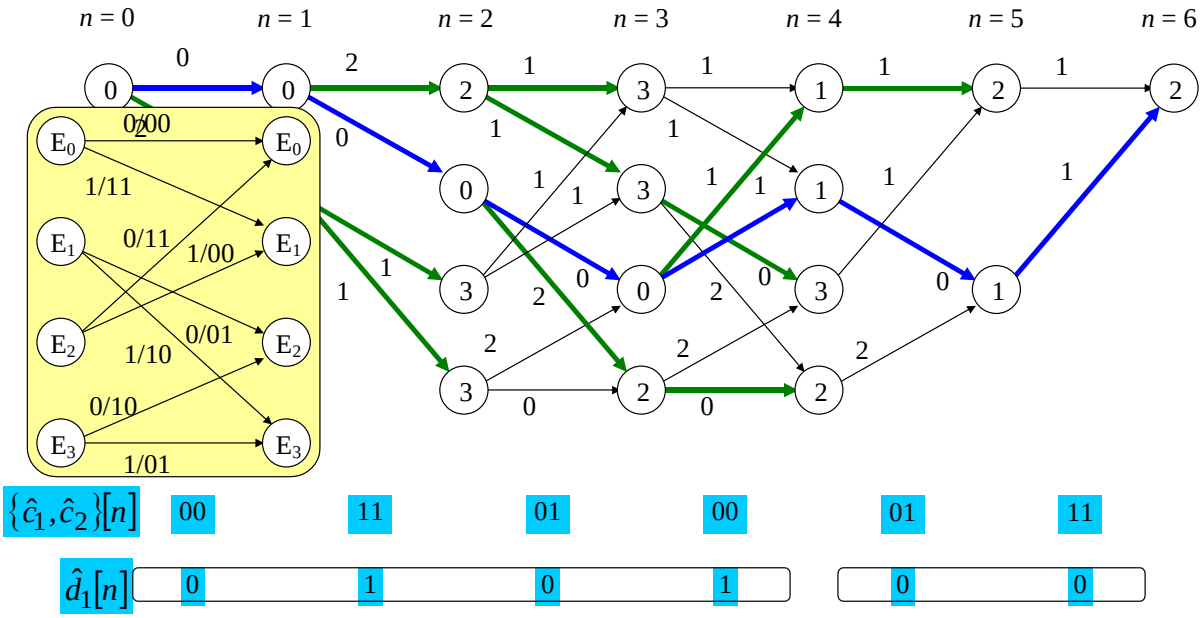
Universidad de Málaga
Escuela Técnica Superior de Ingeniería de Telecomunicación
Departamento de Ingeniería de Comunicaciones

Máster en Ingeniería de Telecomunicación

Técnicas de tratamiento de señal y comunicaciones

4 Codificación de canal



José Tomás Entrambasaguas

2018

Tema 4 Codificación de canal

Conceptos

- Codificación de canal
- Distancia de Hamming
- Prestaciones
- Diseño del código

Codificación bloque (1)

- Codificación bloque binaria lineal
- Decodificación. Comprobación de paridad
- Generalización a más errores
- Recuperación de bits borrados

Codificación bloque (2)

- Códigos cíclicos
- Codificación/Decodificación
- Realización

Códigos cíclicos no binarios

- Codificación bloque no binaria
- Códigos cíclicos no binarios.
- Códigos Reed-Solomon

Códigos cíclicos no binarios

- Codificación bloque no binaria
- Códigos cíclicos no binarios.
- Códigos Reed-Solomon

Codificación concatenada

Codificación convolucional

- Concepto
  - Codificación de secuencias
  - Codificación bloque y convolucional
  - Codificación convolucional $k = 1$ y $k > 1$
- Comportamiento del codificador
  - Representación con diagrama de estados y rejilla
  - Representación de la evolución del codificador
- Decodificación
  - Decodificación de secuencias
  - Distancia entre secuencias
  - Decodificación ML
  - El problema del camino más corto
- Decodificación con algoritmo de Viterbi
  - Fundamento del algoritmo
  - Ejemplo
  - Decodificación de secuencias indefinidas
- Códigos perforados

Decodificación blanda

- Conceptos
  - Ejemplo: código de repetición.
  - Prestaciones
  - Eficacia. Ganancia de codificación
- Aplicación a codificación convolucional
  - Algoritmo de Viterbi con entrada blanda
  - Algoritmo de Viterbi con salida blanda
- Decodificación iterativa
  - Codificación concatenada paralelo
  - Decodificación iterativa

Codificación de canal

Conceptos

Codificación bloque

Codificación convolucional

Decodificación blanda

Codificación de canal

Conceptos

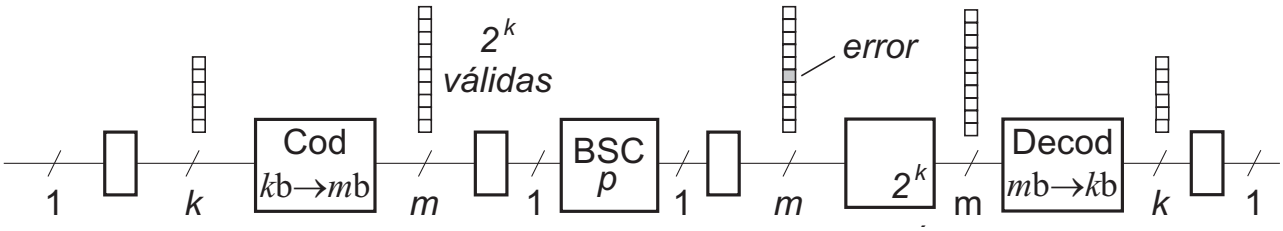
Codificación de canal
Distancia de Hamming
Prestaciones
Diseño del código

Codificación bloque

Codificación convolucional

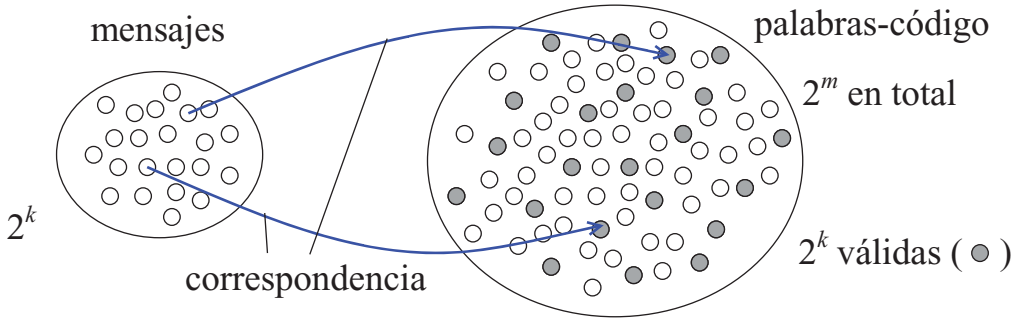
Decodificación blanda

Codificación de canal



- Bits se agrupan en "**mensajes**" de k bits
- Cada mensaje se "codifica" con una "**palabra-código**" de m bits
  (aunque hay 2^m posibles palabras-código sólo se usan 2^k **válidas**)
- Supone una pérdida de velocidad neta de información
  se transmite a ***k*/*m*** (tasa de codificación) por la velocidad binaria del canal (BSC)
- A cambio el receptor puede detectar errores en las palabras-código recibidas incluso corregirlas (con lo que disminuye la BER entre extremos).
- Capacidad de detección /correción aumenta con el núm. bits añadidos ($m - k$)
- si $m = k$ no se pueden detectar errores ni corregir

Codificador



- Hay 2^k posibles mensajes y 2^m posibles palabras-código
- Cada mensaje se "codifica" con una "**palabra-código**"
- Aunque hay 2^m posibles palabras-código sólo se usan 2^k **válidas**)

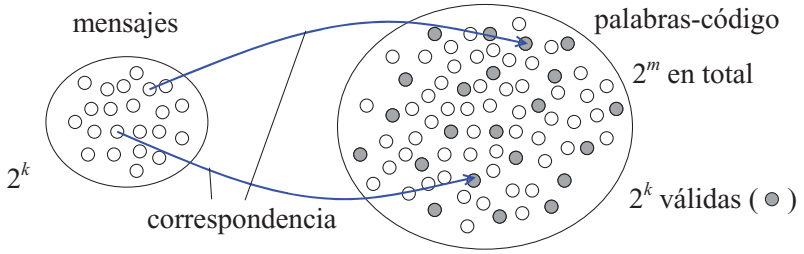
- **código**: conjunto de palabras-código válidas (sólo 2^k)

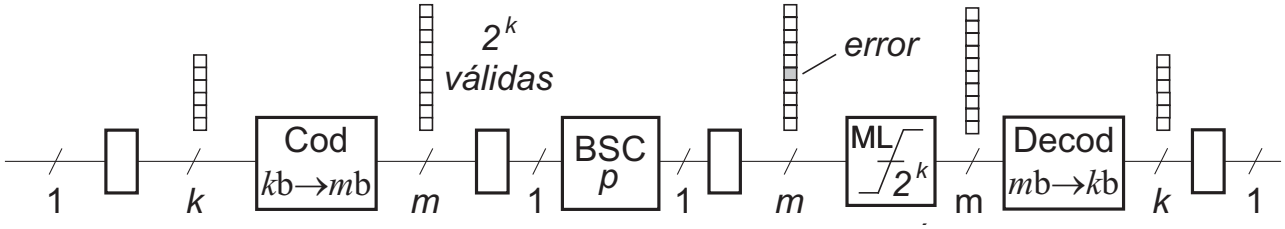
○ no pertenece al código

● pertenece al código

Codificación de canal

¿Cómo funciona?





- Al transmitir una palabra-código, los errores hacen que se reciba otra distinta
- Si la palabra-código recibida es válida (aunque distinta a la transmitida) no se puede saber si ha habido errores o es que se transmitió esa palabra-código
- Si la palabra-código recibida no es válida (no pertenece al código), se sabe que ha habido errores y en ese caso...
- Se decide una de los 2^k palabra-código válidas: **la más "parecida"** (**ML**)

Codificación de canal

Conceptos

Codificación de canal

Distancia de Hamming

Prestaciones

Diseño del código

Distancia de Hamming

- **Distancia de Hamming d_H** entre dos palabras-código: número de bits distintos

  Ejemplo

  0 1 1 0 0 0 1 1
  0 0 1 1 1 0 1 1 $\quad d_H = 3$

- **Mide la semejanza:** cuanto menor d_H más se parecen

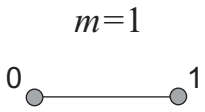
- **código**: conjunto de palabras-código válidas (2^k)

- **Distancia de Hamming d_H** de un código: la mínima entre sus palabras-código

Código	*Distancia de Hamming*
No codificación	$d_H = 1$
Repetición $\times 3$	$d_H = 3$
Repetición $\times N$	$d_H = N$

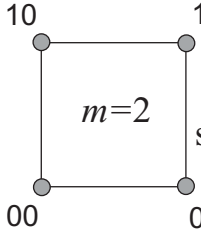
Representación de la Distancia de Hamming de un código (○)

***m* = 1**

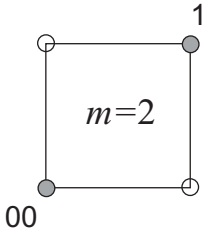


$k=1 \quad d_H=1$
sin codificar

***m* = 2**



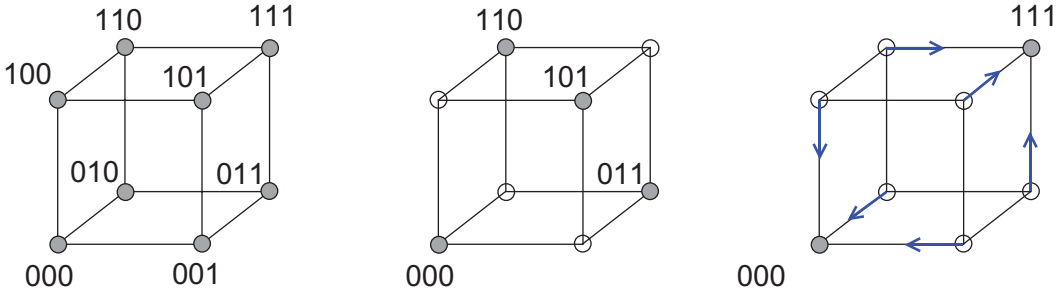
$k=2 \quad d_H=1$
sin codificar



$k=1 \quad d_H=2$
repetición
paridad par

Representación de la Distancia de Hamming de un código (●)

***m* = 3**



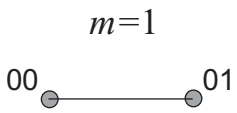
k=3 d_H=1
sin codificar

k=2 d_H=2
paridad par

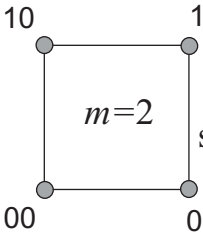
k=1 d_H=3
repetición

- Es complicada la representación para códigos de $m > 3$ bits

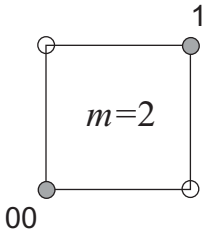
Representación de la Distancia de Hamming de un código (●)



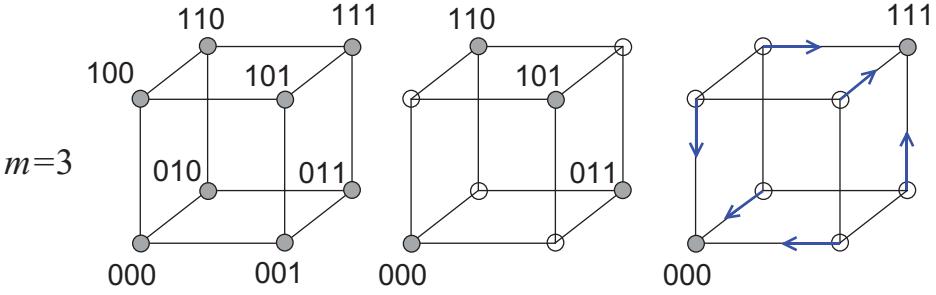
k=1 d_H=1
sin codificar



k=2 d_H=1
sin codificar



k=1 d_H=2
repetición
paridad par



k=3 d_H=1
sin codificar

k=2 d_H=2
paridad par

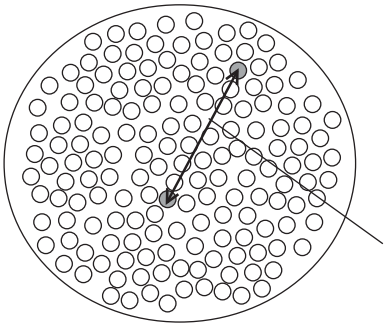
k=1 d_H=3
repetición

- Es complicada la representación para códigos de $m > 3$ bits

Representación de la Distancia de Hamming de un código

Para $m > 3$

- Se usará una representación en un plano
- Poniendo más lejos las palabras-codigo que disten más en distancia de Hamming



d_H entre dos palabras-código

- En 2D no se puede representar bien cuántas palabras-código están a cada distancia

Codificación de canal

Conceptos

Codificación de canal

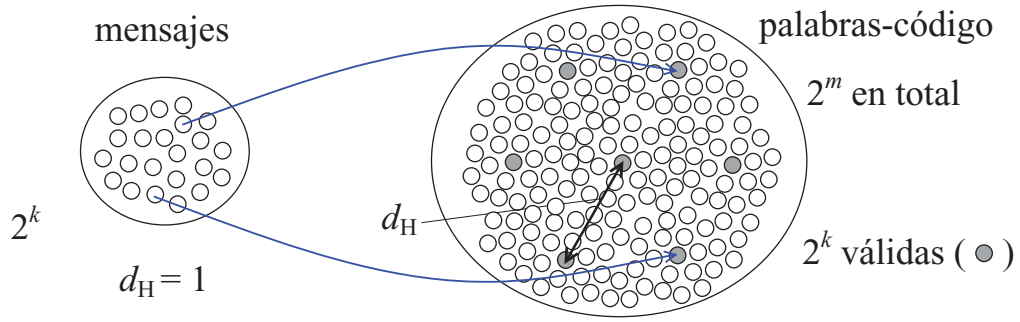
Distancia de Hamming

Prestaciones

- Detección de errores
- Corrección de errores
- Recuperación de bits borrados

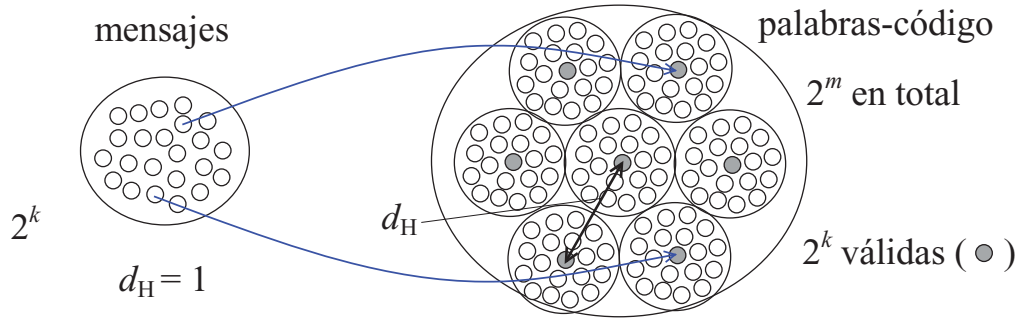
Diseño del código

Prestaciones. Detección de errores



- **Distancia de Hamming** d_H de un código: la mínima entre sus palabras-código
- Si hay errores pero **menos de d_H**: la palabra recibida no es válida: se detecta error
- **d_H** errores pueden resultar en una palabra válida: no se detecta error

Prestaciones. Corrección de errores



- **Distancia de Hamming** d_H de un código: la mínima entre sus palabras-código
- Si la palabra recibida no es válida ...
- Se decide la palabra-código válida más parecida (ML) a la recibida:
  la más cercana en distancia de Hamming
  la que difiere en menos bits (círculos de decisión en figura)
- Si hay **menos de $d_H/2$** la palabra decidida será la correcta: se corrigen los errores
- Si hay **más de $d_H/2$** la palabra decidida será incorrecta no se corrigen los errores

Prestaciones. Recuperación de bit borrados

Bit borrados

- Se recibe una palabra código sin errores pero...
  ... con $\boldsymbol{b}$ bits borrados
  no se sabe si son '0' o '1' pero se sabe dónde están

- Se prueba con las $2^{\boldsymbol{b}}$ combinaciones de los bits borrados y
  se elige la combinación que da una palabra-código válida

- Las palabras código ensayadas estarán a distancias 0,1, 2... $\boldsymbol{b}$ de la correcta
- Si $\boldsymbol{d_H} \geq \boldsymbol{b} + 1$ lá unica válida encontrada será la de distancia 0

- La capacidad de recuperar borrados (posición conocida)
  es mayor que la de corregir errores (sin saber dónde están)

Prestaciones

- **Distancia de Hamming** d_H de un código: la mínima entre sus palabras-código
  - Para detectar $\boldsymbol{\lambda}$ errores: $\boldsymbol{d_H} \geq \boldsymbol{\lambda} + 1$
  - Para corregir $\boldsymbol{t}$ errores: $\boldsymbol{d_H} \geq 2\boldsymbol{t} + 1$
  - Para recuperar $\boldsymbol{b}$ borrados: $\boldsymbol{d_H} \geq \boldsymbol{b} + 1$

		Repetición $(3, 1)$	Repetición $(N, 1)$	Paridad $(N, N+1)$
		$d_H = 3$	$d_H = N$	$d_H = 1$
detecta errores	$\boldsymbol{\lambda}$	2	$N-1$	1
corrige errores	$\boldsymbol{t}$	1	$E[(N-1)/2]$	0
recupera borrados	$\boldsymbol{b}$	*2*	$N-1$	1

- La capacidad de recuperar borrados es mayor que la de corregir errores
- Si no se está seguro del valor de un bit es mejor borrarlo que intentar corregirlo

Codificación de canal

Conceptos

Codificación de canal

Distancia de Hamming

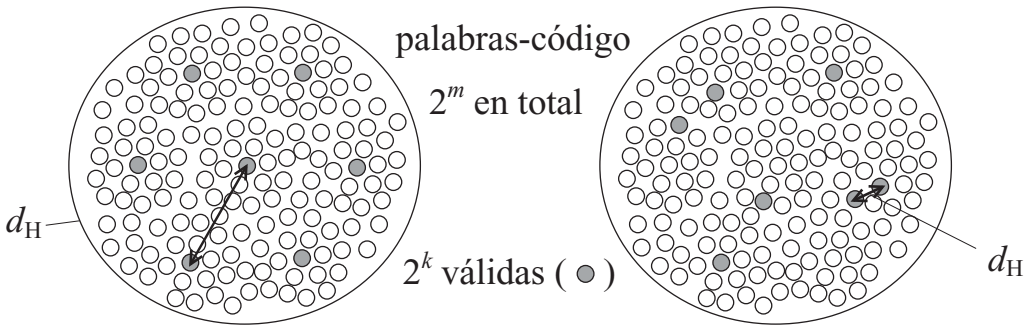
Prestaciones

Diseño del código

Diseño del código

Selección de las (2^k) palabras-código válidas

- Se eligen para maximizar la distancia mínima entre dos de ellas.

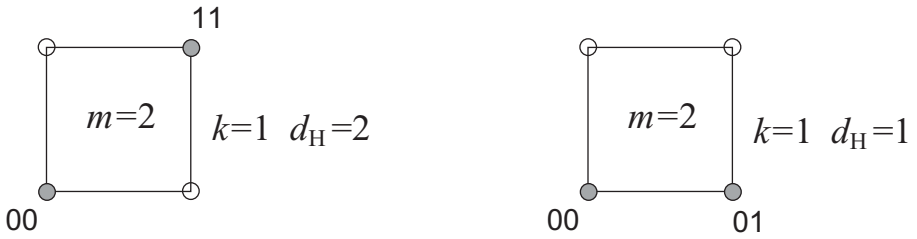


- la máxima distancia que se puede conseguir es $m - k + 1$ (equidistancia)
- Los códigos de repetición son de máxima distancia $\boldsymbol{d_H} = m - k + 1$
- No es fácil encontrar códigos de máxima distancia

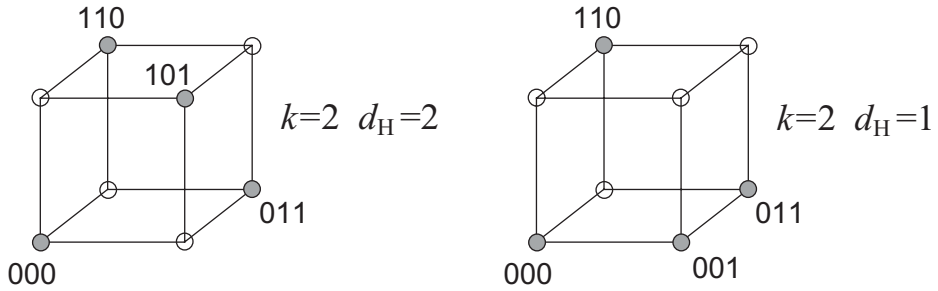
Diseño del código

Selección de las (2^k) palabras-código válidas

***m* = 2, *k* = 1**



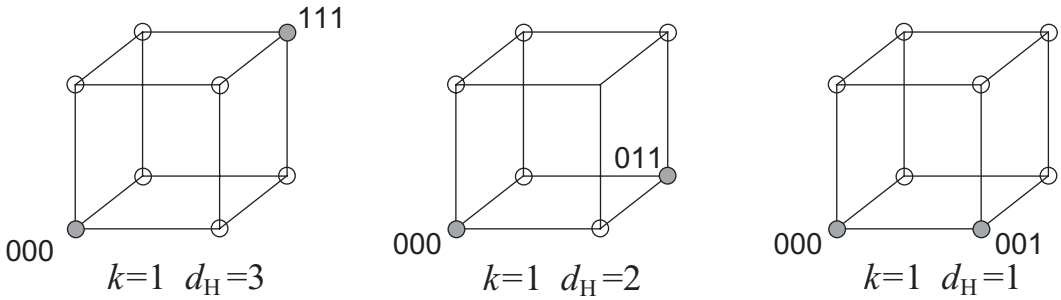
***m* = 3, *k* = 2**



Diseño del código

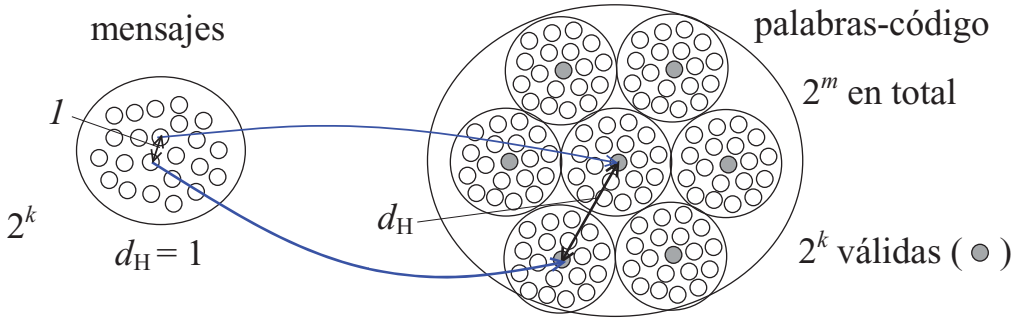
Selección de las (2^k) palabras-código válidas

***m* = 3, *k* = 1**



Diseño del código

Correspondencia mensaje → palabra-código (función del codificador)

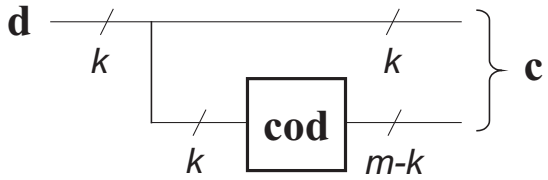


- Palábras-código contiguas (a distancia $\boldsymbol{d_H}$) se asignan a mensajes que difieren en un solo bit (a distancia 1)
- Si no se corrige el error (número de erores > $t = E[(\boldsymbol{d_H} - 1)/2]$:
  - la palabra-código decidida es contigua y
  - sólo hay un bit erróneo en el mensaje decodificado
- Igual que en codificación Gray de símbolos QAM

Diseño del código

Correspondencia mensaje → palabra-código (función del codificador)

Codificación sistemática



- Los primeros k bits de cada palabra-código son los mismos del mensaje asociado
- Sólo hay que calcular $m - k$ bits
- Los códigos de paridad y repetición son sistemáticos

	mensaje	*palabra-código*
Paridad par k=3, m=4 $(m,k) = (4,3)$	000	000-0
	001	001-1
	010	010-1
	011	011-0
	100	100-1
	101	101-0
	110	110-0
	111	111-1
Repetición k=1, m=3 $(m,k) = (3,1)$	0	0-00
	1	1-11

Codificación de canal

Conceptos

Codificación bloque

Codificación convolucional

Decodificación blanda

Codificación de canal

Conceptos

Codificación bloque (1)

- Codificación bloque binaria lineal
- Decodificación. Comprobación de paridad
- Generalización a más errores
- Recuperación de bits borrados

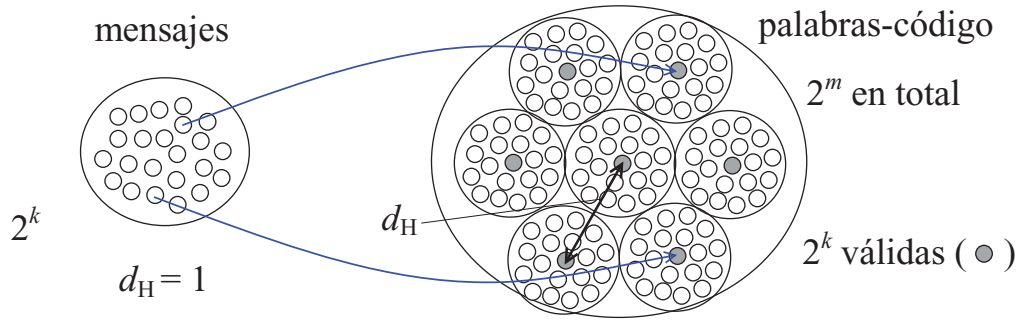
Codificación bloque (2)

- Códigos cíclicos

Codificación convolucional

Decodificación blanda

Realización de codificadores y decodificadores



¿Cómo codificar y decodificar?

Codificador

2^k *mensajes*	→	2^k *palabras-código*

Decodificador

2^m *palabra-código*	→	2^k mensajes o detección de error

Opción 1: Tablas de correspondencia

Opción 2: Algoritmos

Opción 3: Fórmulas → ej: con operaciones binarias

Realización de codificadores y decodificadores

Codificación bloque binaria lineal

- **c** = g (**d**) función **lineal**
- con operaciones suma y multiplicación binaria "**internas**" (módulo 2: el resultado es 0 ó 1)

$a, b \in \{0,1\}$

$a+b = (a+b)_2$ $\quad 0+0=0 \quad 0+1=1+0=1 \quad 1+1=0$

$a \times b = (a \times b)_2$ $\quad 0 \times 0 = 0 \quad 1 \times 0 = 0 \times 1 = 0 \quad 1 \times 1 = 1$

- La función lineal se puede definir con una matriz binaria

$$\mathbf{c} = \mathbf{G} \cdot \mathbf{d} \qquad c_i = \sum_j G_{ij} d_j$$

donde las sumas son internas (modulo 2) y $G_{ij} = 1$ ó 0

- **G** matriz generadora (del código)

Ejemplo Hamming (7,4) m=7, k=4,

$$\mathbf{d} = [d_1\ d_2\ d_3\ d_4]^T$$

$$\mathbf{c} = [c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7]^T$$

$$\mathbf{c} = \mathbf{G}\cdot\mathbf{d}$$

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}^T$$

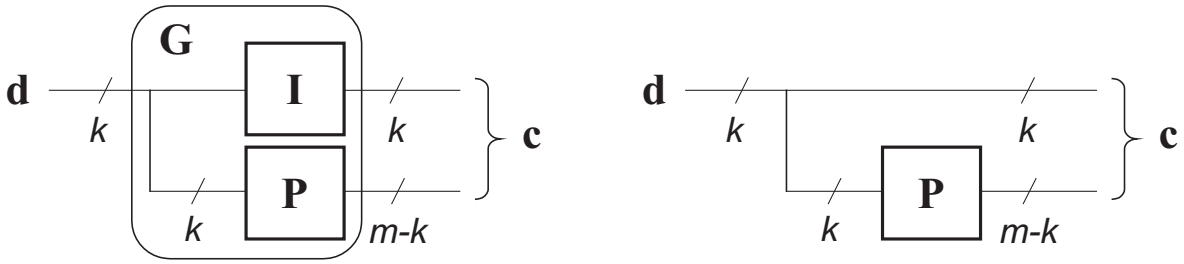
Ejemplo de codificación

$\mathbf{d} = [1\ 0\ 0\ 1]^T$

$$\mathbf{d} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad \mathbf{c} = \mathbf{G}\cdot\mathbf{d} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1+0+0+0 \\ 0+0+0+0 \\ 0+0+0+0 \\ 0+0+0+1 \\ 1+0+0+1 \\ 1+0+0+0 \\ 0+0+0+1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Codificación bloque binaria lineal

Codificación sistemática



- los primeros k bits son los mismos del mensaje
- los m - k bits restantes son combinaciones de los primeros ("paridades")

$$[c_1\, c_1 \ldots c_k]^T = \mathbf{I}\cdot\mathbf{d} \qquad [c_{k+1}\, c_{k+2} \ldots c_m]^T = \mathbf{P}\cdot\mathbf{d}$$

- **P**: matriz <u>generadora de paridades</u>

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} \\ \hline \mathbf{P} \end{bmatrix}$$

Codificación bloque binaria lineal

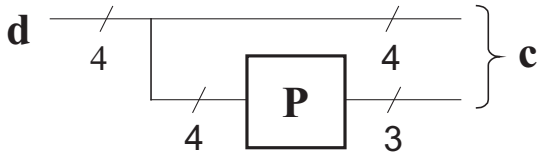
Codificación sistemática Ejemplo Hamming (7,4)

$$\mathbf{G} = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{array}\right] = \left[\begin{array}{c} \mathbf{I} \\ \hline \mathbf{P} \end{array}\right]$$

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \mathbf{I} \cdot \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} \qquad \begin{bmatrix} c_5 \\ c_6 \\ c_7 \end{bmatrix} = \mathbf{P} \cdot \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}$$

Matriz <u>generadora de paridades</u> (genera m-k bits de paridad: c_5, c_6, c_7)

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \qquad \begin{aligned} c_5 &= d_1 + d_3 + d_4 \\ c_6 &= d_1 + d_2 + d_3 \\ c_7 &= d_2 + d_3 + d_4 \end{aligned}$$



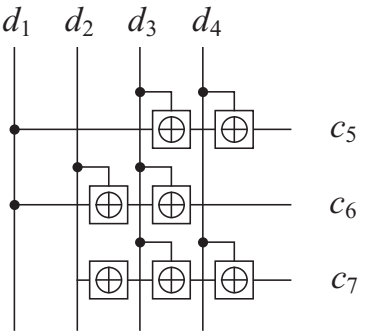
Codificación bloque binaria lineal

Codificación sistemática. Realización con "circuitos"

- Codificador: sólo hay que calcular los $m - k$ bits de paridad

Hamming (7,4)

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \qquad \begin{aligned} c_5 &= d_1 + d_3 + d_4 \\ c_6 &= d_1 + d_2 + d_3 \\ c_7 &= d_2 + d_3 + d_4 \end{aligned}$$



Codificación de canal

Codificación bloque (1)

- Codificación bloque binaria lineal
- Decodificación. Comprobación de paridad
- Generalización a más errores
- Recuperación de bits borrados

Codificación bloque binaria lineal

Decodificación

- Matriz comprobadora de paridades $\mathbf{H}^T = [\mathbf{P} \mid \mathbf{I}_{m-k}]$

- Ejemplo Hamming (7,4)

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \qquad \mathbf{H}^T = \left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

- Propiedad: Aplicada a la matriz generadora da cero

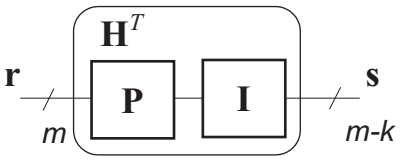
$$\mathbf{H}^T \cdot \mathbf{G} = \left[\mathbf{P} \mid \mathbf{I}\right] \cdot \begin{bmatrix} \mathbf{I} \\ \mathbf{P} \end{bmatrix} = \mathbf{P} \cdot \mathbf{I} + \mathbf{I} \cdot \mathbf{P} = \mathbf{P} + \mathbf{P} = \mathbf{0}$$

- Aplicada a cualquier palabra del código $\mathbf{c} = \mathbf{G} \cdot \mathbf{d}$ sale cero

$$\mathbf{H}^T \cdot \mathbf{c} = \mathbf{H}^T \cdot (\mathbf{G} \cdot \mathbf{d}) = (\mathbf{H}^T \cdot \mathbf{G}) \cdot \mathbf{d} = \mathbf{0}$$

Codificación bloque binaria lineal

Decodificación



- Aplicando la matriz comprobadora al vector recibido:

$$\mathbf{s} = \mathbf{H}^T \cdot \mathbf{r}$$

- si no sale cero, **r** no es una palabra válida (del código): ha habido errores

- "**Síndrome**" **s** (*m*-*k* bits)
  puede decir algo más, además de si ha habido o no errores

Codificación bloque binaria lineal

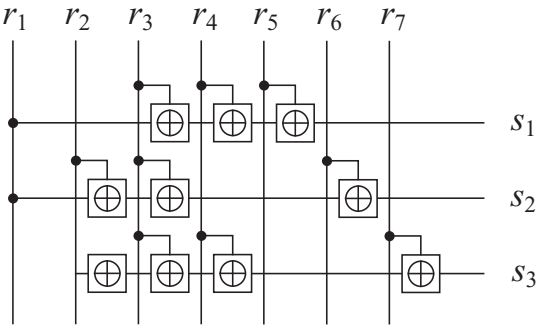
Decodificación. Realización con "circuitos"

- Decodificador: sólo hay que calcular los $m - k$ bits del **síndrome**

Hamming (7,4)

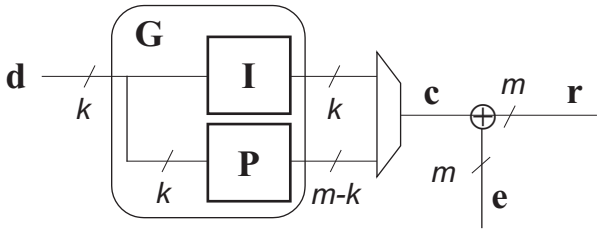
$$\mathbf{H}^T = \left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

$$s_1 = r_1 + r_3 + r_4 + r_5$$
$$s_2 = r_1 + r_2 + r_3 + r_6$$
$$s_3 = r_2 + r_3 + r_4 + r_7$$



Codificación bloque binaria lineal

Decodificación

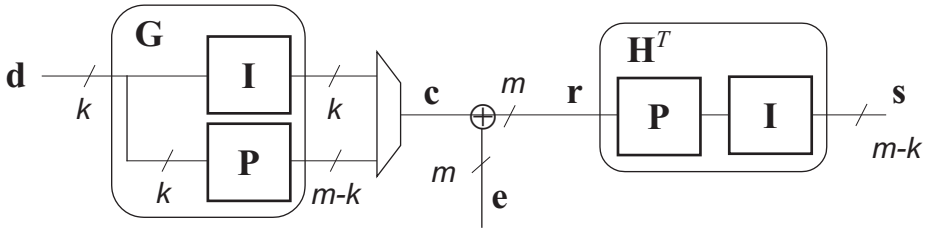


- Si ha habido errores el vector recibido es $\mathbf{r} = \mathbf{c} + \mathbf{e} = (\mathbf{G}\cdot\mathbf{d}) + \mathbf{e}$
  - siendo **e** un vector con "1s" en las posiciones con error
    (**r** tiene cambiados los bits donde **e** vale "1")

c	0	0	1	0	1	1	0
r	0	1	1	0	0	1	0
e	0	1	0	0	1	0	0

Codificación bloque binaria lineal

Decodificación



- Aplicando a **r** la matriz comprobadora se obtiene un ***síndrome***

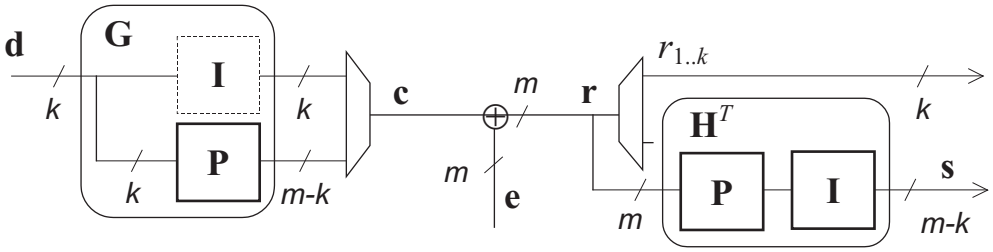
$$\mathbf{s} = \mathbf{H}^T \cdot \mathbf{r} = \mathbf{H}^T \cdot \mathbf{c} + \mathbf{H}^T \cdot \mathbf{e} = \mathbf{H}^T \cdot \mathbf{e} \qquad \text{(recordar: } \mathbf{H}^T \cdot \mathbf{c} = 0\text{)}$$

- El síndrome sólo depende de los errores $\mathbf{s} = \mathbf{H}^T \cdot \mathbf{e}$ y es cero si no hay errores

- A partir del síndrome se puede intentar averiguar el vector de errores $\hat{\mathbf{e}}$

Codificación bloque binaria lineal

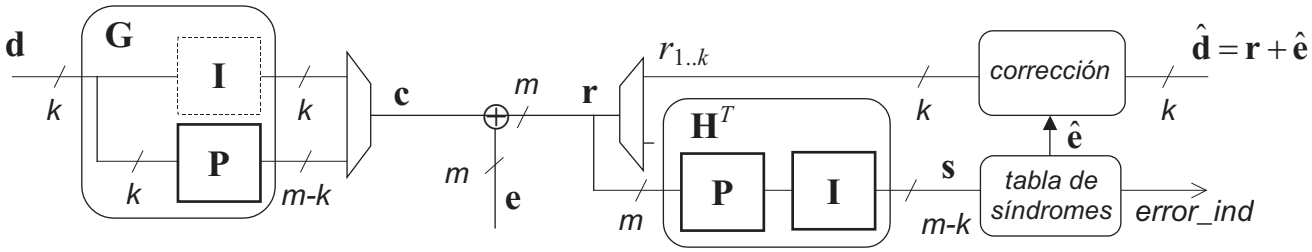
Codificación sistemática. Decodificación con síndrome



- **Sídrome** $\mathbf{s} = \mathbf{H}^T \mathbf{r} = [\mathbf{P} \mid \mathbf{I}]\, \mathbf{r}$ conjunto de ***síntomas*** sobre errores producidos
- 2^{m-k} síndromes no pueden representar todas las posibles combinaciones de errores
- Si se hacen **hipótesis** se puede reducir el número de combinaciones de error posibles
- Si las combinaciones se reducen a $< 2^{m-k}$ se puede hacer un **diagnósico** $\mathbf{s} \rightarrow \hat{\mathbf{e}}$
- El diagnóstico permite hacer una **corrección** (tratamiento) $\mathbf{r} + \hat{\mathbf{e}}$

