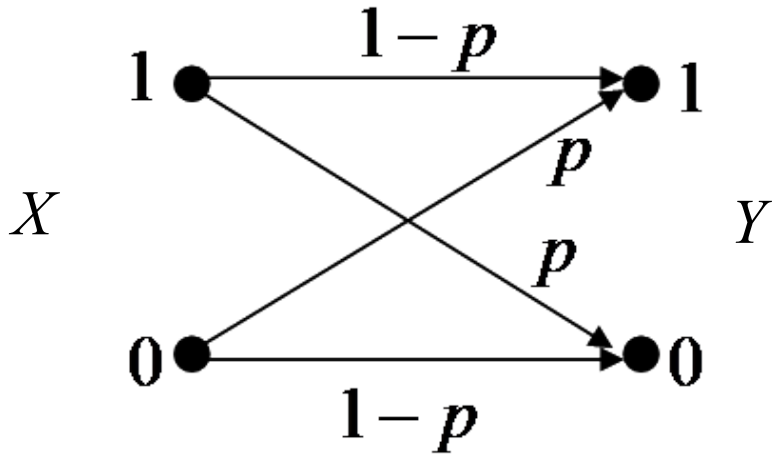
Máster en Multimedia y Comunicaciones

Comunicaciones Digitales

Luca Martino

Codificación de Canal

- Supongamos tener un canal binario discreto, simétrico sin memoria:

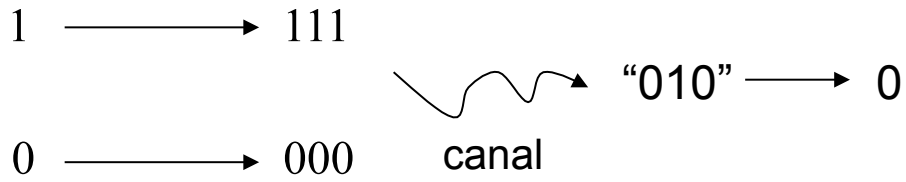


$$C_b = 1 - H_b(p)$$

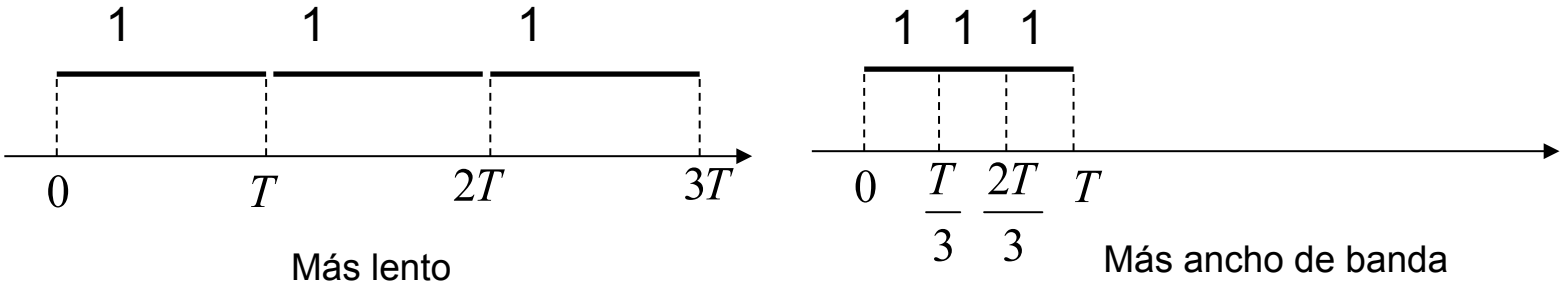
- **Objetivo:** encontrar una “manera” de enviar bits, para tener una probabilidad de error en recepción menor que *p*.

Códigos de Repetición

- La idea más sencilla: “Repetir” y elegir por mayoria.



- Para enviar un bit de información utilizo 3 veces el canal: **transmisión más lenta (o ancho de banda mayor…)**.



