Universidad de Granada

decsai.ugr.es

Teoría de la Información y la Codificación

Grado en Ingeniería Informática

Tema 4.- Información en canales con ruido.

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial

Índice

1. Utilidad de la detección de errores
2. El modelo de comunicaciones con ruido
3. Fundamentos para transmisión con ruido
4. Bits de paridad
5. Códigos de verificación de cuenta fija
6. Códigos de redundancia cíclica

1. **Utilidad de la detección de errores**
2. El modelo de comunicaciones con ruido
3. Fundamentos para transmisión con ruido
4. Bits de paridad
5. Códigos de verificación de cuenta fija
6. Códigos de redundancia cíclica

– **¿Qué es el ruido en la transmisión de datos por un canal?**

- Ruido en una señal telefónica.
- Errores físicos en un disco.
- Tinta *corrida* en un papel.
- … ¡Mala letra al escribir, incluso!
- Un acento muy cerrado de algún sitio, difícil de entender.

A: Sí, señor. Me llamo Ama.
B: Eso, Ana he dicho antes.
A: No, señor, es AMA, con “M”.
B: Eso es, con “N”.
A: No, con “N” no. Con “M”.
B: ¿Con “N” de “Navarra”?.
A: No, con “M” de MENDRUGO.

– ¿Qué es el ruido en la transmisión de datos por un canal?

- Ruido en una señal telefónica.
- Errores físicos en un disco.
- Tinta *corrida* en un papel.
- … ¡Mala letra al escribir, incluso!
- Un acento muy cerrado de algún sitio, difícil de entender.

A: Sí, señor. Me llamo Ama.
B: Eso, Ana he dicho antes.
A: No, señor, es AMA, con “M”.
B: Eso es, con “N”.
A: No, con “N” no. Con “M”.
B: ¿Con “N” de “Navarra”?.
A: No, con “M” de MENDRUGO.

La redundancia: Papel esencial en la detección y corrección de errores.

– **Ejemplos de redundancia para detección de errores en aplicaciones cotidianas:**

  – La letra del DNI → Se utiliza para verificar que el NIF es correcto.

  – Se divide el número entre 23 y el resto se sustituye por una letra que se determina por inspección mediante la siguiente tabla:

RESTO	0	1	2	3	4	5	6	7	8	9	10	11
LETRA	T	R	W	A	G	M	Y	F	P	D	X	B

RESTO	12	13	14	15	16	17	18	19	20	21	22
LETRA	N	J	Z	S	Q	V	H	L	C	K	E

- **Ejemplos de redundancia para detección de errores en aplicaciones cotidianas:**

  - El código de Cuenta Corriente.
    - Los primeros cuatro dígitos son el Código de la Entidad, que coincide con el Número de Registro de Entidades del Banco de España (NRBE).
    - Los siguientes cuatro dígitos identifican la oficina.
    - Los siguientes dos dígitos son los llamados dígitos de control, que sirven para validar el CCC.
    - Los últimos diez dígitos identifican unívocamente la cuenta.

  - Los dígitos situados en las posiciones novena y décima se generan a partir de los demás dígitos del CCC, permitiendo comprobar la validez del mismo.

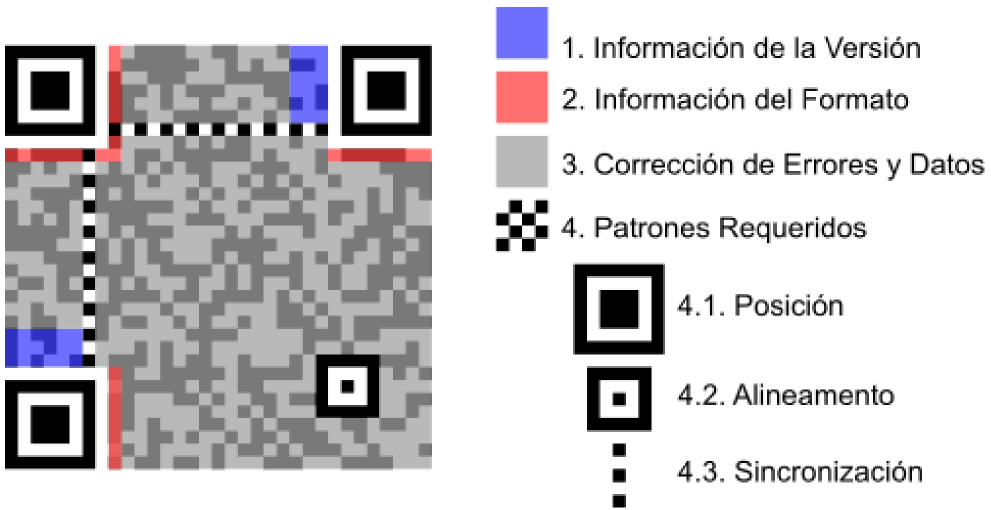
– **Ejemplos de redundancia para detección de errores en aplicaciones cotidianas:**

  – Los códigos de barras de los productos que compramos.

– ¿Porqué es útil la detección de errores?

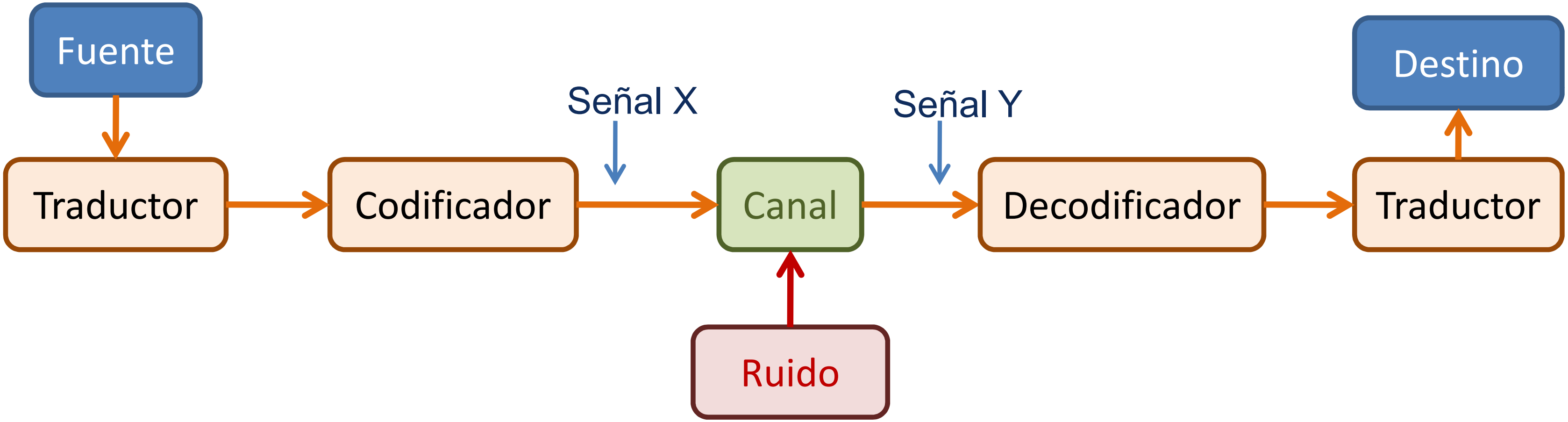
- Para asegurar que la transmisión del código, por el canal que se produzca, es válido cuando se recibe. Informar de la existencia del error para tomar las medidas oportunas (alarmas, notificaciones, …).

- Para efectuar una detección de errores, debemos introducir redundancia en el código.