Codificación bloque binaria lineal

Codificación sistemática. Decodificación con síndrome



- **Sídrome** $\mathbf{s} = \mathbf{H}^T \mathbf{r} = [\mathbf{P} \mid \mathbf{I}]\, \mathbf{r}$ conjunto de ***síntomas*** sobre errores producidos
- **Hípótesis** reduce combinaciones de error (ejemplo: no hay más de un error)
- **Tabla de síndromes** contiene los ***diagnósticos*** $\hat{\mathbf{e}}$ (dónde están los errores) para cada síndrome
- Los diagnósticos $\hat{\mathbf{e}}$ permiten **corregir** los **errores localizados** (e_i=1)

Codificación bloque binaria lineal

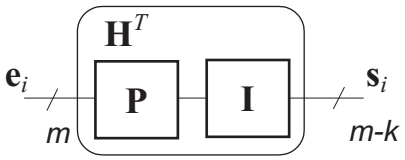
Decodificación con síndrome

Hamming (7,4) Hipótesis: no hay más de un bit erróneo

8 patrones con uno o ninguno bits erróneos

bit erróneo		síndrome
Ninguno	→	0 0 0
1	→	1 1 0
2	→	0 1 1
3	→	1 1 1
4	→	1 0 1
5	→	1 0 0
6	→	0 1 0
7	→	0 0 1

distintos



$$\mathbf{H}^T = \left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

Codificación bloque binaria lineal

Decodificación con síndrome

Hamming (7,4)

Tabla de síndromes ←

síndrome		bit erróneo
0 0 0	→	Ninguno
0 0 1	→	7
0 1 0	→	6
0 1 1	→	2
1 0 0	→	5
1 0 1	→	4
1 1 0	→	1
1 1 1	→	3

bit erróneo		síndrome
Ninguno	→	0 0 0
1	→	1 1 0
2	→	0 1 1
3	→	1 1 1
4	→	1 0 1
5	→	1 0 0
6	→	0 1 0
7	→	0 0 1

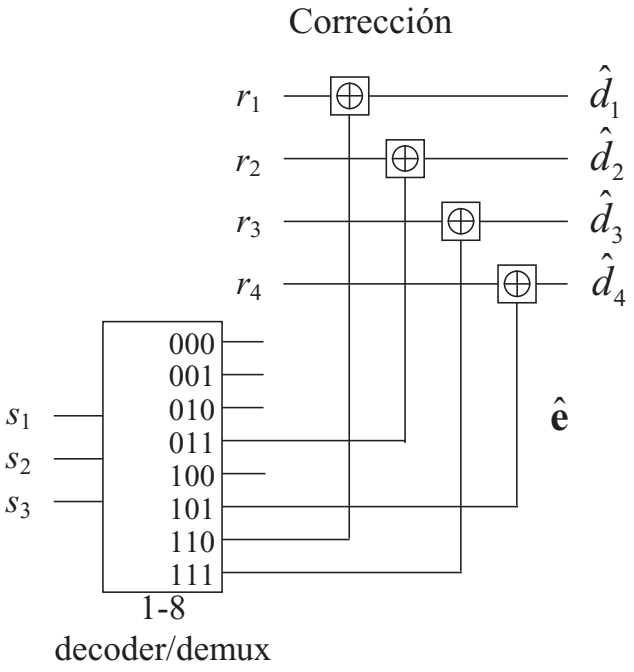
Codificación bloque binaria lineal

Decodificación con síndrome. Realización con "circuitos"

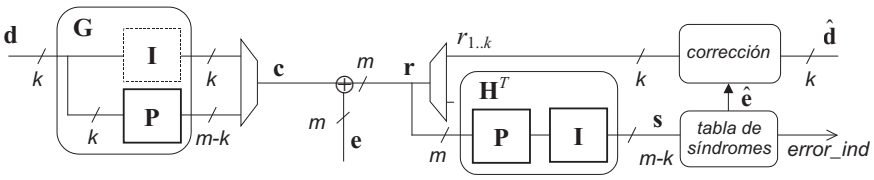
Hamming (7,4)

Tabla de síndromes

síndrome		bit erróneo
0 0 0	→	Ninguno
0 0 1	→	7
0 1 0	→	6
0 1 1	→	2
1 0 0	→	5
1 0 1	→	4
1 1 0	→	1
1 1 1	→	3



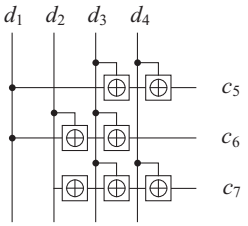
Codificación sistemática



Realización con "circuitos" Hamming (7,4)

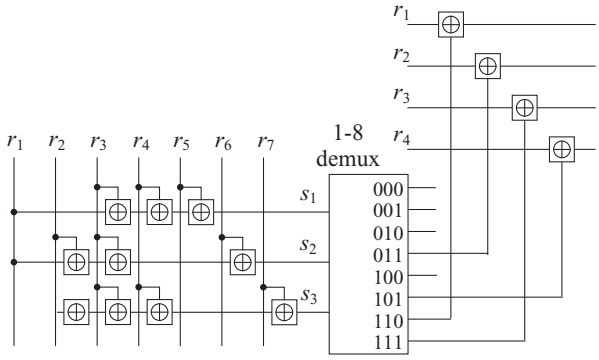
$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$$c_5 = d_1 + d_3 + d_4$$
$$c_6 = d_1 + d_2 + d_3$$
$$c_7 = d_2 + d_3 + d_4$$



$$\mathbf{H}^T = \left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

$$s_1 = r_1 + r_3 + r_4 + r_5$$
$$s_2 = r_1 + r_2 + r_3 + r_6$$
$$s_3 = r_2 + r_3 + r_4 + r_7$$



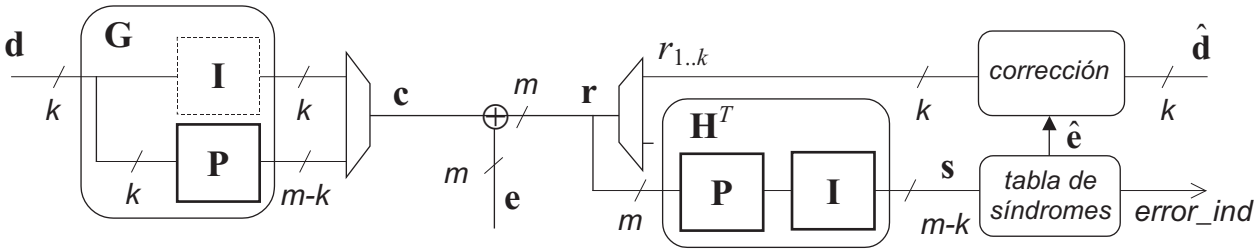
Codificación de canal

Codificación bloque (1)

- Codificación bloque binaria lineal
- Decodificación. Comprobación de paridad
- Generalización a más errores
- Recuperación de bits borrados

Codificación bloque binaria lineal

Codificación sistemática. Decodificación con síndrome



- **Sídrome** $\mathbf{s} = \mathbf{H}^T \mathbf{r} = [\mathbf{P} \mid \mathbf{I}]\, \mathbf{r}$ conjunto de ***síntomas*** sobre errores producidos
- **Hípótesis** reduce combinaciones de error (ejemplo: no hay más de un error)
- **Tabla de síndromes** contiene los ***diagnósticos*** $\hat{\mathbf{e}}$ (dónde están los errores) para cada síndrome
- Los diagnósticos $\hat{\mathbf{e}}$ permiten **corregir** los **errores localizados** (e_i=1)

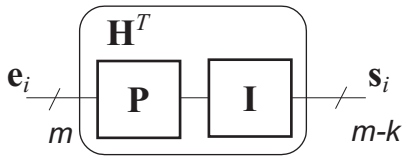
Codificación bloque binaria lineal

Generalización a más errores

Para diseñar la tabla de síndromes hasta t errores:

- Aplicar sucesivamente a $\mathbf{H}^T$ todos los patrones de error a diagnosticar (de 1, 2, ... t errores)
- Obtener el síndrome de cada uno:

$$\mathbf{e}_i \rightarrow \mathbf{s}_i$$



- Invertir la tabla $\mathbf{s}_i \rightarrow \hat{\mathbf{e}}_i$
- Hace falta que
  - Los patrones de error produzcan síndromes distintos (elección adecuada de **P**)
  - Haya suficientes síndromes para todos los patrones de error (¿cuántos?)

Codificación bloque binaria lineal

Generalización a más errores

- El **síndrome** debe ser capaz de representar **todos los patrones de error** que se quiere poder **diagnosticar**
- Si se quieren **corregir hasta t errores** en palabras de m bits, el número de patrones de error es:

$$\binom{m}{0}+\binom{m}{1}+\binom{m}{2}+\Lambda+\binom{m}{t} = \sum_{j=0}^{t}\binom{m}{j}$$

(patrones con 0 errores, con 1 error, con 2 errores , ... con t errores)

- El síndrome (m-k bits) puede representar hasta 2^{m-k} patrones de error

Para que permita diagnosticar **hasta t errores**, (m-k) debe cumplir:

$$2^{m-k} \geq \sum_{j=0}^{t}\binom{m}{j} \quad \Rightarrow \quad m-k \geq \log_2\left(\sum_{j=0}^{t}\binom{m}{j}\right)$$ número mínimo de paridades

Codificación bloque binaria lineal

Generalización a más errores

$$2^{m-k} \geq \sum_{j=0}^{t} \binom{m}{j} \quad \Rightarrow \quad m-k \geq \log_2 \left(\sum_{j=0}^{t} \binom{m}{j} \right) \text{ número mínimo de paridades}$$

- No son códigos de "distancia máxima"
- Si se conoce $\boldsymbol{d_H}$, la capacidad de corrección es $t = \mathrm{E}[(\boldsymbol{d_H} - 1)/2]$
- Para corregir t errores hace falta $\boldsymbol{d_H} - 1 \geq 2t$
- Si el código fuera de distancia máxima $\boldsymbol{d_H} = m - k + 1$ ...
  ... para corregir t errores haría falta sólo

$$m - k \geq 2t$$

Codificación bloque binaria lineal

Generalización a más errores

Ejemplos

$$2^{m-k} \geq \sum_{j=0}^{t} \binom{m}{j} \qquad m-k \geq \log_2 \left(\sum_{j=0}^{t} \binom{m}{j} \right)$$

m	t	patrones de error		$2^{(m-k)}$ (min)	$m-k$ (min)	$m-k$ min (si $\boldsymbol{d_H}$ max)	t
32	1	$\binom{32}{0}+\binom{32}{1}$	1+32 = **33**	64	6	2	1
32	2	$\binom{32}{0}+\binom{32}{1}+\binom{32}{2}$	1+32+496 = **529**	1024	10	4	2
32	3	$\binom{32}{0}+\binom{32}{1}+\binom{32}{2}+\binom{32}{3}$	1+32+496+ +4960 = **5489**	8192	13	6	3

- No son de "distancia máxima" ($\boldsymbol{d_H} = m - k + 1$)

Codificación bloque binaria lineal

Codificación sistemática

Ejemplos de códigos posibles

$$2^{m-k} \geq \sum_{j=0}^{t} \binom{m}{j} \qquad m-k \geq \log_2\left(\sum_{j=0}^{t} \binom{m}{j}\right)$$

m palabra-código	t errores que corrige	$m-k$ (min) paridades	k mensaje	(m,k)	k/m tasa cod.
32	1	6	26	(32,26)	0,81
24	1	5	19	(24,19)	0,79
16	1	5	11	(16,11)	0,69
32	2	10	22	(32,22)	0,69
24	2	9	15	(24,15)	0,62
16	2	8	8	(16, 8)	0,50
32	3	13	19	(32,19)	0,59
24	3	12	12	(24,12)	0,50
16	3	10	6	(16, 6)	0,37

No son de máxima distancia ($\boldsymbol{d}_{\mathbf{H}} < m-k+1$) pero son de fácil realización

Codificación de canal

Codificación bloque (1)

- Codificación bloque binaria lineal
- Decodificación. Comprobación de paridad
- Generalización a más errores
- Recuperación de bits borrados

Codificación bloque binaria lineal

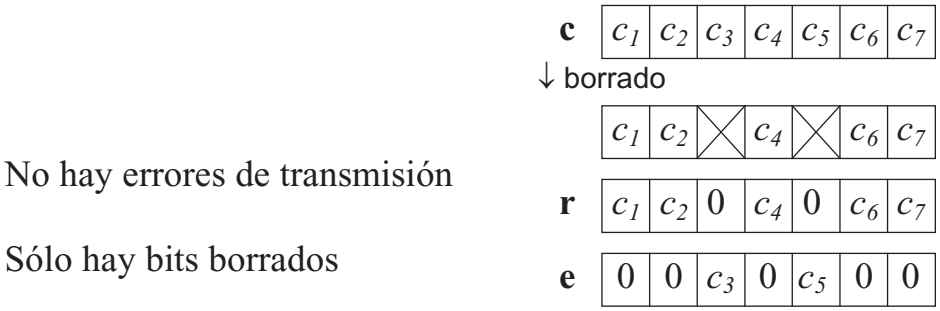
Codificación sistemática. Recuperación de bit perdidos (borrados)

- Bit perdido o "**borrado**": no se sabe su valor
- Codificación de canal permite recuperar bits borrados de los que:

  se sabe su posición pero no su valor (no hay errores en los no borrados)

¿Cómo?

- Se ponen en **r** los bit borrados (valor desconocido) a "0"
- Si **r** = **c** + **e**, el vector de error **e** es "0" en los bit no borrados y "1" ó "0" en los borrados, según el valor transmitido (**c**)



Codificación bloque binaria lineal

Codificación sistemática. Recuperación de bit perdidos (borrados)

- Se calcula el síndrome.
- Si el síndrome es capaz de representar todos los posibles patrones de error se podrán recuperar los bits borrados
- Para b bits borrados: el número de patrones de error distintos es 2^q

  ejemplo $b = 2$: **e** | 0 | 0 | e_3 | 0 | e_5 | 0 | 0 |
- Para que el síndrome puede representar todos los posibles patrones de error:

  $m\text{-}k \geq b$ (se pueden recuperar tantos bits borrados como redundantes hay)

(si no hay errores de transmisión):

- Se pueden borrar $m\text{-}k$ bits "sistemáticos" (los del mensaje $c_1 \ldots c_k$) y recuperarlos con los demás
- y, naturalmente,... se pueden borrar los $m\text{-}k$ bits "de paridad"

Codificación bloque binaria lineal

Codificación sistemática. Recuperación de bit perdidos (borrados)

Ejemplo Hamming (7,4) puede recuperar hasta 3 bits borrados

	Patrones error (**e**)	síndromes $\mathbf{s} = \mathbf{H}^T\mathbf{e}$
2 bits borrados (3 y 4)	0 0 **0 0** 0 0 0	0 0 0
	0 0 **0 1** 0 0 0	0 1 0
	0 0 **1 0** 0 0 0	0 0 1
	0 0 **1 1** 0 0 0	0 1 1

	Patrones error (**e**)	síndromes $\mathbf{s} = \mathbf{H}^T\mathbf{e}$
3 bits borrados (3, 4 y 5)	[0 0 **0 0 0** 0 0]	0 0 0
	[0 0 **0 0 1** 0 0]	1 0 0
	[0 0 **0 1 0** 0 0]	1 0 1
	[0 0 **0 1 1** 0 0]	0 0 1
	[0 0 **1 0 0** 0 0]	1 1 1
	[0 0 **1 0 1** 0 0]	0 1 1
	[0 0 **1 1 0** 0 0]	0 1 0
	[0 0 **1 1 1** 0 0]	1 1 0

Codificación de canal

Conceptos

Codificación bloque (1)

- Codificación bloque binaria lineal
- Decodificación. Comprobación de paridad
- Generalización a más errores
- Recuperación de bits borrados

Codificación bloque (2)

- Códigos cíclicos

Codificación convolucional

Decodificación blanda

Codificación de canal

Conceptos

Codificación bloque (1)

- Codificación bloque binaria lineal
- Decodificación. Comprobación de paridad
- Capacidad de corrección de errores

Codificación bloque (2)

- Códigos cíclicos

Codificación convolucional

Decodificación blanda

Codificación bloque (2)

Códigos cíclicos binarios

- Códigos cíclicos
- Codificación/Decodificación
- Realización

Códigos cíclicos no binarios

- Codificación bloque no binaria
- Códigos cíclicos no binarios.
- Códigos Reed-Solomon

Codificación concatenada

Codificación bloque binaria lineal

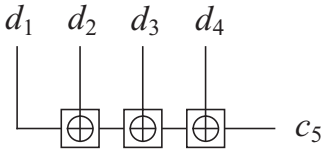
Codificación sistemática. Realización con "circuitos"

- Codificador: sólo hay que calcular los $m - k$ bits de paridad
- Decodificador: calcular los $m - k$ bits del síndrome

Paridad (4,3)

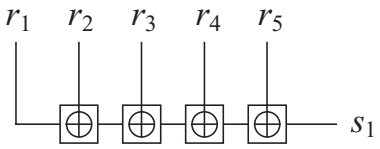
$$\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}$$

$$c_5 = d_1 + d_2 + d_3 + d_4$$



$$\mathbf{H}^T = \left[\begin{array}{cccc|c} 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

$$s_1 = r_1 + r_2 + r_3 + r_4 + r_5$$



Codificación bloque binaria lineal

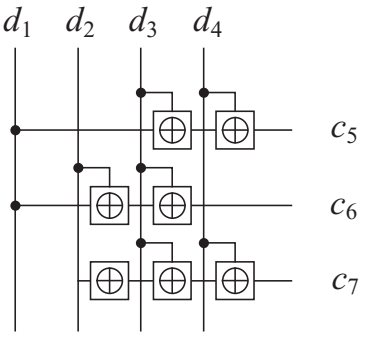
Codificación sistemática

Realización con "circuitos"

Hamming (7,4)

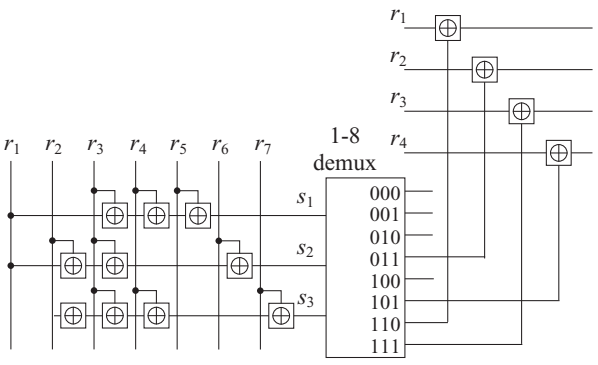
$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

$$c_5 = d_1 + d_3 + d_4$$
$$c_6 = d_1 + d_2 + d_3$$
$$c_7 = d_2 + d_3 + d_4$$



$$\mathbf{H}^T = \left[\begin{array}{cccc|ccc} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

$$s_1 = r_1 + r_3 + r_4 + r_5$$
$$s_2 = r_1 + r_2 + r_3 + r_6$$
$$s_3 = r_2 + r_3 + r_4 + r_7$$



Codificación lineal binaria

Complejidad

- Codificación: calcular las paridades $\mathbf{P}_{m\text{-}k,\,k} \cdot \mathbf{d}_k$
- Decodificación: calcular el síndrome $\mathbf{s}_{m\text{-}k} = \mathbf{H}^T_{m,\,m\text{-}k} \cdot \mathbf{r}_m$

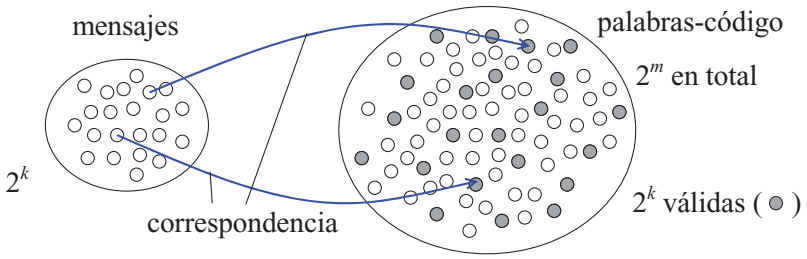
Ejemplo (1000, 984)

$k = 984$	bit de los mensajes
$m = 1000$	bits de las palabras-código
$m - k = 16$	bits de las paridades

- Codificación: calcular las paridades $\mathbf{P}_{16,\,984} \cdot \mathbf{d}_{984}$ $\approx$ 8.000 ex-or
- Decodificación: calcular el síndrome $\mathbf{H}^T_{1000,\,16} \cdot \mathbf{c}_{1000}$ $\approx$ 8.000 ex-or

- Para códigos muy largos (m) la complejidad es muy grande
- **códigos cíclicos** permiten el reducir la complejidad

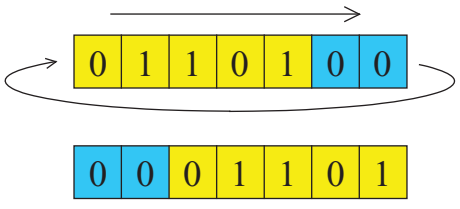
Códigos binarios cíclicos



Códigos cíclicos son los que cumplen esta propiedad:

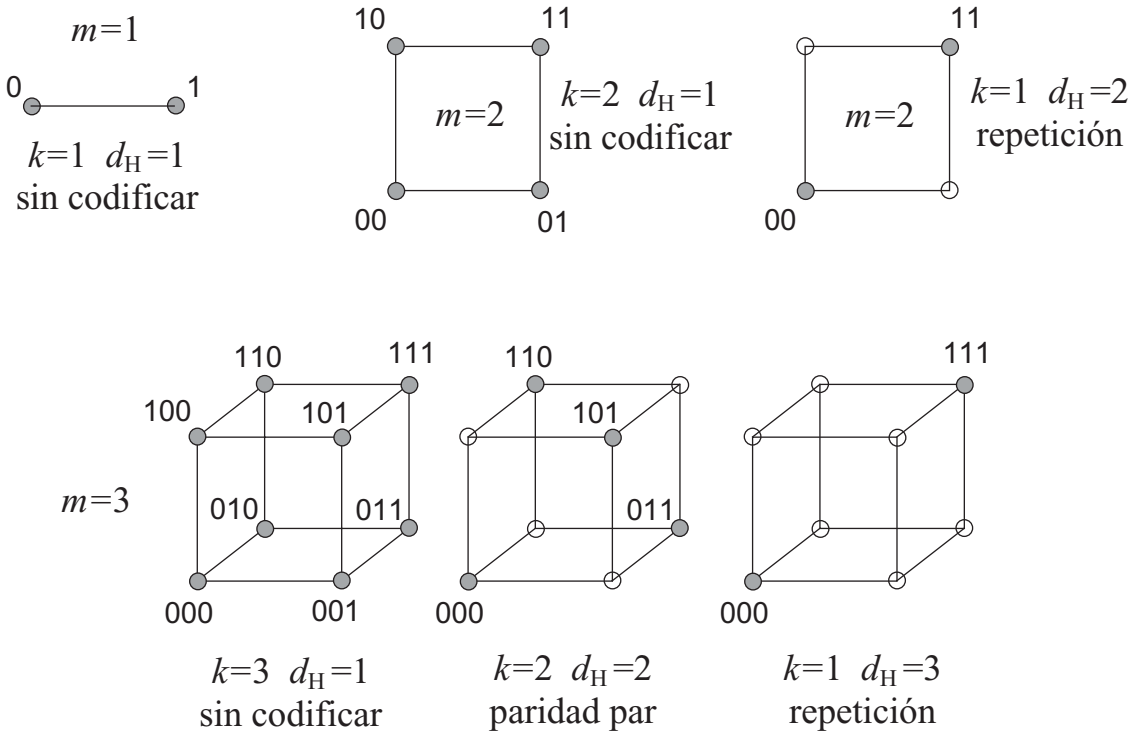
Si $\mathbf{c}=[c_1\ c_2\ \ldots c_{m\text{-}1}\ c_m\ ]^T$ pertenece al código (es una palabra válida)

cualquier **rotación** $[c_p\ c_{p+1}\ldots c_m\ c_1\ldots c_{p\text{-}1}]^T$ también pertenece al código

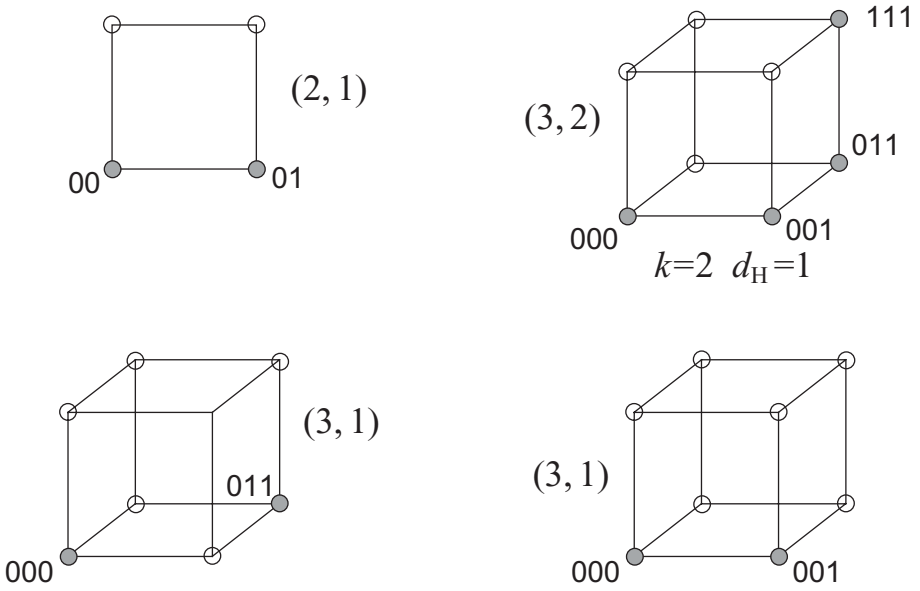


- Permiten reducir mucho la complejidad de los codificadores y decodificadores

Códigos binarios cíclicos



Códigos binarios no cíclicos



Códigos binarios cíclicos

Representación con polinomios

- Se pueden representar las palabras-código binarias (y también los mensajes) ...
  ... como si fueran los **coeficientes binarios de un polinomio** de grado $m-1$

$$c(X) = c_{m-1}X^{m-1} + \ ... \ + c_1X + c_0 \qquad \text{donde } c_i \text{ es } \textbf{binario} \text{ (0 ó 1)}$$

palabra-código

c_{m-1}	...	c_2	c_1	c_0

vector $\mathbf{c} = [c_0\ c_1\ c_3\ \ldots\ c_{m-1}]^T$

polinomio $c(X) = c_{m-1}X^{m-1} + \ ... \ + c_1X + c_0$

Codificación/decodificación:

- se puede hacer con operaciones con polinomios
- suma, multiplicación (convolución), división, etc.

Códigos binarios cíclicos

Representación con polinomios

- Producto de polinomios $c(X) = g(X) \cdot d(X)$ equivale a
- Convolución de vectores $\mathbf{c} = \mathbf{g} * \mathbf{d}$ y también a
- Producto de vector $\mathbf{d}$ por matriz de filas $\mathbf{g}$ repetidas cíclicamente
  $\mathbf{c} = \mathbf{G} \cdot \mathbf{d}$

$$\mathbf{G} = \begin{bmatrix} g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 & 0 \\ 0 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 \\ 0 & 0 & 0 & 0 & 0 & g_4 & g_3 & g_2 & g_1 & g_0 \end{bmatrix}^T$$

Códigos binarios cíclicos

Representación con polinomios

Propiedad

- Si un código formado por 2^k palabras válidas de m bits es cíclico ...

  ... existe **un único polinomio** $g(X)$, de grado $m - k$, tal que ...

  ... **todos los polinomios del código son divisibles** por él

$$g(X) = g_{m-k}X^{m-k} + g_{m-k-1}X^{m-k-1} + \ ... \ + g_1X + g_0 \qquad g_0 = g_{m\text{-}k} = 1$$

- Si se conoce $g(X)$ ...

  ... se puede saber si una palabra pertenece al código ...

  ... dividiendo su polinomio $c(X)$ por $g(X)$ y comprobando si el resto es cero

Codificación de canal - Códigos cíclicos

Códigos cíclicos binarios

- Códigos cíclicos
- Codificación/Decodificación
- Realización

Códigos cíclicos no binarios

- Codificación bloque no binaria
- Códigos cíclicos no binarios.
- Códigos Reed-Solomon

Codificación concatenada

Códigos binarios cíclicos

Codificación

- Los mensajes y palabras-código se consideran:
  coeficientes binarios de polinomios de grado $k-1$ y $m-1$, respectivamente

- $g(X)$ (de grado $m-k$) se denomina **polinomio generador del código**

- Multiplicando el polinomio de un mensaje (grado $k-1$) ...
  ... por el polinomio generador (grado m-k) se obtiene un ...
  ... polinomio de grado $m-k+k-1=m-1$ ...
  ... que pertenece al código al ser divisible por g(X)

- Una forma de codificación (no sistemática) puede ser esa:
  El polinomio de la palabra-código se obtiene
  multiplicando el polinomio mensaje por el polinomio generador

Códigos binarios cíclicos. Codificación (no sistemática)

- Un **mensaje** de k bits se representa con un polinomio de grado k-1:

$$d(X) = d_{k-1}X^{k-1} + \ldots + d_1X + d_0$$

mensaje

d_{k-1}	…	d_2	d_1	d_0

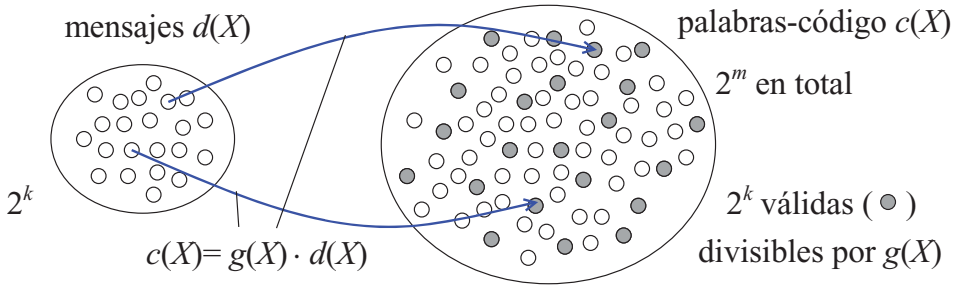
- Su **palabra-código asociada**, de m bits, es la representada por el polinomio que se obtiene **multiplicando por el polinomio generador**:

$$c(X) = g(X)\cdot d(X) = c_{m-1}X^{m-1} + \ldots + c_1X + c_0$$

palabra-código

c_{m-1}	…	c_2	c_1	c_0

- En vez de la matriz generadora **G** se usa el **polinomio generador**



Códigos binarios cíclicos. Codificación (no sistemática)

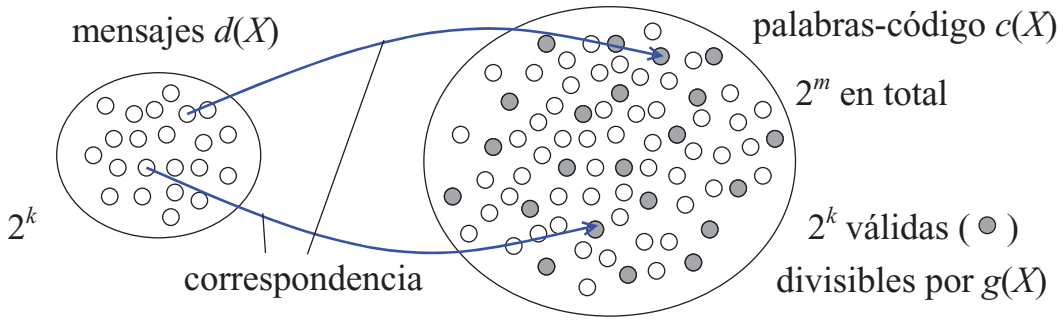
- Codificar con $c(X) = g(X) \cdot d(X)$ equivale a usar la siguiente
- **Matriz generadora G**:
  todas sus filas iguales al vector de coeficientes del polinomio generador, con sucesivos desplazamientos:

- Ejemplo: código (10,6) con polinomio generador $g(X)$ de grado $m - k = 4$

$$\mathbf{G} = \begin{bmatrix} g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 & 0 \\ 0 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 & 0 \\ 0 & 0 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & g_4 & g_3 & g_2 & g_1 & g_0 & 0 \\ 0 & 0 & 0 & 0 & 0 & g_4 & g_3 & g_2 & g_1 & g_0 \end{bmatrix}^T$$

Códigos binarios cíclicos.

Codificación sistemática



- Usar el código generado por $g(X)$ pero ...
  ... cambiar la codificación para que sea sistemática

- En vez de $c(X) = g(X) \cdot d(X)$ usar otra correspondencia para que:
  $c(X)$ tenga sus primeros k coeficientes iguales a los de $d(X)$

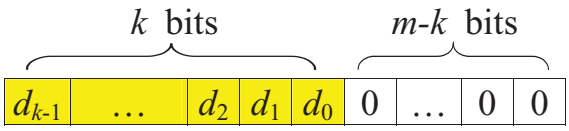
¿Cómo?

Códigos binarios cíclicos

Codificación sistemática

- Formar un polinomio de grado m-1 con el mensaje desplazado m-k bits:

$$d(X)\cdot X^{m-k} = \\ = d_{k-1}X^{m-1} + d_{k-2}X^{m-2} + \ \ldots\ + d_0X^{m-k}$$



- Dividir por $g(X)$ y obtener el resto

$$\frac{d(X)\cdot X^{m-k}}{g(X)} = q(X) + \frac{p(X)}{g(X)} \qquad d(X)\cdot X^{m-k} = g(X)\cdot q(X) + p(X)$$

donde q(X) es el cociente entero y $p(X)$ es el resto (grado < grado de $g(X)$)

- Sumar $p(X)$ (Nótese que $p(X) + p(X) = 0$)

$$d(X)\cdot X^{m-k} + p(X) = g(X)\cdot q(X) + p(X) + p(X) \ = \ g(X)\cdot q(X)$$

$c(X) = d(X)\cdot X^{m-k} + p(X)$ - es divisible por $g(X)$ $\Rightarrow$ pertenece al código

Códigos binarios cíclicos

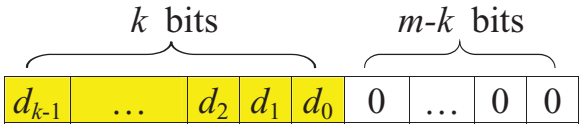
Codificación sistemática d → c

$c(X) = d(X)\cdot X^{m-k} + p(X)$ - es divisible por $g(X)$ $\Rightarrow$ pertenece al código

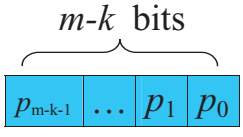
Si a un número se le resta (= suma) el resto de dividir por D ...
... el resultado es divisible por D

¿Cómo es ?

$$d(X)\cdot X^{m-k} = \\ = d_{k-1}X^{m-1} + d_{k-2}X^{m-2} + \ \ldots\ + d_0X^{m-k}$$



$$p(X) = \\ = p_0 + p_1X + \ \ldots\ + p_{m-k-1}X^{m-k-1}$$

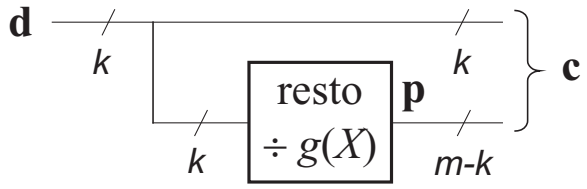


sumar equivale a concatenar:

- los primeros k bits son iguales a los del mensaje: codificación sistemática

Códigos binarios cíclicos

Codificación sistemática



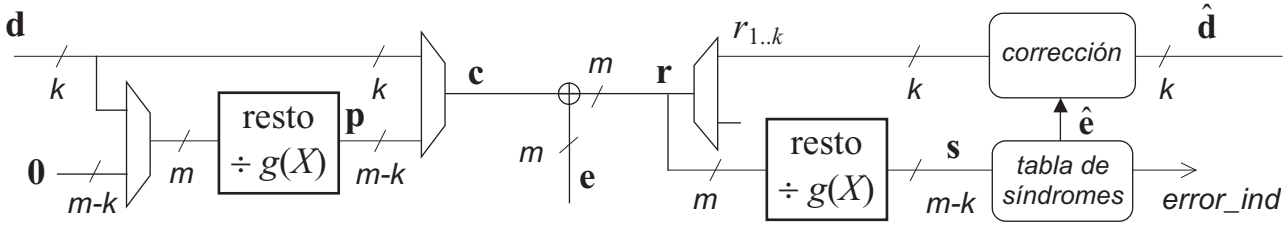
- Los bits de paridad se obtienen calculando el **resto de dividir por $g(X)$**
- La palabra código se obtiene concatenando el mensaje con los m-k bits de paridad

Decodificación

- El **síndrome** (m-k bits) se puede obtener mediante el **resto de dividir el vector recibido por el polinomio generador $g(X)$**
- Si **no hay errores**, el vector recibido pertenece al código, es divisible por $g(X)$ y el **sindrome es cero**
- Si hay errores el **síndrome** puede permitir calcular algunos **patrones de error**

Códigos binarios cíclicos

Codificación sistemática



Codificador: cálculo de la paridad (m-k) bits	**resto de dividir** un polinomio de grado m-1 por el polinomio generador de grado m-k
Decodificador cálculo del síndrome (m-k) bits	

- **Resto de dividir**: se puede realizar fácilmente con **registros de desplazamiento**

Codificación de canal - Códigos cíclicos

Códigos cíclicos binarios

- Códigos cíclicos
- Codificación/Decodificación
- Realización

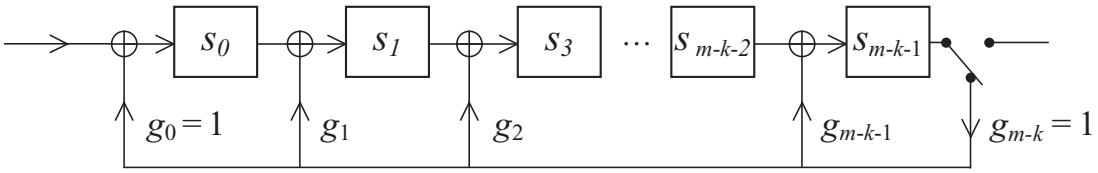
Códigos cíclicos no binarios

- Codificación bloque no binaria
- Códigos cíclicos no binarios.
- Códigos Reed-Solomon

Codificación concatenada

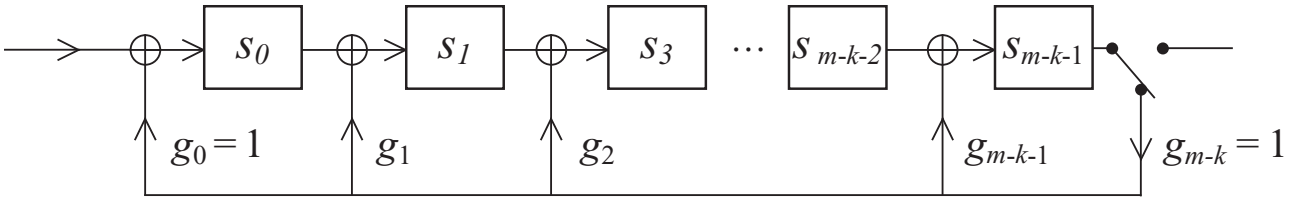
Códigos binarios cíclicos

- **"Circuito"** para realización del **cálculo del resto de dividir** por el polinomio $g(X)$
- Registro de desplazamiento con EX-OR intercaladas

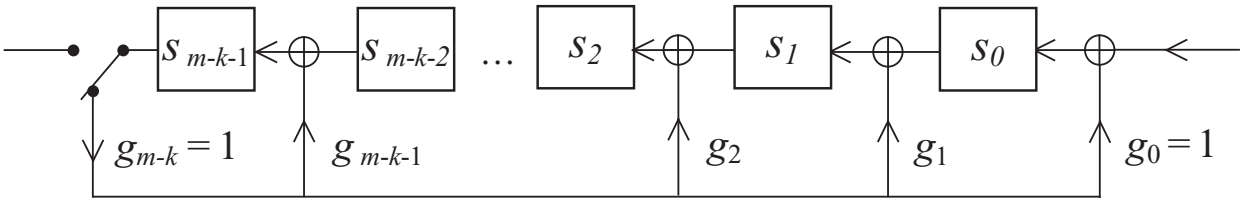


- El registro se inicializa a cero
- Los m bits del dividendo entran en serie empezando por el coeficiente de X^{m-1}
  (al mismo tiempo se pueden ir transmitiendo y ya no es necesario guardarlos)
- Al final se tienen en el registro los m-k bits del resto
  (paridades en el transmisor o síndrome en el receptor)
- Se puede sacar el resto en serie desconectando la realimentación

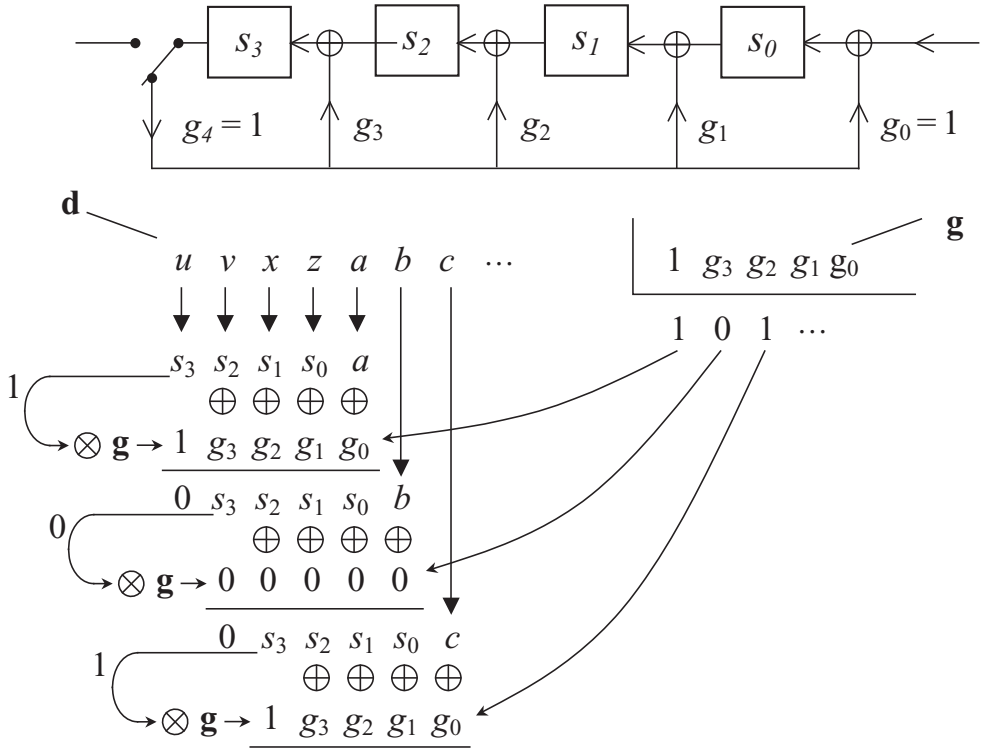
Códigos binarios cíclicos. Cálculo del resto



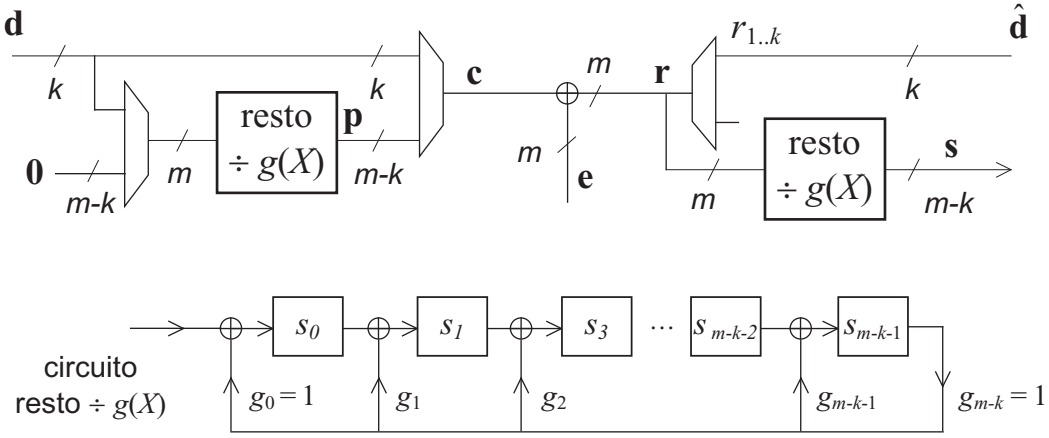
Al revés...