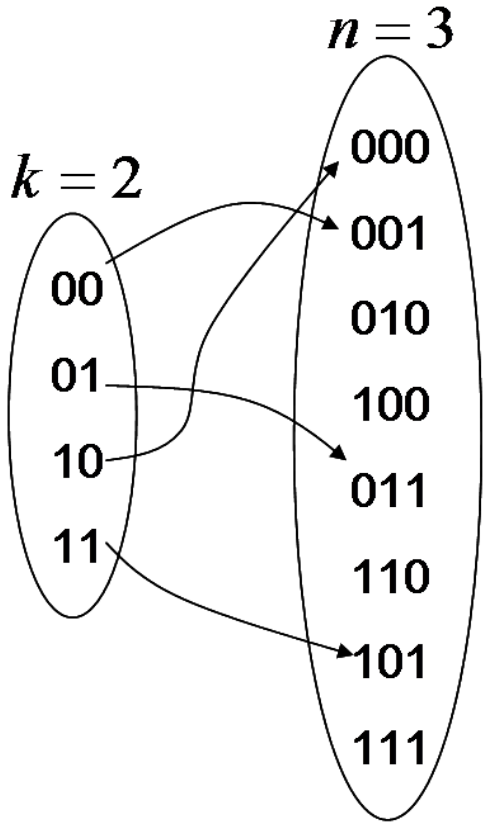
Trade off

- El ejemplo anterior nos muestra que hay que encontrar un **compromiso** entre baja probabilidad de error y velocidad de transmisión:

Fiabilidad ⟺ *Velocidad*

Codificador de Canal

- En general, la tarea de un codificador consiste en ***elegir 2^k secuencias de n bits***.



Códigos Aleatorios

- Si la relación biunívoca está elegida aleatoriamente, para decodificar necesitaríamos **la comparación con todas las posibles *k* palabras códigos** (hay que almacenar 2^k palabras código).

- Complejidad creciente en decodifica: no útil.

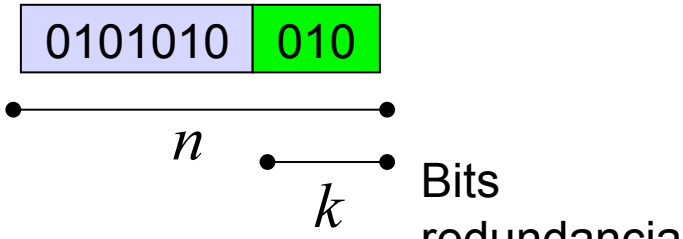
Redundancia: propiedades

- Para disminuir la complejidad del decodificador, hay que añadir bits (**redundancia**) a los mensajes en modo “inteligente”. *Los bits de redundancia* tienen que tener estas propiedades:

1. Ser fácil en generarse (*baja complejidad en codifica*).
2. Maximizar la distancia (diferencia en bits) entre dos palabras códigos.
3. Tener una cierta “estructura” que, a lo mejor, permita individuar donde se han producido los errores.
4. Permitir la decodifica sin comparar con todas las posibles palabras códigos (*baja complejidad en decodifica*).

Código ideal

- Tasa de un código: $R = \frac{k}{n}$



1. complejidad lineal en codifica.
2. complejidad lineal en decodifica.
3. probabilidad de error que tiende a cero $P_e \to 0$ por $n \to \infty$.
4. Una tasa R más alta posible (más cerca posible del máximo C_b).

Teorema de Codificación de Canal

- Si $R \leq C_b$ (capacidad) es posible disminuir la P_e aumentando *n,* con tasa *R constante.*

- En general, por un canal genérico, este teorema proporciona un limite máximo *C para la velocidad de transmisión*, para lograr una comunicación fiable.

Códigos Lineales

- Dentro de los *lineales*, hay 2 grupos principales que interpretan dos filosofías distintas:

1. Códigos *Bloque* (la decodifica de un bloque de bits se hace de modo independiente de las otras secuencias enviadas).

2. Códigos *Convolucionales* (sistema con *memoria*).

Código Bloque Lineales

- Cada palabra código se puede expresar como una combinación lineal con coeficientes 0 y 1 de unas palabras de base:

$$\vec{c} = a_1 \cdot \vec{c}_1 + a_2 \cdot \vec{c}_2 + a_3 \cdot \vec{c}_3 + a_3 \cdot \vec{c}_3 + \ldots + a_k \cdot \vec{c}_k$$
$$\vec{c} = \vec{a} \cdot G$$