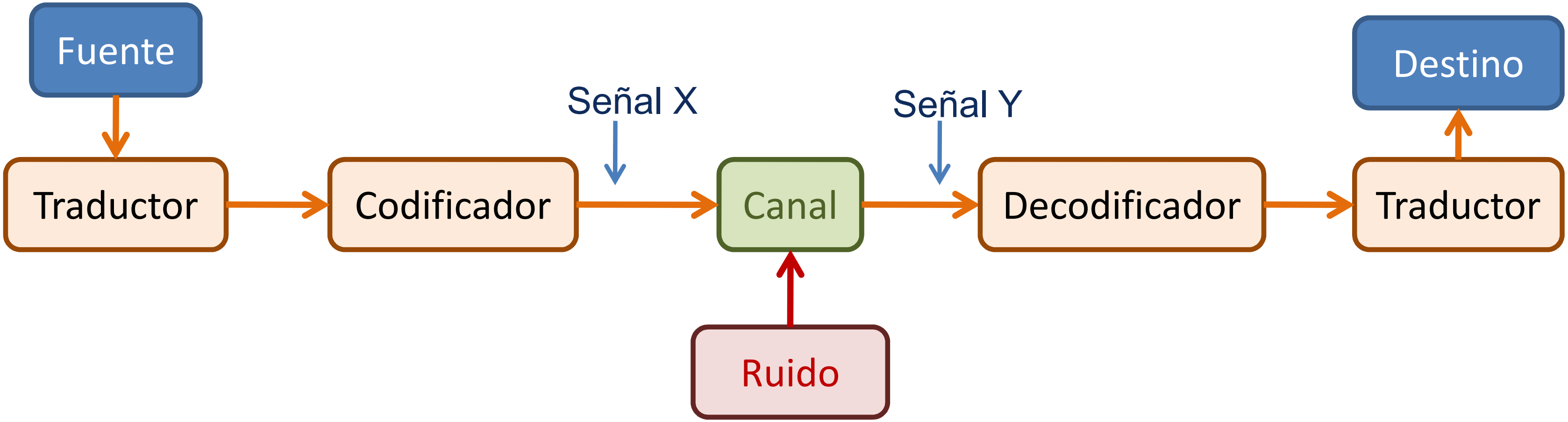
1. Utilidad de la detección de errores
2. **El modelo de comunicaciones con ruido**
3. Fundamentos para transmisión con ruido
4. Bits de paridad
5. Códigos de verificación de cuenta fija
6. Códigos de redundancia cíclica

– **Base para la codificación en un canal con ruido:**



– Cuando el emisor envía la información al receptor, lo hace a través de un canal.
– Este canal puede verse afectado por alteraciones, perturbaciones o interferencias de entidades externas (o del propio canal), que hacen que la señal Y recibida en el receptor sea diferente a la señal X enviada por el emisor.
– Como desconocemos el fenómeno que provoca las perturbaciones, entenderemos que estas **tienen un carácter aleatorio → Ruido.**

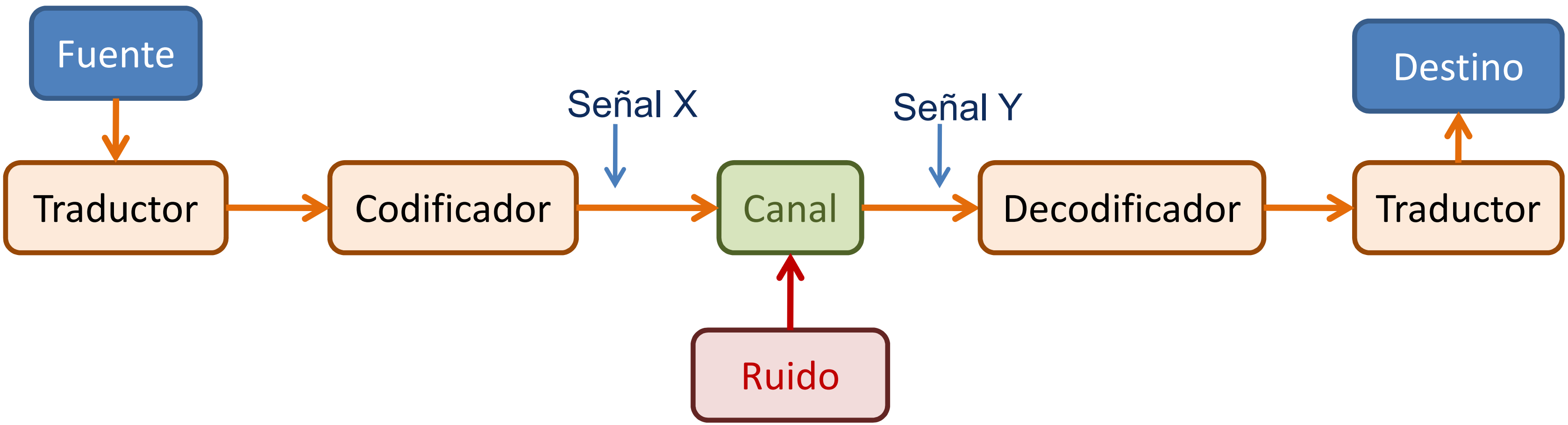
– **Base para la codificación en un canal con ruido:**



– En el próximo tema, nos interesará **comprobar la veracidad de la señal y, si ha sufrido alteraciones, reconstruirla**.
– En este caso, la teoría de la probabilidad nos ayuda a modelar esta situación a través de las probabilidades condicionadas:

  – **P(X|Y)** → Sabiendo que he recibido **Y**, ¿Qué probabilidad hay de que el emisor haya enviado **X**?

– **Base para la codificación en un canal con ruido:**



– En este tema, nos interesará **comprobar la veracidad de la señal**. Pera ello, hay que **detectar que se ha producido una perturbación en el canal** introduciendo redundancia en la información.

– La Teoría de la Probabilidad también jugará un papel importante?:

¿Qué probabilidad hay de que un símbolo se vea alterado? → **A mayor probabilidad, mayor redundancia deberemos incluir en el código para detectar el error.**

- **Tipos de errores:**

  - **Aislados:** Los bits afectados son contiguos a bits correctos.
    - **Simples**: Un único bit del código se ve afectado por el error
    - **Múltiples**: Varios bits del código se ven afectados por el error.

  - **Ejemplo: Cadena inicial**

0	0	1	0	1	1	1	0	1	0

  - **Error aislado simple:**

0	0	1	0	1	1	1	1	1	0

  - **Error aislado múltiple:**

1	0	0	0	1	1	1	0	0	0

– Tipos de errores:

– **Ráfagas de errores:** Secuencias de bits erróneos.

– Ejemplo: Cadena inicial

0	0	1	0	1	1	1	0	1	0

– **Ráfaga de errores:**

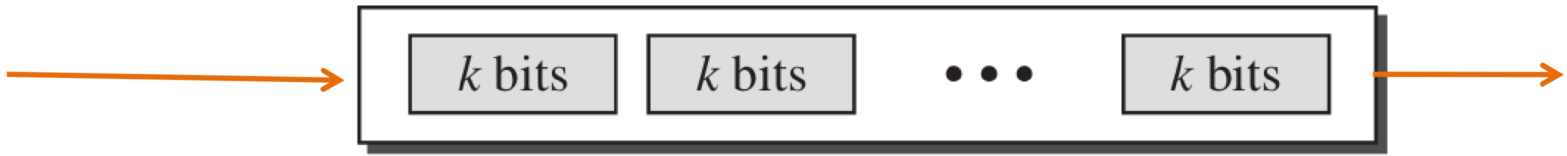
0	0	1	1	0	0	0	1	1	0

– **Supondremos, por simplicidad que los códigos a enviar son uniformes.**

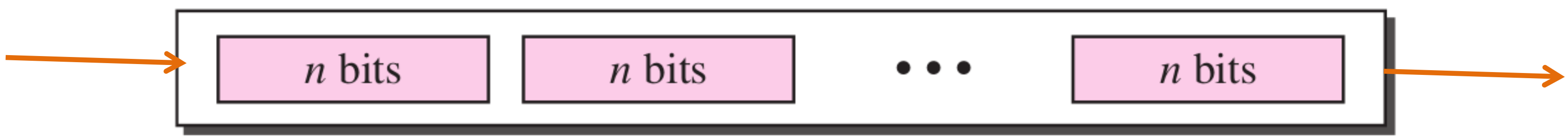
– **Codificación por bloques:**

  – **Tenemos un código con M palabras, $M<=2^k$**

  – La codificación por bloques envía el mensaje como secuencias de bloques de *k bits.*