Códigos binarios cíclicos. Cálculo del resto



Detección de error con códigos binarios cíclicos



- (Tx y Rx): el cociente no se necesita; sólo requiere memoria para almacenar el **resto**
- (Tx y Rx): se puede **realizar "en serie"**: mientras se envían los bits del dividendo se van introduciendo en el divisor que al final contiene el resto
- (Tx): al final se extrae y se transmite el resto en serie (paridades)
- Se puede usar para **cualquier longitud** (k) de los mensajes (pueden ser ~1000)

Códigos binarios cíclicos

CRC: **C**yclic **R**edundancy **C**heck

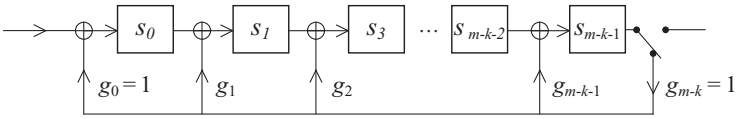
	m-k	*Polinomio generador*	*Sistema*
CRC-16-IBM	16	$x^{16} + x^{15} + x^{13} + x^7 + x^4 + x^2 + 1$	SDLC
CRC-16-ANSI	16	$x^{16} + x^{15} + x^2 + 1$	USB
CRC-16-CCITT	16	$x^{16} + x^{12} + x^5 + 1$	X.25, *Bluetooth*
CRC-16-DECT	16	$x^{16} + x^{10} + x^8 + x^7 + x^3 + 1$	DECT
CRC-32-CCITT	32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+$ $+x^{10}+x^8+x^7+x^5+x^4+x^2+x^1+1$	*Ethernet*, Serial ATA, MPEG-2, PNG
ITU-T I.432.1	8	$x^8 + x^2 + x + 1$	ISDN-BB, GPON
GPRS	12 8	$x^{12} + x^{11} + x^{10} + x^8 + x^5 + x^4 + 1$ $x^8 + x^6 + x^3 + 1$	RLC - datos RLC - cabecera

- La distancia de Hamming del código depende de $(m - k)$ y de la elección de $g(X)$
- Longitud variable m
- Los códigos se pueden **acortar** y **alargar** manteniendo $(m - k)$

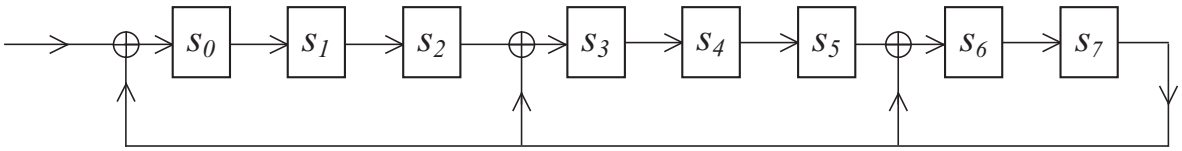
Códigos binarios cíclicos

CRC: **C**yclic **R**edundancy **C**heck

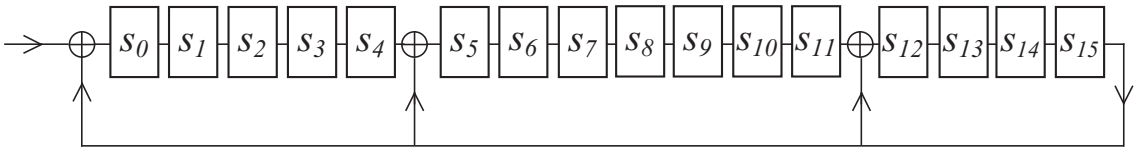
Ejemplos de divisores para cálculo de resto



Cabecera RLC EDGE $g(x) = x^8 + x^6 + x^3 + 1$



CRC-16-CCITT $g(x) = x^{16} + x^{12} + x^5 + 1$



Codificación bloque (2)

Códigos cíclicos binarios

- Códigos cíclicos
- Codificación/Decodificación
- Realización

Códigos cíclicos no binarios

- Codificación bloque no binaria
- Códigos cíclicos no binarios.
- Códigos Reed-Solomon

Codificación concatenada

Codificación bloque (2)

Códigos cíclicos binarios

- Códigos cíclicos
- Codificación/Decodificación
- Realización

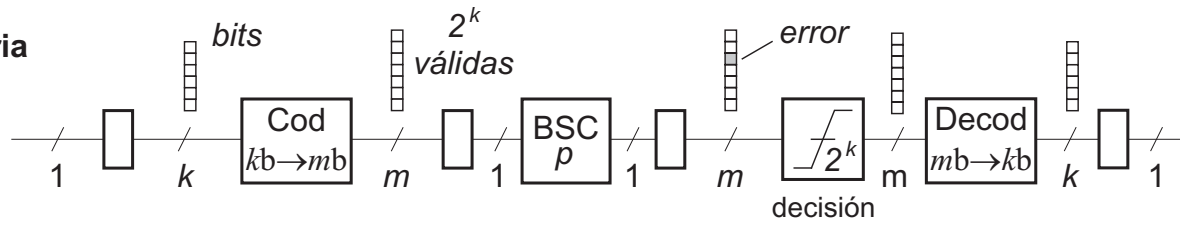
Códigos cíclicos no binarios

- Codificación bloque no binaria
- Códigos cíclicos no binarios.
- Códigos Reed-Solomon

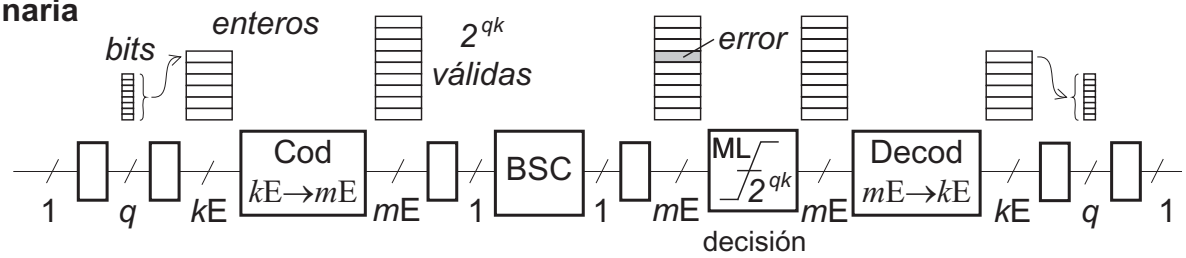
Codificación concatenada

Codificación no binaria

Binaria



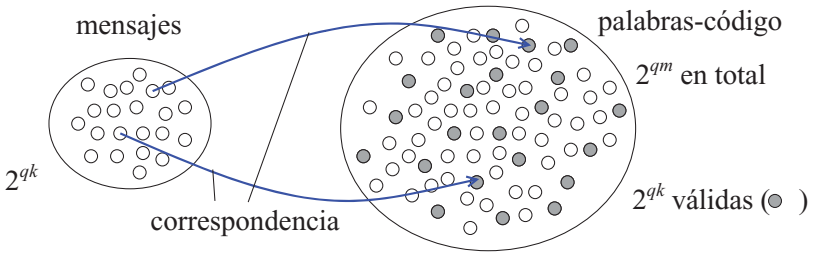
No binaria



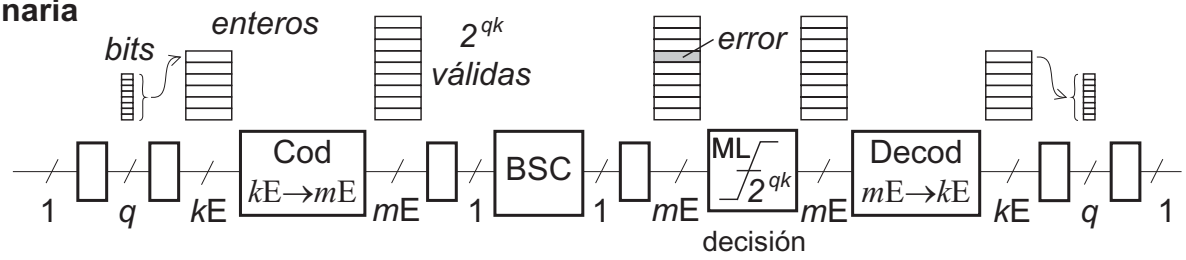
- Se **agrupan** q **bits** y se codifican con un **número entero** (de 0 a 2^q-1)
- Se forman **mensajes** de k enteros y se asocian a **palabras-código** de m enteros
- El **código** tiene sólo 2^{qk} **palabras-código** de enteros de las 2^{qm} posibles

Codificación de canal

¿Cómo funciona?



No binaria



- Al transmitir una palabra-código, los errores hacen que se reciba otra distinta
- Si la palabra-código recibida no es válida (no pertenece al código), se sabe que ha habido errores y en ese caso...
- Se decide una de las 2^{qk} palabra-código válidas: **la más "parecida"** (**ML**)

Codificación no binaria

Realización de codificadores y decodificadores

- ¿Se puede hacer la codificación y decodificación con fórmulas $\mathbf{c} = g(\mathbf{d})$?
  con $\mathbf{d}$ vector de k enteros (0 a 2^q-1) (mensaje)
  $\mathbf{c}$ vector de m enteros (0 a 2^q-1) (palabra-código)

- ¿Se pueden representar las operaciones con productos de matrices $\mathbf{c} = \mathbf{G}\cdot\mathbf{d}$?

- Para asegurar que el resultado ($\mathbf{c}$) sea un entero (0 a 2^q-1) es necesario que …
  … todas las operaciones suma y multiplicación sean "**internas**":
  resulten en un entero (0 a 2^q-1)

- Se utilizan **cuerpos de Galois** GF(M) que son
  conjuntos finitos de $M = 2^q$ enteros
  con operaciones + y × internas

Codificación no binaria

Codificación bloque entera no binaria

$\mathbf{c} = \mathbf{G} \cdot \mathbf{d}$ $\quad c_i = \sum_j G_{ij} d_j$

d (mensaje)	vector de k enteros (0 a 2^q-1)
c (palabra-código)	vector de m enteros (0 a 2^q-1)
G matriz generadora	$m \times k$ enteros (0 a 2^q-1)

- Los elementos de **d**, **c** y **G** son de un cuerpo de Galois
- Todas las operaciones suma y multiplicación son "**internas**": resultan en un entero entero del conjunto (0 a 2^q-1)

codificación binaria	**G** sólo tiene "0" ó "1" → **c** es una suma de elementos de **d** (paridades)
codificación no binaria	**G** tiene valores mayores de "1" → **c** se calcula con productos y sumas internos

Codificación bloque no binaria

Cuerpos de Galois

- GF(M): conjunto finito de M enteros con operaciones + y × internas
- Las operaciones suma y multiplicación resultan en un elemento del conjunto
- Se usan con $M = 2^q$ que representan un "alfabeto" de palabras binarias de q bits

GF($2^1 = 2$) 1 bit 0, 1

$0 + 0 = 0$	$0 \times 0 = 0$
$0 + 1 = 1$	$0 \times 1 = 0$
$1 + 0 = 1$	$1 \times 0 = 0$
$1 + 1 = 0$	$1 \times 1 = 1$

GF($2^2 = 4$) 2 bits 0, 1, 2, 3

$0 + 0 = 0$	$2 + 0 = 2$
$0 + 1 = 1$	$2 + 1 = 3$
$0 + 2 = 2$	$2 + 2 = 0$
$0 + 3 = 3$	$2 + 3 = 1$
$1 + 0 = 1$	$3 + 0 = 3$
$1 + 1 = 0$	$3 + 1 = 2$
$1 + 2 = 3$	$3 + 2 = 1$
$1 + 3 = 2$	$3 + 3 = 0$

$0 \times 0 = 0$	$2 \times 0 = 0$
$0 \times 1 = 0$	$2 \times 1 = 2$
$0 \times 2 = 0$	$2 \times 2 = 3$
$0 \times 3 = 0$	$2 \times 3 = 1$
$1 \times 0 = 0$	$3 \times 0 = 0$
$1 \times 1 = 1$	$3 \times 1 = 3$
$1 \times 2 = 2$	$3 \times 2 = 1$
$1 \times 3 = 3$	$3 \times 3 = 2$

GF($2^3 = 8$) 3 bits 0, 1, 2, 3, 4, 5, 6, 7

$0+0=0$	$2+0=2$	$4+0=4$	$6+0=6$
$0+1=1$	$2+1=3$	$4+1=5$	$6+1=7$
$0+2=2$	$2+2=0$	$4+2=6$	$6+2=4$
$0+3=3$	$2+3=1$	$4+3=7$	$6+3=5$
$0+4=4$	$2+4=6$	$4+4=0$	$6+4=2$
$0+5=5$	$2+5=7$	$4+5=1$	$6+5=3$
$0+6=6$	$2+6=4$	$4+6=2$	$6+6=0$
$0+7=7$	$2+7=5$	$4+7=3$	$6+7=1$
$1+0=1$	$3+0=3$	$5+0=5$	$7+0=7$
$1+1=0$	$3+1=2$	$5+1=4$	$7+1=6$
$1+2=3$	$3+2=1$	$5+2=7$	$7+2=5$
$1+3=2$	$3+3=0$	$5+3=6$	$7+3=4$
$1+4=5$	$3+4=7$	$5+4=1$	$7+4=3$
$1+5=4$	$3+5=6$	$5+5=0$	$7+5=2$
$1+6=7$	$3+6=5$	$5+6=3$	$7+6=1$
$1+7=6$	$3+7=4$	$5+7=2$	$7+7=0$

$0\times0=0$	$2\times0=0$	$4\times0=0$	$6\times0=0$
$0\times1=0$	$2\times1=2$	$4\times1=4$	$6\times1=6$
$0\times2=0$	$2\times2=4$	$4\times2=3$	$6\times2=7$
$0\times3=0$	$2\times3=6$	$4\times3=7$	$6\times3=1$
$0\times4=0$	$2\times4=3$	$4\times4=6$	$6\times4=5$
$0\times5=0$	$2\times5=1$	$4\times5=2$	$6\times5=3$
$0\times6=0$	$2\times6=7$	$4\times6=5$	$6\times6=2$
$0\times7=0$	$2\times7=5$	$4\times7=1$	$6\times7=4$
$1\times0=0$	$3\times0=0$	$5\times0=0$	$7\times0=0$
$1\times1=1$	$3\times1=3$	$5\times1=5$	$7\times1=7$
$1\times2=2$	$3\times2=6$	$5\times2=1$	$7\times2=5$
$1\times3=3$	$3\times3=5$	$5\times3=4$	$7\times3=2$
$1\times4=4$	$3\times4=7$	$5\times4=2$	$7\times4=1$
$1\times5=5$	$3\times5=4$	$5\times5=7$	$7\times5=6$
$1\times6=6$	$3\times6=1$	$5\times6=3$	$7\times6=4$
$1\times7=7$	$3\times7=2$	$5\times7=6$	$7\times7=3$

Codificación de canal - Códigos cíclicos

Códigos cíclicos binarios

- Códigos cíclicos
- Codificación/Decodificación
- Realización

Códigos cíclicos no binarios

- Codificación bloque no binaria
- Códigos cíclicos no binarios
- Códigos Reed-Solomon

Codificación concatenada

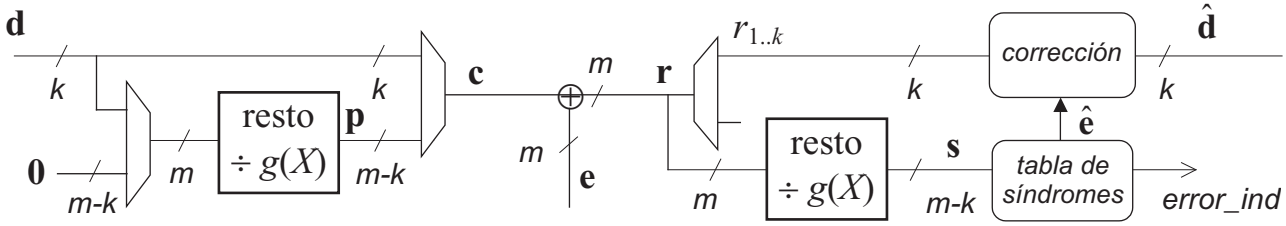
Codificación bloque no binaria

Códigos cíclicos no binarios

- Para reducir operaciones se pueden usar **códigos cíclicos**
- El **polinomio generador** $g(X)$ es de coeficientes enteros en GF(2^q)

Codificación/decodificación sistemática:

- Igual que en caso binario pero números y con operaciones en GF(2^q)

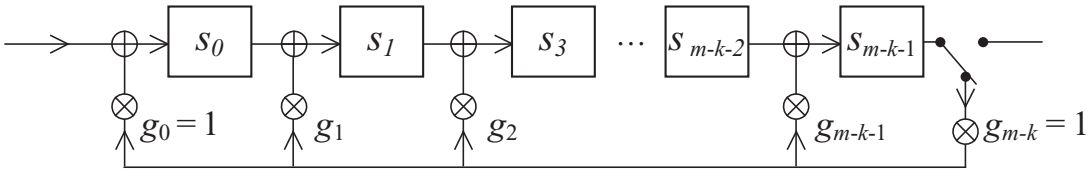


- Diagnósticos (**ê**): localización y corrección

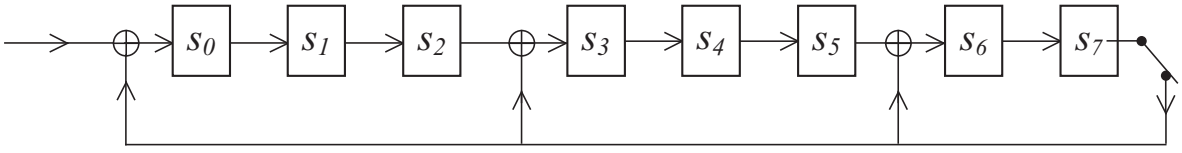
Codificación bloque no binaria

Códigos cíclicos no binarios

Circuito para realización del **cálculo del resto de dividir** por el polinomio g(X)



- Números y operaciones en GF(2^q)
- En el caso binario GF(2^1) las multiplicaciones son sólo conexiones que están o no están:



Codificación de canal - Códigos cíclicos

Códigos cíclicos binarios

- Códigos cíclicos
- Codificación/Decodificación
- Realización

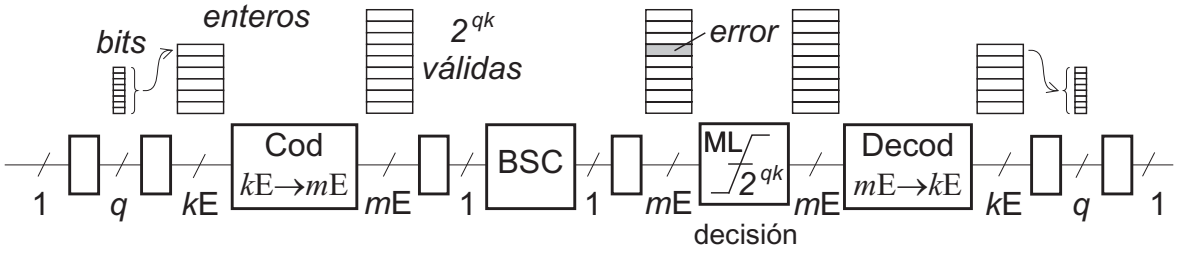
Códigos cíclicos no binarios

- Codificación bloque no binaria
- Códigos cíclicos no binarios
- Códigos Reed-Solomon

Codificación concatenada

Códigos Reed-Solomon

- Son códigos <u>cíclicos no binarios</u> con $m = 2^q - 1$



- Óptimos: **máxima distancia** $d_H = m - k + 1$ (número de símbolos distintos)

Parámetros: *típico*

q	número de bits de alfabeto de símbolos	8
2^q	tamaño del alfabeto de símbolos enteros	256
$m = 2^q - 1$	número de símbolos de las palabras-código	255
$k\ (< m)$	número de símbolos de los mensajes ($m - k$ siempre par)	

Códigos Reed-Solomon

Capacidad de detección y corrección

- **Máxima distancia** $d_H = m - k + 1$ (número de símbolos distintos)
- Puede **corregir** hasta $t = (m-k)/2$ símbolos erróneos (no sabiendo dónde están)
- Puede **detectar** hasta $m-k$ símbolo erróneos (no sabiendo dónde están)
- Puede **recuperar** hasta $m-k$ símbolos **borrados** (sabiendo dónde están)

Ejemplos

	corrección errores t	detección errores $m-k$	recuperación borrados $m-k$	tasa de codificación k/m
(255,223)	16	32	32	0,87
(255,239)	8	16	16	0,94
(255,251)	2	4	4	0,98

Códigos Reed-Solomon

Acortamiento

- Si se necesita codificador con m y k más pequeños …
  … se puede "acortar" el de $m = 255$
- Acortar: quitar símbolos de entrada y de salida
- Si se acortan por igual m y k: $m-k$ (número de "paridades") no varía y se sigue teniendo la misma capacidad absoluta de corrección/detección/recuperación

Ejemplos

	Código madre	Acortado a	Acorta-miento	Bloque k	corrección t	detección $m-k$	tasa
ADSL	(255, 223)	-	-	223	16	32	0,87
TDT	(255, 239)	(204,188)	51	188	8	16	0,92
CD	(255,251)	(32,28)	223	28	2	4	7/8
CD	(255,251)	(28,24)	227)	24	2	4	6/7
DVD	(255,239)	(208,192)	47	192	8	16	0,92
DVD	(255,239)	(182,172)	67 y 6 perf	172	5	10	0,94

Codificación de canal - Códigos cíclicos

Códigos cíclicos binarios

- Códigos cíclicos
- Codificación/Decodificación
- Realización

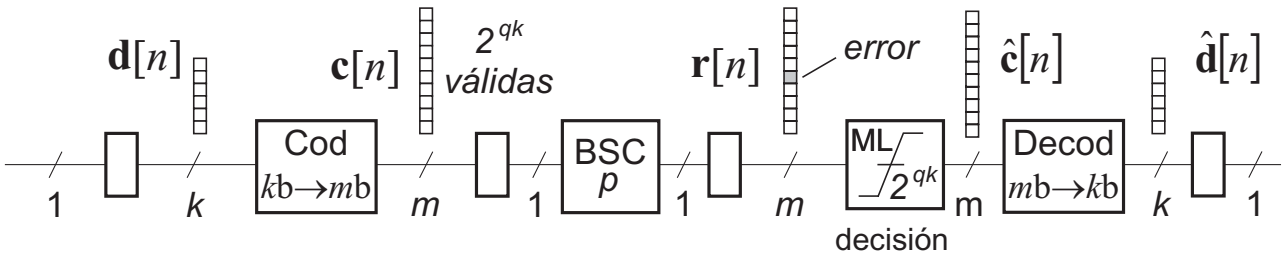
Códigos cíclicos no binarios

- Codificación bloque no binaria
- Códigos cíclicos no binarios
- Códigos Reed-Solomon

Codificación concatenada

Codificación de secuencias

Codificación (m, k) de una secuencia de bits (larga o indefinida)



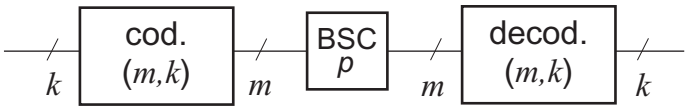
- Cada intervalo de codificación [n] se toman k nuevos bits de entrada
- Se puede usar un codificador/decodificador bloque (Hamming, cíclico, etc.)

  $\mathbf{c}[n]=f(\mathbf{d}[n])$ la palabra código en [n] se calcula con el mensaje en [n]
- El proceso puede ser indefinido

Codificación bloque binaria lineal

Mejora de la tasa de errores. Canal binario simétrico

Ejemplo: Codificador (*m*,*k*) que corrige hasta *t* errores



- Probabilidad de <u>bloque</u> no corregido ($t+1$ o más errores en bloque de m):

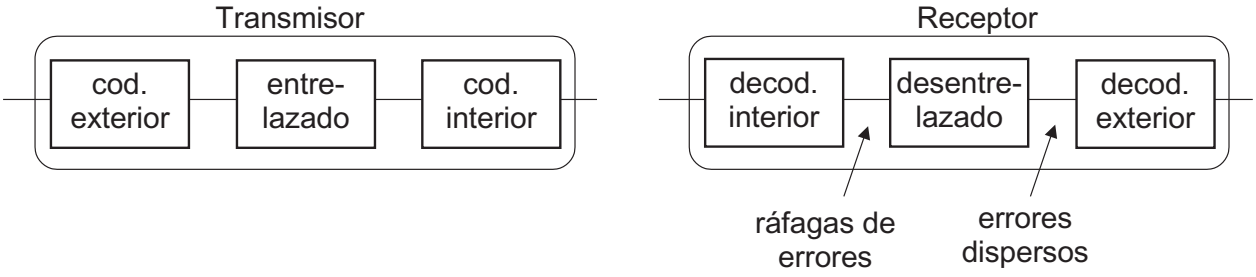
$$p_B = \binom{m}{t+1} p^{t+1}(1-p)^{m-t-1} + \binom{m}{t+2} p^{t+2}(1-p)^{m-2-1} + \Lambda \approx \binom{m}{t+1} p^{t+1}$$

- Probabilidad de error de bit de mensaje p_B
  (caso peor: en bloque no corregido, todos los bits erróneos)

$$\text{Ejemplo:} \quad m = 32 \quad t = 3 \quad p = 10^{-3} \rightarrow p_B \approx \binom{32}{4} p^4 = 3{,}6 \cdot 10^{-8}$$

Codificación de secuencias

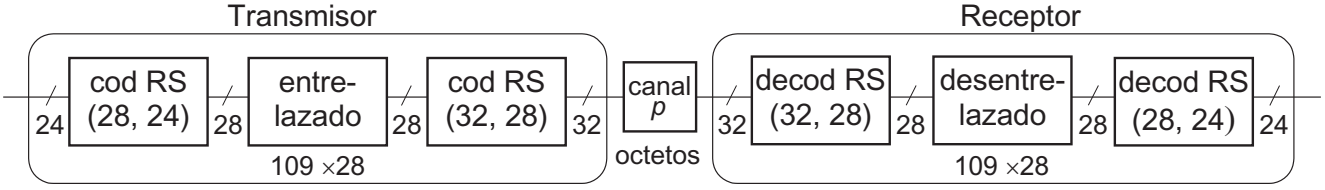
Codificación concatenada con entrelazado de muchos bloques



- **Dos** codificaciones: **interior** y **exterior** + **entrelazado (barajado)**
- El decodificador **interior** elimina los **errores dispersos** pero no las ráfagas
- El **desentrelazador dispersa** los errores (no corregidos) de las ráfagas
- El decodificador **exterior** elimina los errores ya dispersados

Codificación concatenada

Ejemplo



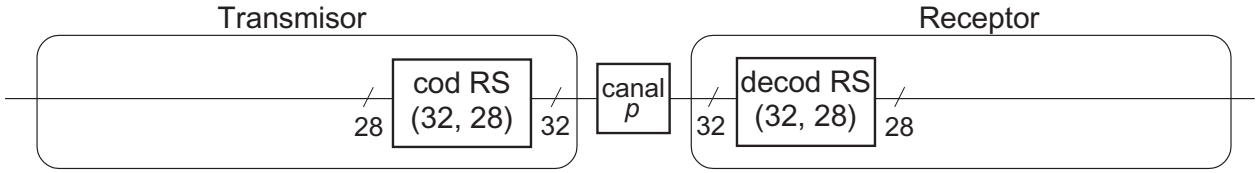
Compact Disk (Philips-Sony - 1982) dos Reed-Solomon (255,251) acortados

codificador interior	RS (32,28)	$m-k=4$	corrige 2 errores en bloque de 32
codificador exterior	RS (28,24)	$m-k=4$	corrige 2 errores en bloque de 28

Entrelazado: 109 bloques de 28

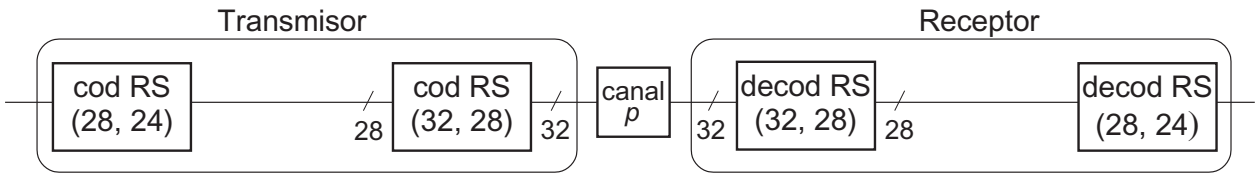
Codificación concatenada

Ejemplo (Compact Disk)



- Tasa de errores de símbolo en el canal: p
- Tasa de errores de símbolo a la salida: ($m = 32$, $t = 2$)

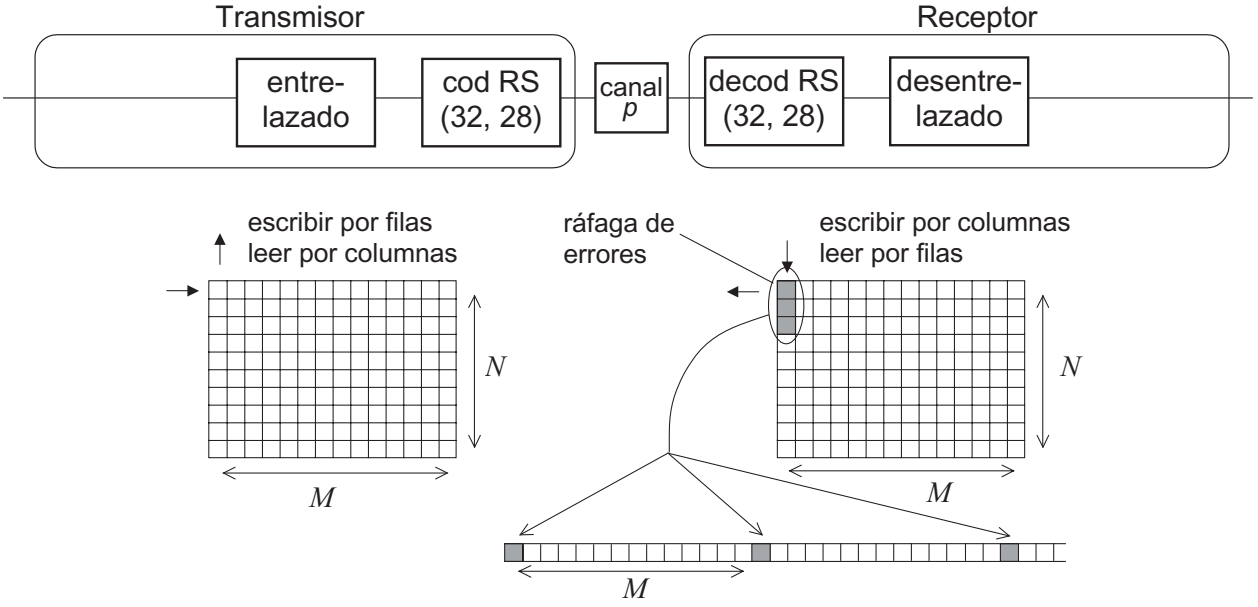
$$p_I \approx \binom{32}{3} p^3 = 5\times(10p)^3$$



- Si se pasan los bloque erróneos al decodificador exterior, tampoco puede corregirlos (tiene 3 o más errores)

Codificación concatenada

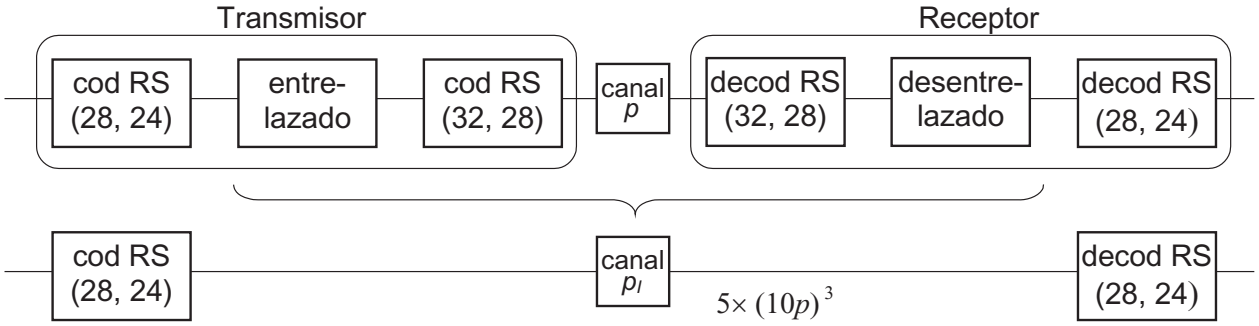
Ejemplo (Compact Disk)



- Desentrelazado dispersa los errores resultando un canal sin memoria

Codificación concatenada

Ejemplo (Compact Disk)



Desentrelazado dispersa los errores

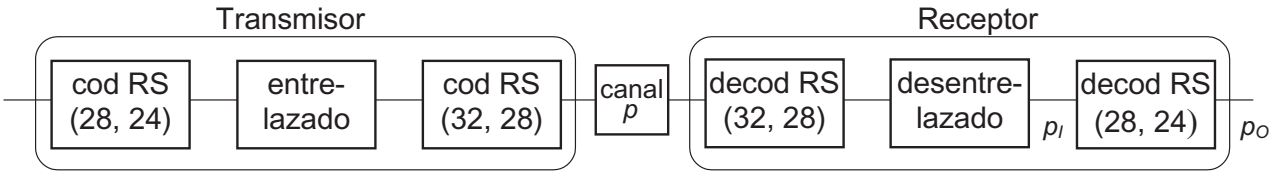
- Resulta un canal sin memoria con $p_I = 5\times (10p)^3$

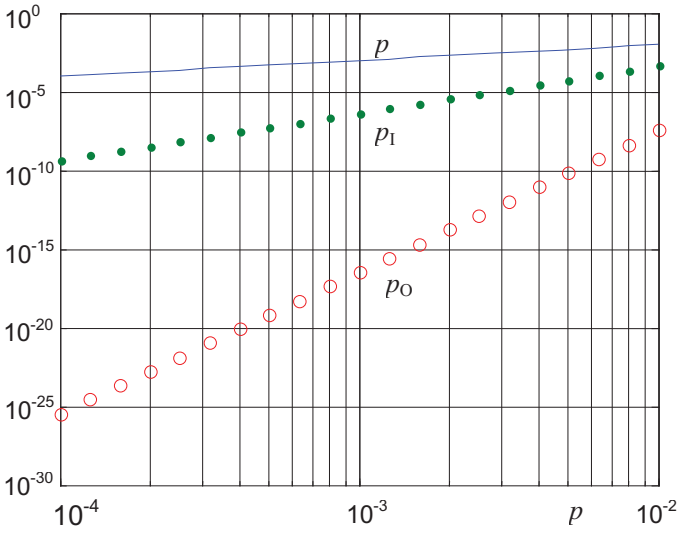
Después del decodificador exterior

- Tasa de errores de símbolo ($m = 28$, $t = 2$): $p_O \approx \binom{28}{3} p_I^3 = 3{,}3\times(10p)^9$

Codificación concatenada

Ejemplo (Compact Disk)





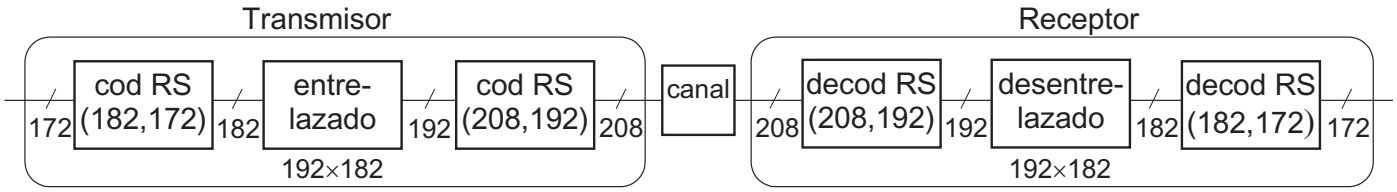
$$p_I \approx 5\times(10p)^3$$

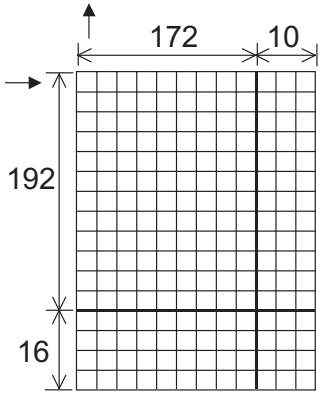
$$p_O \approx 3{,}3\times(10p)^9$$

Codificación concatenada

Ejemplo

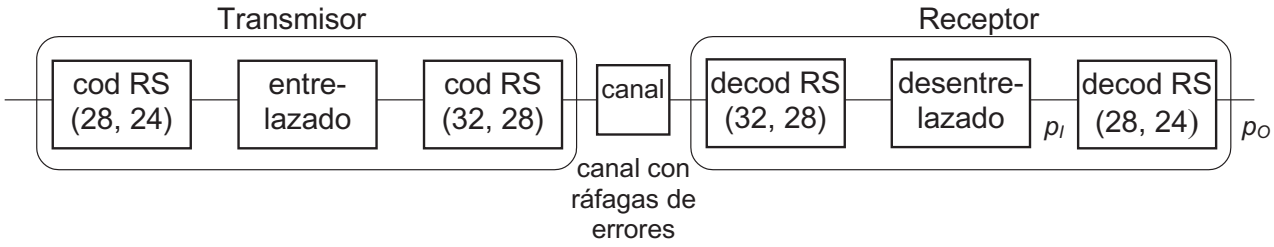
DVD (*DVD consortium* - 1995) dos Reed-Solomon (255,239) acortados / perforados





Codificación concatenada

Ejemplo (Compact Disk)



Protección contra ráfagas de errores

- Ráfagas producen errores agrupados y bloques que no se corrigen
- Decodificador interior elimina errores dispersos pero no las ráfagas
- Entrelazado dispersa los errores de ráfaga
- Decodificador exterior elimina errores ya dispersados

Codificación bloque (2)

Códigos cíclicos binarios

Códigos cíclicos
Codificación/Decodificación
Realización

Códigos cíclicos no binarios

Codificación bloque no binaria
Códigos cíclicos no binarios.
Códigos Reed-Solomon

Codificación concatenada

Codificación de canal

Codificación convolucional

Concepto

Codificación de secuencias

Codificación bloque y convolucional

Codificación convolucional $k = 1$ y $k > 1$

Comportamiento del codificador

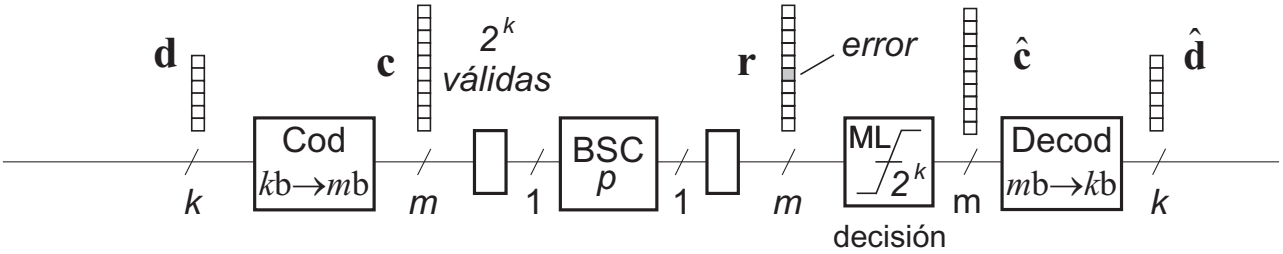
Decodificación

Decodificación con algoritmo de Viterbi

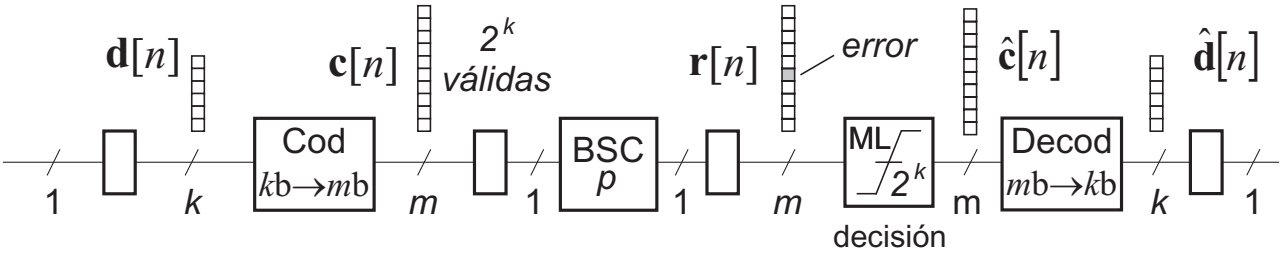
Códigos perforados

Codificación de secuencias

Codificación (m, k) de un vector de k bits



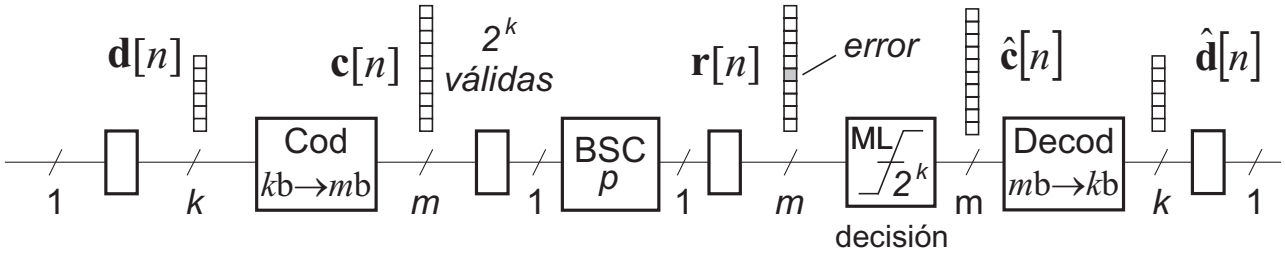
Codificación sucesiva (m, k) de una secuencia de bits (larga o indefinida)



- Secuencia larga: longitud >> k
- El proceso de codificación de un vector se repite cada k nuevos bits de entrada

Codificación de secuencias

Codificación (m, k) de una secuencia de bits (larga o indefinida)

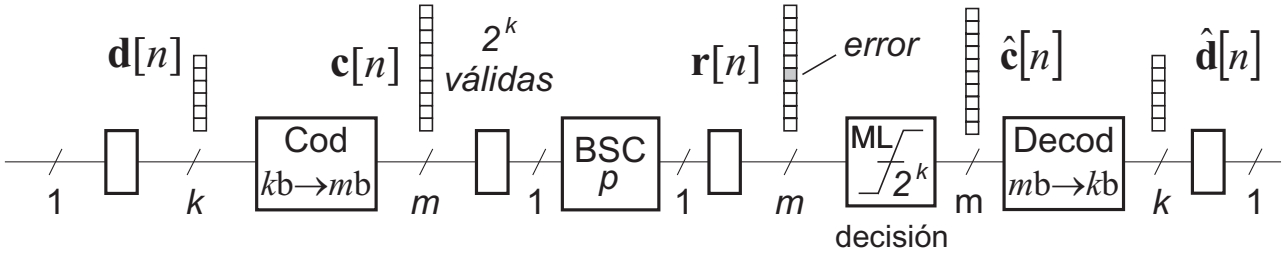


- Cada intervalo de codificación [n] se toman k nuevos bits de entrada
- Se puede usar un codificador/decodificador bloque (Hamming, cíclico, etc.)