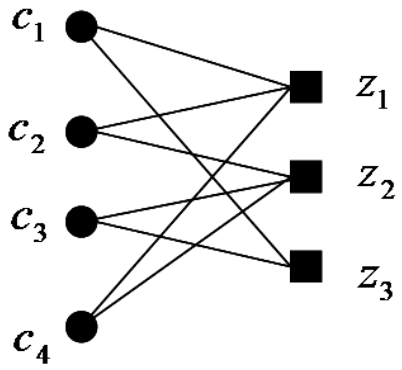
- para la decodifica se utiliza la matriz *H*:

$$\vec{c} \cdot H^T = \vec{0}$$
$$G \cdot H^T = 0$$

Gráfico de Tanner

- A cada matriz de paridad *H* está asociado un gráfico compuesto por 2 conjuntos de nodos:

$$H_{3\times4} = \begin{bmatrix} 1\,1\,0\,1 \\ 0\,1\,1\,1 \\ 1\,0\,1\,0 \end{bmatrix}$$



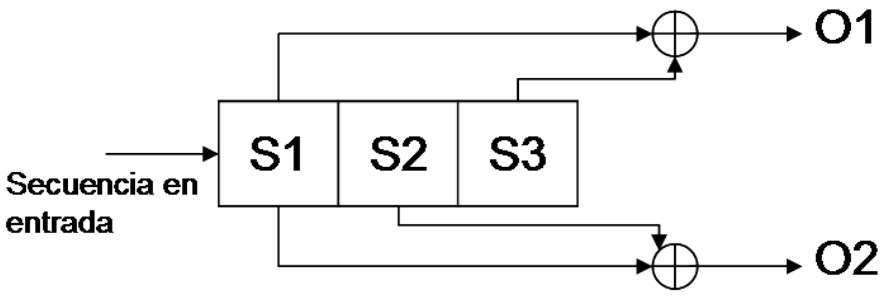
$$\begin{cases} c_1 + c_2 + c_4 = 0 \\ c_2 + c_3 + c_4 = 0 \\ c_1 + c_3 = 0 \end{cases}$$

- se ve que c_1 interviene en el nodo z_1 , z_3.

Códigos Convolucionales

- Siguen siendo *lineales* (es decir una combinación de dos palabras código cualquiera es también una palabra código).

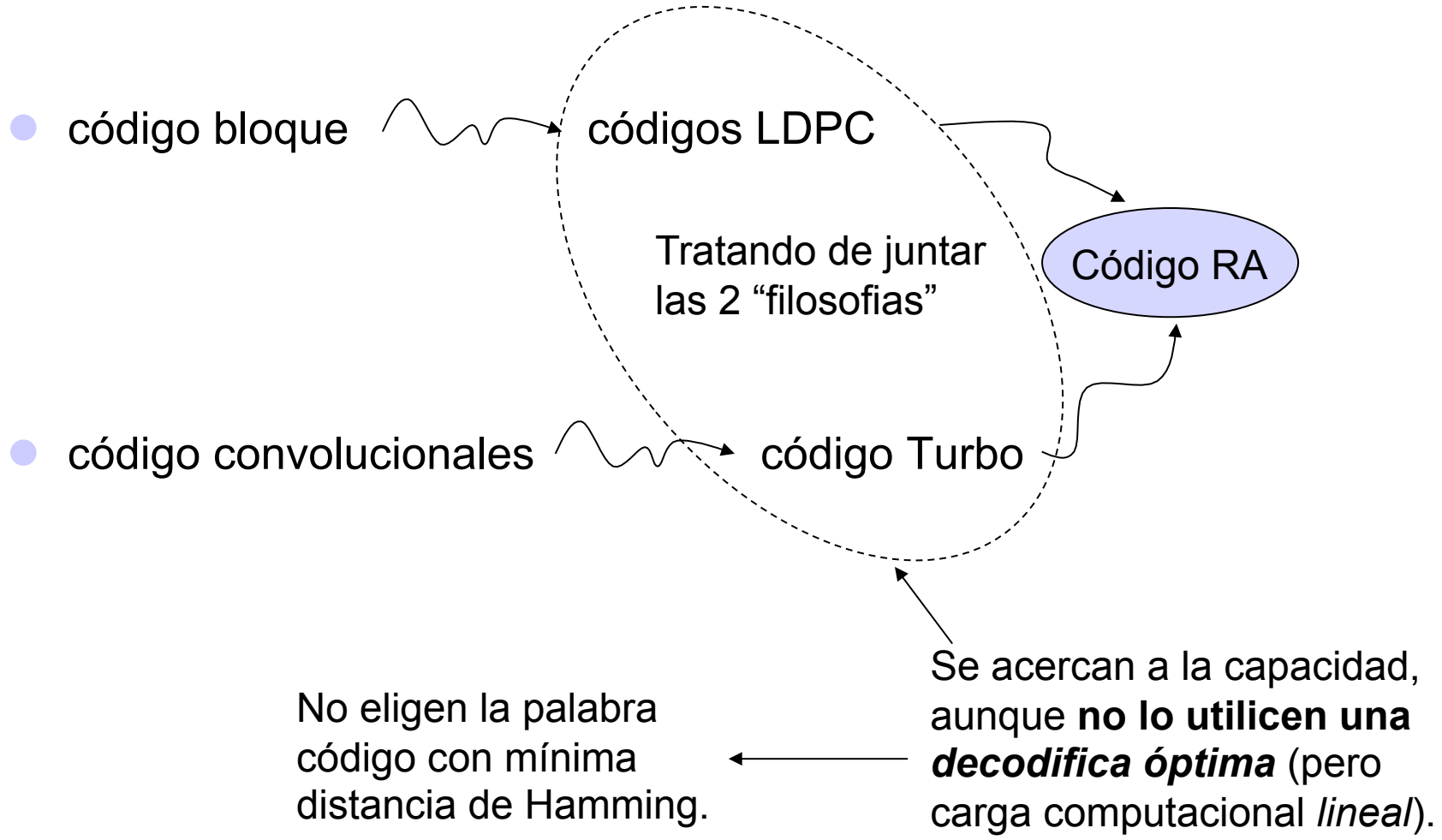
- Sistema con *memoria*: convoluciona la secuencia en entrada con un filtro FIR binario.