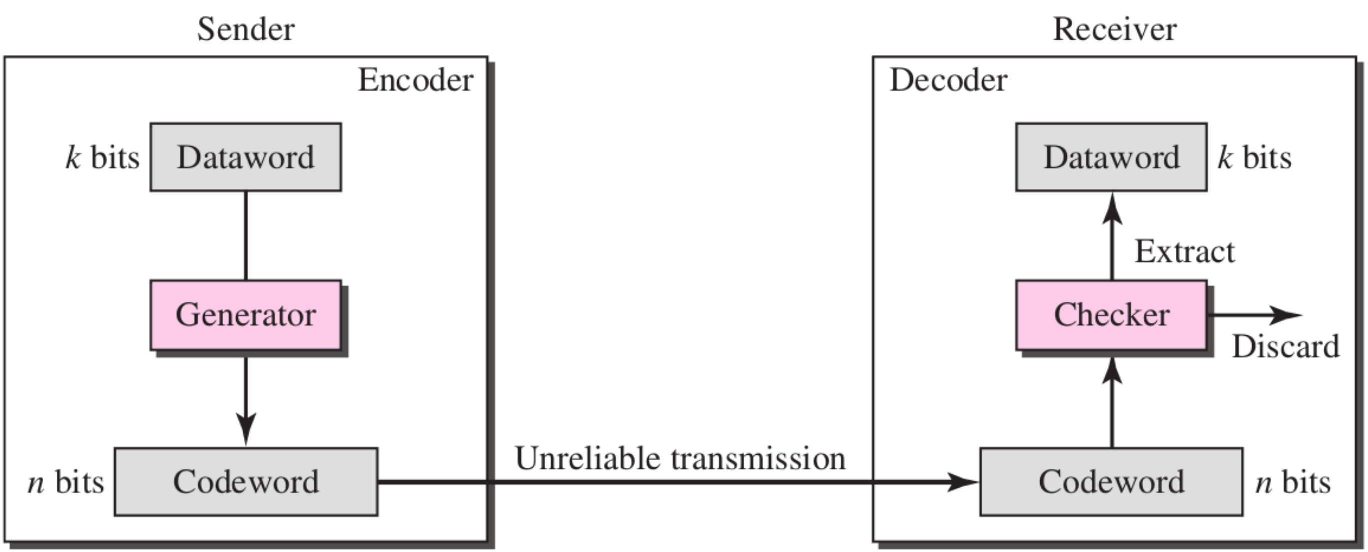


  – Como estamos en canales con ruido, a los *k* bits del código habrá que introducir *r* bits de redundancia, emitiendo códigos de ***n=k+r* bits.**



– Esquema de transmisión:

- **Tenemos un código con M palabras, M<=2^k** (k bits)**.**
- Se le añade la redundancia **r** hasta completar los **n=k+r** bits.
- ***Se envía el bloque, y se recibe en el receptor.***
- El receptor **comprueba la validez del mensaje,**
- **y lo decodifica en el código original de *k* bits.**



– **Matriz de codificación: Generalizamos y suponemos que el alfabeto de la fuente y del destino pueden no ser el mismo.**

  – **Llamamos matriz de codificación a la matriz que representa a las probabilidades de recibir un símbolo (salida del canal) y su asociación con el símbolo enviado (entrada del canal).**

  – **Ejemplo: Supongamos que se envía por un canal valores 'A', 'B', C', pero que un receptor sólo es capaz de recibir símbolos ''1" y "2".**

  – La matriz de codificación indica cuál es la asociación de **lo que recibe el receptor** con respecto a **lo que envía el emisor.**

	1	2
A	0	1
B	1	0
C	1	0

  – **Si en envía 'A', el receptor percibirá un 2. Si se envía tanto 'B' como 'C', el receptor percibirá un 1.**

- Matriz de codificación: Generalizamos y suponemos que el alfabeto de la fuente y del destino pueden no ser el mismo.

  - Ante la existencia de ruido, la matriz de codificación contiene las probabilidades de recibir un símbolo cuando el emisor ha enviado algún otro.

  - Ejemplo:

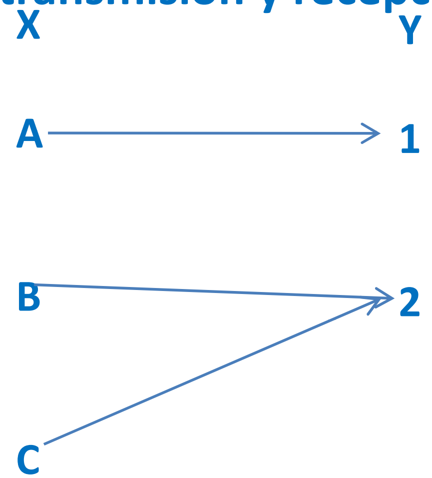
	1	2
A	0.05	0.95
B	1	0
C	0,8	0,2

  - Para cada símbolo de entrada al canal, su fila debe sumar 1 (son probabilidades asociadas a la recepción de ese símbolo).

  - P(Y|X)= Probabilidad de que el receptor perciba “Y” cuando el emisor envíe X.
  - P(‘1’|‘A’): Probabilidad de que se perciba ‘1’ cuando se envíe ‘A’.

- **Modelos de canales con ruido:**

  - **Canales deterministas:** La entrada determina unívocamente la salida.
    - **Diagrama de transmisión y recepción de símbolos:**

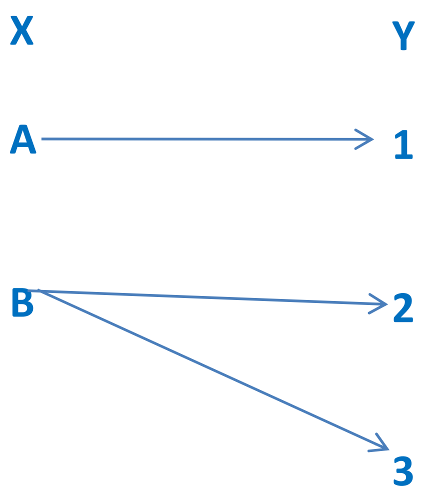


  - En estos casos, es posible que un mismo símbolo de **salida** se asocie a varios símbolos de **entrada**.

– **Modelos de canales con ruido:**

– **Canales sin pérdida:** Al conocer la salida, se conoce unívocamente la entrada.

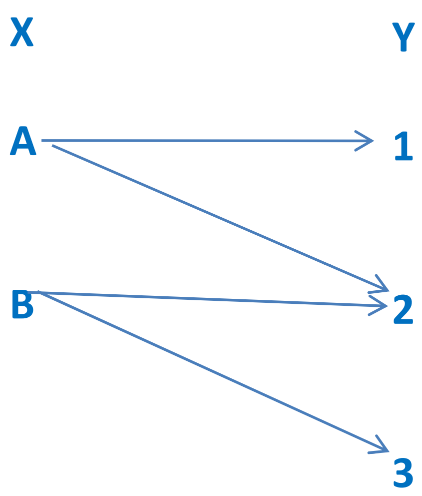
– **Diagrama de transmisión y recepción de símbolos:**



– En estos casos, es posible que un mismo símbolo de **entrada** se asocie a varios símbolos de **salida**.

– **Modelos de canales con ruido:**

– **Canales sin ruido:** Simultáneamente determinista y sin pérdida.
  – **Diagrama de transmisión y recepción de símbolos:**



– Contiene las propiedades de los canales deterministas y de canales sin pérdida.

- **Modelos de canales con ruido:**
  - **Canales simétricos:** Todas las filas (columnas) de la matriz de codificación contienen los mismos elementos, aunque en distinto orden.
  - **Ejemplo: Canal binario simétrico (BSC). Alfabeto= {0, 1}**

	0	1
0	0.05	0.95
1	0.95	0.05

– Modelos de canales con ruido:

- **Canales inútiles:** Todas las celdas de la matriz de codificación contienen los mismos valores.