  $\mathbf{c}[n]=f(\mathbf{d}[n])$ la palabra código en [n] se calcula con el mensaje en [n]
- El proceso puede ser indefinido
  - para secuencias indefinidas de bits
  - tiene un retardo de k bits (adquisición del mensaje)

Codificación de secuencias

Codificación (m, k) de una secuencia de bits (larga o indefinida)



Codificación sin memoria

- La palabra código en [n] se calcula con el mensaje en [n]

  $\mathbf{c}[n]=f(\mathbf{d}[n])$

Codificación con memoria

- La palabra código en [n] se calcula con el mensaje en [n] y los anteriores

  $\mathbf{c}[n]=f(\mathbf{d}[n], \mathbf{d}[n\text{-}1], \mathbf{d}[n\text{-}2], \mathbf{d}[n\text{-}2],...)$
- Puede ser FIR o IIR

Codificación de canal

Codificación convolucional

Concepto

- Codificación de secuencias
- Codificación bloque y convolucional
- Codificación convolucional $k = 1$ y $k > 1$

Comportamiento del codificador

Decodificación

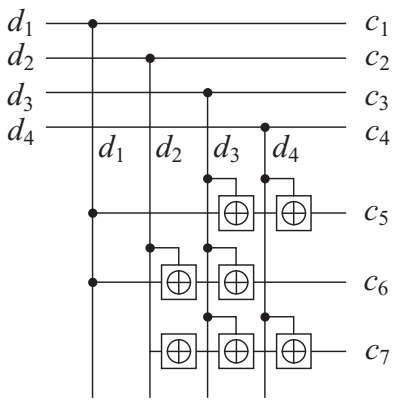
Decodificación con algoritmo de Viterbi

Códigos perforados

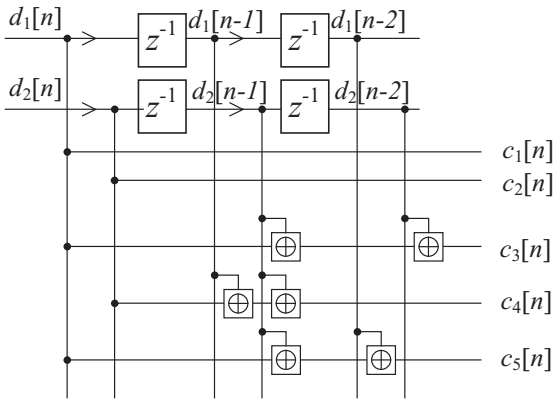
Codificación bloque y convolucional

codificación bloque	sin memoria	• la palabra-código asignada a cada mensaje sólo depende de ese mensaje
codificación convolucional	con memoria	• la palabra-código asignada a cada mensaje depende tambien de mensajes anteriores

Codificación bloque (7,4)



Codificación convolucional (5,2)



Codificación de canal

Codificación convolucional

Concepto

- Codificación de secuencias
- Codificación bloque y convolucional
- Codificación convolucional $k = 1$ y $k > 1$

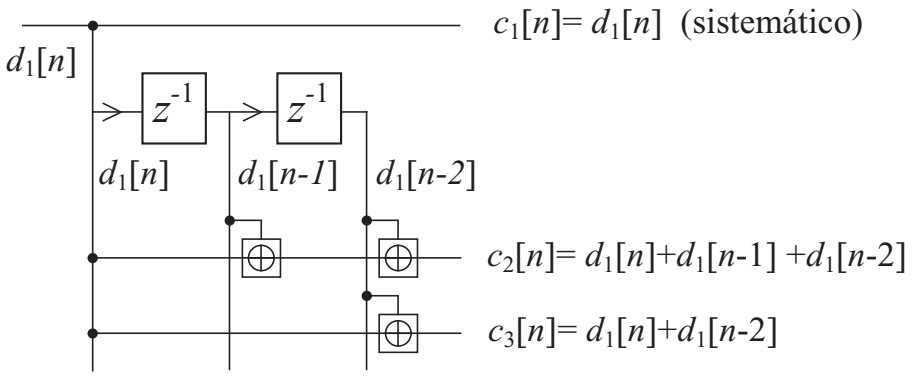
Comportamiento del codificador

Decodificación

Decodificación con algoritmo de Viterbi

Códigos perforados

Codificación convolucional $k = 1$ Ejemplo (3,1)



- Los bits de la palabra código se calculan a partir de los valores actuales y **anteriores**
- En general para $k = 1$ (mensajes de 1 bit $\mathbf{d}[n]= \{d_1[n]\}$):

$$c_i[n]= \sum_{p=0}^{K-1} g_i[p]d_1[n-p] \quad i=1..m$$

- La secuencia $c_i[n]$ es la **convolución** de la secuencia $d_1[n]$ con la "respuesta al impulso" $g_i[p]$

Codificación convolucional

- La secuencia $c_i[n]$ es la **convolución** de la secuencia $d_1[n]$ con la "respuesta al impulso" $g_i[p]$

$$c_i[n] = \sum_{p=0}^{K-1} g_i[p] d_1[n-p] \quad i = 1..m$$

- El cálculo de cada paridad equivale a un **filtro FIR** de **respuesta al impulso** $g_i[p]$ (los coeficientes valen sólo "0" o "1" y las sumas son or-ex)

- Cada paridad $c_i[n]$ se calcula con una respuesta al impulso diferente $g_i[p]$

- La máxima longitud de las **respuestas al impulso** (K) se denomina "**longitud de restricción**" (***constraint length***) (K=3 en el ejemplo anterior)

- Si la entrada es un impulso $d_1[n]=\delta[n]$ las salidas son $g_i[p]$ (= "0" para $n \geq K$)

Codificación convolucional $k > 1$

- Para $k > 1$ se suman las respuestas
  a cada una de las k secuencias de entrada $d_j[n]$ j =1..k:

$k = 1$	$c_i[n] = \sum_{p=0}^{K-1} g_i[p] d_1[n-p] \quad i = 1..m$	donde $g_i[k]$ son "0" ó "1"
$k > 1$	$c_i[n] = \sum_{j=1}^{k} \sum_{p=0}^{K-1} g_{ij}[p] d_j[n-p] \quad i = 1..m$	donde $g_{ij}[k]$ son "0" ó "1"

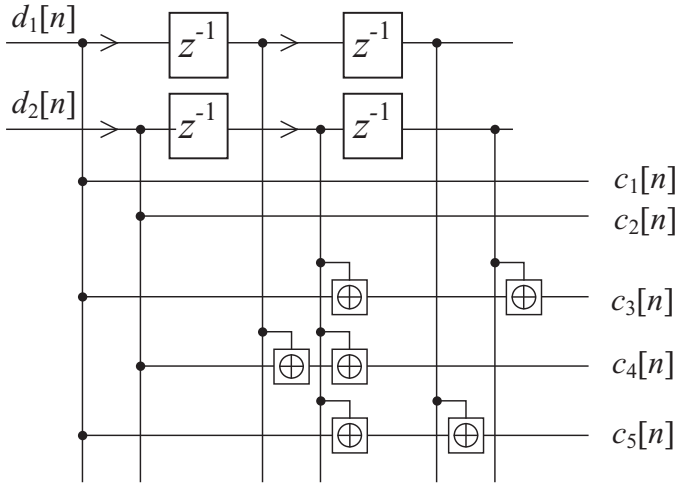
- Si todas las entradas son impulsos $d_j[n] = \delta[n]$ las salidas son

$$c_i = \sum_{j=1}^{k} g_{ij}[n] \qquad (= \text{"0" para } n > K)$$

- La **longitud de restricción** (K) es la máxima de todas las **respuestas al impulso**

Codificación convolucional $k > 1$

Ejemplo (5,2) K=3



$$c_1[n] = d_1[n]$$

$$c_2[n] = d_2[n]$$

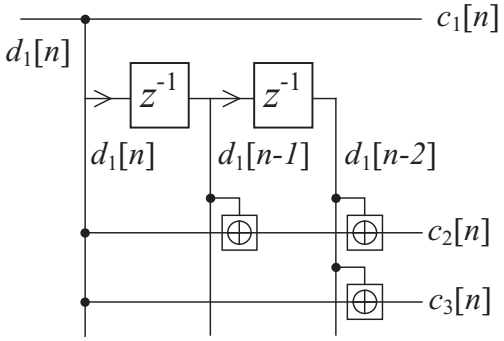
$$c_3[n] = d_1[n] + d_2[n-1] + d_2[n-2]$$

$$c_4[n] = d_1[n-1] + d_2[n] + d_2[n-1]$$

$$c_5[n] = d_1[n] + d_1[n-2] + d_2[n-1]$$

Codificación convolucional

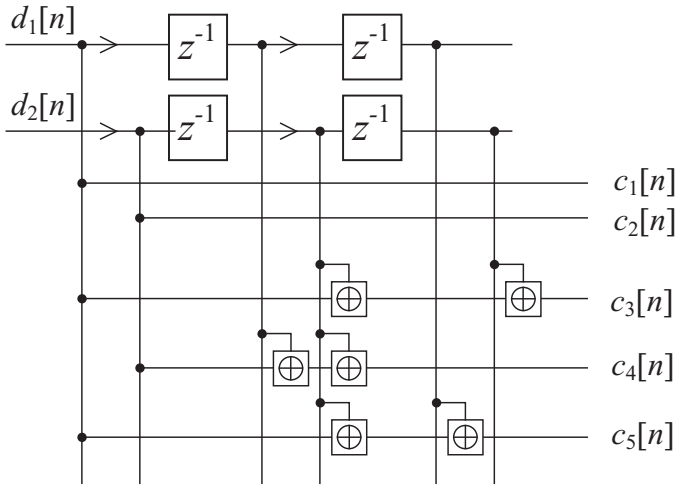
$k = 1$: (3,1) K=3



$$c_2[n] = d_1[n] + d_1[n-1] + d_1[n-2]$$

$$c_3[n] = d_1[n] + d_1[n-2]$$

$k > 1$: (5,2) K=3



$$c_3[n] = d_1[n] + d_2[n-1] + d_2[n-2]$$

$$c_4[n] = d_1[n-1] + d_2[n] + d_2[n-1]$$

$$c_5[n] = d_1[n] + d_1[n-2] + d_2[n-1]$$

Codificación de canal

Codificación convolucional

- Concepto
- Comportamiento del codificador
  - Representación con diagrama de estados y rejilla
  - Representación de la evolución del codificador
- Decodificación
- Decodificación con algoritmo de Viterbi
- Códigos perforados

Codificación de canal

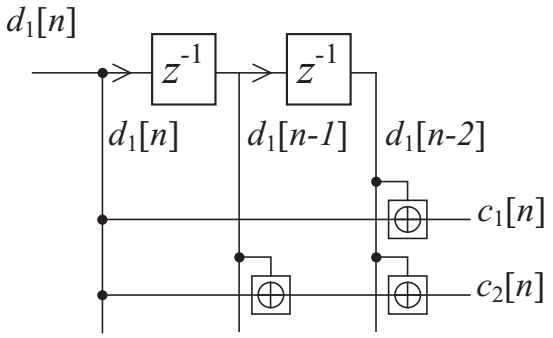
Codificación convolucional

- Concepto
- Comportamiento del codificador
  - Representación con diagrama de estados y rejilla
  - Representación de la evolución del codificador
- Decodificación
- Decodificación con algoritmo de Viterbi
- Códigos perforados

Codificación convolucional

Representación del comportamiento: diagramas de estado

Ejemplo codificador no sistemático (2,1) de tasa 1/2 y K=3



$c_1[n] = d_1[n]+d_1[n\text{-}2]$

$c_2[n]= d_1[n]+d_1[n\text{-}1] +d_1[n\text{-}2]$

las salidas dependen de la entrada y del estado del sistema:

$d_1[n\text{-}1]$

$d_1[n\text{-}2]$

Estado	$d_1[n\text{-}1]$	$d_1[n\text{-}2]$
E_0	0	0
E_1	1	0
E_2	0	1
E_3	1	1

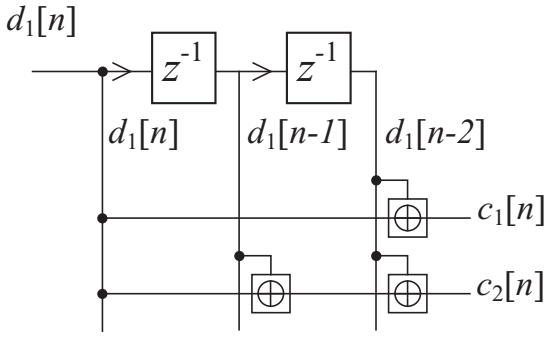
	c_1	c_2
E_0	d_1	d_1
E_1	d_1	$\overline{d_1}$
E_2	$\overline{d_1}$	$\overline{d_1}$
E_3	$\overline{d_1}$	d_1

Salidas $\{c_1\, c_2\}[n]$	$d_1[n]$= 0	$d_1[n]$= 1
E_0	00	11
E_1	01	10
E_2	11	00
E_3	10	01

Codificación convolucional

Representación del comportamiento: diagramas de estado

Ejemplo codificador no sistemático (2,1) de tasa 1/2 y K=3



$c_1[n] = d_1[n]+d_1[n\text{-}2]$

$c_2[n]= d_1[n]+d_1[n\text{-}1] +d_1[n\text{-}2]$

las salidas dependen de la entrada y del estado del sistema:

$d_1[n\text{-}1]$

$d_1[n\text{-}2]$

Estado	$d_1[n\text{-}1]$	$d_1[n\text{-}2]$
E_0	0	0
E_1	1	0
E_2	0	1
E_3	1	1

E[n+1]	$d_1[n]$= 0	$d_1[n]$= 1
E_0	E_0	E_1
E_1	E_2	E_3
E_2	E_0	E_1
E_3	E_2	E_3

Salidas $\{c_1\, c_2\}[n]$	$d_1[n]$= 0	$d_1[n]$= 1
E_0	00	11
E_1	01	10
E_2	11	00
E_3	10	01

Codificación convolucional

Ejemplo codificador no sistemático (2,1) de tasa 1/2 y K=3

Tabla 1

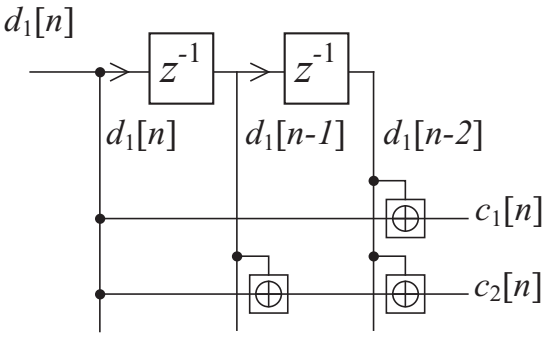
Estado	$d_1[n\text{-}1]$	$d_1[n\text{-}2]$
E_0	0	0
E_1	1	0
E_2	0	1
E_3	1	1

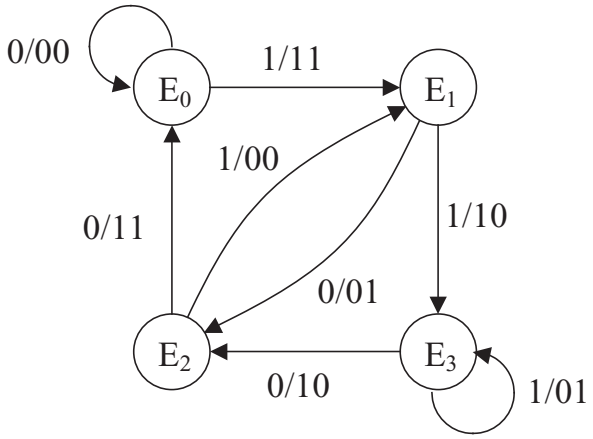
Tabla 2 E[n+1]

	$d_1[n]$= 0	$d_1[n]$= 1
E_0	E_0	E_1
E_1	E_2	E_3
E_2	E_0	E_1
E_3	E_2	E_3

Tabla 3 Salidas $\{c_1\, c_2\}[n]$

	$d_1[n]$= 0	$d_1[n]$= 1
E_0	00	11
E_1	01	10
E_2	11	00
E_3	10	01

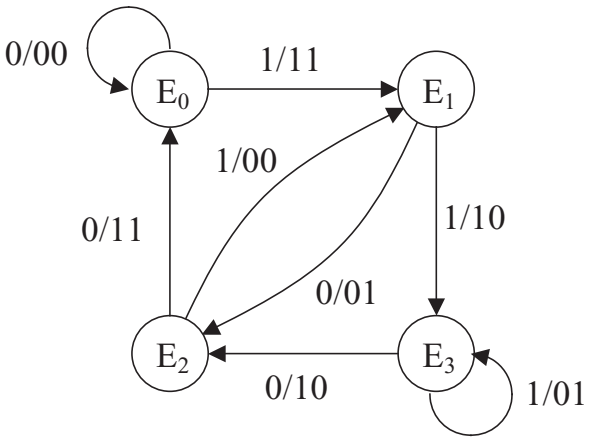


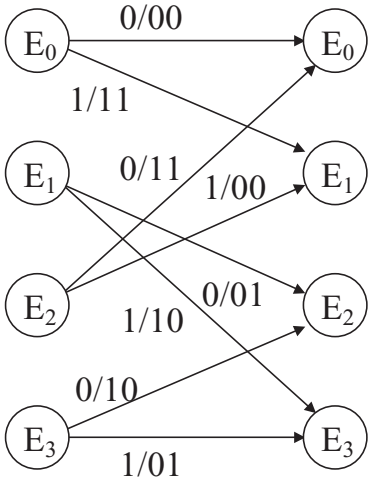


Codificación convolucional

Representación del comportamiento: diagramas de rejilla (*trellis*)

Ejemplo codificador no sistemático (2,1) de tasa 1/2 y K=3





- Ambos indican salidas $c_1[n]$ $c_2[n]$ y estado siguiente E[n+1] a partir de estado actual E[n] y entrada $d_1[n]$
- Diagrama de rejilla permite representar la <u>evolución temporal</u>

Codificación convolucional

Tabla 2 E[n+1]

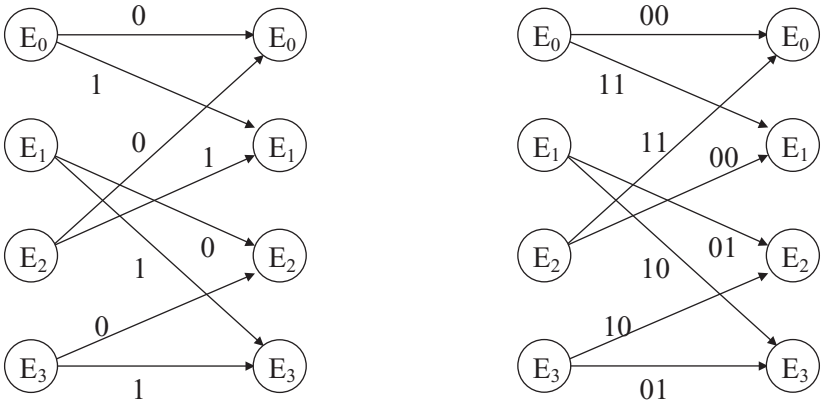
	$d_1[n]$= 0	$d_1[n]$= 1
E_0	E_0	E_1
E_1	E_2	E_3
E_2	E_0	E_1
E_3	E_2	E_3

Tabla 3 Salidas $\{c_1\, c_2\}[n]$

	$d_1[n]$= 0	$d_1[n]$= 1
E_0	00	11
E_1	01	10
E_2	11	00
E_3	10	01

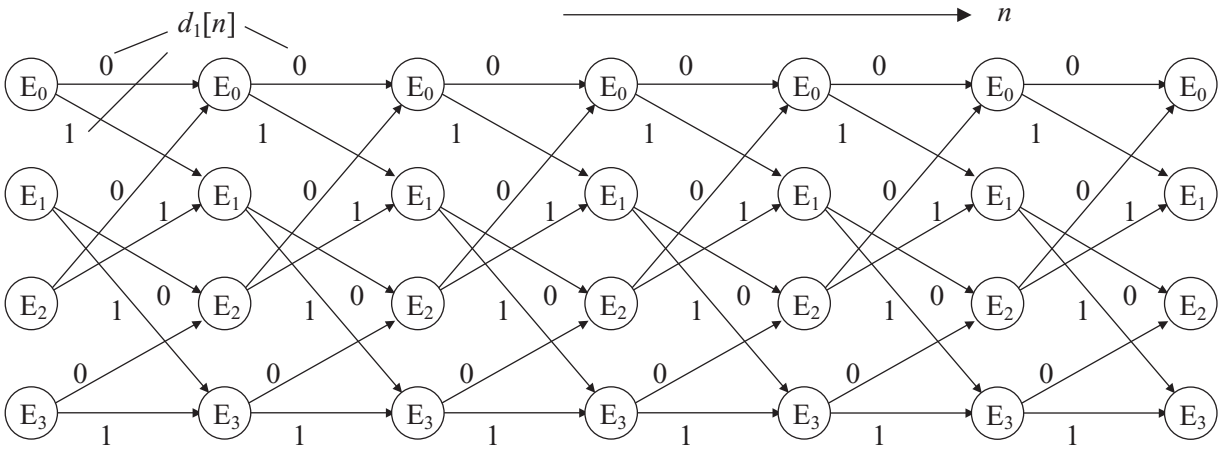
entradas determinan la transición entre estados

transición entre estados determina las **salidas**



Codificación convolucional

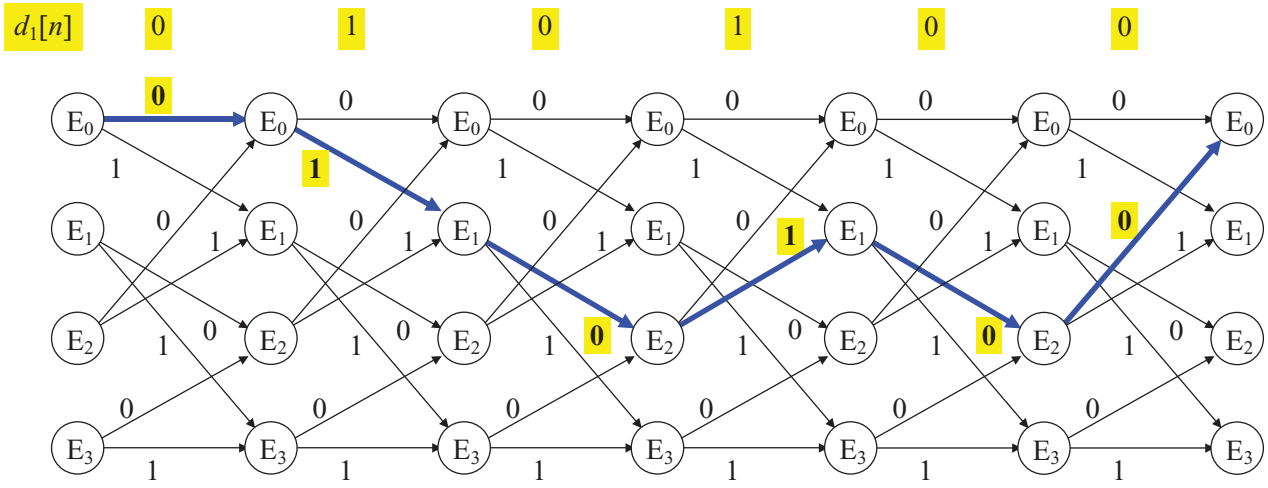
Diagrama de rejilla (grafo orientado)



- Evolución temporal del estado del codificador según la secuencia de bits de entrada
- **Grafo orientado** con:
  - **nodos** (estados posibles en cada instante [n] y
  - **ramas orientadas** (transición a estado siguiente según valor de la entrada)

Codificación convolucional

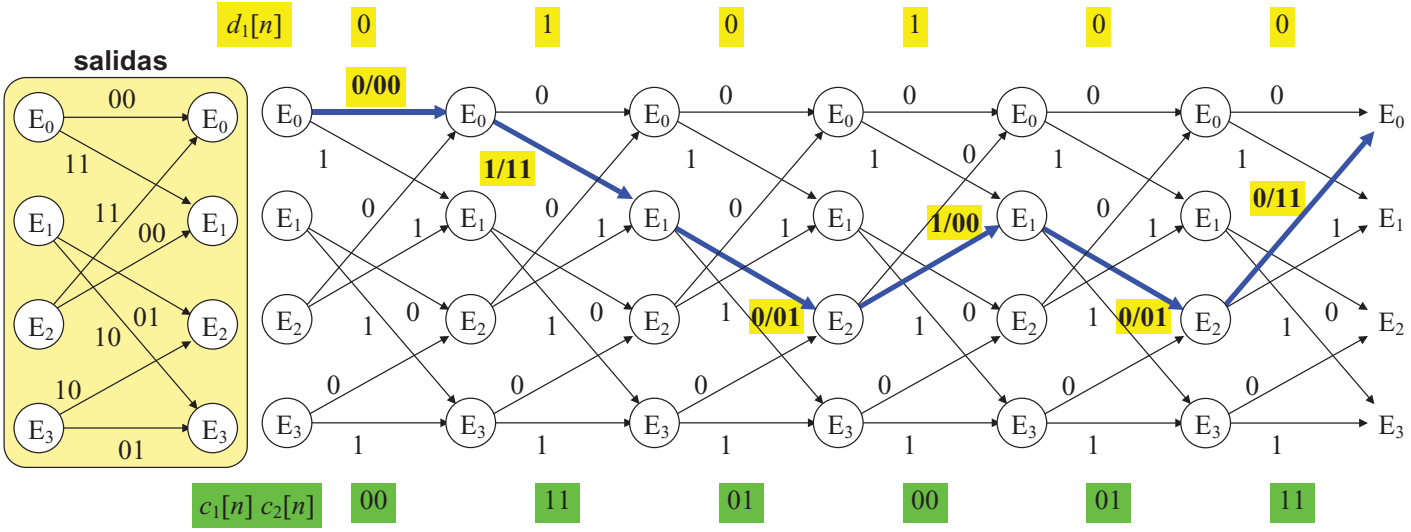
Evolución temporal. Ejemplo



- la evolución de los estados se representa con un **camino** a través del grafo

Codificación convolucional

Evolución temporal. Ejemplo

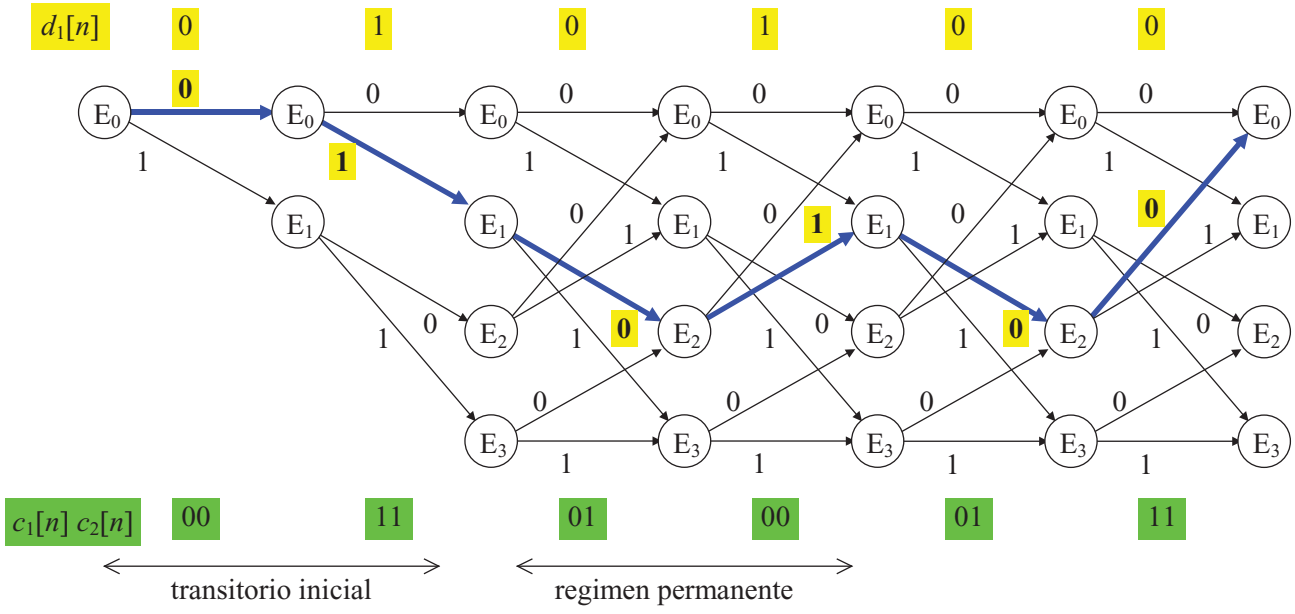


- La evolución de los **estados** se representa con un **camino** a través del grafo
- La **salida** $c_1[n]\ c_2[n]$ se puede extraer con las **asociadas a cada rama**

Codificación convolucional

Evolución temporal. Transitorio inicial

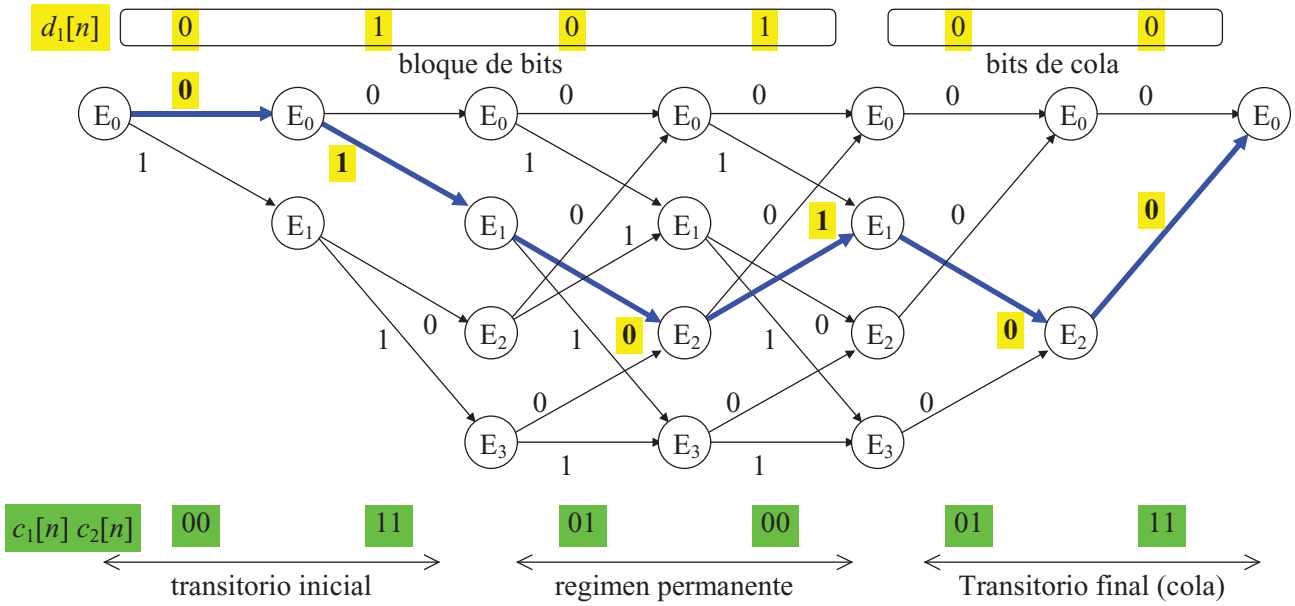
- Si se empieza en E_0 algunas ramas no son posibles (K-1 primeras etapas)
- Después del transitorio inicial → **régimen permanente**: todas las ramas son posibles



Codificación convolucional

Evolución temporal. Transitorio final

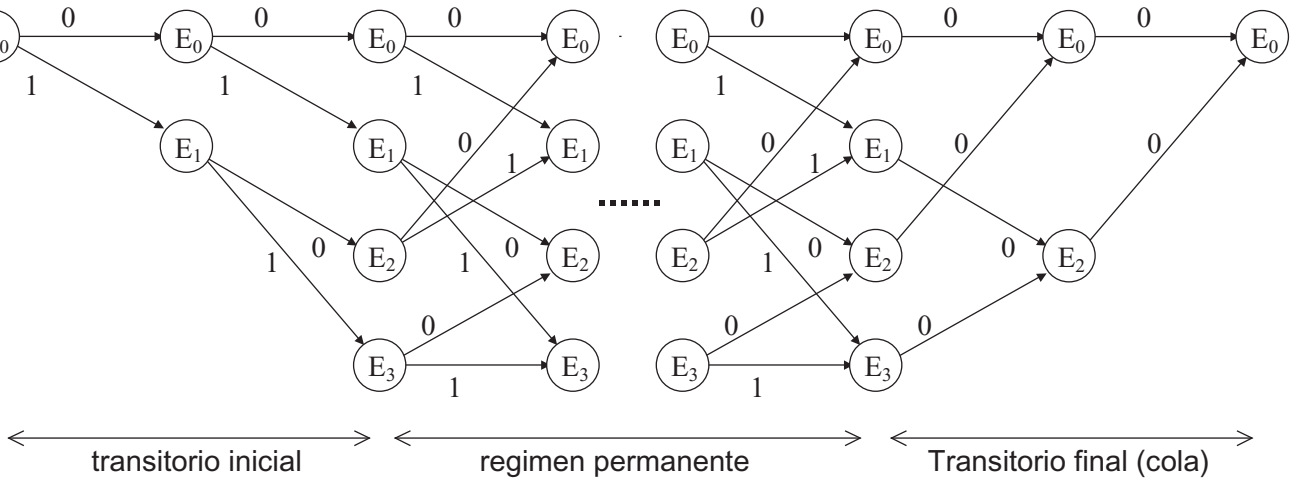
- Si se fuerza a que termine en E_0 algunas ramas no son posibles (últimas K-1 etapas)
- Suele hacerse transmitiendo al final K-1 "0s" (***bits de cola***):



Codificación convolucional

Evolución temporal

- Cuando se transmiten **bloques de bits** se terminan al final en el estado E_0



- También se pueden transmitir **secuencias indefinidas** sin terminación

Codificación de canal

Codificación convolucional

Concepto

Comportamiento del codificador

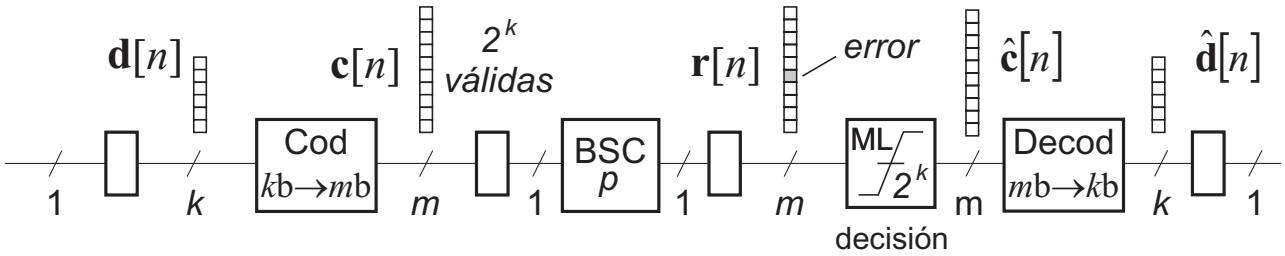
Decodificación

- Decodificación de secuencias
- Distancia entre secuencias
- Decodificación ML
- El problema del camino más corto

Algoritmo de Viterbi

Códigos perforados

Codificación de secuencias



Codificación sin memoria

- La palabra código en [n] se calcula con el mensaje en [n]

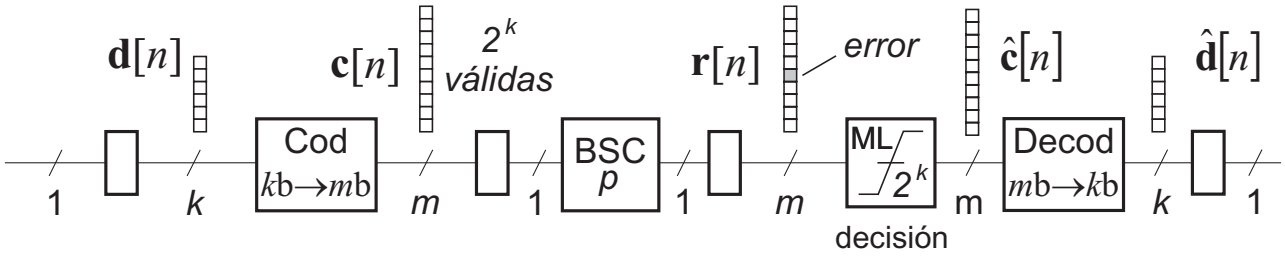
$$\mathbf{c}[n]=f(\mathbf{d}[n])$$

Codificación con memoria

- La palabra código en [n] se calcula con el mensaje en [n] y los anteriores

$$\mathbf{c}[n]=f(\mathbf{d}[n], \mathbf{d}[n\text{-}1], \mathbf{d}[n\text{-}2], \mathbf{d}[n\text{-}2],...)$$

Codificación de secuencias. Decodificación



Decodificación (codificación sin memoria)

- Se hace independientemente para cada intervalo de mensaje [n]

Decodificación (codificación con memoria)

- Se hace teniendo en cuenta las palabra-código en [n] y anteriores
- Si es transmisión indefinida se utilizan varias palabras-código anteriores
- Si se transmite un bloque de bits a veces se procesa toda la secuencia conjuntamente

Codificación de canal

Codificación convolucional

Concepto
Comportamiento del codificador

Decodificación
- Decodificación de secuencias
- Distancia entre secuencias
- Decodificación ML
- El problema del camino más corto

Algoritmo de Viterbi
Códigos perforados

Codificación convolucional

Distancia entre secuencias

- Al transmitir un **bloque de N bits**...
  no son posibles todas las secuencias binarias a la salida del codificador (debido a su memoria) (sólo hay 2^N secuencias posibles)

- El receptor puede buscar, de las 2^N secuencias posibles, ...
  ... la **secuencia más parecida** (ML) a la recibida

- La **capacidad de corrección** de errores depende de la distancia de Hamming entre las secuencias posibles (número total de bits diferentes)

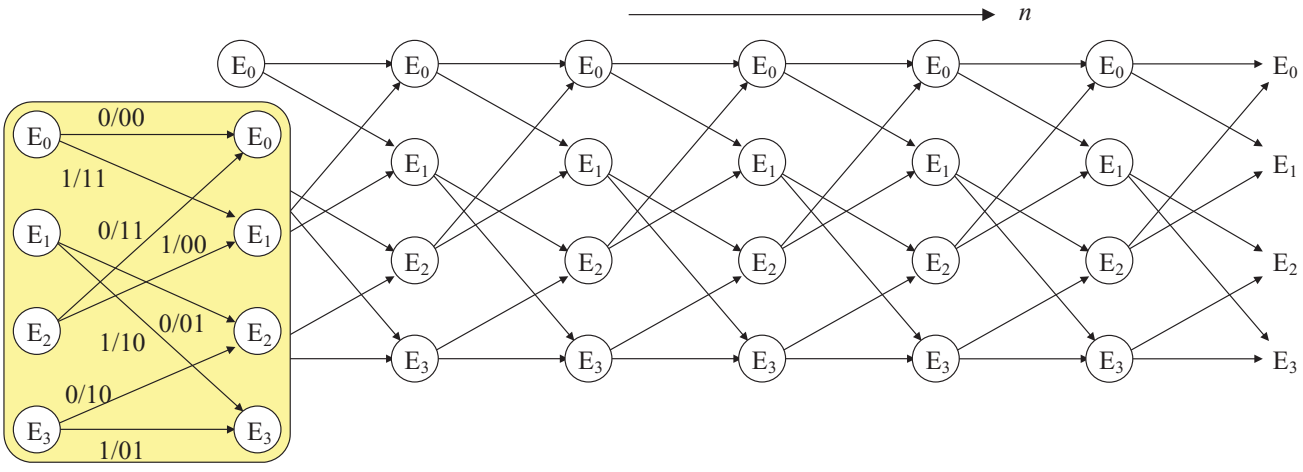
- ¿Cuáles son las secuencias posibles?