S1,S2,S3	01,02
0,0,0	0,0
0,0,1	1,0
0,1,0	0,1
0,1,1	1,1
1,0,0	1,1
1,0,1	0,1
1,1,0	1,0
1,1,1	0,0

- Decodifica con *Algoritmo de Viterbi* (max *verosimilitud*).

Altas prestaciones



Distancia Mínima en códigos bloque

- La distancia mínima (mínima diferencia en bits) entre 2 palabras código es igual al:

1. Número mínimo de 1 entre todas las combinaciones lineales de las *filas* de *G* (todas las palabras código).
2. Número mínimo de *columnas* de *H* que sumadas dan más que el vector nulo.

- El cualquier caso con “pocos” 1 se puede lograr d_{min} grandes.

LDPC

- Fueron propuesto la primera vez por Gallager en los años 60.
- Código bloque lineal, caracterizados por matriz de chequeo *H dispersa* (pocos unos en comparación con los ceros); para lograr altas prestaciones (d_{min} elevada), *H* tiene que ser muy grande.
- Para hallar *G* se utiliza el método de *eliminación de Gauss* partiendo de *H* hasta llegar a la forma $H=[I_k\ P]$, así que $G=[P^T\ I_r]$.
- No fueron utilizados al principio: demasiada carga computacional y de almacenamiento de datos.

Decodifica LDPC

- Con síndrome (realmente imposibles por *H* grandes):

$$\vec{c} \cdot H^T = \vec{0}$$

- Algoritmo basados en grafos: *BCJR* con grafo Trellis, y *Message passing algorithms* para grafos bipartidos (por ejemplo, Tanner).

- Dentro de los Message passing algorithms está el *Belief Propagation* que puede verse como una extensión del algoritmo de Viterbi, que proporciona las *probabilidades a posteriori* (en vez que las verosimilitudes).

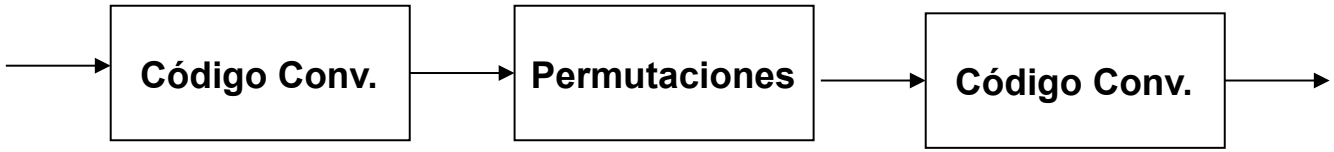
Tipos de LDPC

- *Regular*: numero *constante* de 1 en la columnas de *H.*

- *Irregular*: numero de 1 variable en las columnas de *H* (suelen tener mejores prestaciones de los regulares).