- **Ejemplo: Canal inútil. Alfabeto de fuente= {1, 2, 3}, Alfabeto de destino= {0,1}**

	0	1
1	0.5	0.5
2	0.5	0.5
3	0.5	0.5

- **Se denominan canales inútiles porque no se puede enviar ni recibir nada con fiabilidad suficiente como para decodificarlo.**

- Ejercicio: Supongamos un alfabeto de fuente {1, 2, 3} y de destino {0, 1}.

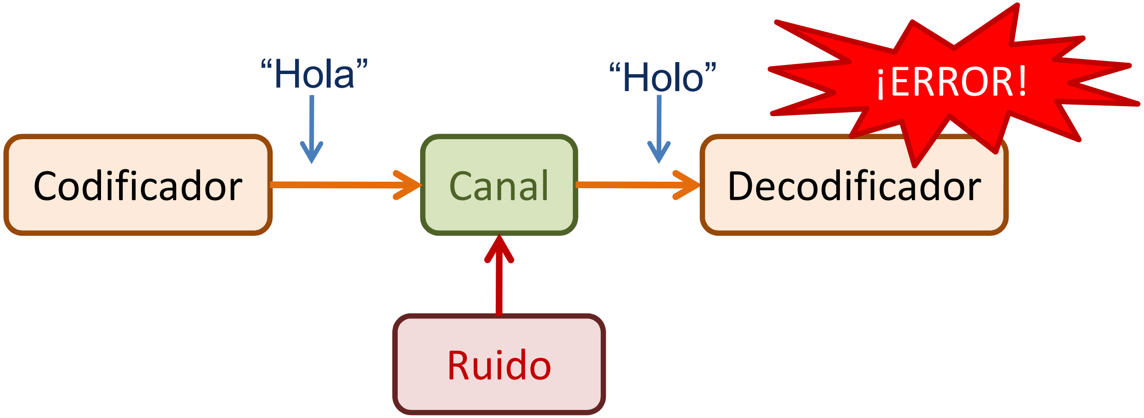
  - Dibujar, atendiendo a la siguiente tabla, el diagrama de transmisión y recepción de símbolos.

  - ¿Qué tipo/s de canal se puede/n asociar al esquema?

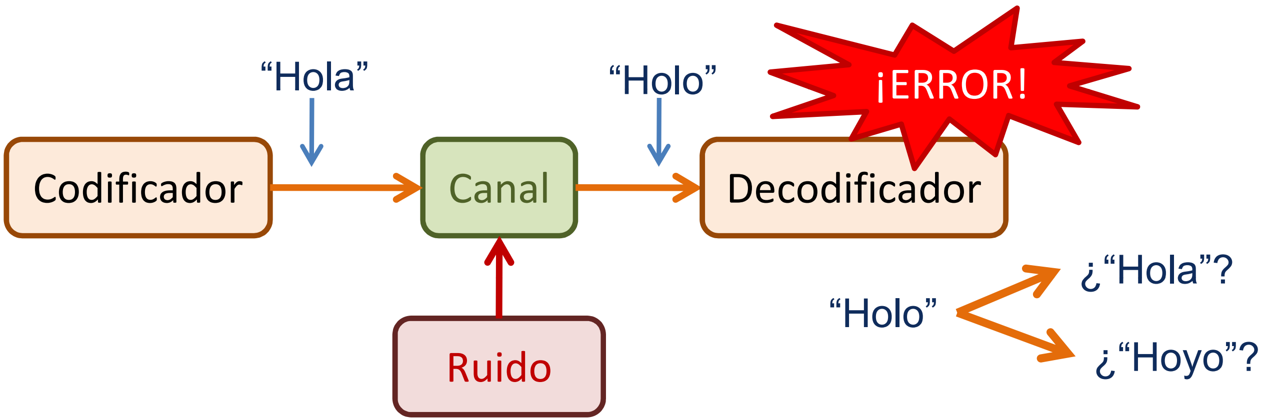
	0	1
1	0.1	0.9
2	0.9	0.1
3	0.9	0.1

1. Utilidad de la detección de errores
2. El modelo de comunicaciones con ruido
3. **Fundamentos para transmisión con ruido**
4. Bits de paridad
5. Códigos de verificación de cuenta fija
6. Códigos de redundancia cíclica

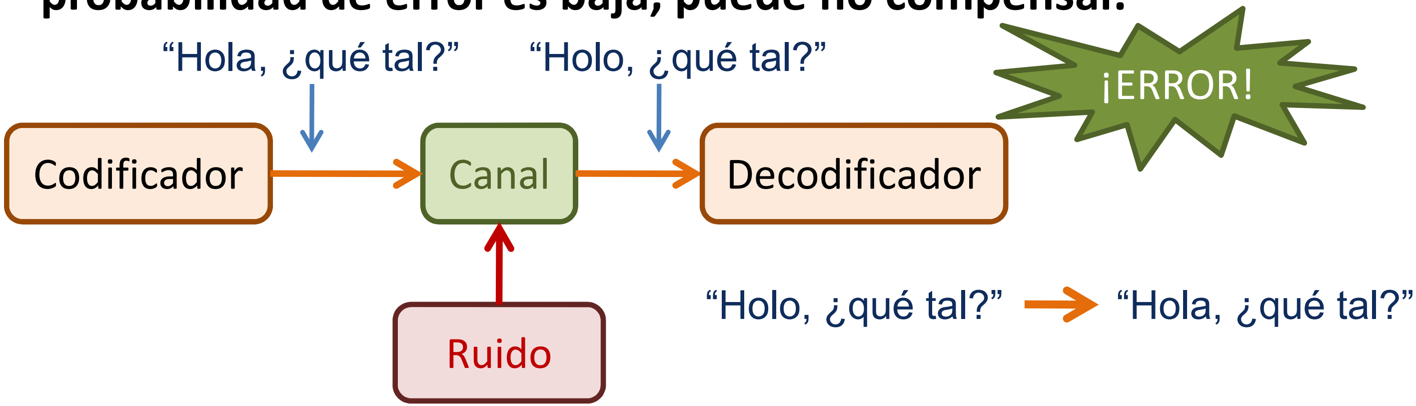
- **Aspectos prácticos de la detección de errores:**
  - Detectar el error es útil cuando sólo la detección es suficiente para la aplicación que tenemos entre manos.
  - O cuando la probabilidad de error es muy alta y nos podemos permitir pedirle al emisor que retransmita el mensaje de nuevo.
- Normalmente, la detección de errores requiere menor redundancia.



– Si deseamos reconstruir el mensaje, necesitamos no sólo detectarlo, sino saber dónde está el error y corregirlo.



– **La corrección de errores requiere una mayor redundancia. Si la probabilidad de error es baja, puede no compensar.**



- Lo primero que se preguntaron los padres de la Teoría de la Información es:

  - Si un canal posee ruido, **¿Puedo enviar información por el canal y que se reciba correctamente en el destino?**

  - **Existen dos teoremas de gran utilidad para responder esta cuestión:**

    - **El teorema para codificación con ruido de Shannon.**

    - **El teorema de Hamming para detección y corrección de errores.**

  - **Shannon, además, relacionó las propiedades de un canal con las medidas de incertidumbre y de información que propuso.**

– Lo primero que se preguntaron los padres de la Teoría de la Información es:

  – Si un canal posee ruido, **¿Puedo enviar información por el canal y que se reciba correctamente en el destino?**

  – **Existen dos teoremas de gran utilidad para responder esta cuestión:**

    – **El teorema para codificación con ruido de Shannon.**

    – **El teorema de Hamming para detección y corrección de errores.**

  – **Shannon, además, relacionó las propiedades de un canal con las medidas de incertidumbre y de información que propuso.**

- La **distancia de Hamming** nos servirá para detectar errores en la transmisión de códigos.

- Por simplicidad en la explicación y en el desarrollo de la teoría, supondremos que **todos los símbolos del alfabeto de la fuente se codificarán con un código uniforme**, de longitud **n**.

- En la asignatura nos centraremos en códigos binarios uniformes de longitud **n**.