- ¿Cuáles son las **distancias de Hamming** entre las secuencias posibles?

Codificación convolucional

Distancia entre secuencias

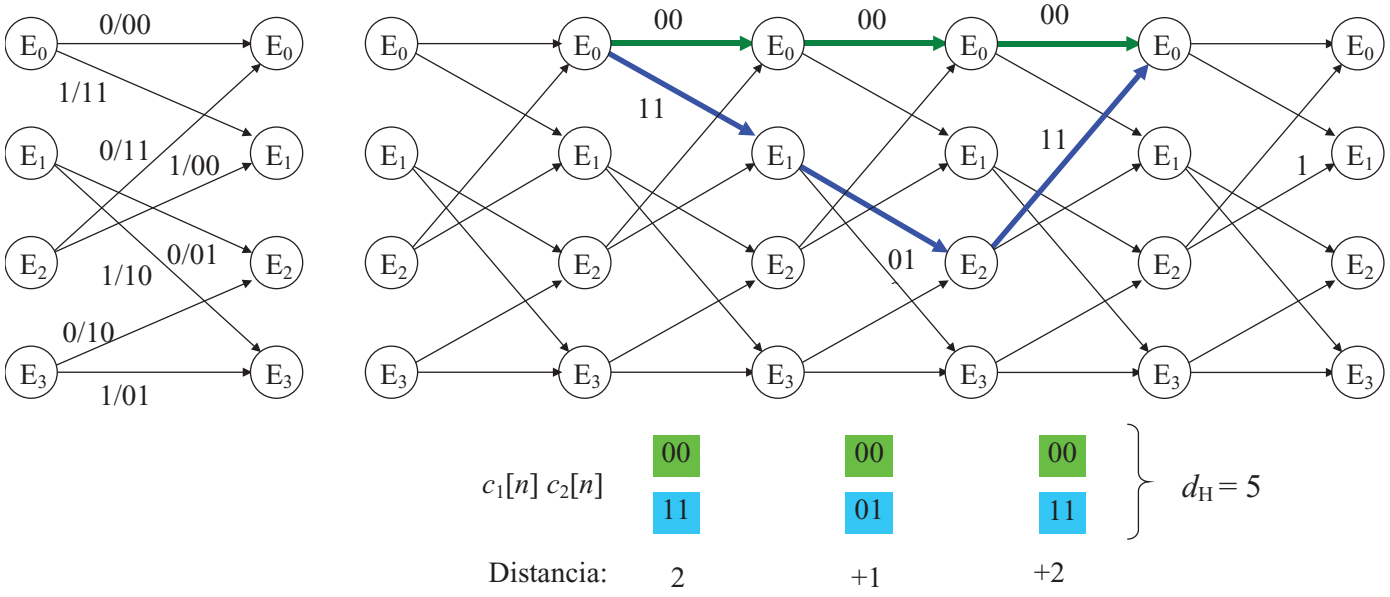
- ¿Cuáles son las secuencias posibles?
  Las que corresponden a caminos en el grafo



- ¿Cuáles son las **distancias de Hamming** entre las secuencias posibles?

Codificación convolucional

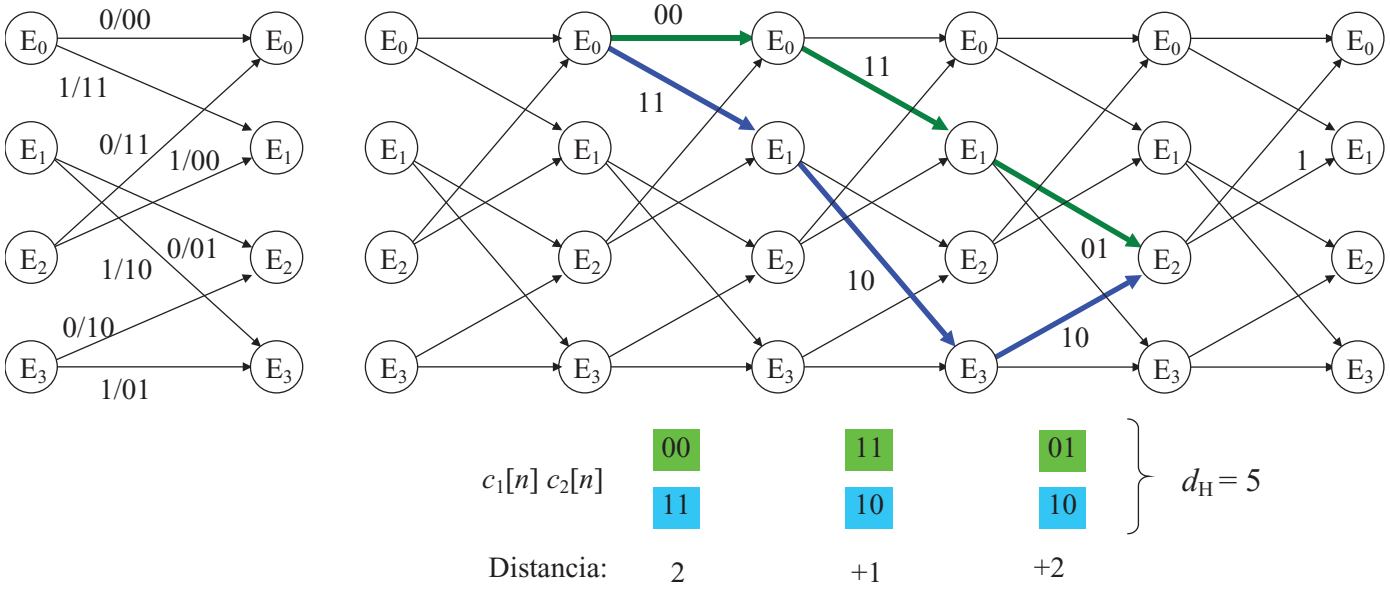
Distancia entre secuencias. Ejemplo 1 $d_1[n]$ = [.. **0** 0 0 ..] y [.. **1** 0 0 ..]



- Un sólo bit de diferencia en entrada provoca $d_H = 5$ en secuencia codificada

Codificación convolucional

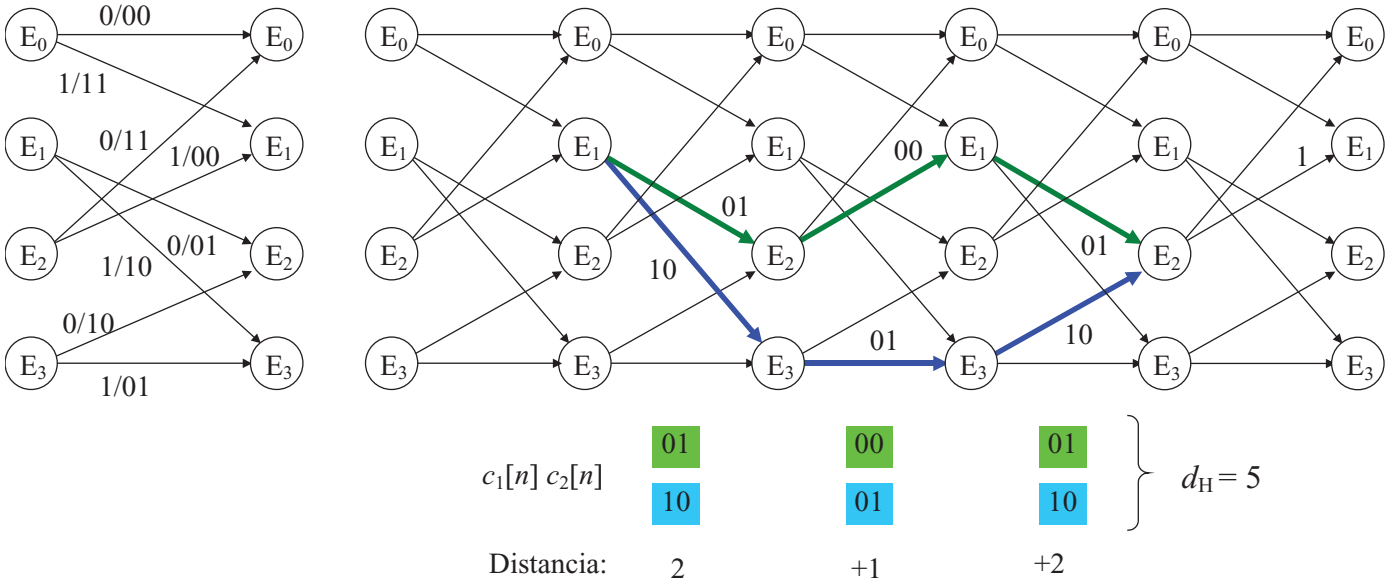
Distancia entre secuencias. Ejemplo 2 $d_1[n]$ = [.. **0** 1 0 ..] y [.. **1** 1 0 ..]



- Un sólo bit de diferencia en entrada provoca $d_H = 5$ en secuencia codificada

Codificación convolucional

Distancia entre secuencias. Ejemplo 3 $d_1[n]$ = [.. **0** 1 0 ..] y [.. **1** 1 0 ..]



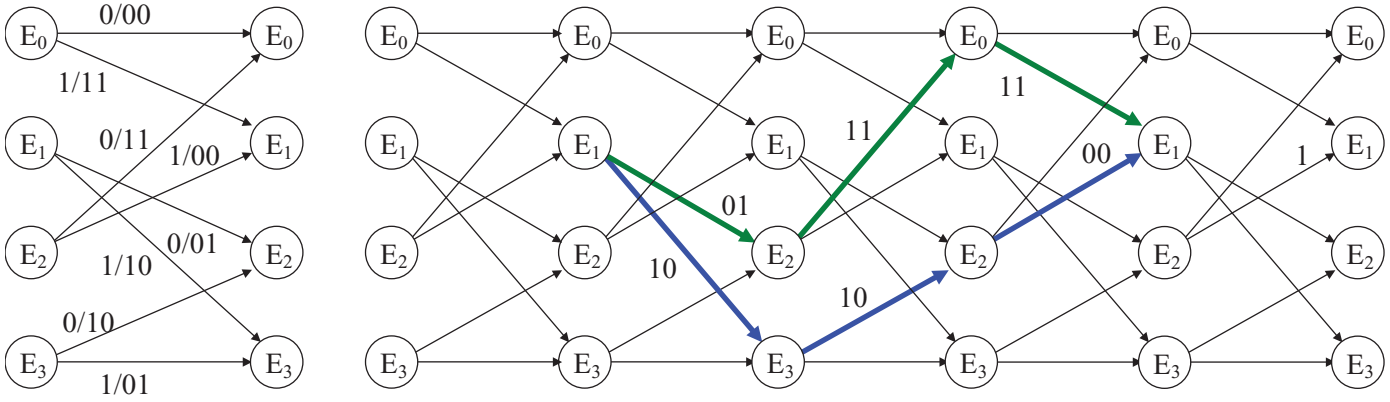
$c_1[n]$ $c_2[n]$	01	00	01
	10	01	10
Distancia:	2	+1	+2

$d_H = 5$

- Un sólo bit de diferencia en entrada provoca $d_H = 5$ en secuencia codificada

Codificación convolucional

Distancia entre secuencias. Ejemplo 4 $d_1[n]$ = [.. **0** 0 1 ..] y [.. **1** 0 1 ..]



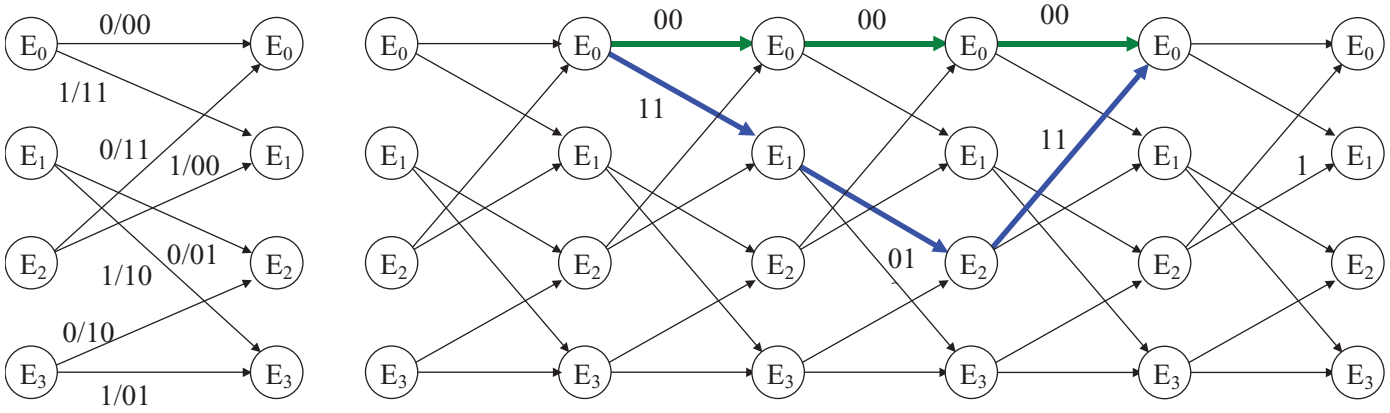
$c_1[n]$ $c_2[n]$	01	11	11
	10	10	00
Distancia:	2	+1	+2

$d_H = 5$

- Un sólo bit de diferencia en entrada provoca $d_H = 5$ en secuencia codificada

Codificación convolucional

Distancia entre secuencias. Ejemplo 1 $d_1[n]$ = [.. **0** 0 0 ..] y [.. **1** 0 0 ..]



- Las dos ramas que salen de, o llegan a, un estado tienen distancia 2
- Cuando se sale de un estado no se puede volver inmediatamente
- Habrá distancia 2 (al salir) +2 (al regresar) +1 en la etapa intermedia: min 5

Codificación de canal

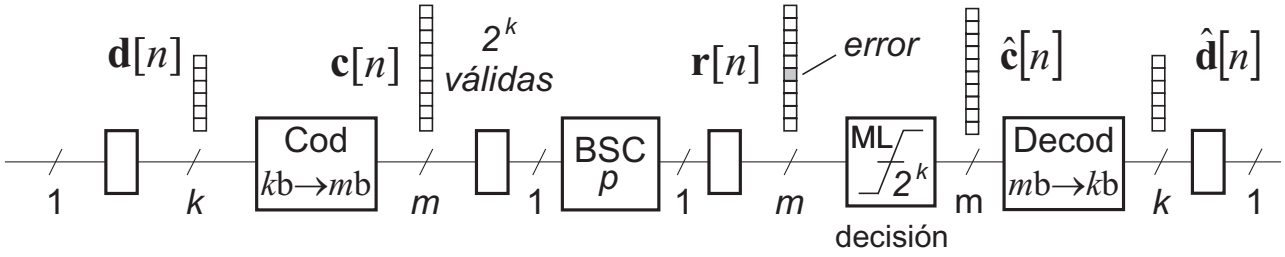
Codificación convolucional

Concepto
Comportamiento

Decodificación
- Decodificación de secuencias
- Distancia entre secuencias
- Decodificación ML
- El problema del camino más corto

Algoritmo de Viterbi
Códigos perforados

Codificación convolucional. Decodificación ML



Ejemplo (2,1) K=3

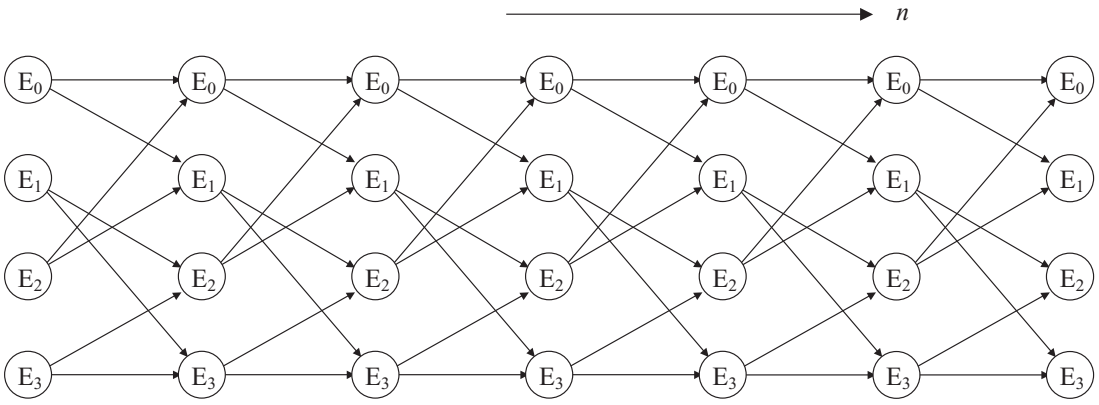
- Se ha recibido la secuencia $\mathbf{r}[n] = \{r_1, r_2\}[n]$ n=0..N ($2N$ bits)
- De las posibles secuencias transmitidas válidas $\mathbf{c}[n] = \{c_1, c_2\}[n]$
  ¿cuál es la más parecida $\hat{\mathbf{c}}[n]$?
  ¿a qué mensaje $\hat{\mathbf{d}}[n]$ corresponde?

Solución ML:

- Medir la **distancia** de Hamming ...
  ... de la secuencia **recibida** a cada una de las (2^N) **secuencias posibles** y
- Seleccionar **la más cercana**

Codificación convolucional. Decodificación ML

Ejemplo (2,1) K=3



- Identificar todos los **posibles caminos** en el diagrama de rejilla
  - (corresponden a todas las **posibles secuencias** de salida del codificador)
- Para cada **camino posible** obtener las **distancia** entre
  - su correspondiente secuencia de salida y
  - la secuencia recibida $\mathbf{r}[n]$:

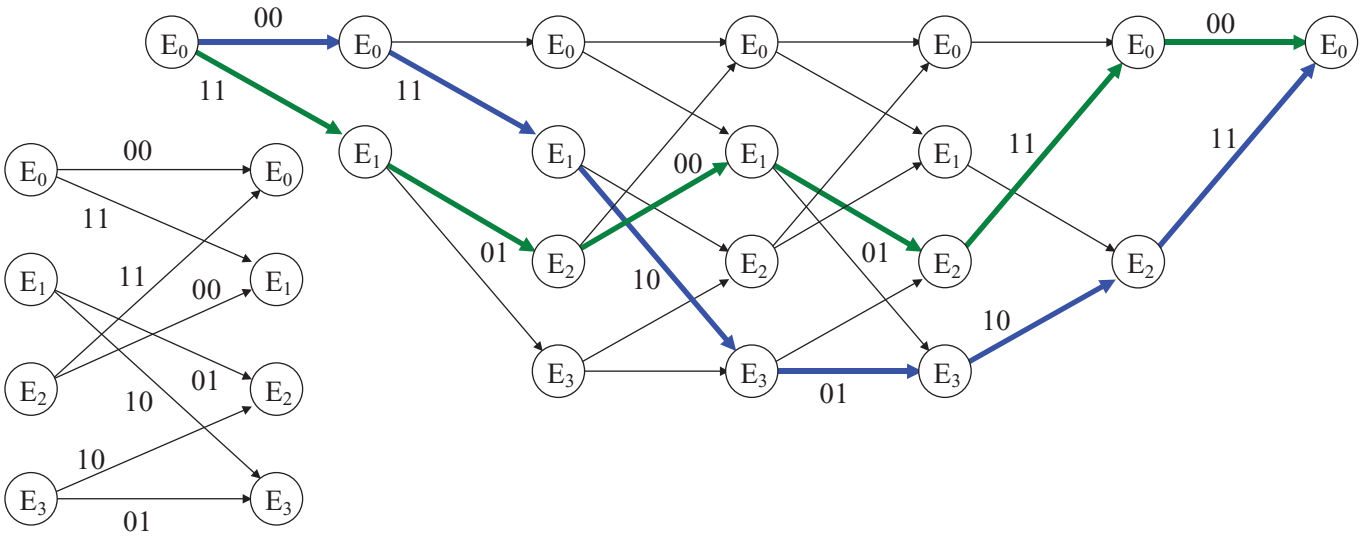
Codificación convolucional. Decodificación ML

Ejemplo

- El mensaje 0 1 0 1 0 0
  se codifica como: 00 11 01 00 01 11
  pero se ha recibido: 00 11 01 0**1** 01 **0**1 (2 errores)

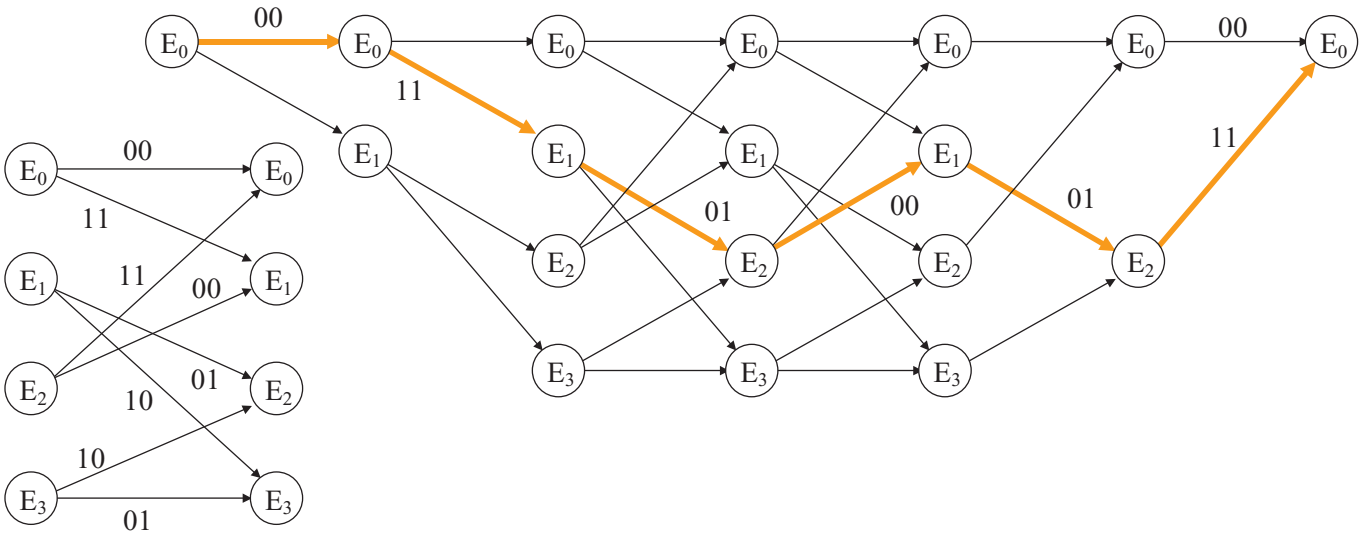
- Ejemplo de medida de distancia de posibles caminos

Codificación convolucional. Decodificación ML. Ejemplo



recibida	00	11	01	**0**1	01	**0**1	suma
prueba 1	00 (0)	11 (0)	10 (2)	01 (0)	10 (2)	11 (1)	d_H=5
prueba 2	11 (2)	01 (1)	00 (1)	01 (0)	11 (1)	00 (1)	d_H=6

Codificación convolucional. Decodificación ML. Ejemplo



recibida	00	11	01	01	01	01	
prueba 1	00 (0)	11 (0)	10 (2)	01 (0)	10 (2)	11 (1)	d_H=5
prueba 2	11 (2)	01 (1)	00 (1)	01 (0)	11 (1)	00 (1)	d_H=6
prueba 3 (enviada)	00 (0)	11 (0)	01 (0)	00 (1)	01 (0)	11 (1)	d_H=2

Codificación convolucional. Decodificación ML

Ejemplo

- El mensaje 0 1 0 1 0 0
  se codifica como: 00 11 01 00 01 11
  pero se ha recibido: 00 11 01 0**1** 01 **0**1 (2 errores)

- Ejemplo de medida de distancia de posibles caminos
  - las secuencias azul y verde están a distancias 5 y 6 respecto a la recibida
  - Las secuencia verdadera está a distancia 2 (los dos errores producidos)
  - Se puede comprobar que todas las demás secuencias están a mayor distancia

Problema:

- En bloques largos hay **muchas** posibles secuencias y distancias a calcular

Codificación de canal

Codificación convolucional

Concepto

Comportamiento

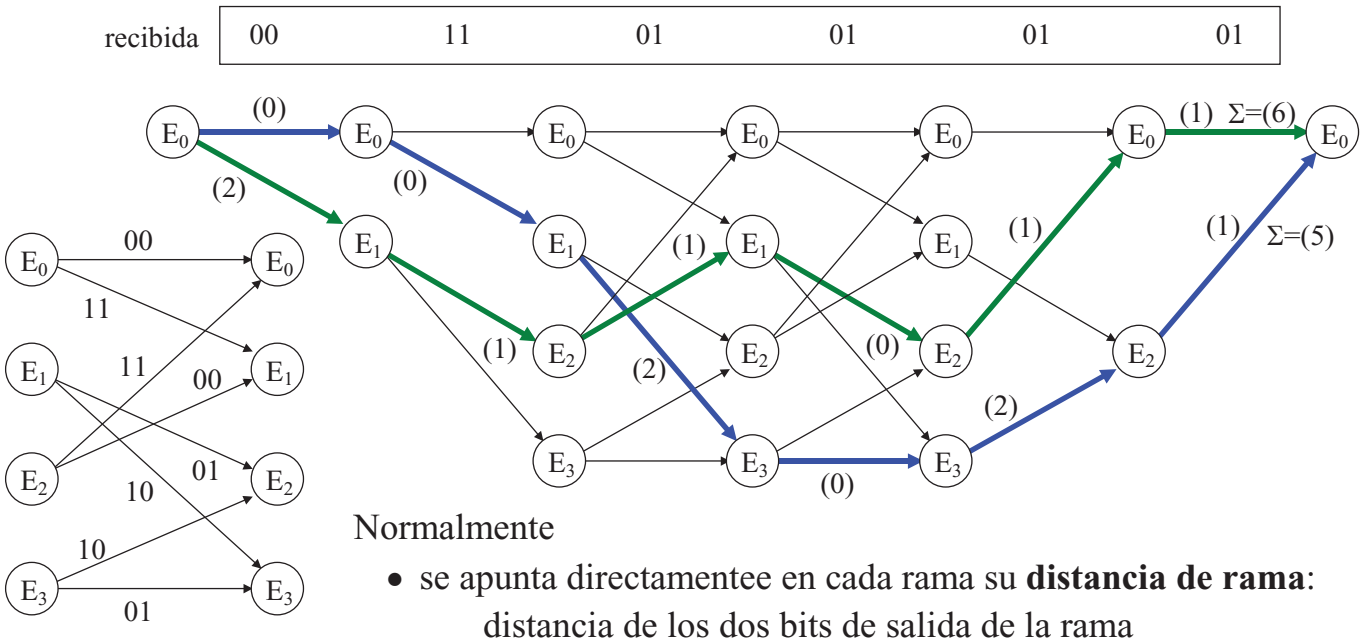
Decodificación

- Decodificación de secuencias
- Distancia entre secuencias
- Decodificación ML
- El problema del camino más corto

Algoritmo de Viterbi

Códigos perforados

Codificación convolucional. Decodificación ML

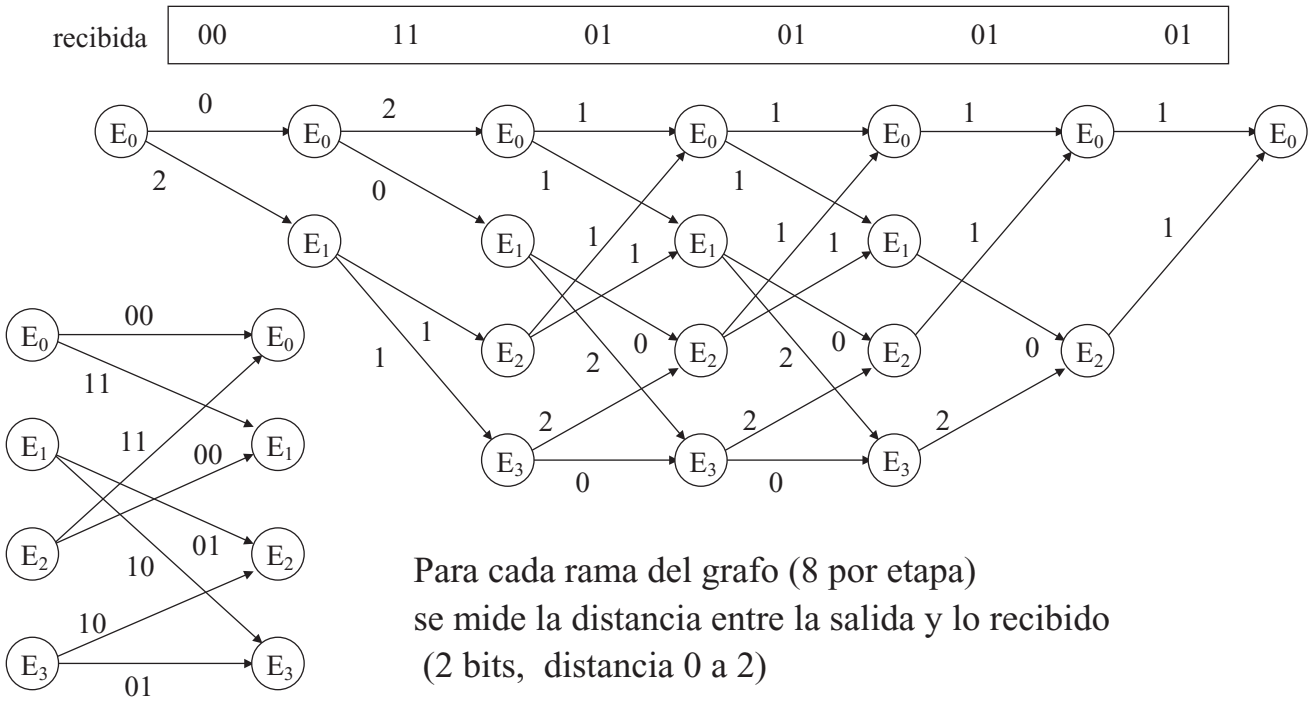


Normalmente

- se apunta directamentee en cada rama su **distancia de rama**:
  distancia de los dos bits de salida de la rama
  a los dos bits recibidos
- se suman las distancias de rama de **cada camino**
- se selecciona el camino más corto

Codificación convolucional. Decodificación ML

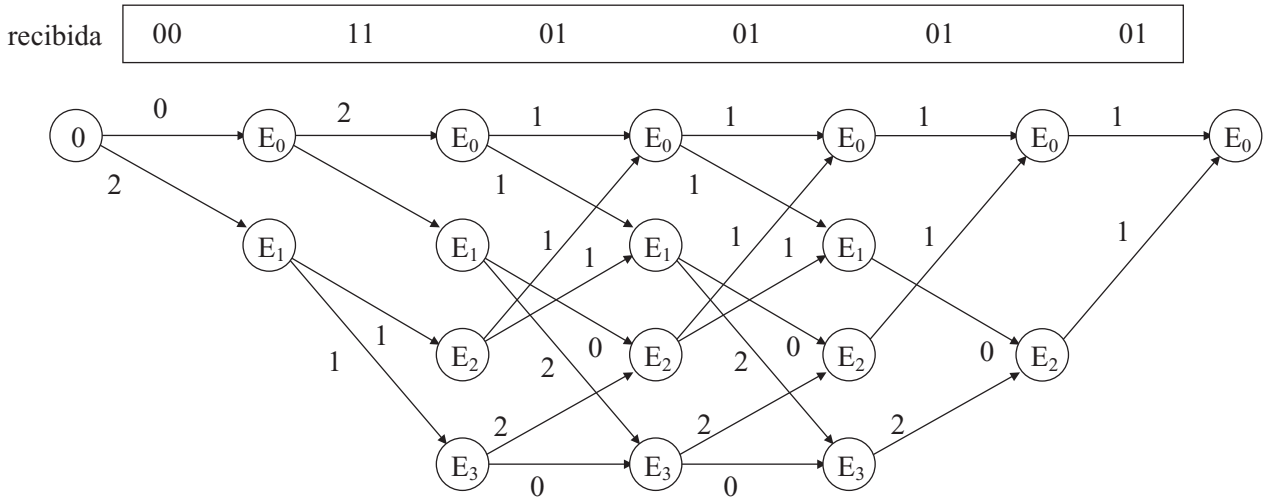
Cálculo de las medidas de rama



Para cada rama del grafo (8 por etapa)
se mide la distancia entre la salida y lo recibido
(2 bits, distancia 0 a 2)

Codificación convolucional. Decodificación ML

Busqueda del camino más corto



- Para todos los posibles caminos del grafo ¿cuál es el más corto?
- ¿Cuántos posibles caminos hay?
- ¿Hay que evaluarlos todos?

Codificación de canal

Codificación convolucional

- Concepto
- Comportamiento
- Decodificación
- Algoritmo de Viterbi
  - Fundamento del algoritmo
  - Ejemplo
  - Decodificación de secuencias indefinidas
- Códigos perforados

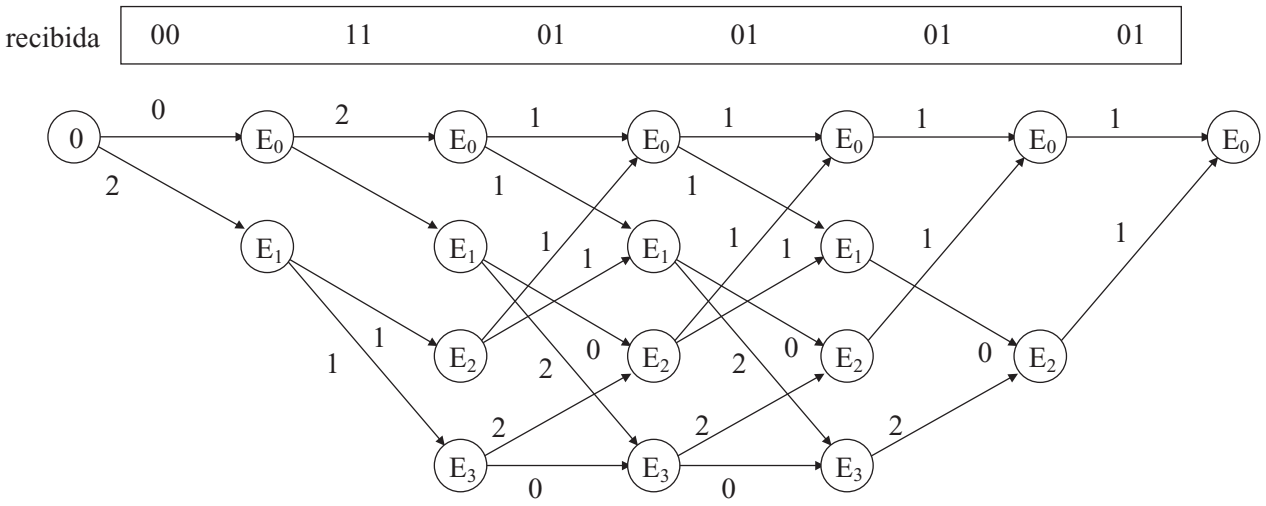
Codificación de canal

Codificación convolucional

- Algoritmo de Viterbi
  - Fundamento del algoritmo
  - Ejemplo
  - Decodificación de secuencias indefinidas

Codificación convolucional. Decodificación ML

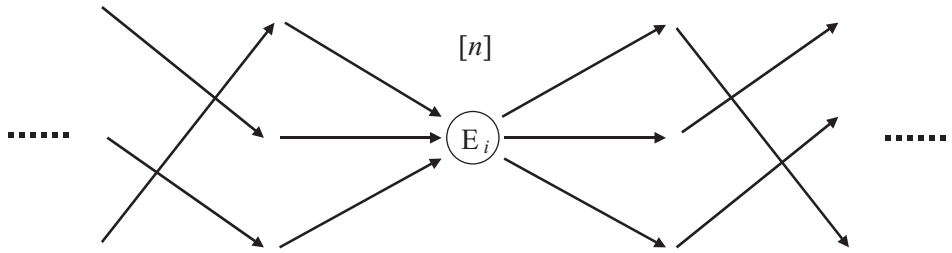
Busqueda del camino más corto



- Para todos los posibles caminos del grafo ¿cuál es el más corto?
- ¿Cuántos posibles caminos hay?
- ¿Hay que evaluarlos todos?

Codificación convolucional. Algoritmo de Viterbi

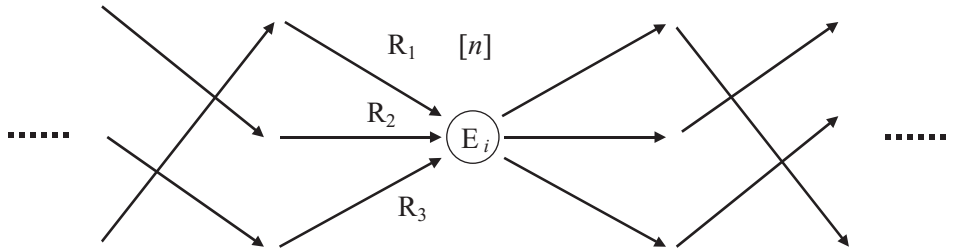
Busqueda del camino más corto Fundamento del algoritmo



- Fijemos un instante $[n]$ y un estado $E_i[n]$
- De todos los caminos que pasan por el **nodo** $E_i[n]$ ¿cuál es el de distancia mínima?
- Se han identificado todos los posibles caminos desde $E[n=0]=E_0$ hasta $E_i[n]$
- Se han identificado todos los posibles caminos desde $E_i[n]$ hasta el final (E_0)
- ¿Hay que considerar todas las combinaciones de caminos que llegan y que salen y calcular las distancias suma?