Códigos Turbo

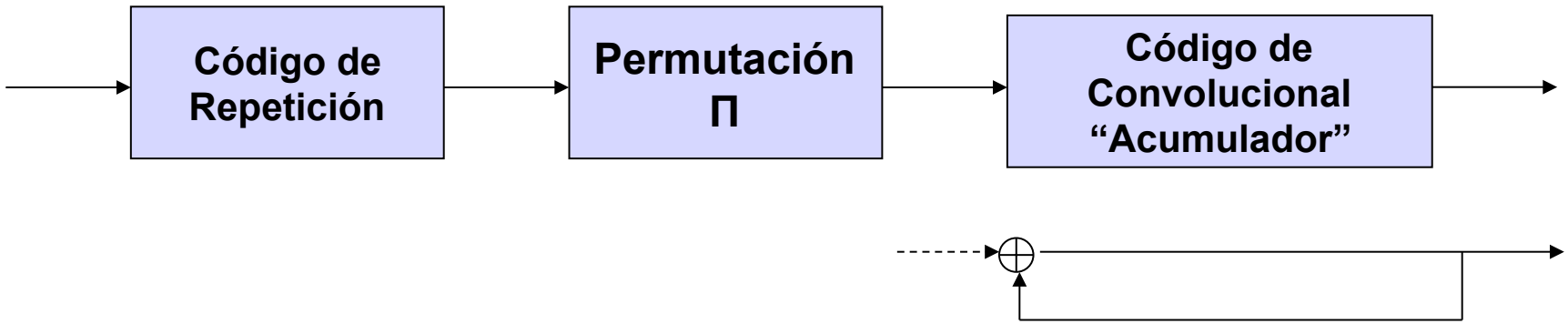
- Concatenación de 2 códigos convolucionales sistemáticamente recursivos.



- Decodificación subóptima (con infinita iteraciones llagaría a la solución *MAP*).

Códigos RA

- “Repeated Accumulate”:



- Parece tener un cierto parecido con los códigos Turbo….

Ejemplo: RA

- Tenemos 2 palabras *mensajes* m_1, m_2, y utilizamos un código de repetición (1,3):

$$m_1, m_1, m_1, m_2, m_2, m_2$$

- Permutamos según $\Pi = (1,4,6,2,3,5)$:

$$m_1, m_2, m_2, m_1, m_1, m_2$$

- Y la redundancia (o la salida final) se logra acumulando:

$$p_1 = m_1, p_1 = m_2 + p_1, p_3 = m_3 + p_2 ....$$

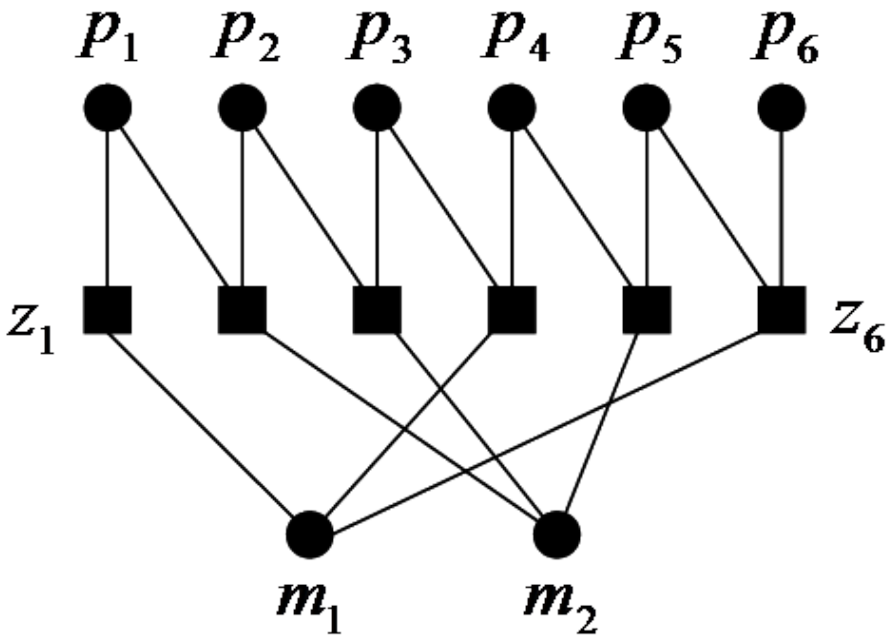
- La codifica, en modo sistemático, sería:

$$c_1 = m_1, c_1 = m_2, c_3 = p_1, c_3 = p_1, c_4 = p_2, c_5 = p_3, ...$$

$$...c_6 = p_4, c_7 = p_5, c_8 = p_6$$

Representación con Tanner

- Lo que es muy interesante es que los Códigos RA pueden representar con gráfico de Tanner (se pueden utilizar ciertos algoritmos de decodificación):



- Gráfico del ejemplo.

Representación con Tanner (II)

- Hemos divido en bits de paridad (arriba) y de mensaje (abajo), por claridad:

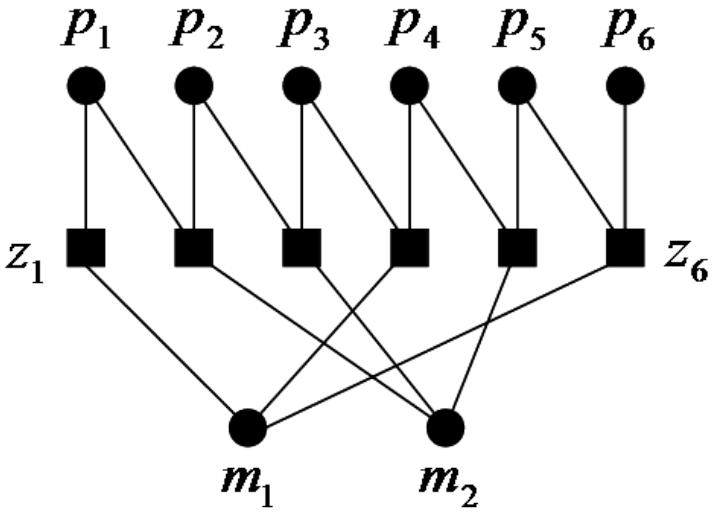
(operaciones binarias)

$$z_1 : p_1 + m_1 = 0$$

$$z_2 : p_1 + p_2 + m_1 = 0$$

$$z_3 : p_2 + p_3 + m_1 = 0$$

....



- Se pueden ver como cada bits de mensaje tiene exactamente 3 conexiones cada uno.

- con está representación recuerdan más los LDPC…

Irregular RA

- Podemos generalizar la estructura anterior, dividiendo en q grupos los k bits de información (mensaje) y *repitiendo los bits en cada bloque un número de veces distintos*.

- En ejemplo anterior de código RA, los k=2 bits de mensaje estaban repetidos, ambos de igual manera, 3 veces. Tienen El mismo número de conexiones con los nodos de chequeo z_j.

Irregular RA: fracciones

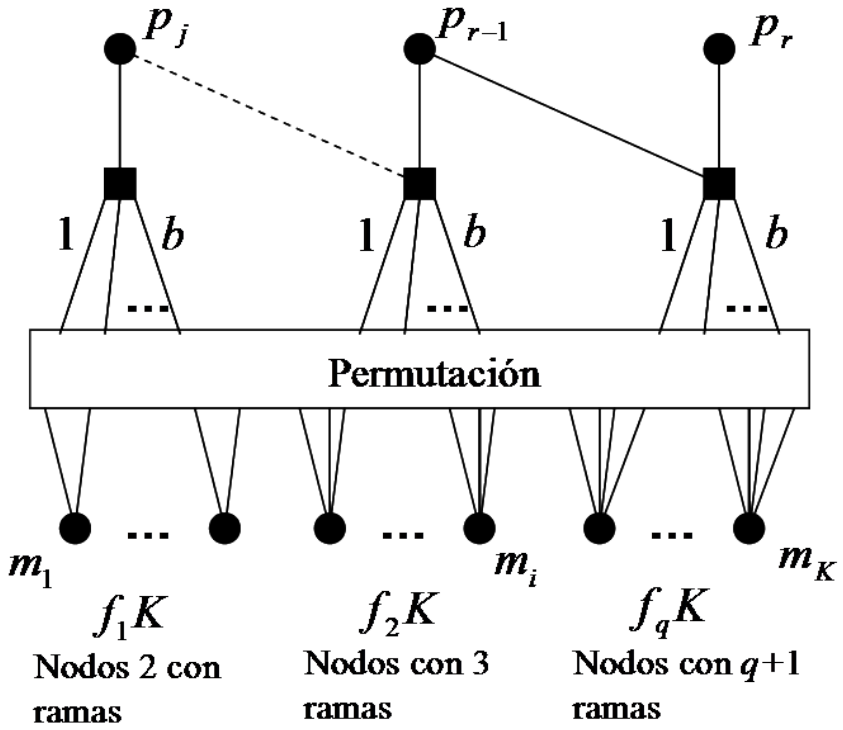
- En concreto, se elegirán *q-fracciones* f_i (una por cada grupo de bits de mensaje):