- **Ejemplo: Código binario uniforme de longitud 2 para codificar el alfabeto {R, G, B}.**
- Distancia de Hamming entre dos palabras del código **x** e **y**: **D(x, y)**

Símbolo	Código
R	00
G	11
B	01

D(00, 01)= 1
D(00, 11)= 2
D(11, 01)= 1

– La **distancia de Hamming de un código** **se define como** **la mínima distancia de Hamming** **existente entre dos palabras de ese código.**

– **En el ejemplo anterior:**

Símbolo	Código
R	00
G	11
B	01

D(00, 01)= 1
D(00, 11)= 2
D(11, 01)= 1

– **La distancia de Hamming del código es 1:** **La mínima distancia de Hamming entre 2 palabras del código.**

- **El Teorema de Hamming para detección de errores:**

  - **Sea un código con una distancia d. Entonces, es posible detectar d-1 errores de 1 bit en una palabra de dicho código.**

Símbolo	Código
R	00
G	11
B	01

D(00, 01)= 1
D(00, 11)= 2
D(11, 01)= 1

- **La distancia de Hamming del código es 1: No se asegura poder detectar error en ningún bit.**

- **Ejemplo: Se transmite "11" pero hay un error en el bit más significativo y se recibe "01" → No se detecta error ninguno.**

– El Teorema de Hamming para detección de errores:

– Sea un código con una distancia d. Entonces, es posible detectar d-1 errores de 1 bit en una palabra de dicho código.

Símbolo	Código
R	001
G	010
B	100

D(001, 010)= 2
D(001, 100)= 2
D(010, 100)= 2

– La distancia de Hamming del código es 2: Se asegura poder detectar error en 1 bit.

– Ejemplo: Se transmite “001” pero hay un error en un bit al azar, y se recibe “011” → Se detecta error.

– Ejemplo: Se transmite “001” pero hay un error en 2 bits al azar, y se recibe “010” → No se detecta error.

– **Ejercicio:**

– **Sea una fuente F que emite símbolos {A, B, C, D}**
– **Diseñar un código con distancia de Hamming igual a 3.**

– Indique, para dicho código, cuántos errores pueden detectarse con el mensaje según el Teorema de Hamming para detección de errores.

– Indique un ejemplo para detección de un **error simple**. ¿Sería posible dar un ejemplo de error simple que no pudiese ser detectado por el código?

– Indique un ejemplo para detección de un **error doble**. ¿Sería posible dar un ejemplo de error doble que no pudiese ser detectado por el código?

– Indique un ejemplo para detección de un **error triple**. ¿Sería posible dar un ejemplo de error triple que no pudiese ser detectado por el código.

- Lo primero que se preguntaron los padres de la Teoría de la Información es:
  - Si un canal posee ruido, **¿Puedo enviar información por el canal y que se reciba correctamente en el destino?**
  - **Existen dos teoremas de gran utilidad para responder esta cuestión:**
    - **El teorema para codificación con ruido de Shannon.**
    - **El teorema de Hamming para detección y corrección de errores.**
  - **Shannon, además, relacionó las propiedades de un canal con las medidas de incertidumbre y de información que propuso.**

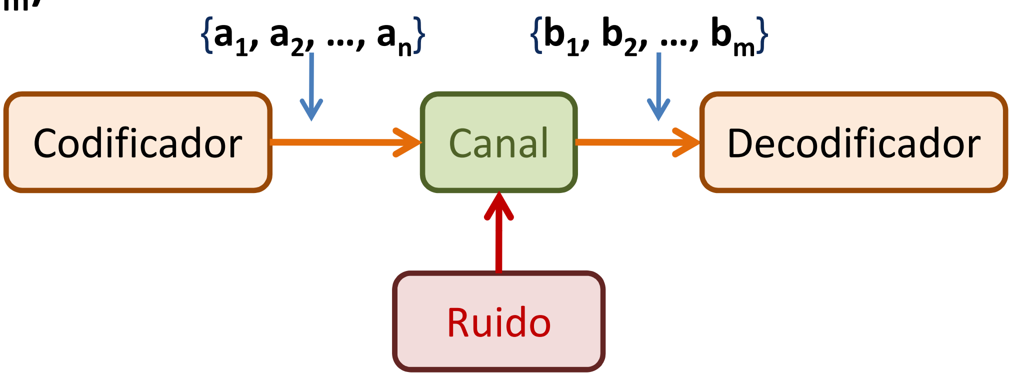
- **La idea de Shannon: Aplicar la detección de errores en transmisiones.**

  - Demostrar que se puede enviar datos en canales con ruido.
  - Intentar acotar o encontrar límites para la transmisión.

  - Vamos a suponer una fuente **F** emisora de símbolos, y un receptor **S**.

  - Supondremos también el alfabeto de la fuente **F** tiene **n** símbolos $\mathbf{a_1, a_2, \ldots, a_n}$, con probabilidades $\mathbf{p(a_1), p(a_2), \ldots, p(a_n)}$.

  - Para generalizar, también supondremos que el receptor **S** puede percibir **m** símbolos $\mathbf{b_1, b_2, \ldots, b_m}$, con probabilidades $\mathbf{p(b_1), p(b_2), \ldots, p(b_m)}$



- Si la fuente F envía un símbolo a_i, el receptor S recibirá un símbolo b_j con cierta probabilidad…
- … que es diferente a la probabilidad de percibir el mismo símbolo b_j si F enviase algún otro símbolo a_k → Existe dependencia S de frente a F.

- Nuestro estudio se centrará en las matrices de probabilidades condicionadas $p(b_j \mid a_i)$, que hemos visto previamente. Ejemplo:

	1	2
A	0.05	0.95
B	0.95	0.05

  - Ejemplo: $P(S=2 \mid F=A) = 0,95$

- También supondremos canales simétricos.

– Si la fuente F envía un símbolo a_i, el receptor S recibirá un símbolo b_j con cierta probabilidad…
– … que es diferente a la probabilidad de percibir el mismo símbolo b_j si F enviase algún otro símbolo a_k → Existe dependencia S de frente a F.

– Nuestro estudio se centrará en las matrices de probabilidades condicionadas $p(b_j \mid a_i)$, que hemos visto previamente. Ejemplo:

	1	2
A	0.05	0.95
B	0.95	0.05

  – Ejemplo: P(S=2 | F=A) = 0,95

– También supondremos canales simétricos.
– Así, si queremos calcular la probabilidad de recibir un b_j en S, aplicaremos la fórmula general:

$$p(b_j) = \sum_{i=1}^{n} p(b_j \mid a_i) p(a_i)$$

– **Ejemplo:**
  – **Supongamos que la fuente F emite símbolos con las probabilidades p(A)= 0.3 y p(B)= 0,7. ¿Cuál es la probabilidad de recibir un '2' en S? ¿y de recibir un '1'?**

	1	2
A	0.05	0.95
B	0.95	0.05

$$p(b_j) = \sum_{i=1}^{n} p(b_j \mid a_i) p(a_i)$$

$$p('2') = \sum_{i=1}^{2} p('2' \mid a_i) p(a_i) = p('2' \mid 'A') p('A') + p('2' \mid 'B') p('B')$$

$$p('2') = 0.95 * 0.3 + 0.05 * 0.7 = 0{,}32$$

$$p('1') = \sum_{i=1}^{2} p('1' \mid a_i) p(a_i) = p('1' \mid 'A') p('A') + p('1' \mid 'B') p('B')$$

$$p('2') = 0.05 * 0.3 + 0.95 * 0.7 = 0{,}68$$

– Si queremos conocer la información en S para un valor emitido por F, ¿cómo se calcularía?