Codificación convolucional. Algoritmo de Viterbi

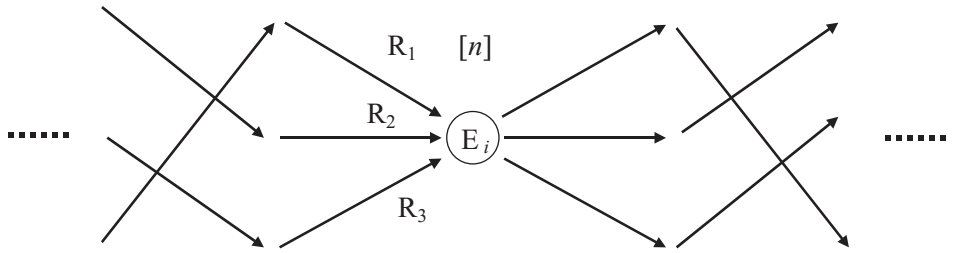
Busqueda del camino más corto Fundamento del algoritmo



- Se ha calculado la **distancia acumulada** de los caminos desde E[n=0] hasta $E_i[n]$ (sumando las distancias de rama) (R_1, R_2, R_3,...)
- Sea R_{min} la menor de distancias acumuladas hasta $E_i[n]$
- El camino global óptimo que <u>pasa</u> por $E_i[n]$ es:
  el que llega a $E_i[n]$ acumulando R_{min},
  combinado con el óptimo desde $E_i[n]$ hasta el final (E_0)

Codificación convolucional. Algoritmo de Viterbi

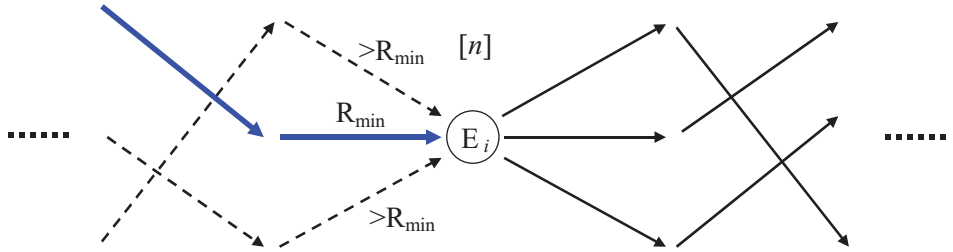
Busqueda del camino más corto Fundamento del algoritmo



- Los caminos que llegan a $E_i[n]$ acumulando $R > R_{min}$:
  <u>no pueden formar parte del camino global óptimo:</u> se pueden **descartar**
- Se pueden eliminar del grafo todas las ramas que llegan a $E_i[n]$ menos una:
  la rama que llega acumulando menor distancia R_{min}:
  la **rama superviviente**

Codificación convolucional. Algoritmo de Viterbi

Busqueda del camino más corto Fundamento del algoritmo



- Los caminos que llegan a $E_i[n]$ acumulando $R > R_{min}$:
  no pueden formar parte del camino óptimo: se pueden **descartar**

- Se pueden eliminar del grafo todas las ramas que llegan a $E_i[n]$ menos una:
  la rama que llega acumulando menor distancia R_{min}:
  la **rama superviviente**

Codificación de canal

Codificación convolucional

Decodificación con algoritmo de Viterbi

Fundamento del algoritmo

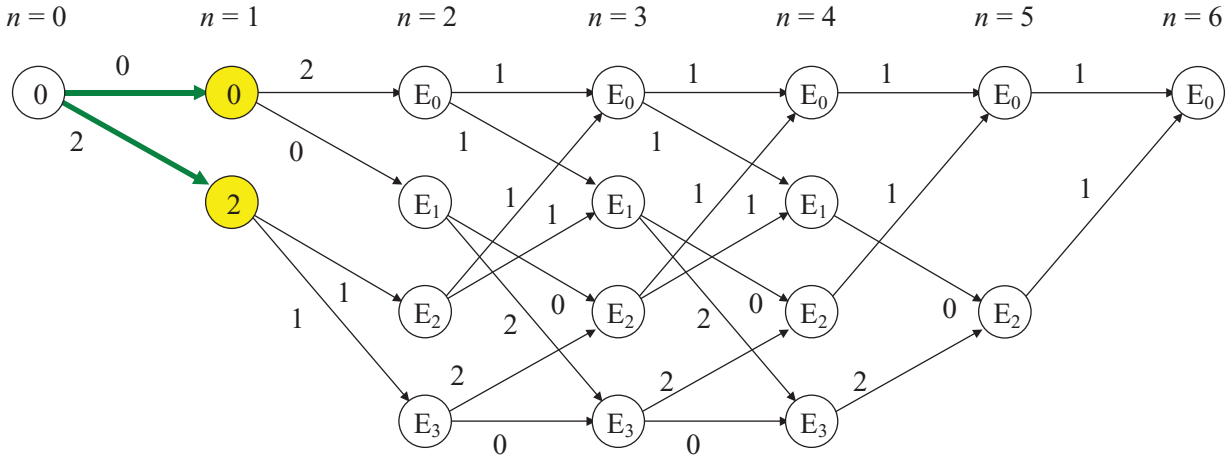
Ejemplo. Recorrido hacia adelante: Obtención de ramas supervivientes

Ejemplo. Recorrido hacia atrás: Obtención del camino y decisión

Decodificación de secuencias indefinidas

Codificación convolucional. Decodificación ML. Ejemplo

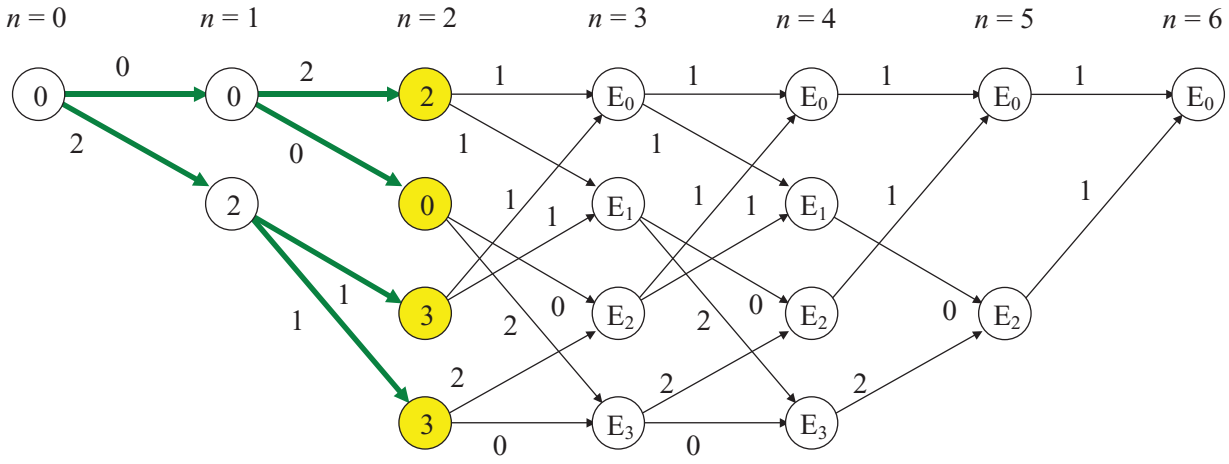
Distancias acumuladas hasta [n=1]



estado		rama	Distancia acumulada	
[n = 0]	[n = 1]	[n] 0→1	[n=0]	[n=1]
E_0	E_0	0	0	0
E_0	E_1	2	0	2

Codificación convolucional. Decodificación ML. Ejemplo

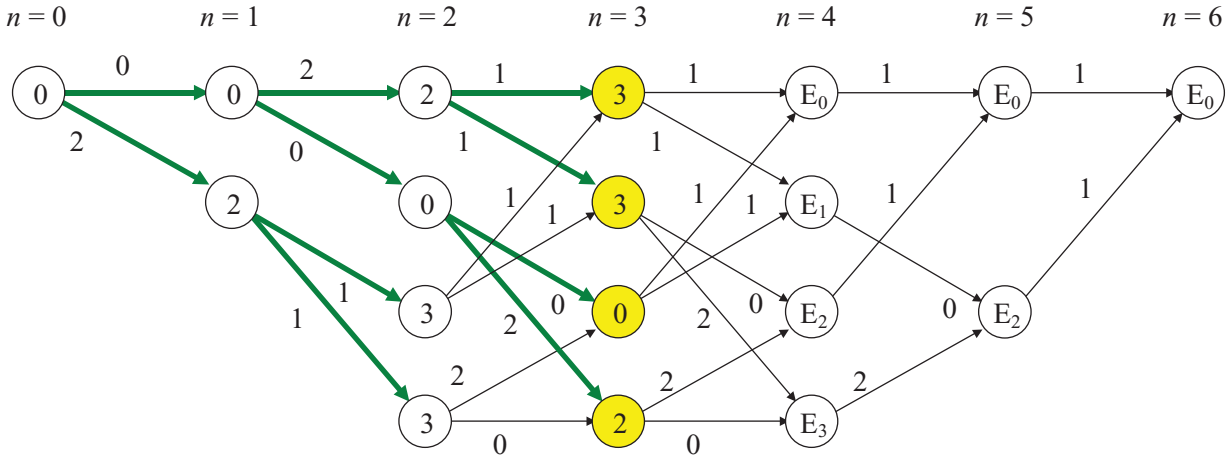
Distancias acumuladas hasta [n=2]



estado		rama	Distancia acumulada	
[n = 1]	[n = 2]	[n] 1→2	[n=1]	[n=2]
E_0	E_0	2	0	2
E_0	E_1	0	0	0
E_1	E_2	1	2	3
E_1	E_3	1	2	3

Codificación convolucional. Decodificación ML. Ejemplo

Distancias acumuladas hasta [n=3]. Selección de ramas supervivientes

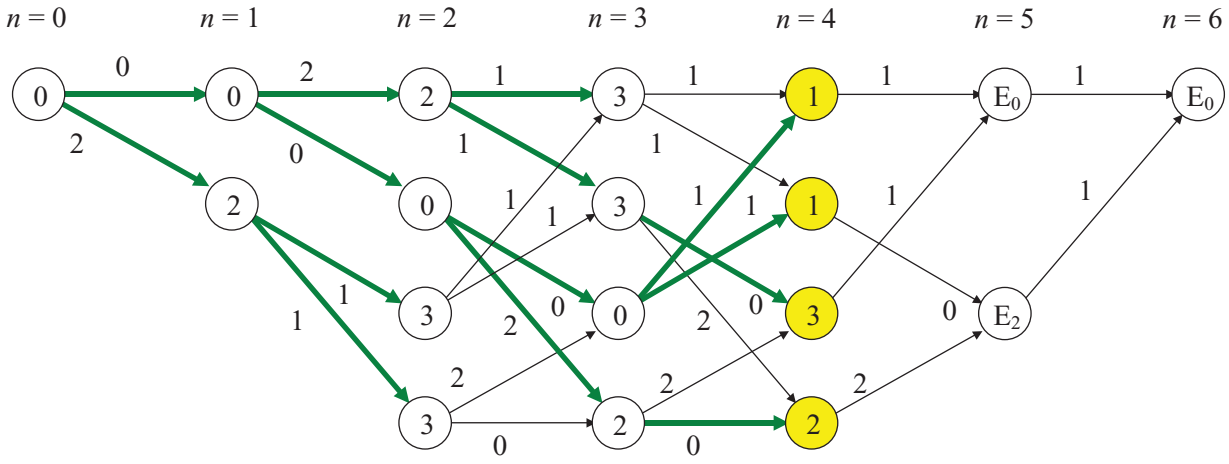


estado		rama	Distancia acumulada	
[n = 2]	[n = 3]	[n] 2→3	[n=2]	[n=3]
E_0	E_0	1	2	3
E_2		1	3	4
E_0	E_1	1	2	3
E_2		1	3	4

estado		rama	Distancia acumulada	
[n = 2]	[n = 3]	[n] 2→3	[n=2]	[n=3]
E_1	E_2	0	0	0
E_3		2	3	5
E_1	E_3	2	0	2
E_3		0	3	3

Codificación convolucional. Decodificación ML. Ejemplo

Distancias acumuladas hasta [n=4]. Selección de ramas supervivientes

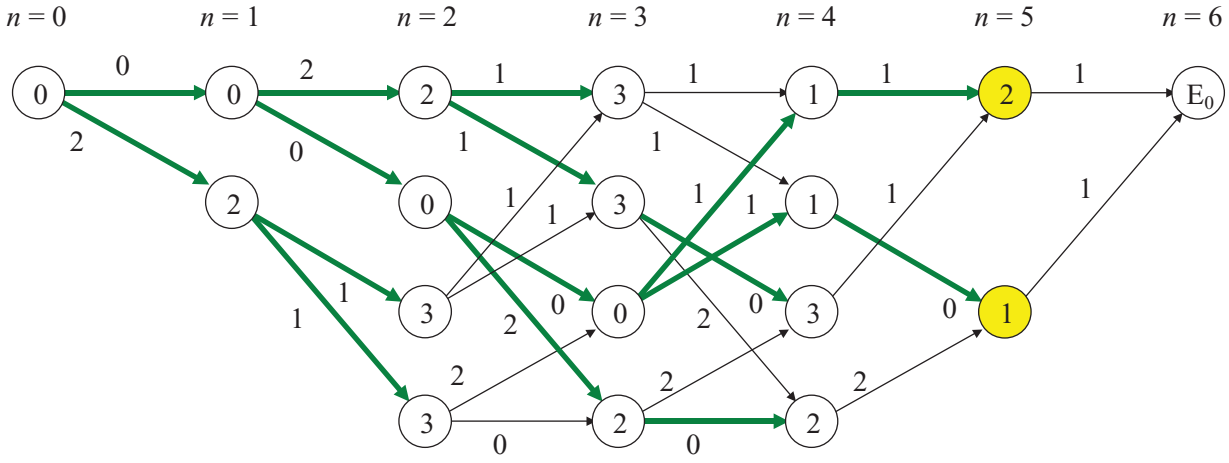


estado		rama	Distancia acumulada	
[n = 3]	[n = 4]	[n] 3→4	[n=3]	[n=4]
E_0	E_0	1	3	4
E_2		1	0	1
E_0	E_1	1	3	4
E_2		1	0	1

estado		rama	Distancia acumulada	
[n = 3]	[n = 4]	[n] 3→4	[n=3]	[n=4]
E_1	E_2	0	3	3
E_3		2	2	4
E_1	E_3	2	3	5
E_3		0	2	2

Codificación convolucional. Decodificación ML. Ejemplo

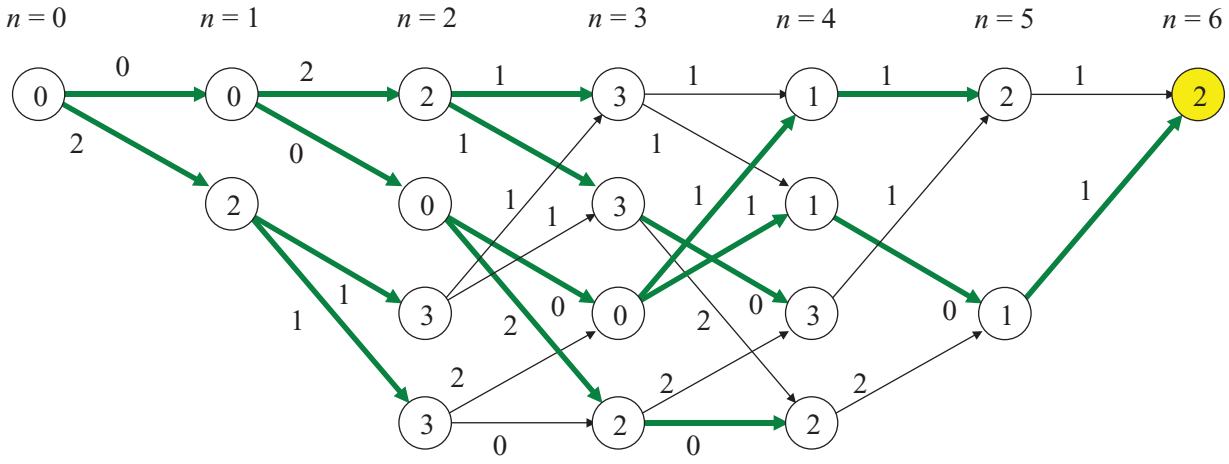
Distancias acumuladas hasta [n=5]. Selección de ramas supervivientes



estado		rama	Distancia acumulada	
[n = 4]	[n = 5]	[n] 4→5	[n=4]	[n=5]
E_0	E_0	1	1	2
E_2		1	3	4
E_1	E_2	0	1	1
E_3		2	2	4

Codificación convolucional. Decodificación ML. Ejemplo

Distancias acumuladas hasta [n=5]. Selección de ramas supervivientes



estado		rama	Distancia acumulada	
[n = 5]	[n = 6]	[n] 5→6	[n=5]	[n=6]
E_0	E_0	1	2	3
E_2	E_0	1	1	2

Decodificación con algoritmo de Viterbi

Fundamento del algoritmo

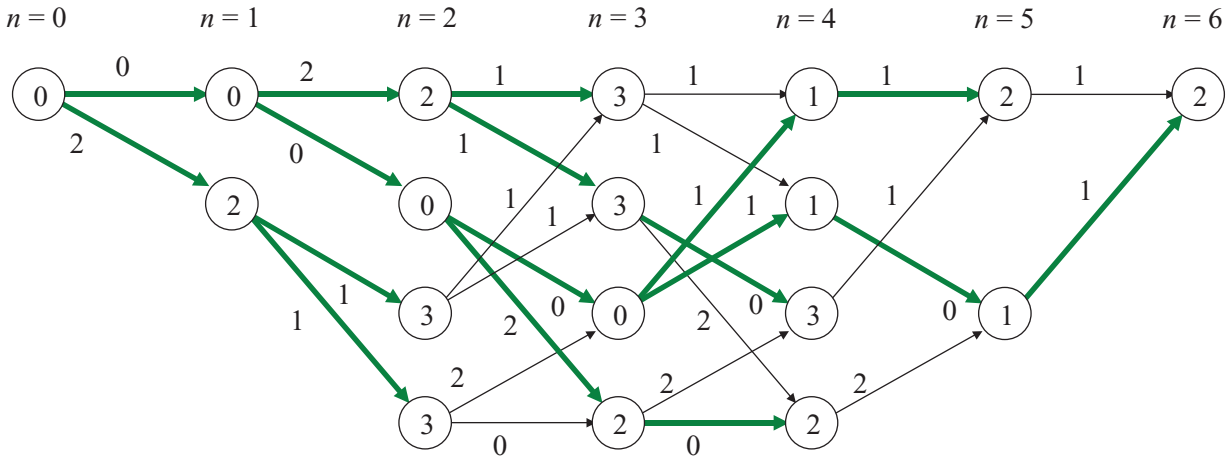
Ejemplo. Recorrido hacia adelante: Obtención de ramas supervivientes

Ejemplo. Recorrido hacia atrás: Obtención del camino y decisión

Decodificación de secuencias indefinidas

Codificación convolucional. Decodificación ML. Ejemplo

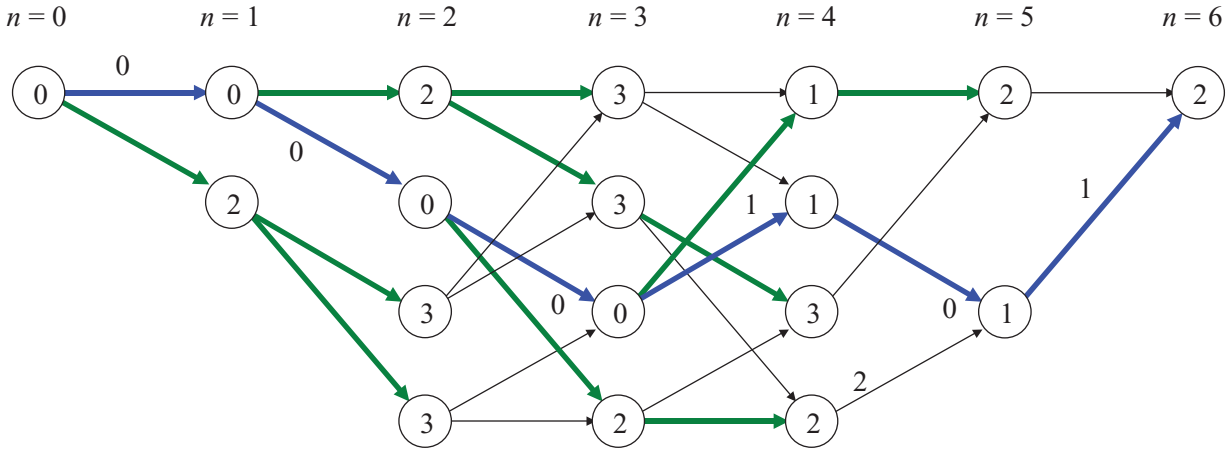
Recorrido hacia atrás



- Recorriendo hacia atrás por las ramas seleccionadas se encuentra el camino (d_H=2)

Codificación convolucional. Decodificación ML. Ejemplo

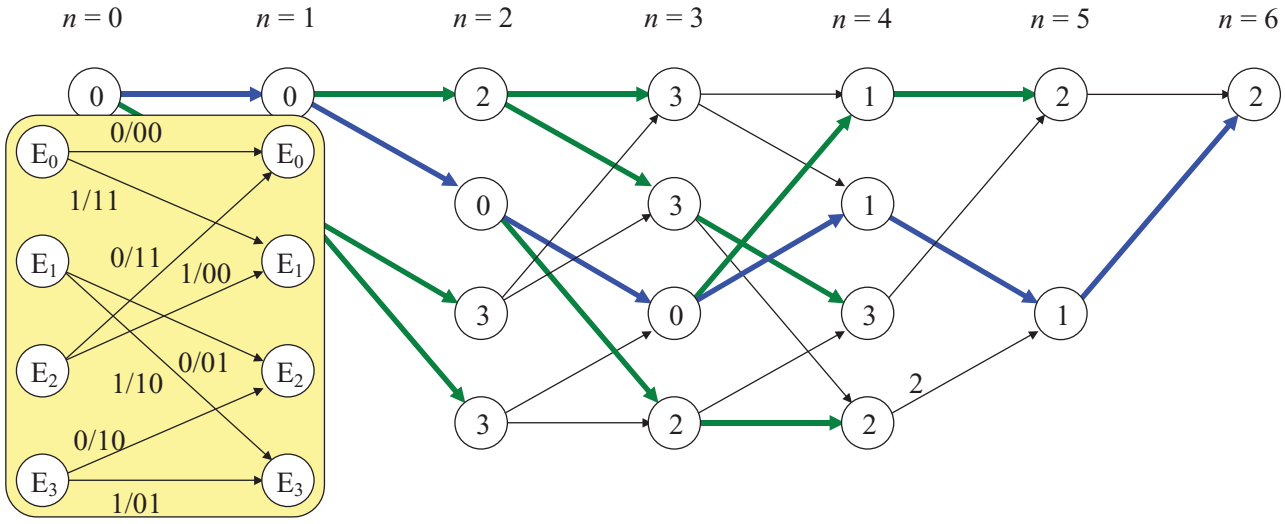
Recorrido hacia atrás



- Recorriendo hacia atrás por las ramas seleccionadas se encuentra el camino (d_H=2)

Codificación convolucional. Decodificación ML. Ejemplo

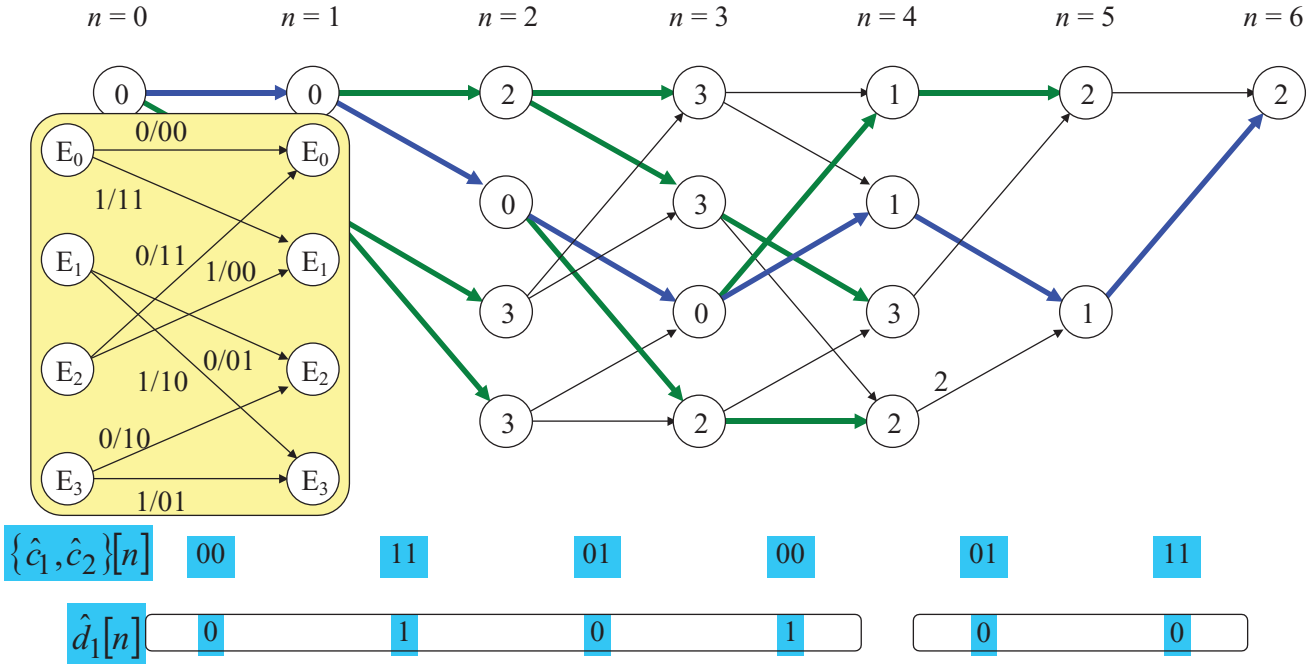
Recorrido hacia atrás



- Recorriendo hacia atrás por las ramas seleccionadas se encuentra el camino (d_H=2)
  - el camino de ramas permite obtener la secuencia $\{c_1,c_2\}[n]$ decidida
  - y decodificar las secuencia de bits de mensaje $d_1[n]$

Codificación convolucional. Decodificación ML. Ejemplo

Recorrido hacia atrás. Decisión y obtención de los bits de mensaje



Codificación de canal

Codificación convolucional

Algoritmo de Viterbi

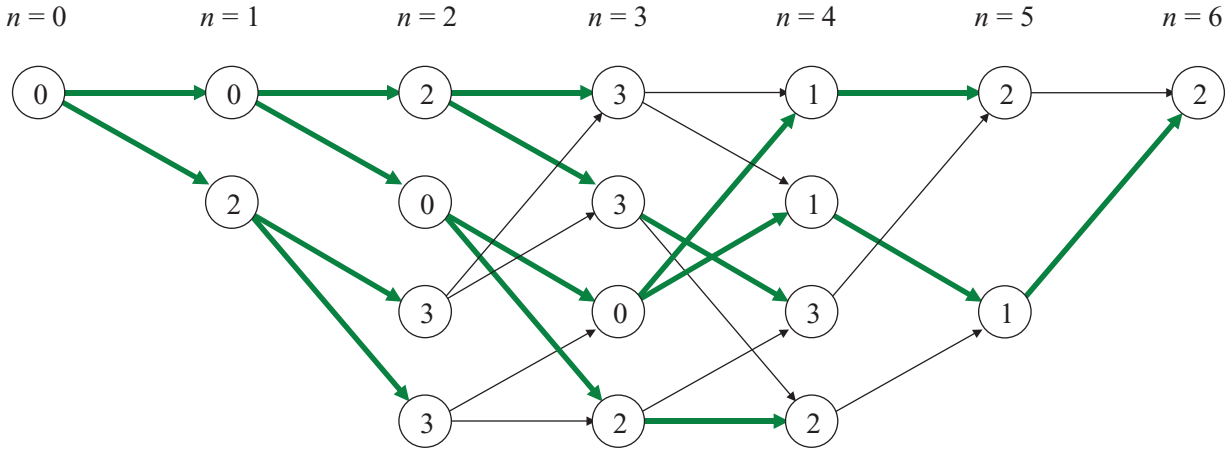
Fundamento del algoritmo

Ejemplo

Decodificación de secuencias indefinidas

Codificación convolucional. Decodificación ML.

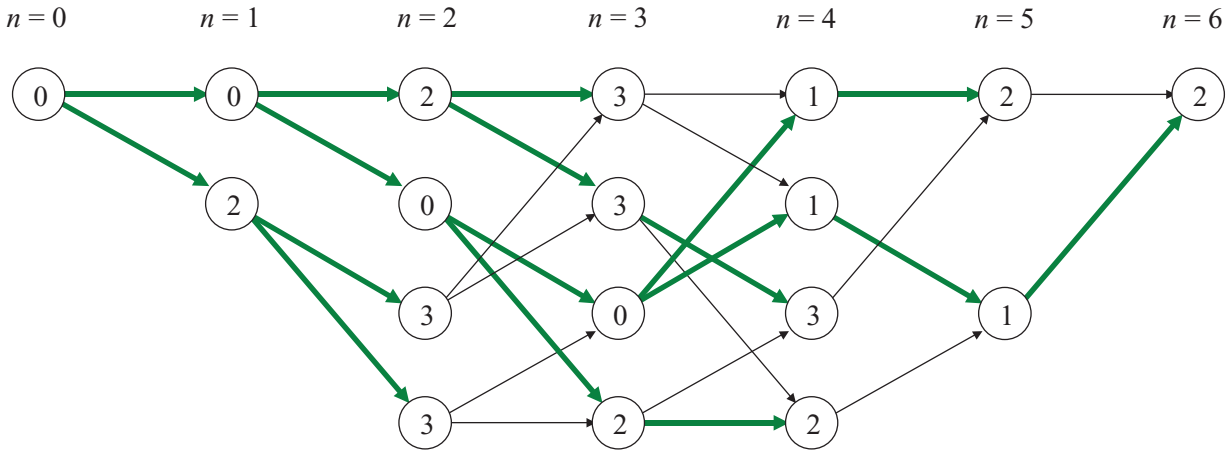
Algoritmo de Viterbi. Secuencias indefinidas



- Para decidir los bits de las primeras etapas ...
  ... ¿Hay que esperar a tener toda la secuencia?
- Si es una secuencia indefinida ... ¿no se puede usar el algoritmo?

Codificación convolucional. Decodificación ML.

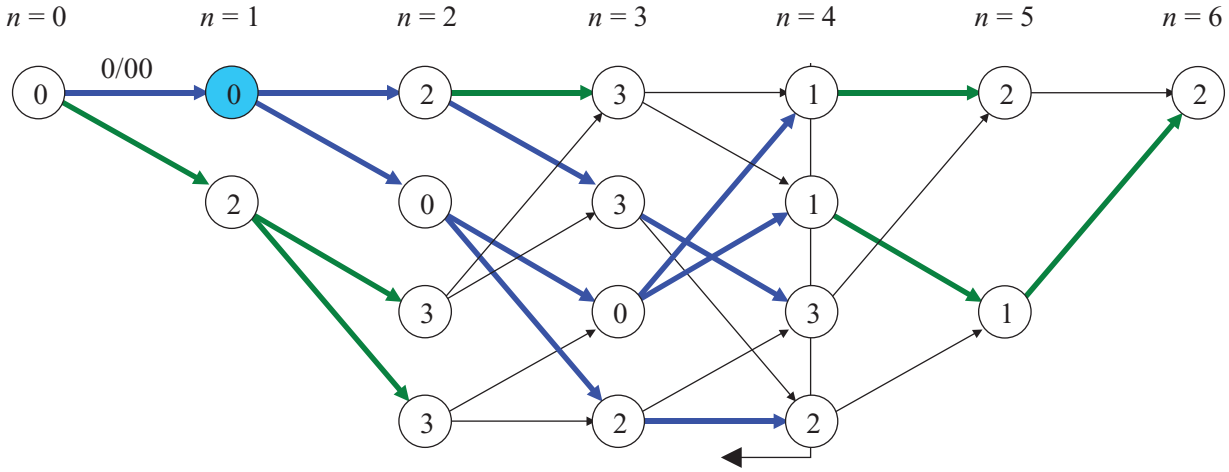
Algoritmo de Viterbi. Secuencias indefinidas



- Si desde un estado intermedio se recorren todos los caminos supervivientes hacia atrás, ... **se encuentran** a una cierta distancia

Codificación convolucional. Decodificación ML.

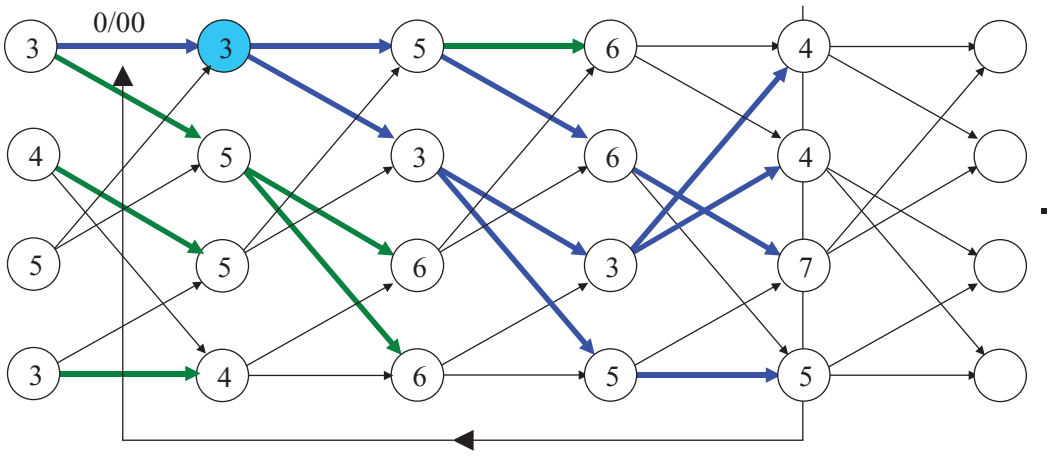
Algoritmo de Viterbi. Secuencias indefinidas



- Si desde E[n=4] se recorre hacia atrás $K = 3$ etapas, ...
  - ... todos los caminos supervivientes se encuentran: E[n=1]=E_0 y ...
  - ... se puede saber la rama anterior (0/00) y ...
  - ... decidir $\{c_1,c_2\}[n=1] = \{0,0\}$ $d_1[n=0] = 0$

Codificación convolucional. Decodificación ML.

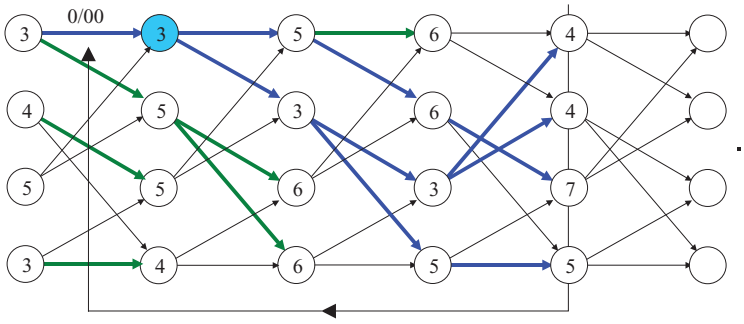
Algoritmo de Viterbi. Secuencias indefinidas



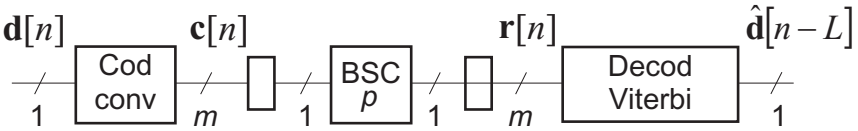
- En el instante [n] se puede tomar la decisión sobre L instantes antes
  $\{c_1,c_2\}[n-L]$ $d_1[n-L]$
- L : **profundidad del recorrido hacia atrás**: basta algo mayor que K
  (K: longitud de restricción, memoria del sistema)

Codificación convolucional. Decodificación ML.

Algoritmo de Viterbi. Secuencias indefinidas



- Profundidad del recorrido hacia atrás $L > K$
- Se pueden procesar secuencias indefinidas con retardo L



Codificación de canal

Codificación convolucional

Concepto
Comportamiento
Decodificación

Algoritmo de Viterbi

Códigos perforados

Codificación convolucional. Perforado

Codificadores convolucionales

- Suelen ser simples $(m,k) = (2,1)$ $(3,1)$
- Normalmente se describen con su **tasa de codificación**: 1/2, 1/3 k/m (m,k)
- Suelen ser de 4 u 8 estados (2 ó 3 registros de bit)

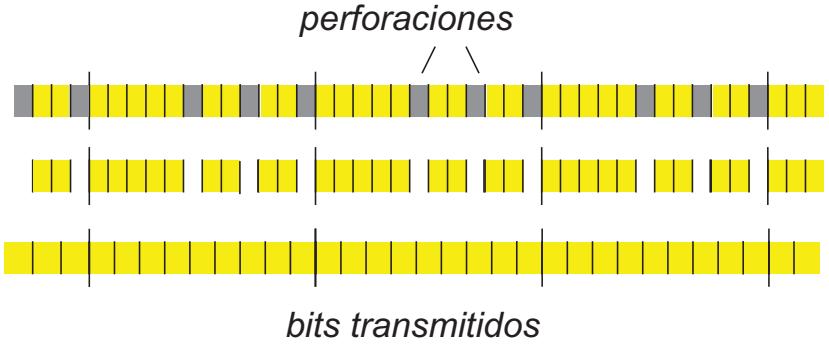
Tasa de codificación y complejidad

- Un código de tasa 1/2 ó 1/3 tiene mucha redundancia y capacidad de corrección (a cambio de una gran pérdida de velocidad binaria)
- Frecuentemente no es necesaria tanta capacidad de corrección (cuando la BER sin codificar es baja: $<10^{-2}$ ó 10^{-3})
- Solución: usar códigos **perforados**

Codificación convolucional. Perforado

Código perforado

- Se utiliza un codificador simple de tasa 1/2 ó 1/3 k/m (***código madre***)
- Después del codificador se eliminan algunos bits de salida que no se transmiten
- La perforación se hace siguiendo un **patrón periódico:**
  En cada periodo de p bits se eliminan q bits
  Ejemplo $p = 12$, $q = 3$



- La tasa de codificación resultante es $(k/m) \times p/(p-q)$ ej.: $1/3 \times 12/9 = 4/9$

Codificación convolucional. Perforado

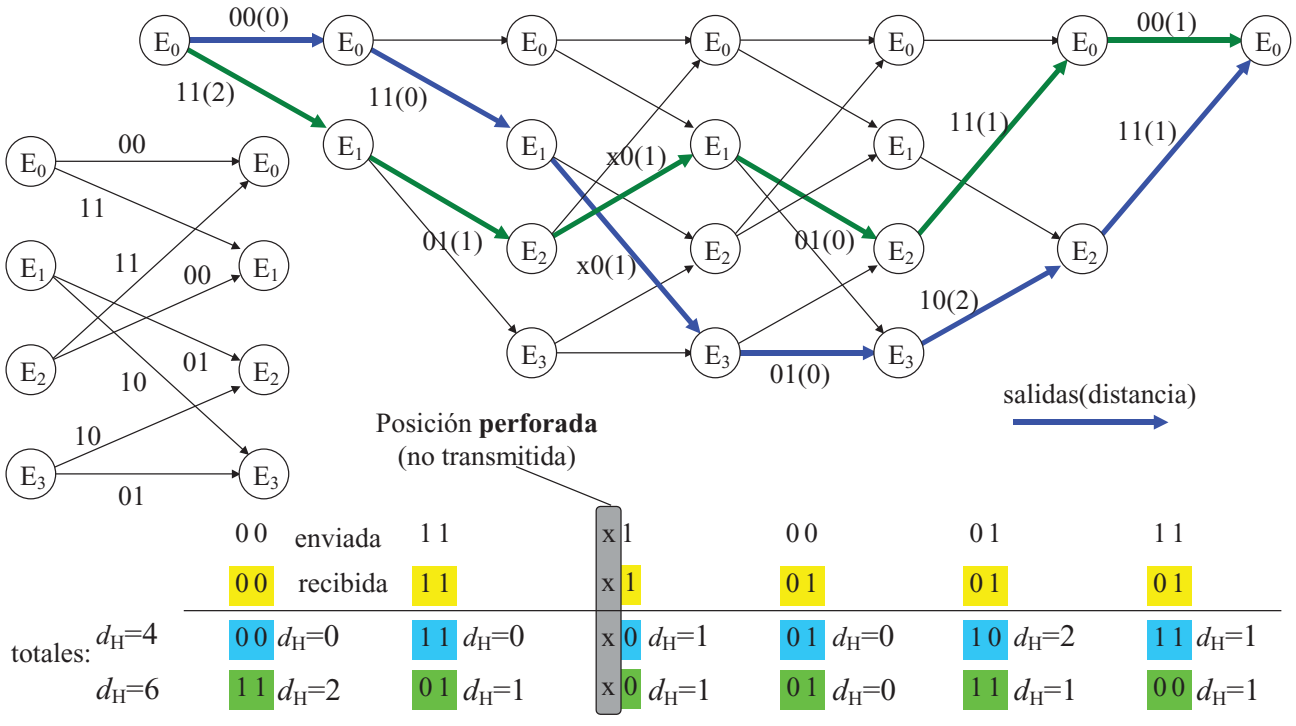
Código perforado

Decodificación

- El decodificador no recibe esos bits pero sabe dónde se han eliminado
- Se compara la secuencia recibida con las posibles que son:
  - las que el codificador genera, eliminando de ellas los bits perforados
  - (no cuentan para calcular la distencia entre secuencias)
- Si se utiliza el grafo de distancias y la búsqueda del camino más corto ...
  - En bits perforados: se elimina la distancia entre bit recibido y posible generado
  - (al calcular distancias de rama se ignora la de los bits perforados)
  - (los bits perforados no cuentan a favor ni en contra de un camino)

Codificación convolucional. Decodificación ML Ejemplo

Código perforado



Codificación convolucional. Perforado

- La **perforación** reduce la redundancia y la capacidad de corrección de errores ...
  ... a cambio de aumentar la velocidad binaria transmitida

Ejemplo 1

- Codificador sistemático de tasa 1/3 periodo de perforación $p = 12$ bits

patrón	*bits 1 a 12*	(*q*) *bits perforados*	(*p-q*) *bits transmitidos*	*tasa de codificación*	$\frac{1}{3}\cdot\frac{p}{p-q}$
1	d d d d	0	12	1/3	0,33
2	d d d d	1	11	4/11	0,36
3	d d d d	2	10	2/5	0,40
4	d d d d	3	9	4/9	0,44
5	d d d d	4	8	1/2	0,50
6	d d d d	5	7	4/7	0,57
7	d d d d	6	6	2/3	0,67
8	d d d d	7	5	4/5	0,80
9	d d d d	8	4	1	1,00

bits perforados d bits del mensaje

Codificación convolucional. Perforado

Ejemplo 2 TDT

- Codificador no sistemático de tasa 1/2.
- Periodo de perforación distinto para cada patrón

patrón	(*p*)	(*q*) *bits perforados*	(*p-q*) *bits transmitidos*	*tasa de codificación*	
1 (sin perforación)	2	0	2 de 2	1/2	0,50
2	4	1 de 4	3 de 4	2/3	0,67
3	6	2 de 6	4 de 6	3/4	0,75
4	10	4 de 10	6 de 10	5/6	0,83
5	14	6 de 14	8 de 14	7/8	0,87

Tasa de codificación: $k/m \times \dfrac{\text{longitud patrón } (p)}{\text{bits transmitidos } (p\text{-}q)}$

Ejemplos: patrón 4: $1/2 \times 10/6 = 5/6$ patrón 5: $1/2 \times 14/8 = 7/8$

Codificación de canal

Conceptos

Codificación bloque

Codificación convolucional

Decodificación blanda

Codificación de canal

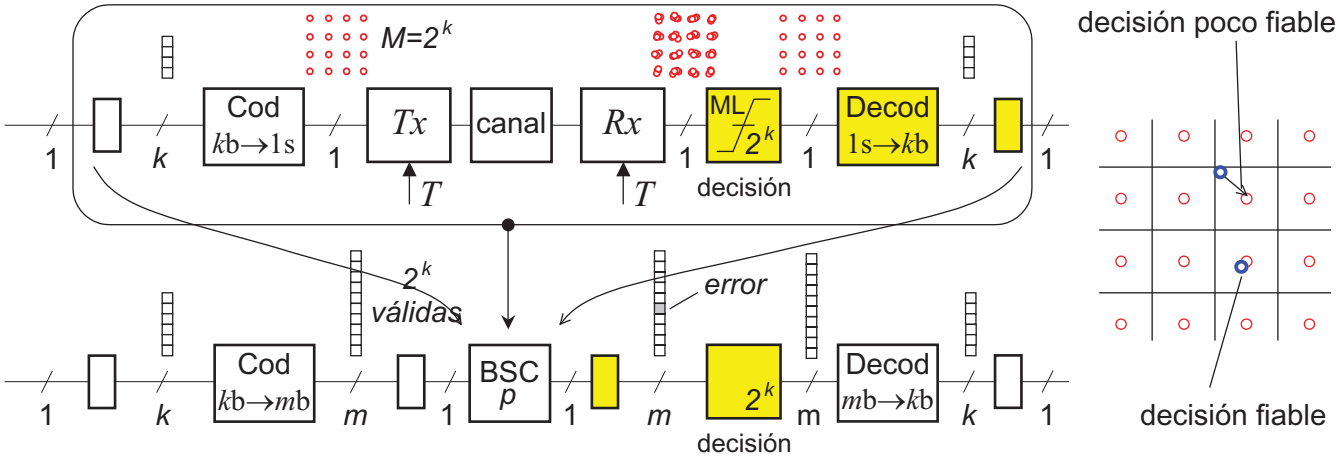
Decodificación blanda

Concepto. Prestaciones y eficacia

Aplicación a codificación convolucional

Decodificación iterativa

Codificación de canal con decisión “dura”

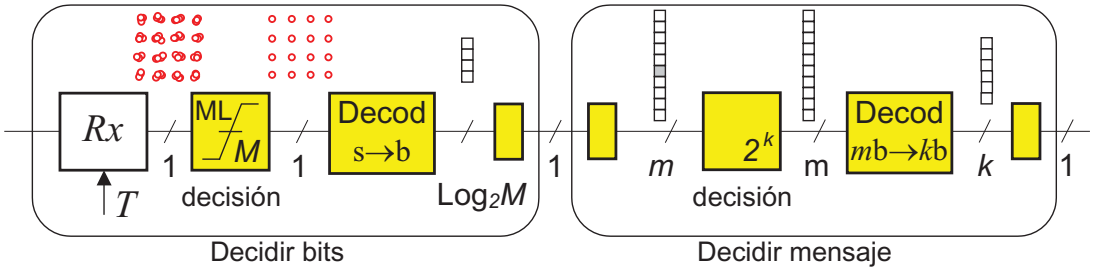


- Demodulador QAM **decide símbolos**, decodifica en bits y, a partir de ellos ...
- Decodificador de canal **decide** la **palabra-código**
- Una decisión de símbolo “poco fiable” (a distancia similar de dos símbolos) ¿debe ser igualmente considerada que una más fiable?