$$\sum_{i=1}^{q} f_i = 1$$

- De manera, que los $f_1 k$ bits del primer grupo se repetirán 2 veces, los $f_2 k$ bits del primer grupo se repetirán 3 veces hasta los $f_q k$ bits del primer grupo se repetirán q+1 veces.

Irregular RA: gráfico de Tanner

- Otra propiedad: cada nodo de chequeo estará conectado a un número constante (*b*) de bits de información.



- (Hemos añadido un bloque de permutación porque nos referimos a un caso genérico).

Parámetros y propiedades

- Parámetros:

q = grupos

b = conexiones nodo chequeo - bits de mensaje

k = bits de mensaje

- Número nodo de chequeo y de los bits de paridad será:

$$r = \frac{k}{b} \cdot \sum_{i=1}^{q} (i+1) \cdot f_i$$

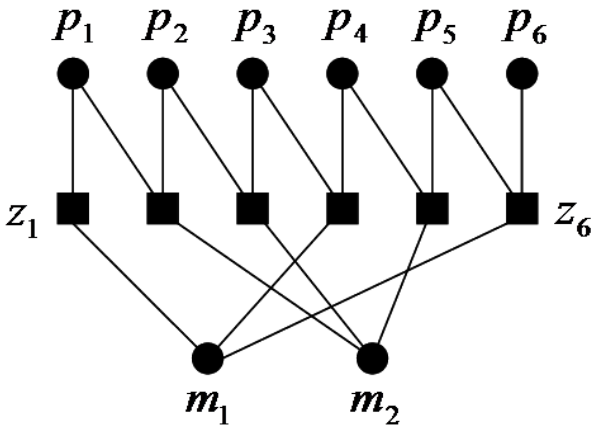
- Es más fácil para entender el caso *b*=1: $r = \sum_{i=1}^{q} (i+1) \cdot (f_i k)$

Caso particular: RA

- Para un RA tendríamos:

1. Cada nodo chequeo conectado solo a un nodo de mensaje:

$$b = 1$$



2. Habrá *k* bloque solo de 1 bits de mensaje:

$$q = k$$

3. Las fracciones serán todas nula menos una:

$$f_{i-1} = \delta_{i-n}$$

Es decir todos los bits de mensaje están repetidos *n* veces. Substituyendo en caso en figura:

$$r = \frac{k}{b} \cdot \sum_{i=1}^{q} (i+1) \cdot f_i = \frac{2}{1} \cdot (2+1) \cdot f_2 = 2 \cdot 3 = 6$$

Tasa

- El número el nodo de chequeo es igual al numero de nodo de paridad; podemos tener una versión *sistemática*:

$$\{m_1, m_2, ...m_k, p_1, p_2, ..., p_r\}$$

- Con tasa *R* será entonces:

$$R = \frac{k}{k+r} = \frac{k}{k + \frac{k}{b}\sum_{i=1}^{q}(i+1)f_i} = \frac{b}{b + \sum_{i=1}^{q}(i+1)f_i}$$

- En modo *no sistemático*:

$$\{p_1, p_2, ..., p_r\}$$

$$R = \frac{k}{r} = \frac{k}{\frac{k}{b}\sum_{i=1}^{q}(i+1)f_i} = \frac{b}{\sum_{i=1}^{q}(i+1)f_i}$$

Observaciones

- Los IRA suelen tener mejor prestaciones de los RA.
- Aunque son un caso particular de los LDPC (y de los Turbo), tienen una complejidad menor a paridad de prestaciones.
- Los códigos RA-IRA tienen complejidad *lineal* en codifica y en decodifica.