– Tema 2:
  – I(S; F)= H(S)-H(S|F)= H(S)-H(S,F)+H(F)

– Recordemos las fórmulas de la entropía:

$$H(S) = -\sum_{i=1}^{n} p(S = s_i)\cdot\log_2(p(S = s_i))$$

$$H(Y \mid X = x_i) = -\sum_{j=1}^{m} P(Y = y_j \mid X = x_i)\log_2(P(Y = y_j \mid X = x_i))$$

$$H(Y \mid X) = \sum_{i=1}^{n} P(X = x_i)H(Y \mid X = x_i)$$

– **En nuestro ejemplo, ¿cuál sería la información mutua entre S y F?**

– **Primero: Calculemos H(S | F=‘A’) y H(S | F=‘B’)**

$$H(Y \mid X = x_i) = -\sum_{j=1}^{m} P(Y = y_j \mid X = x_i)\log_2(P(Y = y_j \mid X = x_i))$$

$$H(S \mid F = 'A') = -\sum_{i=1}^{2} P(S = b_i \mid F = 'A')\log_2(P(S = b_i \mid F = 'A'))$$

$$H(S \mid F = 'A') = -p(S = '1' \mid F = 'A')\log_2(p(S = '1' \mid F = 'A')) - p(S = '2' \mid F = 'A')\log_2(p(S = '2' \mid F = 'A'))$$
$$= -0.95 * \log_2(0{,}95) - 0{,}05\log_2(0{,}05) = 0.2864$$

$$H(S \mid F = 'B') = -p(S = '1' \mid F = 'B')\log_2(p(S = '1' \mid F = 'B')) - p(S = '2' \mid F = 'B')\log_2(p(S = '2' \mid F = 'B'))$$
$$= -0.05 * \log_2(0{,}05) - 0{,}95\log_2(0{,}95) = 0.2864$$

– **En nuestro ejemplo, ¿cuál sería la información mutua entre S y F?**

– **Segundo: Calculemos H(S | F)**

$$H(S \mid F) = \sum_{i=1}^{n} p(F = a_i) H(S \mid F = a_i) =$$

$$- p(F =' A') H(S \mid F =' A') - p(F =' B') H(S \mid F =' B')$$

$$= 0.3 * 0.2864 + 0.7 * 0.2864 = 0.2864$$

– **Tercero: Calculemos H(S), sabiendo que ya hemos calculado antes p('1')= 0.68 y p('2')=0.32**

$$H(S) = -\sum_{i=1}^{n} p(S = s_i) \cdot \log_2(p(S = s_i)) =$$

$$- p(S ='1') \log_2(p(S ='1')) - p(S ='2') \log_2(p(S ='2')) =$$

$$- 0.68 \log_2(0.68) - 0.32 \log_2(0.32) = 0.9044$$

– **En nuestro ejemplo, ¿cuál sería la información mutua entre S y F?**

– **Por último: Calculemos I(S ; F)**

$$I(X;Y) = H(X) - H(X|Y)$$

$$I(S;F) = H(S) - H(S|F) = 0.9044 - 0.2864$$

$$I(S;F) = 0.618$$

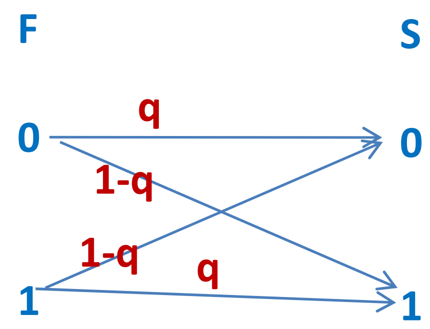
– **¿Y todo esto para qué nos sirve?**

– Nos sirve para definir la capacidad **C** de un canal desde el punto de vista de la teoría de la información:

$$C = \max_{p(error)} \{I(X;Y)\}$$

– Simplificación para el caso de un canal binario simétrico

– Tanto la fuente F como el receptor S tienen símbolos {0,1}



– La probabilidad de que F mande un 0 y de que S reciba un 0 es la misma que si F manda 1 y S recibe 1, valor q.

– La probabilidad de que F mande un 0 y de que S reciba un 1 es la misma que si F manda un 1 y S recibe un 0, valor 1-q

– Llamaremos *p* a la probabilidad de error (enviar un símbolo y recibir otro) en canales simétricos binarios.

– En el esquema, p=1-q

– Simplificación para el caso de un canal binario simétrico

  – Vamos a parametrizar C con respecto a la probabilidad de error p, y la notaremos C(p)

  – En el caso de canales simétricos binarios, se tiene que la función C(p)= $\max_p$\{ I(S; F) \} se calcula como:

$$C(p) = 1 + p\log_2(p) + (1-p)\log_2(1-p)$$

  – Si p=0,5, tiene valor máximo. Se tiene que tanto si F emite un 0 como un 1, la probabilidad de que S reciba un 0 o un 1 es la misma, 0.5. Estamos ante un canal inútil.

– ¿Cuál es la capacidad de un canal binario simétrico inútil?

  – C(0,5)= 1+0,5*log2(0,5)+(1-0,5)*log2(1-0,5)= 0 → NO SE PUEDE TRANSMITIR POR EL CANAL

– **Simplificación para el caso de un canal binario simétrico**

– **¿Y si la probabilidad de error es de 0 (canal sin ruido)?**

$$C(p) = 1 + p\log_2(p) + (1-p)\log_2(1-p)$$

– **C(0)= 1+0*log2(0)+(1-0)*log2(1-0)= 1 → 1 BIT.**

– **Se puede transmitir el símbolo 0 o 1 sin problemas.**

– **Simplificación para el caso de un canal binario simétrico**

– **¿Y si la probabilidad de error es de 0.1 (canal con ruido)?**

$$C(p) = 1 + p\log_2(p) + (1-p)\log_2(1-p)$$

– **C(0.1)= 1+0.1*log2(0.1)+(1-0.1)*log2(1-0.1)= 0.531 → 0.531 BITs.**

– **Con una probabilidad de error de 0.1, por cada unidad de tiempo no podremos enviar más de 0.531 bits.**

– **<u>Como mínimo</u> necesitaremos 2 unidades de tiempo (código de longitud n=2) para poder transmitir 1 BIT de información por un canal simétrico binario, si la probabilidad de error p=0.1.**

– **Conclusión: Podemos calcular la longitud mínima que debe tener un código para que exista un método que lo codifique y decodifique correctamente.**

– **Simplificación para el caso de un canal binario simétrico**

– Utilizaremos **códigos uniformes**.

– Un código se dice **uniforme** si todas las palabras del código tienen la misma longitud.

– Ejemplos de códigos uniformes:

Símbolo	Código
R	00
G	11
B	01

Símbolo	Código
R	001
G	010
B	100

– **Notaremos a un código uniforme con M palabras de longitud n y distancia de código d como (n, M, d)-código**.

- Notaremos a un código uniforme con M palabras de longitud n y distancia de código d como (n, M, d)-código.

- Ejemplo:

  - Código (2, 3, 1):

Símbolo	Código
R	00
G	11
B	01

  - Código (3, 3, 2):

Símbolo	Código
R	001
G	010
B	100

– **Simplificación para el caso de un canal binario simétrico**

– También, definimos **e(F)** como el **error cometido por un código de una fuente F con n palabras**.

$$e(F) = \frac{1}{n} \sum_{i=1}^{n} p(error \mid a_i \; transmitido)$$

– En el siguiente código, ¿cuál sería el valor de **e(F)**?

Símbolo	Código
R	AA
G	BB
B	AB

Envía\Recibe	1	2
A	0.05	0.95
B	0.95	0.05

– En el siguiente código, ¿cuál sería el valor de **e(F)**?

Símbolo	Código
R	AA
G	BB
B	AB

Envía\Recibe	1	2
A	0.05	0.95
B	0.95	0.05

– La probabilidad de error de enviar una 'A' es la misma que la probabilidad de error de enviar una 'B': 0.05.