Codificación de canal. Decisión dura y blanda

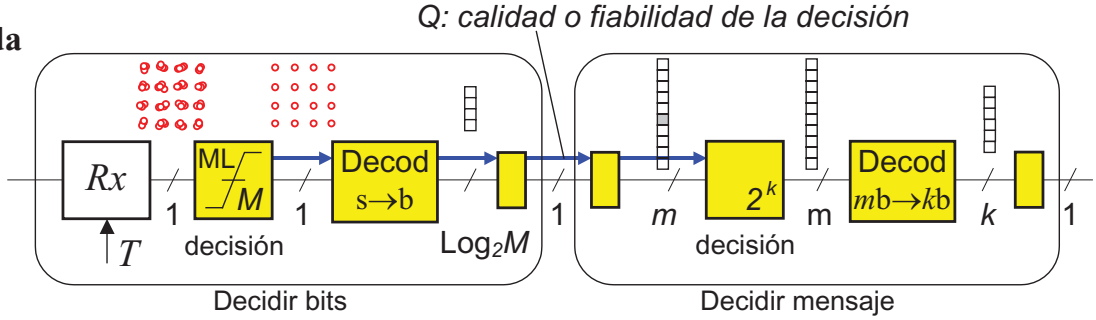
Decisión Dura

Hard Decision **(HD)**



Decisión Blanda

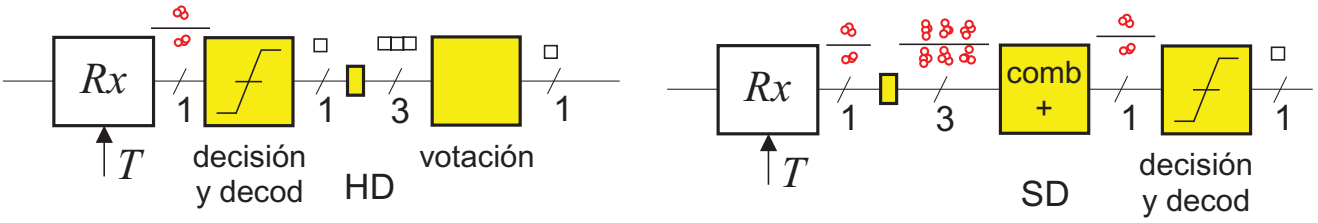
Soft Decisión **(SD)**



- Al decidir el símbolo se indica también la “calidad o fiabilidad” de la decisión
- La “calidad” de los bits decididos se utiliza en la decodificación de canal

Codificación de canal. Decisión dura y blanda

Ejemplo PAM-2 Canal gaussiano Código de repetición (3,1)



Decisión dura (HD)

- El demodulador envía el valor del símbolo recibido {-1,1} + ruido
- Se decide cada bit a partir del signo del símbolo
- Con cada grupo de 3 bits de decodifica el mensaje por votación simple

Decisión blanda (SD)

- El demodulador envía el valor del símbolo recibido {-1,1} + ruido (bit blando)
- Se suman los valores de cada grupo de tres símbolos
- A partir de esa suma (combinación) decide el mensaje a partir del signo

Codificación de canal

Decodificación blanda

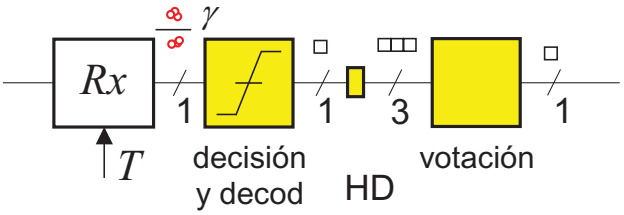
Concepto. Prestaciones y eficacia

Aplicación a codificación convolucional

Decodificación iterativa

Codificación de canal. Decisión dura y blanda

Ejemplo PAM-2 Canal gaussiano Código de repetición (3,1)



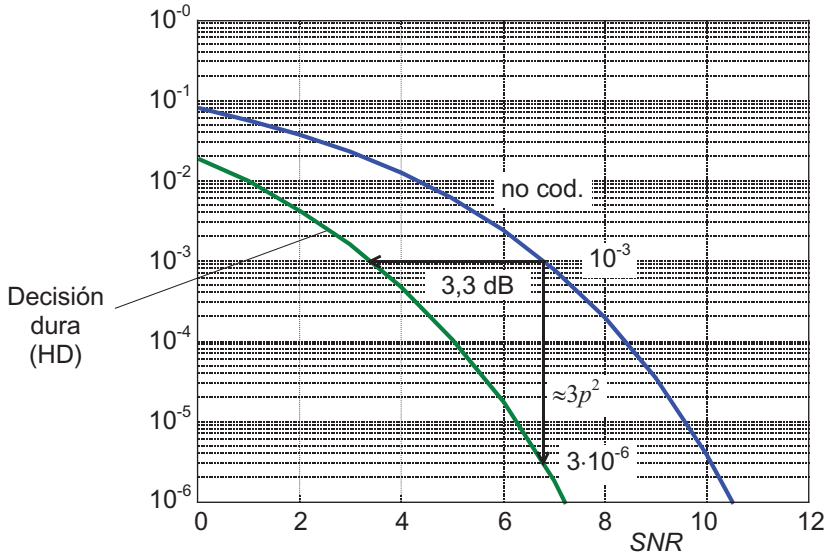
Decisión dura (HD). Probabilidad de error

- PAM-2: probabilidad de error de decisión de bit: $p = Q\left(\sqrt{2\gamma}\right)$
- Repetición (3,1):
  - probabilidad de error de decisión de mensaje (votación) : $3p^2+p^3$
    (*Prob* (2 errores) + *Prob* (3 errores))

$$3\left(Q\left(\sqrt{2\gamma}\right)\right)^2 + \left(Q\left(\sqrt{2\gamma}\right)\right)^3$$

Codificación de canal. Decisión dura y blanda

Ejemplo PAM-2 Canal gaussiano Código de repetición (3,1)



no cod.

$Q\left(\sqrt{2\gamma}\right)$

rep(3,1) HD

$3\left(Q\left(\sqrt{2\gamma}\right)\right)^2 + \left(Q\left(\sqrt{2\gamma}\right)\right)^3$

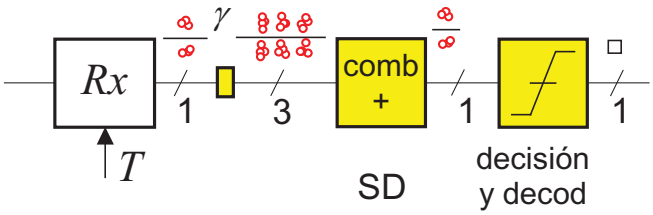
- Con codificación se consigue la misma BER con 3,3 dB menos de SNR

  (a cambio de una pérdida de velocidad de transmisión)

Codificación de canal. Decisión dura y blanda

Ejemplo PAM-2 Canal gaussiano Código de repetición (3,1)

Decisión blanda (SD)



Ejemplos

Bits blandos		
0,6	0,6	-0.9

Decisión blanda

suma	*signo*	*mensaje*
0,3	+	"1"

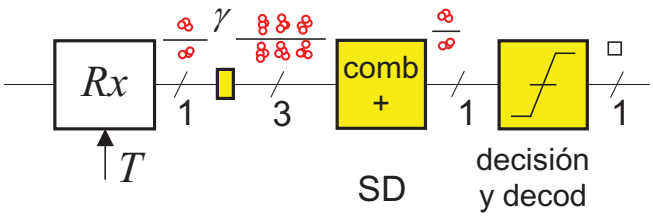
Decisión dura

Bits			*mensaje*
"1"	"1"	"0"	"1"

Codificación de canal. Decisión dura y blanda

Ejemplo PAM-2 Canal gaussiano Código de repetición (3,1)

Decisión blanda (SD)



Ejemplos

Bits blandos		
0,6	0,6	-0.9
-0,1	-0,1	0,7

Decisión blanda

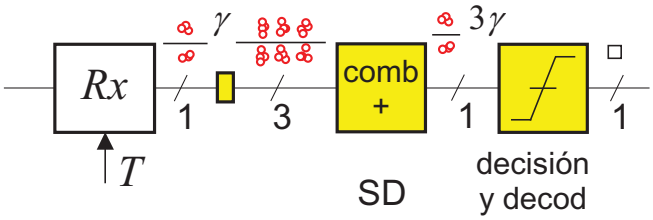
suma	*signo*	*mensaje*
0,3	+	"1"
0,5	+	"1"

Decisión dura

Bits			*mensaje*
"1"	"1"	"0"	"1"
"0"	"0"	"1"	"0"

Codificación de canal. Decisión dura y blanda

Ejemplo PAM-2 Canal gaussiano Código de repetición (3,1) Decisión blanda (SD)



SNR de la combinación

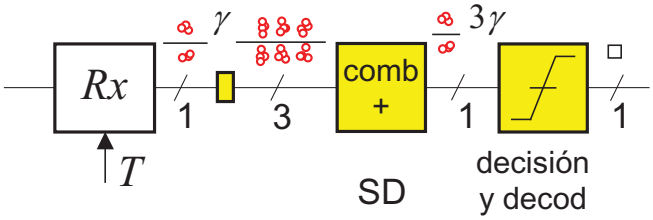
- Bits se envían potencia unidad {-1,1}. Si σ^2 es la potencia del ruido: $\gamma = 1/\sigma^2$
- Los tres bit enviados son iguales: su suma es ± 3 y la potencia de la suma es 9
- Las tres muestras de ruido son independientes: su potencia es $3/\sigma^2$
- Relación señal ruido de la combinación:

$$\frac{9}{3\sigma^2} = \frac{3}{\sigma^2} = 3\gamma$$

Ganancia de SNR por repetición: 3
(muestras de ruido de signo distinto)

Codificación de canal. Decisión dura y blanda

Ejemplo PAM-2 Canal gaussiano Código de repetición (3,1) Decisión blanda (SD)



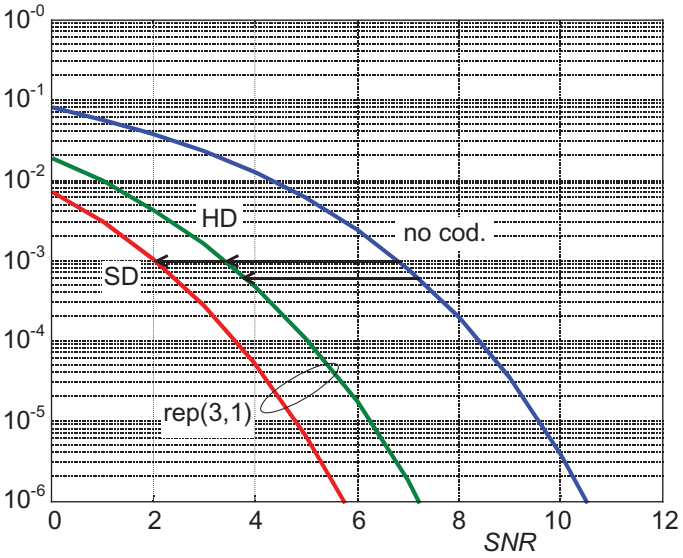
Probabilidad de error

- SNR de la combinación es 3γ :

probabilidad de error de decisión del mensaje $Q\left(\sqrt{2 \cdot 3\gamma}\right)$

Codificación de canal. Decisión dura y blanda

Ejemplo PAM-2 Canal gaussiano Código de repetición (3,1)



no cod.

$$Q\left(\sqrt{2\gamma}\right)$$

rep(3,1) HD

$$3\left(Q\left(\sqrt{2\gamma}\right)\right)^2+\left(Q\left(\sqrt{2\gamma}\right)\right)^3$$

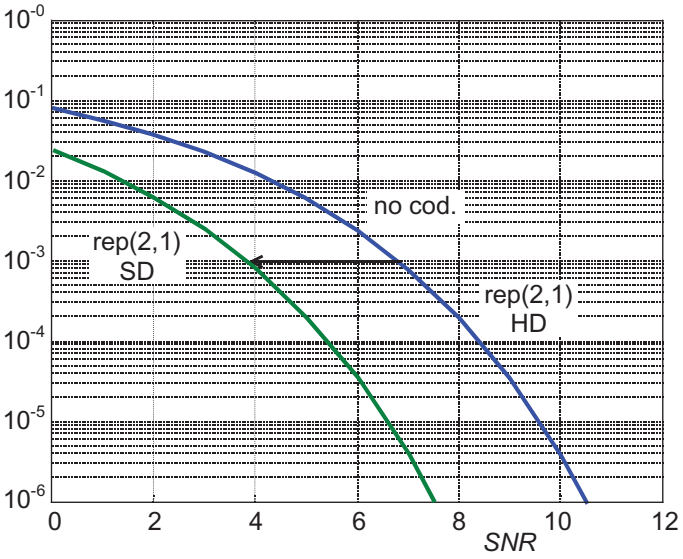
rep(3,1) SD

$$Q\left(\sqrt{6\gamma}\right)$$

- Codificando y con decisión blanda se puede bajar la SNR 4,7 dB ($10\log_{10}3$)
- SD tiene una **ganancia** de 1,4 dB sobre HD
  (sin pérdida de velocidad de transmisión)

Codificación de canal. Decisión dura y blanda

Ejemplo PAM-2 Canal gaussiano Código de repetición (2,1)



no cod.

$$Q\left(\sqrt{2\gamma}\right)$$

rep(2,1) SD

$$Q\left(\sqrt{4\gamma}\right)$$

- El código de repetición (2,1) con HD no mejora la BER pero...
- SD tiene una **ganancia** de 3 dB ($10\log_{10}2$) sobre HD

Codificación de canal

Decodificación blanda

Concepto. Prestaciones y eficacia

Aplicación a codificación convolucional

Decodificación iterativa

Codificación de canal. Decisión dura y blanda

Eficacia de la codificación

- Los códigos de repetición permiten obtener igual P_{err} con menor SNR ...
  ... pero a costa de una reducción de la velocidad binaria
- ¿Como comparar la eficacia real de los códigos?
  - usando la SNR referida a la energía por bit sin codificar (bit del mensaje)
  - en vez de la SNR referida a la energía por bit codificado o por símbolo

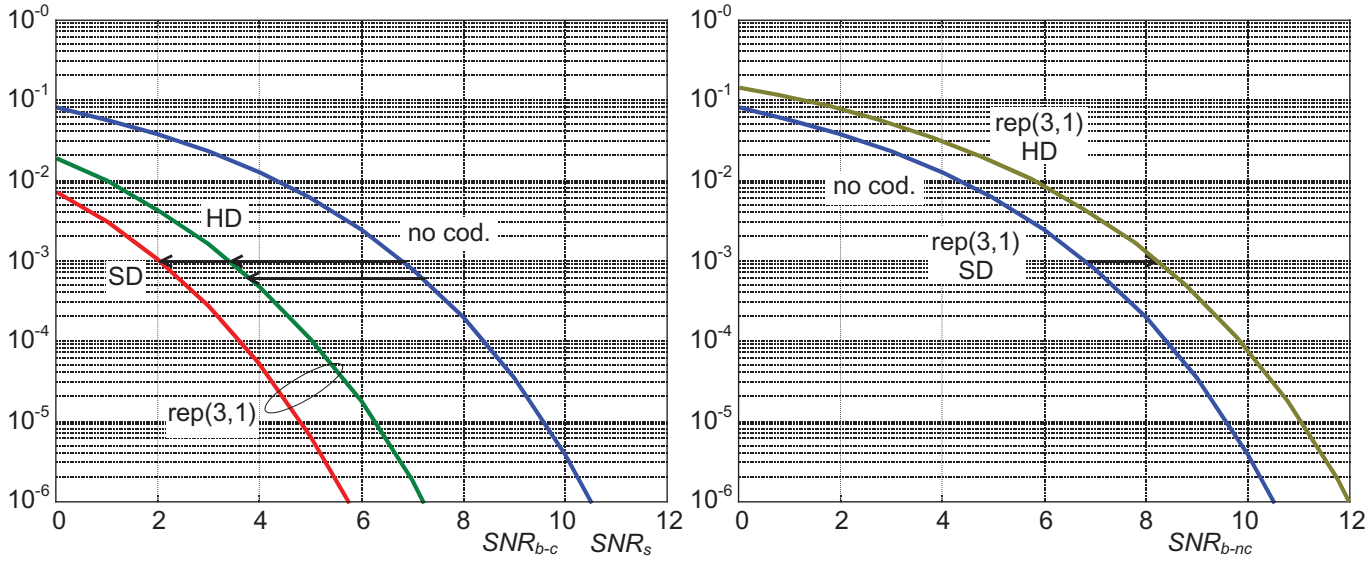
Ejemplo PAM-2 Canal gaussiano Código de repetición (3,1)

- Energía por bit codificado $E_{b\text{-}c}$ = energía por símbolo E_s
- Energía por bit de mensaje (no codificado) $E_{b\text{-}nc} = 3 \times E_{b\text{-}c} = 3 \times E_s$
- SNR referida a la energía por bit sin codificar (de mensaje)

$$\gamma_{b-nc} = \frac{E_{b-nc}}{N_0} \qquad \gamma_{b-nc} = 3 \cdot \gamma_{b-c} = 3 \cdot \gamma_s \qquad + 4{,}7 \text{ dB}$$

Codificación de canal. Decisión dura y blanda

Eficacia de la codificación



- El código de repetición con decisión blanda no tiene ganancia
- El código de repetición con decisión dura tiene una pérdida 1,4 dB
- Códigos de repetición: se usan para aumentar fiabilidad no para aumentar eficacia

Codificación de canal. Decisión dura y blanda

Eficacia de la codificación. Caso general

- Código (m,k) tasa de codificación k/m
- Energía por bit de mensaje (no codificado) $E_{b\text{-}nc} = (m/k) \times E_{b\text{-}c}$
- SNR referida a la energía por bit sin codificar (de mensaje)

$$\gamma_{b-nc} = \frac{E_{b-nc}}{N_0} \qquad \gamma_{b-nc} = \frac{m}{k} \cdot \gamma_{b-c}$$

Ganancia de codificación:

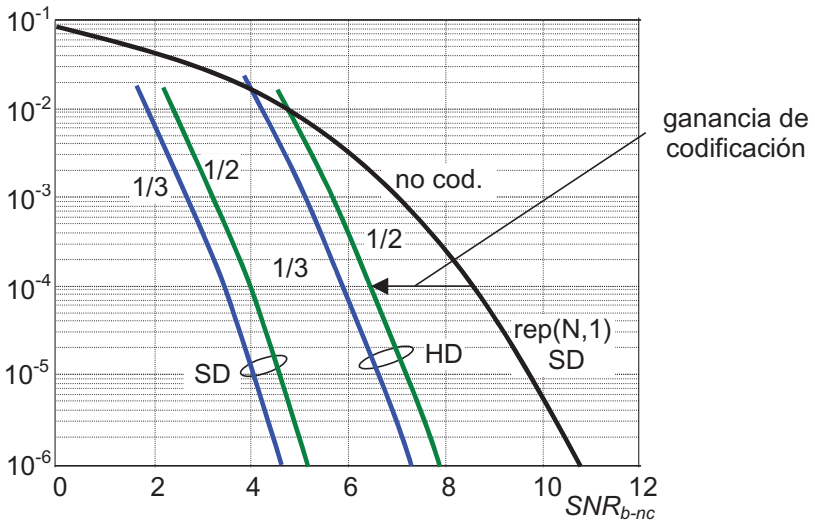
Reducción de $SNR_{b\text{-}nc}$ para obtener una misma P_{err}

- Los códigos de repetición no tienen ganancia de codificación (con HD: pérdida)
- Los códigos convolucionales son más eficaces:
  - tienen ganancia de codificación,
  - sobre todo si usan decisiones blandas

Codificación de canal. Decisión dura y blanda

Eficacia de la codificación

- Los códigos convolucionales tienen ganancia de codificación,
  ... sobre todo si usan decisiones blandas



Codificación de canal

Decodificación blanda

Concepto. Prestaciones y eficacia

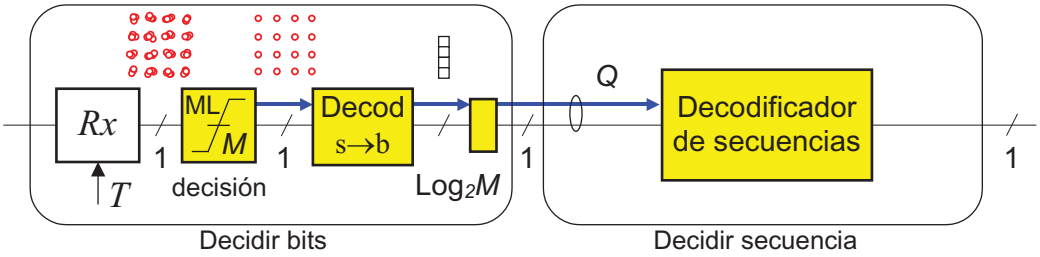
Aplicación a codificación convolucional

Decodificación con decisiones blandas

Decodificación con salida blanda

Decodificación iterativa

Decodificación de secuencias con decisión blanda

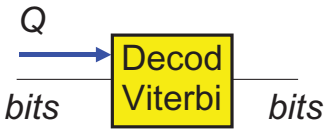


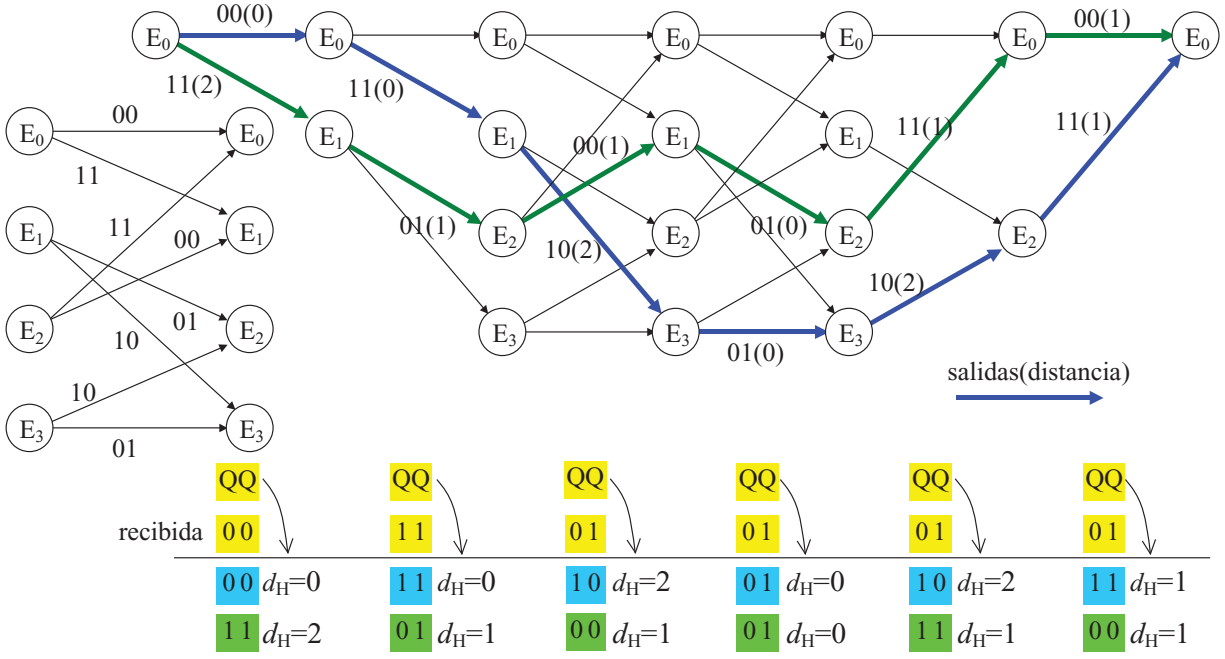
Medidas de distancia

- El decodificador de canal dispone de las **medidas de calidad** de los bits: Q
- El decodificador ML busca la secuencia (de las posibles) **más parecida** a la recibida
- ML: **menor distancia entre secuencias**
  - **suma** de distancias de cada bit recibido a **cada bit de la secuencia**)
- Se puede hacer una **suma ponderada** por la calidad de cada bit
- Los bits más "inciertos" contribuyen menos al cálculo de la **distancia total**

Codificación convolucional

Decodificación ML con decisión blanda



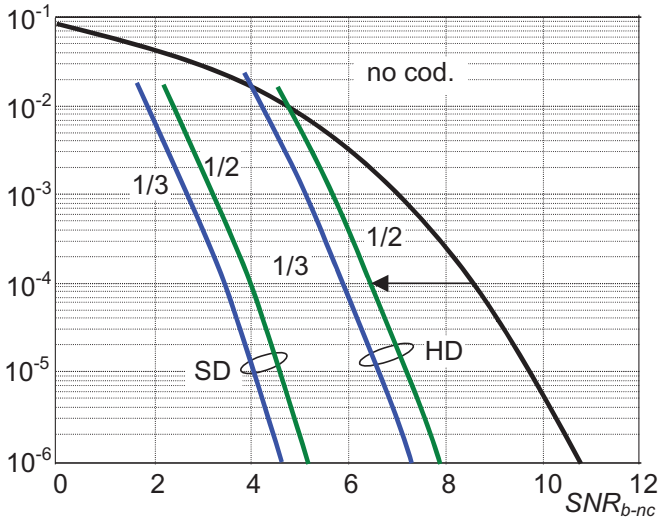


- Si la Q de un bit recibido es ≈ 0 la distancia que se suma por ese bit es ≈ 0

Decodificación de secuencias con decisión blanda

Ejemplos

- Codificadores de tasa 1/2 y 1/3
- Decodificación ML dura y blanda (SD es ~2 dB mejor)



Codificación de canal

Decodificación blanda

Concepto. Prestaciones y eficacia

Aplicación a codificación convolucional

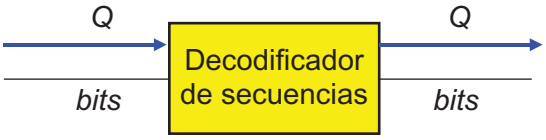
Decodificación con decisiones blandas

Decodificación con salida blanda

Decodificación iterativa

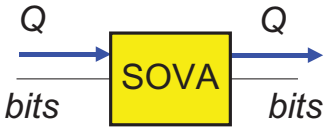
Decodificación de secuencias con decisión blanda

¿Podría entregarse también una indicación de **calidad** de los **bits de salida** del decodificador de secuencias?



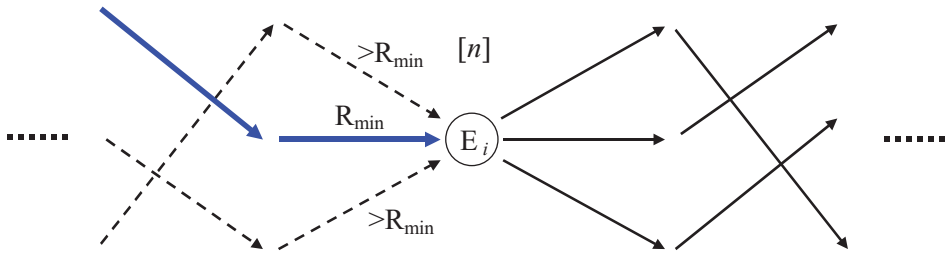
Algoritmo de Viterbi con salida blanda

***Soft Output Viterbi Algorithm* (SOVA)**



Codificación convolucional. Algoritmo de Viterbi

Busqueda del camino más corto Fundamento del algoritmo

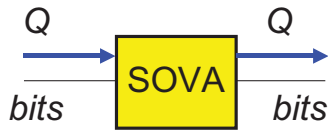


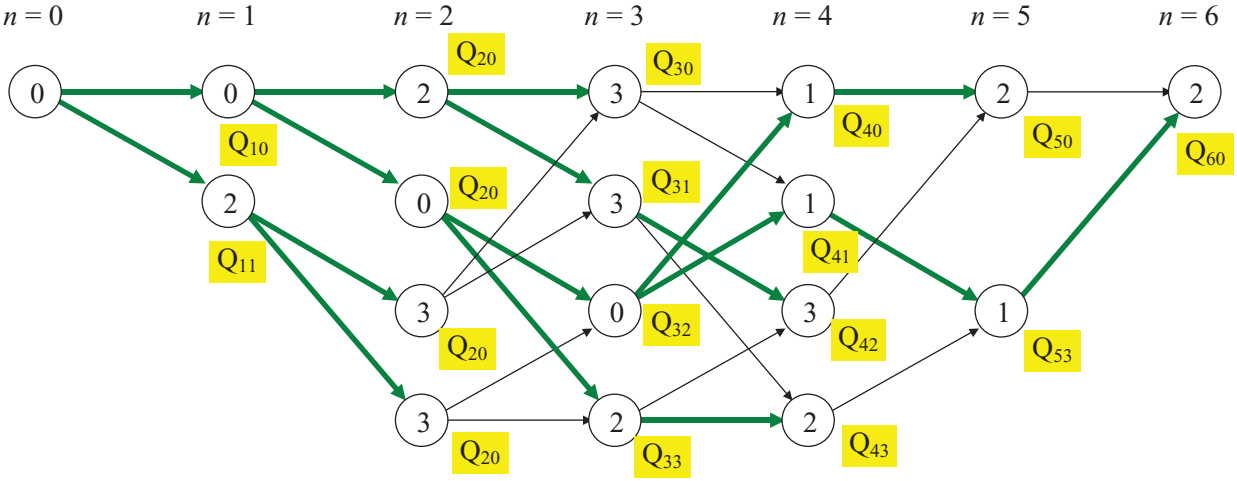
- Desechar ramas y **dejar sólo la superviviente** es una "**decisión dura**"
  (puede haber ramas entrantes de distancia acumulada menor pero similar)

- Se puede dar una calificación "Q" a esa decisión según lo clara que sea
  - si $R_{\min}$ es mucho menor que las demás: Q es grande
  - si son parecidas $Q \to 0$

Codificación convolucional

Decodificación SOVA

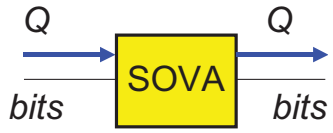


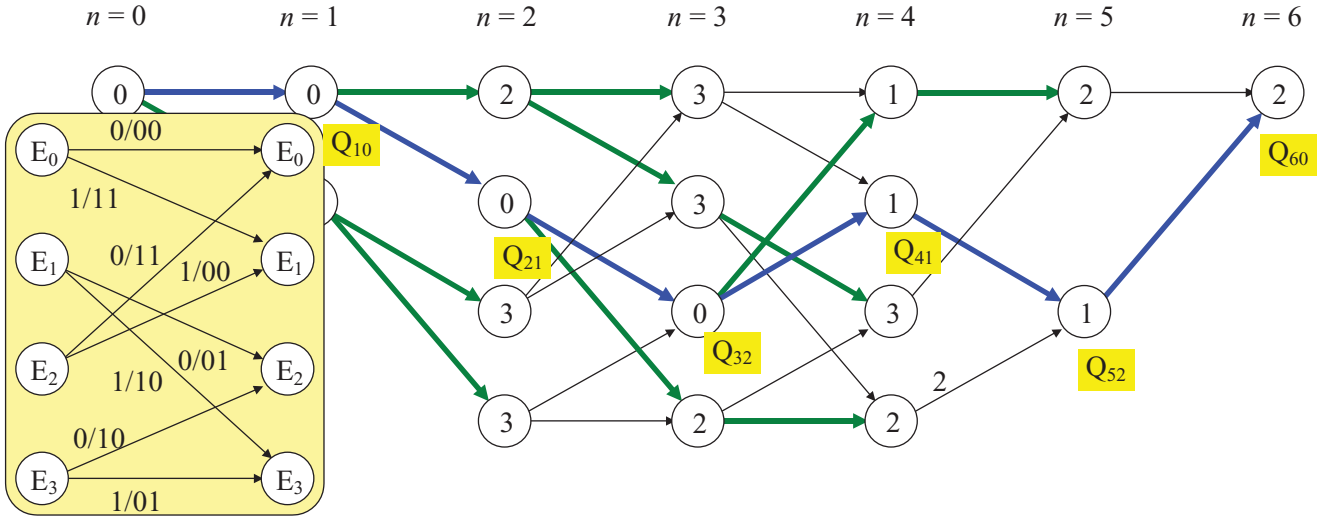


- En el recorrido **hacia adelante**
  - además de identificar las ramas supervivientes y anotar las distancias acumuladas
  - **se anota una calidad** Q de la decisión tomada (selección de superviviente)

Codificación convolucional

Decodificación SOVA

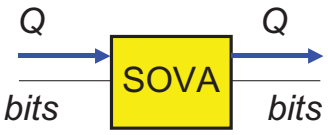


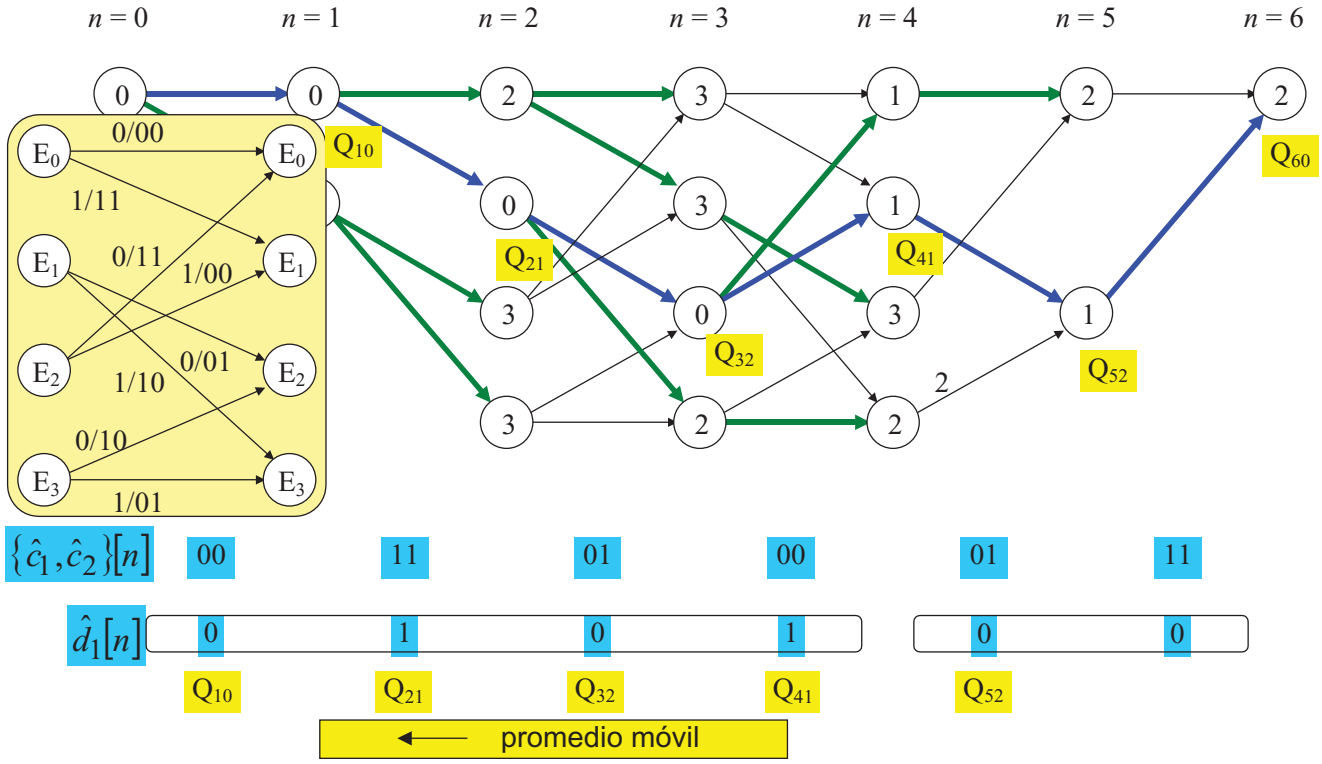


- En el recorrido **hacia atrás**
  - además de obtener la secuencia $\{c_1,c_2\}[n]$ decidida
  - y decodificar las secuencia de bits de mensaje $d_1[n]$
  - **se indica una calidad** asociada a cada bit $d_1[n]$

Codificación convolucional

Decodificación SOVA





Codificación de canal

Decodificación blanda

Concepto. Prestaciones y eficacia

Aplicación a codificación convolucional

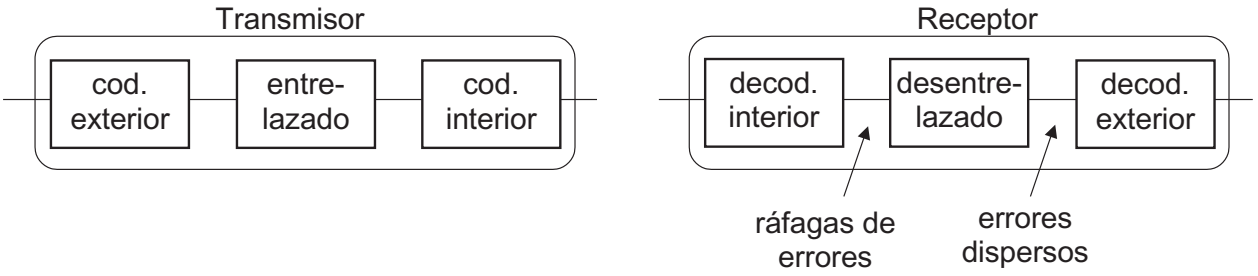
Decodificación con decisiones blandas

Decodificación con salida blanda

Decodificación iterativa

Codificación concatenada

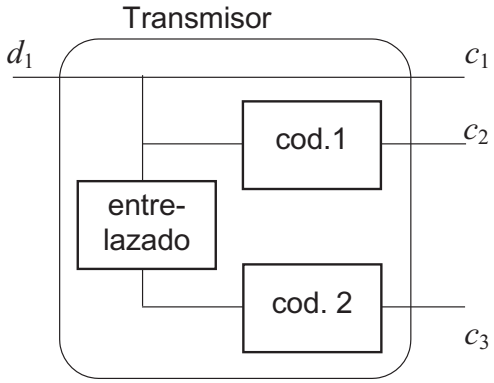
Concatenación serie



- **Dos** codificaciones de canal: **interior** y **exterior** + **entrelazado (barajado)**
- El decodificador **interior** elimina los **errores dispersos** pero no las ráfagas
- El **desentrelazador dispersa** los errores de las ráfagas
- El decodificador **exterior** elimina los errores ya dispersados

Codificación concatenada

Concatenación paralela

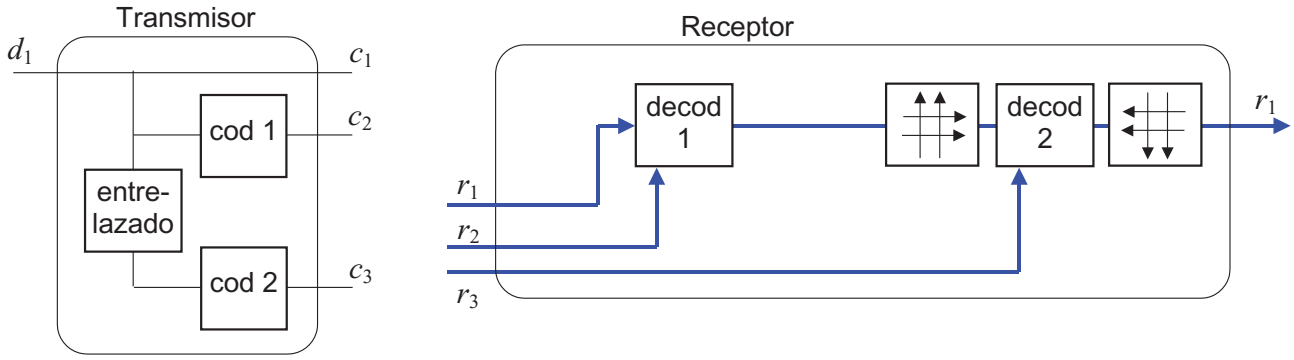


Transmisor:

- dos codificadores convolucionales **iguales sistemáticos** de tasa 1/2
- un entrelazador
- salida: el bit de entrada y los dos bits c_2, c_3 generados por cod.1 y cod.2
- $c_2 \neq c_3$ debido a entrelazado

codificación de tasa 1/3

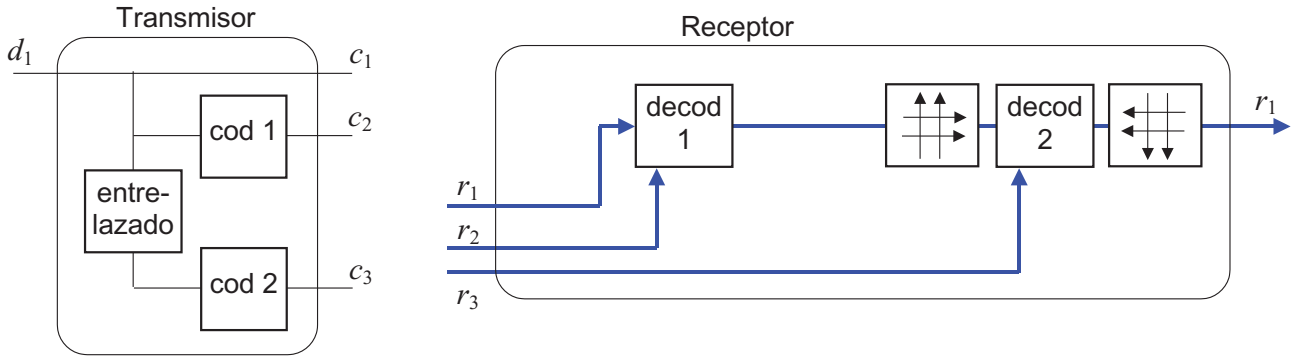
Codificación concatenada paralela



Decodificación secuencial

- El receptor recibe bits con medidas de calidad (bits "blandos")
- Los dos decodificadores son iguales
  - usan medidas blandas de distancia
  - entregan a la salida bits con medidas de calidad (bits "blandos")

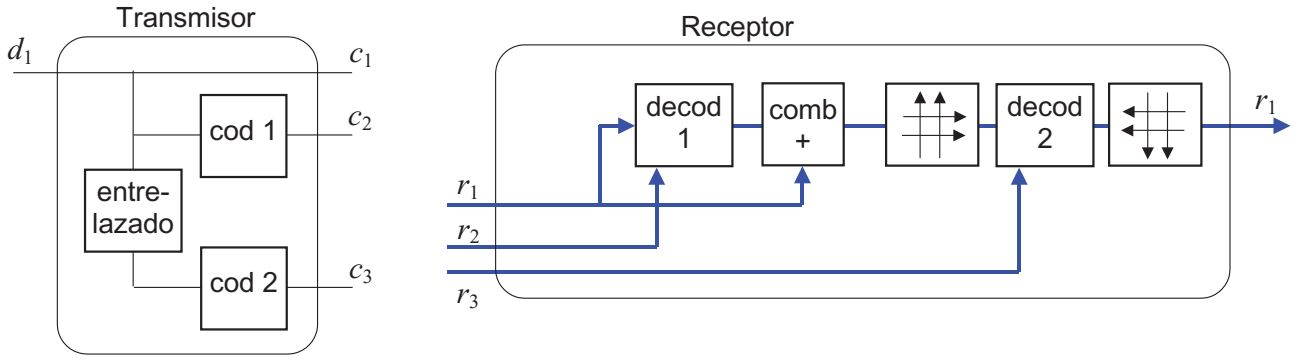
Codificación concatenada paralela



Decodificación secuencial

- **decod.1** entrega ráfagas de errores pero **identifica** esos bits como malos (Q)
- **entrelazado** **dispersa** los errores
- **decod.2** - corrige muchos de esos errores al estar ya dispersos
  - tener identificados los errores (Q) mejora la decodificación ML
- **desentrelazado** vuelve a poner los bits en su sitio para entregarlos a la salida

Codificación concatenada paralela



Decodificación secuencial: Mejora con combinación

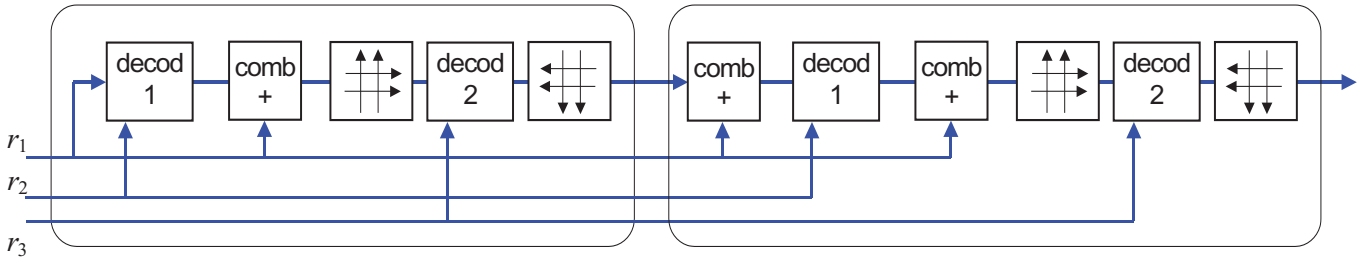
- **decod.2** usa una combinación de la entrada y la salida de decod.1
- Suma de "bits blandos" como en código de repetición (2,1)
  - se puede dar más peso a una de ellas
- Si la calidad de un bit ha mejorado en decod.1 se tendrá más en cuenta
  - la combinación proporciona mejor medida de la fiabilidad (Q)

Codificación concatenada paralela

Decodificación secuencial

Aprovechando que se han obtenido decisiones blandas

¿porqué no hacerlo de nuevo?

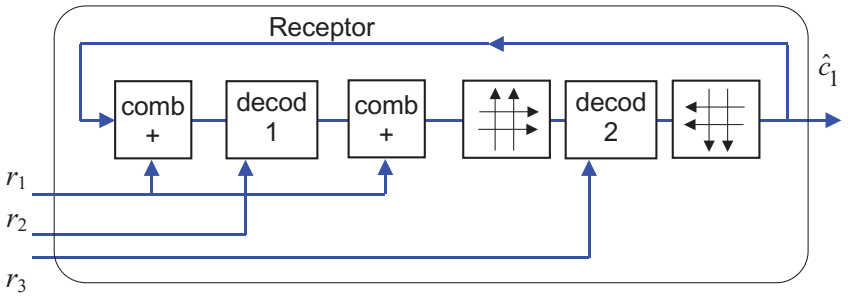


- En cada combinación, la calidad mejora

¿porqué no hacerlo de nuevo?

Codificación concatenada paralela

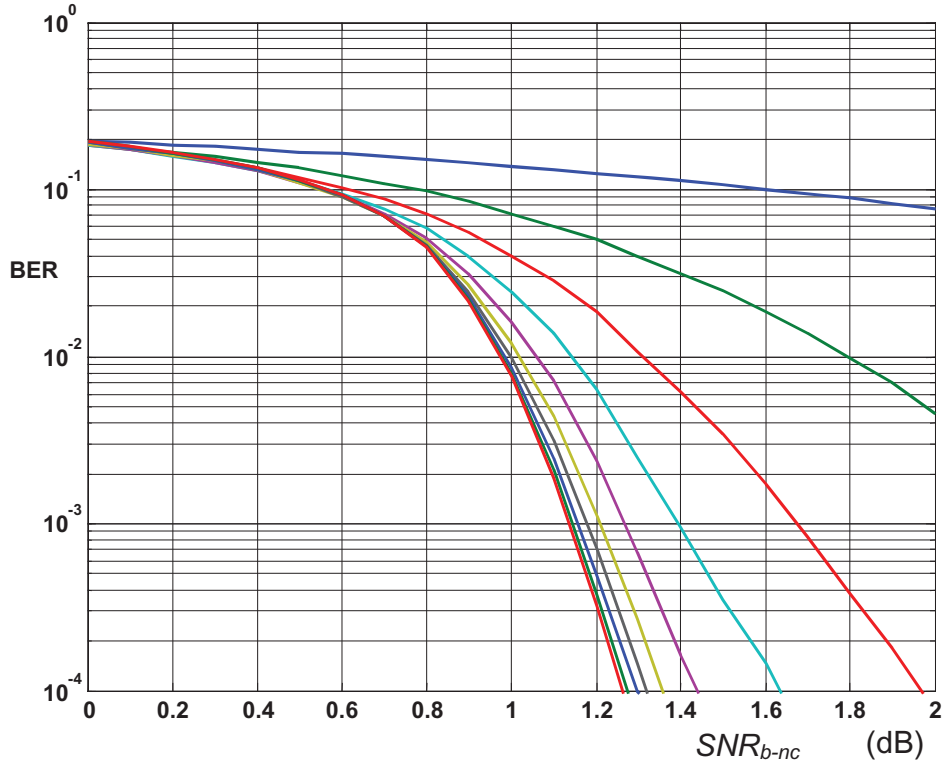
Decodificación iterativa: Turbodecodificación varias iteraciones (típico ~ 6)



- Las medidas de calidad mejoran con las combinaciones en cada iteración
- Ganancia de lazo: peso (en **comb**+) de las estimaciones frente a r_1
- Casi se alcanza el límite de Shannon

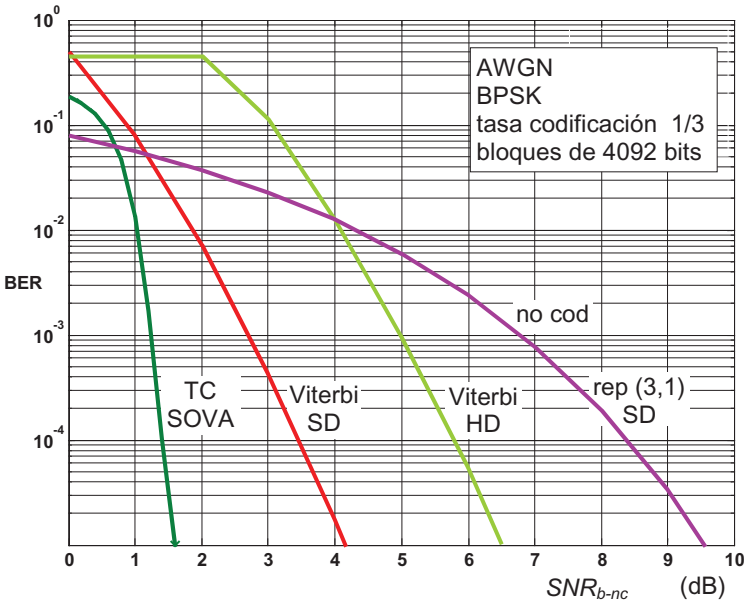
Turbodecodificación (UMTS) **6144 bits 1-10 iteraciones**

TC-SOVA iteraciones **AWGN, BPSK, tasa de codificación de 1/3**



Codificación de canal

Comparación de prestaciones



Ganancia de codificación: (10^{-3})

Viterbi HD	1,8 dB	
Viterbi SD	4,1 dB	(+2,3 dB)
TC-SOVA (6 iteraciones)	5,5 dB	(+1,4 dB)

- La codificación convolucional concatenada con decodificación iterativa puede llegar a casi 1 dB del límite de Shannon

Codificación de canal

Decodificación blanda

Concepto. Prestaciones y eficacia

Aplicación a codificación convolucional

Decodificación iterativa

Universidad de Málaga. Escuela Técnica Superior de Ingeniería de Telecomunicación
Departamento de Ingeniería de Comunicaciones
Master en Ingeniería de Telecomunicación
Ejercicios de *Técnicas de tratamiento de señal y comunicaciones*
Tema 4
José Tomás Entrambasaguas
Ed. 2 4-6-2015

4.1
Indique la capacidad de detección y corrección de errores de los siguientes códigos, de los que se conoce su distancia de Hamming:

	d_H	Número de errores que detecta	Número de errores que corrige
Caso 1	3		
Caso 2	4		
Caso 3	5		

4.2
Un codificador bloque tiene la siguiente matriz generadora:

. $$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}^T$$

Calcule su matriz generadora de paridades **P** y la matriz comprobadora de paridades $\mathbf{H}^T$ del decodificador.

4.3
Con el codificador del ejercicio 4.2, calcule la palabra código correspondiente al mensaje $\mathbf{d}=[1\ 0\ 1]^T$

4.4
Un mensaje de tres bits, $\mathbf{d} = [d_1\ d_2\ d_3]$, se ha codificado con el codificador del ejercicio 4.2 y el resultado se ha transmitido por un canal. Se ha recibido la palabra $\mathbf{r} = [0\ 1\ 1\ 1\ 1\ 1\ 0]^T$. El decodificador utiliza la siguiente tabla de síndromes:

síndrome	bit erróneo
0 0 0 0	ninguno
0 1 1 1	d_2
1 1 0 1	d_1
1 1 1 0	d_3
resto	errores en paridades

a) ¿Ha habido errores de transmisión?
b) ¿Cuál era el mensaje **d**?

4.5

Se quiere diseñar un código bloque lineal de $m = 12$ que pueda corregir todos los bloques que tengan sólo un bit erróneo.

a) calcule el número de síndromes necesarios para representar todos los patrones de error

b) calcule el número de bit de paridad necesarios para obtener esos síndromes

c) calcule el tamaño posible de los mensajes sin codificar (k)

d) calcule la tasa de codificación

4.6

Repita el ejercicio anterior si se quieren corregir todos bloques que tengan uno o dos bits erróneos.

4.7

Un codificador (5,3) genera códigos cíclicos sistemáticos mediante el polinomio generador

$g(X) = X^2 + 1$.

a) Calcule la palabra-código generada a partir del mensaje [1 1 1]

b) Se ha recibido la palabra [1 1 0 1 0] ¿Ha habido errores?

c) Se ha recibido la palabra [1 0 0 0 1] ¿Ha habido errores?

4.8

Repita el ejercicio 4.7 con ayuda de MATLAB [convierta los vectores binarios en vectores de GF(2) con la función "`gf`"

a) genere el mensaje y el polinomio generador en el cuerpo de Galois GF(2):

```
g=gf([1 0 1],1); % polinomio generador en GF(2)
d=gf([1 1 1],1); % mensaje en GF(2)
```

b) calcule la palabra código (desplazar, dividir y sumar el resto):

```
da=[d gf([0 0],1)];  % mensaje desplazado (m-k=2) hacia la izquierda
[cociente,resto]=deconv(da,g); % division
c=da+resto;          % suma del resto
```

c) compruebe que la palabra generada es del código calculando su síndrome (s):

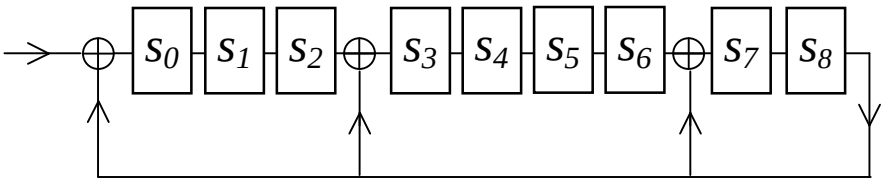
```
[cociente,s]=deconv(c,g);
```

d) compruebe si hay errores en las palabras recibidas [1 1 0 1 0] y [1 0 0 0 1] calculando los síndromes (s):

```
r1=gf([1 1 0 1 0],1);
[cociente,s1]=deconv(r1,g);
r2=gf([1 0 0 0 1],1);
[cociente,s2]=deconv(r2,g);
```

4.9

El circuito de la figura es una realización de un codificador cíclico ¿Cuál es su polinomio generador?



4.10
Dibuje los circuitos que realicen los codificadores cíclicos de los siguientes polinomios generadores:

a) $x^{16} + x^{10} + x^8 + x^7 + x^3 + 1$

b) $x^{16} + x^{15} + x^{13} + x^7 + x^4 + x^2 + 1$

4.11
Obtenga con MATLAB las tablas de multiplicar y de sumar en GF(2^3) y compruébelas con las mostradas en clase (están en el campus virtual)
[En MATLAB se puede definir un cuerpo de Galois con sus correspondientes operaciones]
[Puede usar el siguiente programa]

```
X=gf(0:7,3)  % define un vector fila de elementos GF(2^3)
             % con los 8 valores ordenados
             % las operaciones (*,+) que se hagan con X serán en GF(2^3)

tablax=X'*X  % calcula la tabla de multiplicar

onesGF=gf(ones(1,8),3); % vector fila de ocho "1"
Xrep = X'*onesGF;       % matriz 8x8 rectangular con columnas repetidas

tablasum=Xrep'+Xrep;    % calcula la tabla de sumar
```

4.12
Rellene la siguiente tabla, indicando para cada uno de los códigos Reed-Solomon los datos que se indican

Código madre		(255,251)	(255,239)	(255, 239)	(255,251)	(255, 223)	(255,251)
Acortado a:		NO	NO	(204,188)	(32,28)	NO	(28,24)
Tamaño de bloque sin codificar	octetos						
	bits						
Tamaño de bloque codificado	octetos						
	bits						
Número de octetos erróneos que corrige							
Número de octetos erróneos que detecta							
Número de octetos borrados que recupera							
Tasa de codificación							

4.13
Un codificador convolucional sistemático de tasa 1/3 está definido por las siguientes ecuaciones:

$$c_1[n] = d_1[n]$$
$$c_2[n] = d_1[n]+d_1[n\text{-}2]$$
$$c_3[n] = d_1[n]+d_1[n\text{-}1] +d_1[n\text{-}2]$$

El codificador se inicia siempre con $d_1[n\text{-}1] = d_1[n\text{-}2] = 0$

a) ¿Cuál es su longitud de restricción?

b) Dibuje su diagrama de estados

4.14
Calcule la secuencia de salida del codificador del ejercicio 4.13 para las siguientes secuencias de entrada: a) 1 1 0 0 0; b) 1 0 1 1 1

4.15
En un sistema que utiliza el codificador del ejercicio 4.13 se ha recibido la secuencia

$$\mathbf{r} = \{c_1\ c_2\ c_3\}[n] = \{000\}\ \{111\}\ \{001\}\ \{101\}\ \{001\}\ \{011\}.$$

Calcule la secuencia más probable transmitida que obtendría un decodificador de Viterbi con decisión dura:

a) Dibuje el grafo de evolución del codificador

b) Anote en cada rama las distancias de los tres bits de salida correspondientes a esa rama a los tres bits recibidos en ese instante.

c) Realice el recorrido hacia adelante calculando distancias acumuladas y descartando ramas (márquelas de alguna forma) (es posible que en algún caso no pueda descartarse por dar medidas acumuladas iguales; en ese caso no las descarte pero anote la medida acumulada).

d) Realice el recorrido hacia atrás a partir del estado al que se llegue con menor distancia acumulada.

4.16
El sistema del ejercicio anterior se utiliza con un patrón de perforación de longitud 9, en el que se perfora el 5º bit. Se ha recibido la secuencia:

$$\mathbf{r} = \{000\}\ \{11\}\ \{001\}\ \{101\}\ \{01\}\ \{011\}.$$

Calcule la secuencia más probable transmitida que obtendría un decodificador de Viterbi con decisión dura.

4.17
Un sistema de transmisión de tasa binaria ajustable usa un codificador convolucional de tasa 1/3 y un conjunto de patrones de perforación de diferentes periodos y bits perforados por periodo. Para cada uno de los patrones de perforación calcule la tasa de codificación resultante y la velocidad binaria que se obtiene si los bits codificados se transmiten a 3 Mbit/s.

Patrón de perforación	Periodo de perforación p	bits perforados en un periodo p q	tasa de codificación	velocidad binaria (kbit/s)
1	18	0		
2	18	2		
3	18	4		
4	18	6		
5	18	8		
6	18	9		
7	18	10		
8	18	11		
9	30	19		
10	48	31		

Universidad de Málaga. Escuela Técnica Superior de Ingeniería de Telecomunicación
Departamento de Ingeniería de Comunicaciones

Grado en Ingeniería de Sistemas de Telecomunicación

Ejercicios de *Técnicas de tratamiento de señal y comunicaciones*

Tema 4

Soluciones a los ejercicios

José Tomás Entrambasaguas

Ed. 1 27-5-2017

4.1

	d_H	errores que detecta	errores que corrige
Caso 1	3	2	1
Caso 2	4	3	1
Caso 3	5	4	2

4.2

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \qquad \mathbf{H}t = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

4.3

$\mathbf{c} = [1\ 0\ 1\ 0\ 0\ 1\ 1]^T$

4.4

$\mathbf{s} = \mathbf{H}t * \mathbf{r} = [0\ 1\ 1\ 1]^T \Rightarrow$ error error en 2º bit $\Rightarrow$ $\mathbf{d}=[0\ 0\ 1]^T$

4.5

a) $\binom{12}{0} + \binom{12}{1} = 13$

b) 4

c) $k = 12 - 4 = 8$

d) 8/12 = 2/3

4.6

a) $\binom{12}{0} + \binom{12}{1} + \binom{12}{2} = 1 + 12 + 66 = 79$

b) 7

c) $k = 12 - 7 = 5$

d) 5/12 = 2/3

4.7

a) resto de $[1\ 1\ 1\ 0\ 0] \div [1\ 0\ 1]$ es $[1\ 0]$ $\Rightarrow$ $\mathbf{c} = [1\ 1\ 1\ 1\ 0]^T$

b) resto de $[1\ 1\ 0\ 1\ 0] \div [1\ 0\ 1]$ es $[0\ 1]$ $\Rightarrow$ hay errores

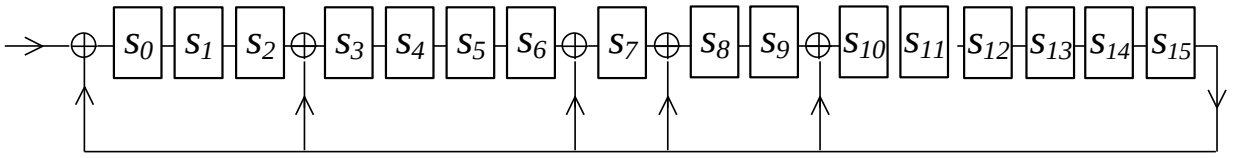
c) resto de $[1\ 0\ 0\ 0\ 1] \div [1\ 0\ 1]$ es $[0\ 0]$ $\Rightarrow$ no hay errores

4.9

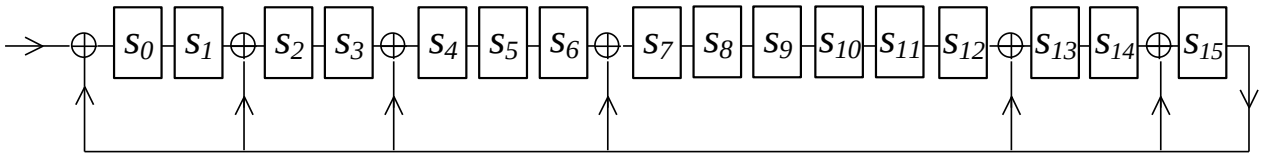
$g(X) = X^9 + X^7 + X^3 + 1$.

4.10

a) $x^{16} + x^{10} + x^8 + x^7 + x^3 + 1$



b) $x^{16} + x^{15} + x^{13} + x^7 + x^4 + x^2 + 1$



4.12

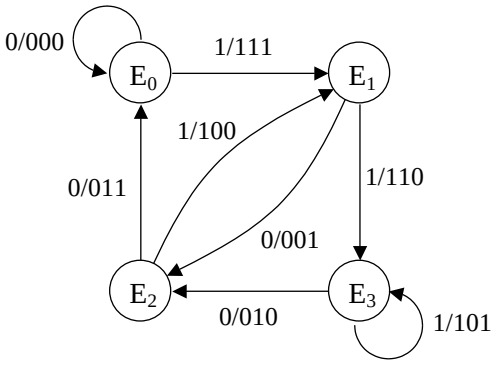
Código madre		(255,251)	(255,239)	(255, 239)	(255,251)	(255, 223)	(255,251)
Acortado a:		NO	NO	(204,188)	(32,28)	NO	(28,24)
Tamaño de bloque sin codificar	octetos	251	239	188	28	223	24
	bits	2008	1912	1504	224	1784	192
Tamaño de bloque codificado	octetos	255	255	204	32	255	28
	bits	2040	2040	1632	256	2040	224
Número de octetos erróneos que corrige		2	8	8	2	16	2
Número de octetos erróneos que detecta		4	16	16	4	32	4
Número de octetos borrados que recupera		4	16	16	4	32	4
Tasa de codificación		0,98	0,94	0,92	7/8	0,87	6/7

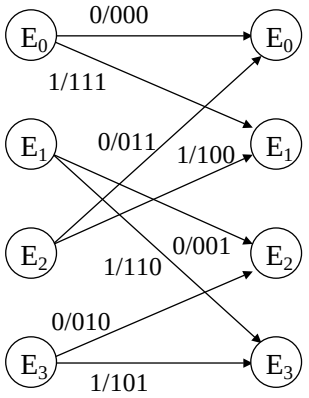
4.13

a) Longitud de restricción: $K = 3$

b)

Estado	$d_1[n\text{-}1]$	$d_1[n\text{-}2]$
E_0	0	0
E_1	1	0
E_2	0	1
E_3	1	1





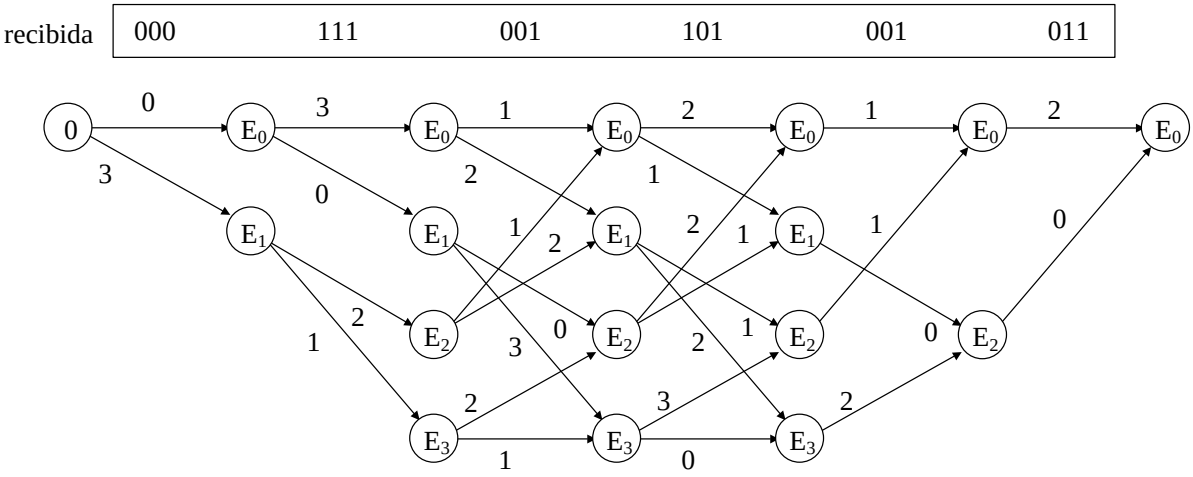
4.14

a) **d**[*n*] = 1 1 0 0 0 -> **c**[*n*] {111} {110} {010} {011} {000}

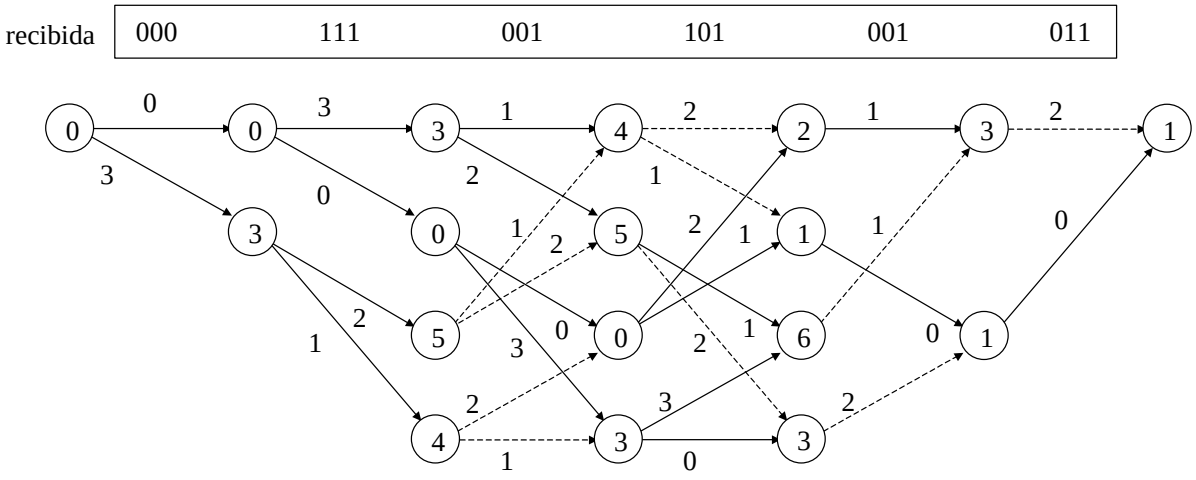
b) **d**[*n*] = 1 0 1 0 0 -> **c**[*n*] {111} {001} {100} {001} {011}

4.15

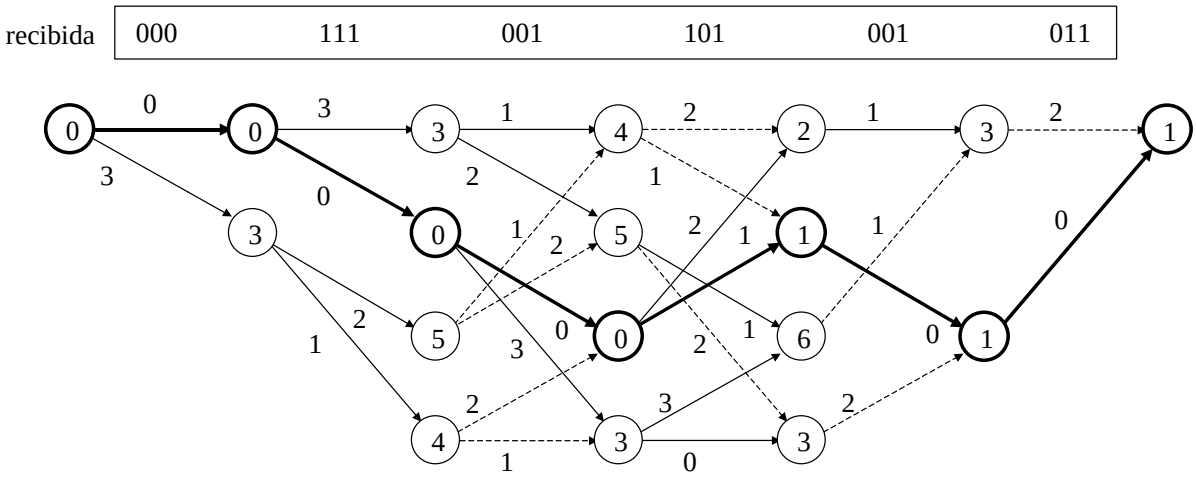
a), b) Grafo con distancias de rama



c) Recorrido hacia adelante



d) Recorrido hacia atras



Secuencia de estados

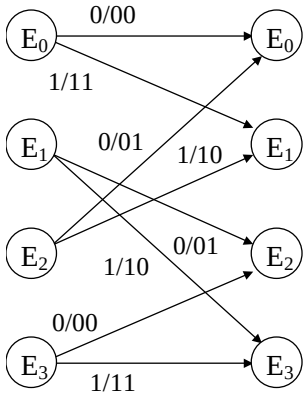
E_0 E_0 E_1 E_2 E_1 E_2 E_0

Secuencia de menor distancia a la recibida

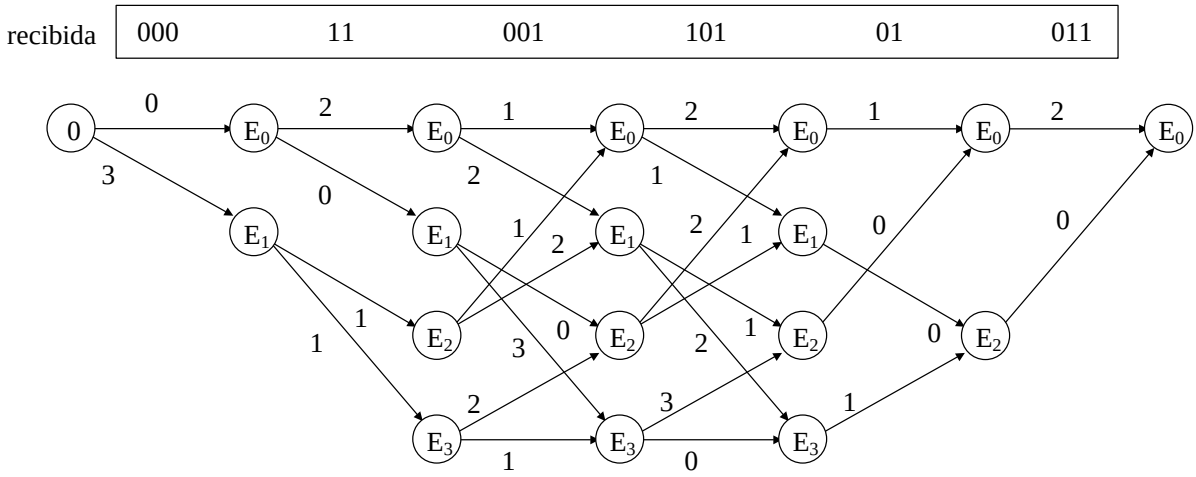
{000} {111} {001} {100} {001} {011}

4.16

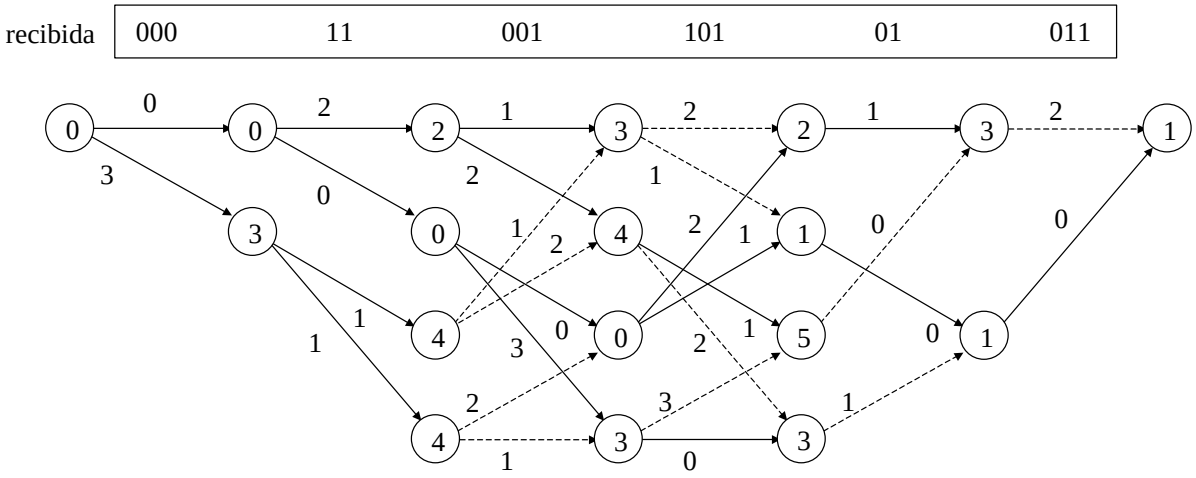
Salidas cuando se perfora el bit central de los tres:



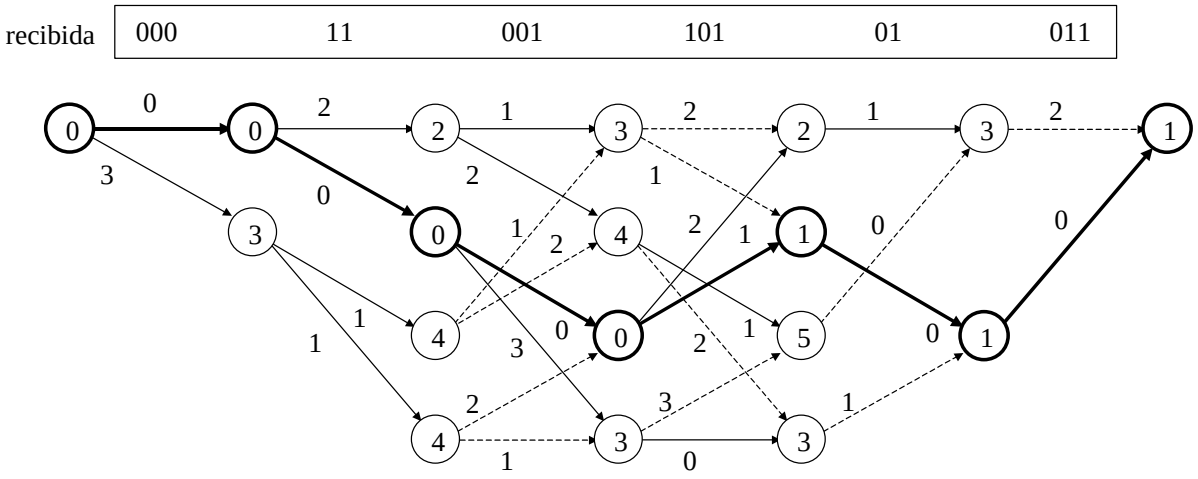
a), b) Grafo con distancias de rama
(igual que en 4.15 excepto las etapas con bis perforados)



c) Recorrido hacia adelante



d) Recorrido hacia atras



Secuencia de estados

E_0 E_0 E_1 E_2 E_1 E_2 E_0

Secuencia de menor distancia a la recibida

{000} {11} {001} {100} {01} {011}

4.17

Periodo de perforación (p)	Bits perforados en un período (q)	Bits transmitidos en un período (p-q)	p/(p-q)	tasa de codificación 1/3 x p/(p-q)		velocidad binaria (kbit/s)
18	0	18	1,00	0,33	1/3	333
18	2	16	1,13	0,38	3/8	375
18	4	14	1,29	0,43	3/7	429
18	6	12	1,50	0,50	1/2	500
18	8	10	1,80	0,60	3/5	600
18	9	9	2,00	0,67	2/3	667
18	10	8	2,25	0,75	3/4	750
18	11	7	2,57	0,86	6/7	857
30	19	11	2,73	0,91	10/11	909
48	31	17	2,82	0,94	16/17	941