– Por tanto:
  – Si se envía R: p(error | R)= 2*0,05= **0,1 (porque no hay redundancia)**
  – Si se envía G: p(error | G)= 2*0,05= **0,1 (porque no hay redundancia)**
  – Si se envía B: p(error | B)= 2*0,05= **0,1 (porque no hay redundancia)**

$$e(F) = \frac{1}{n}\sum_{i=1}^{n} p(error \mid a_i\ transmitido) = \frac{1}{3}(0.1 + 0.1 + 0.1) = 0.1$$

– **TEOREMA DE SHANNON PARA CODIFICACIÓN CON RUIDO**:

  – **Sea un canal simétrico binario con probabilidad de error p y sea R un número que satisface 0 < R < C(p)**

  – **Entonces, para cualquier ε>0, para un *n* lo suficientemente grande existe un (n, 2^k, d)-código con ratio k/n<=R tal que e(C) < ε.**

  – **Según Shannon, si el canal es binario y simétrico, tiene ruido con probabilidad "p" y no es es canal inútil, C(p) > 0,**

  – **... Entonces podemos encontrar un código que se pueda transmitir por el canal...**

  – **... Con la condición de que el código tenga M<=2^k palabras de longitud n, y que k/n < C(p)**

  – **... (buscar dicho código ya es otra historia)**

– **Según Shannon, ¿se podría transmitir este código por el canal?**

Símbolo	Código
R	AA
G	BB
B	AB

Envía\Recibe	1	2
A	0.05	0.95
B	0.95	0.05

– **El código es un (2, 3, 1) → M=3, n=2**

$$C(p) = 1 + p\log_2(p) + (1-p)\log_2(1-p)$$

– La capacidad del canal es:

C(0,05)= **1+0.05*log2(0.05)+(1-0.05)*log2(1-0.05)= 0.7136**

– Por tanto: **$M<=2^k$ → k>=2**

k/n= 2/2= 1

– **El código no podría transmitirse por el canal y ser detectado o corregido en el destino.**

– **Según Shannon, ¿Qué longitud mínima n tendría que tener el código?**

Símbolo	Código
R	?
G	?
B	?

Envía\Recibe	1	2
A	0.05	0.95
B	0.95	0.05

– **El código es un (?, 3, ?) → M=3, n=desconocido, hay que buscarlo**

$$C(p) = 1 + p\log_2(p) + (1-p)\log_2(1-p)$$

– La capacidad del canal es **C(0,05)= 0.7136**

– Por tanto, hay que buscar un **n** que satisfaga:

k/n < C(p) → n > k/C(p)

– **El mínimo valor de n que nos valdría es n>2/0.7136 n>2.8027**

Con una longitud n=3 del código nos bastaría.

– Según Shannon, ¿se podría transmitir este código por el canal?

Símbolo	Código
R	AAB
G	ABA
B	BAA

Envía\Recibe	1	2
A	0.05	0.95
B	0.95	0.05

– **El código es un (3, 3, 2) → M=3, n=3**

– La capacidad del canal es **C(0,05)= 0.7136**

– Por tanto: **$M<=2^k$ → k>=2**

k/n= 2/3= 0.6666 → Se cumple k/n < C(0.05)

– **El código podría transmitirse por el canal y ser detectado en el destino, con un margen de la probabilidad de no recibir bien el mensaje muy cercana a 0.**

1. Utilidad de la detección de errores
2. El modelo de comunicaciones con ruido
3. Fundamentos para transmisión con ruido
4. **Bits de paridad**
5. Códigos de verificación de cuenta fija
6. Códigos de redundancia cíclica

- **Los bits de paridad se utilizan para detectar errores simples o de ocurrencia impar.**

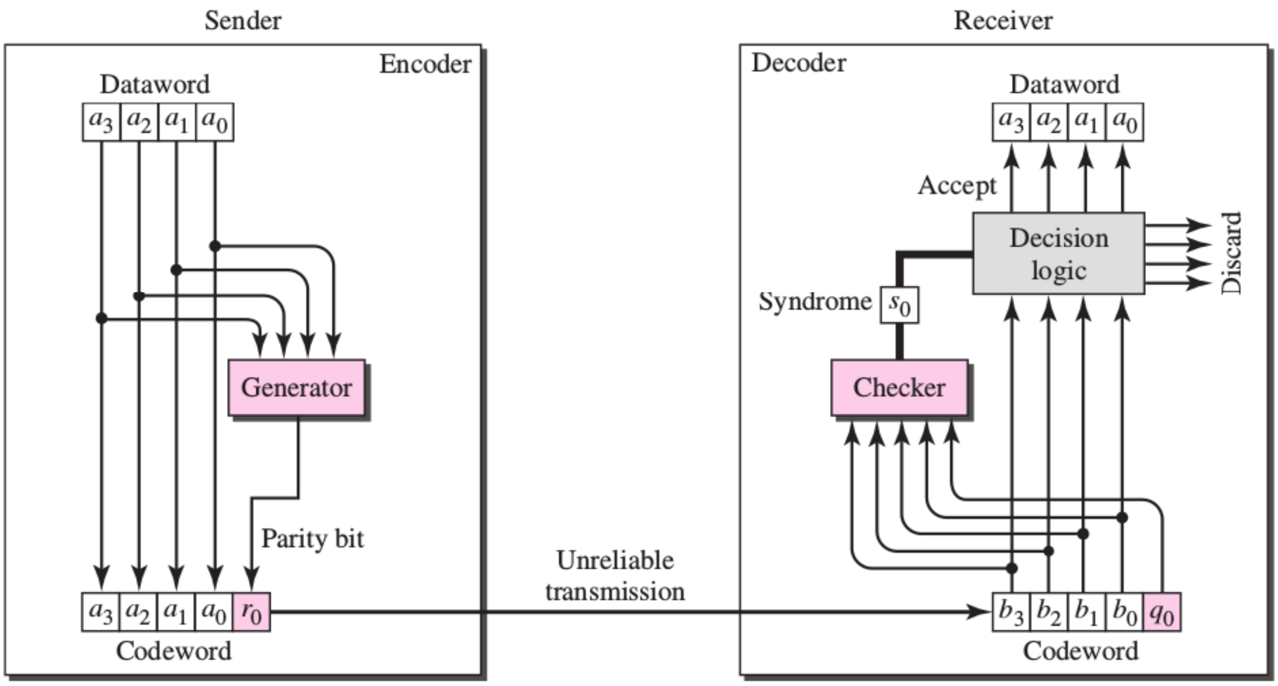
  - Consiste en incorporar bits de más al código y contar el número de 1's y 0's existentes.

  - **Paridad par**: La suma de todos los bits debe ser 0.
  - **Paridad impar**: La suma de todos los bits debe ser 1.

- **Se basa en el Teorema para la detección de errores de Hamming de que, para códigos de distancia de Hamming *d, se pueden detectar d-1 errores*.**

– **Fundamentos de la técnica de detección de errores con bits de paridad.**

– El modelo del esquema de funcionamiento es el siguiente:



- Los bits de paridad se utilizan para detectar errores simples o de ocurrencia impar.

  - Consiste en incorporar bits de más al código y contar el número de 1's y 0's existentes.

  - **Paridad par**: La suma de todos los bits debe ser 0.
  - **Paridad impar**: La suma de todos los bits debe ser 1.

- **Ejemplo: Codificación con Paridad par con 1 bit de paridad, sobre un código uniforme de longitud 9:**

0	0	1	0	1	1	1	0	1	0

Bit de paridad

- **En la paridad par, todos los 1's deben sumar "0".**

– Ejemplo: Codificación con Paridad impar con 1 bit de paridad, sobre un código uniforme de longitud 9:

1	0	1	0	1	1	1	0	1	0

Bit de paridad

– En la paridad impar, todos los 1's deben sumar "1".

– Ejemplo:

?	1	1	0	0	1	0	1	0	1

Bit de paridad : En paridad impar, todo debe sumar 1. Como hay 5 1's, el bit de paridad deberá ser 0 en el ejemplo.

– Ejemplo de detección de errores con paridad par:

0	1	0	1	1	0	1	0	1	0

– Hay 5 “1”’s. Como es paridad par, debería aparecer un 1 en el bit de paridad para que todo sume 0.

– Sin embargo, hay un 0 en el bit de paridad → **Hay un error en el mensaje.**

– **Se pueden incluir varios bits de paridad para detectar en qué zona del mensaje se encuentra el error.**
  – Agrupar por bits pares/impares.
  – Dividir el mensaje en varias partes
  – Etc.

- La forma más común de agrupar bits de paridad es agrupar el mensaje por bloques y aplicar bits de paridad horizontal, vertical y cruzada (diagonal).

- Nos referiremos a estos códigos como **P(m,k)**: Códigos de paridad de **m** bloques con **k** bits por bloque. **Ejemplo de código de paridad par P(3, 3), que envía el mensaje 010100110.**

$$\begin{array}{cc} & \text{K bits por bloque} \\ \text{M bloques} & \left\{\begin{matrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}\right. \end{array}$$

- La forma más común de agrupar bits de paridad es agrupar el mensaje por bloques y aplicar bits de paridad horizontal, vertical y cruzada (diagonal).

- Nos referiremos a estos códigos como **P(m,k)**: Códigos de paridad de **m** bloques con **k** bits por bloque. **Ejemplo de código de paridad par P(3, 3), que envía el mensaje 010100110.**

- **Transmite:**
  0101100111000000

K bits por bloque

M bloques

$$\left\{\begin{array}{ccc|c} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \end{array}\right.$$

Paridad par horizontal (LRC)

Paridad par vertical (VRC)

Paridad par cruzada (paridad de los bits de paridad)

– Ejemplo: Se desean transmitir los bits 011010 por un código de paridad par P(2, 3).

– Hay 2 bloques, cada bloque de 3 bits. **Se añade una fila y una columna más para los bits de paridad.**

– **Transmite:**
  011001010010

$$\left\{\begin{array}{|cccc} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right.$$

Paridad par VRC

– Ejemplo: Se desean transmitir los bits 011010 por un código de paridad par P(2, 3).

– Hay 2 bloques, cada bloque de 3 bits.

– **Transmite:**
  011001010010

$$\left\{ \begin{array}{|cccc} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array} \right.$$

Paridad par LRC

Paridad par VRC

- **Ejemplo: Se desean transmitir los bits 011010 por un código de paridad par P(2, 3).**

- Hay 2 bloques, cada bloque de 3 bits.

- **Transmite:**
  011001010010

$$\left\{\begin{array}{ccc|c} 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right.$$

Paridad par LRC

Paridad par VRC

Paridad par cruzada

– **Ejemplo: Supongamos un alfabeto de la fuente {A, B, C, D}, codificado como:**

Símbolo	Código
A	1010
B	0101
C	1100
D	0011

– Asumiendo una codificación de paridad por bloques P(3, 4), ¿Qué secuencia de bits se transmitiría para enviar el mensaje “ACBBD”?

– **Suponer un símbolo adicional “Símbolo vacío” con código “0000” para la elaboración del ejercicio.**

1. Utilidad de la detección de errores
2. El modelo de comunicaciones con ruido
3. Fundamentos para transmisión con ruido
4. Bits de paridad
5. **Códigos de verificación de cuenta fija**
6. Códigos de redundancia cíclica

- Se denominan también **códigos *i en n***.

- **Son códigos uniformes de longitud n que tienen exactamente *i* bits igual a 1 (el resto a 0).**

- **¿Qué cantidad de códigos de verificación de cuenta fija podemos obtener?**

  - Si tenemos **n bits** → Hay 2^n posibles palabras.

  - **Ejemplo: Longitud de n=5 → 2^5=32 posibles palabras**

  - Si consideramos que **i** de los bits tienen que ser 1 y que **(n-i)** bits tienen que ser 0:

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

  - **Ejemplo: Código 3 en 5. Longitud de n=5 donde cada palabra tiene que tener i=3 unos → 5!/(3!*(5-3)!) =10 posibles palabras**

– Ejemplo: Código 4 en 8.

– Palabras de 8 bits con 4 unos en cada palabra.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	0	0	0	1	1	1	1
0	0	1	0	1	1	1	0
0	0	1	1	0	1	1	0
0	0	1	1	1	0	1	0
0	0	1	1	1	1	0	0
…	…	…	…	…	…	…	…
1	1	1	1	0	0	0	0

– La distancia de Hamming del código es de 2.
– Códigos fáciles de diseñar, que permiten detectar errores de 1 bit por palabra del código.

- La detección del error en el receptor es simple:
  - Se cuenta el números de 1's del mensaje.
  - Si el número de 1's en cada símbolo coincide con *i*, parece que todo ok.
  - Si el código se corresponde con un símbolo, todo ok. Decodificar.
    - Puede que no se detecte el número de 1's igual a *i* → En tal caso, hay detección de error.
    - Puede que se detecte el número de 1's igual a *i* , pero que la decodificación no sea ninguna palabra del código (sólo válido si la distancia de Hamming del código >2).
    - En tal caso, también se detecta error.

- **Ejemplo: Suponga una fuente F que tiene el alfabeto {A, B, C}.**

  - **¿Cuál es la longitud mínima del tamaño del código *n* para que se pueda codificar en un código de verificación de cuenta fija *3 en n*?**

  - **Diseñe el código.**

  - Qué bits se transmitirían para enviar el mensaje {AB}.

  - ¿Es posible detectar 2 bits de error si se manda el mensaje "A"?.

  - ¿Es posible que no se detecten 2 bits de error si se manda el mensaje "B"? Ponga un ejemplo.

  - ¿Qué bits se codificarían en el envío del mensaje "AB"? Introduzca un bit de error en cada código. ¿Cómo se detectaría en el receptor cada error?

1. Utilidad de la detección de errores
2. El modelo de comunicaciones con ruido
3. Fundamentos para transmisión con ruido
4. Bits de paridad
5. Códigos de verificación de cuenta fija
6. Códigos de redundancia cíclica

- **Los códigos de redundancia cíclica son un tipo de códigos polinómicos.**

- Un **código polinómico** está basado en modelar el código con polinomios cuyos **coeficientes se expresan en módulo 2 (← De ahí el término "cíclico").**

  - **Ejemplo: Polinomio p(x) de grado n**

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \ldots + a_2 x^2 + a_1 x + a_0$$

  - **Los coeficientes a_i están restringidos a los valores {0, 1}**

  - **Ejemplo (n=3):** $p(x) = x^3 + x$

  - **Ejemplo (no válido):** $p(x) = x^3 + 2x^2 + x$

Los coeficientes a_i no pueden valer otra cosa que no sea 0 ó 1

– **Los códigos de redundancia cíclica consisten en añadir unos *bits de redundancia* al código, que se calculan a partir de la división del código por un polinomio generador.**

– **Parada estratégica: Recordemos cómo dividir en binario:**

11010110110000 / 10011

Divisor

Dividendo

1 1 0 1 0 1 1 0 1 1 0 0 0 0 | 1 0 0 1 1

. . .
. . .
. . .
. . .

Cociente

(Resto)

- Los códigos de redundancia cíclica consisten en añadir unos *bits de redundancia* al código, que se calculan a partir de la división del código por un polinomio generador.

- Parada estratégica: Recordemos cómo dividir en binario:

Paso 1. Seleccionamos tantos dígitos del dividendo como tenga el divisor.

1 1 0 1 0 1 1 0 1 1 0 0 0 0 | **1 0 0 1 1**

– Los códigos de redundancia cíclica consisten en añadir unos *bits de redundancia* al código, que se calculan a partir de la división del código por un polinomio generador.

– Parada estratégica: Recordemos cómo dividir en binario:

Paso 2. Ponemos en el cociente un 0 o un 1, según lo seleccionado en el dividendo sea igual (1) o menor (0) que el divisor.

1 1 0 1 0 1 1 0 1 1 0 0 0 0	**1 0 0 1 1**
	1

– **Los códigos de redundancia cíclica consisten en añadir unos *bits de redundancia* al código, que se calculan a partir de la división del código por un polinomio generador.**

– **Parada estratégica: Recordemos cómo dividir en binario:**

Paso 3. Si pusimos un 1 en el cociente, rellenamos con el divisor debajo de la parte sombreada del dividendo. Si pusimos un 0, rellenamos con 0's.

1 1 0 1 0 1 1 0 1 1 0 0 0 0	**1 0 0 1 1**
1 0 0 1 1	**1**

– Los códigos de redundancia cíclica consisten en añadir unos *bits de redundancia* al código, que se calculan a partir de la división del código por un polinomio generador.

– Parada estratégica: Recordemos cómo dividir en binario:

Paso 4. Restamos módulo 2 en el dividendo y bajamos la siguiente cifra. Comenzamos desde el Paso 2de nuevo.

1 1 0 1 0 1 1 0 1 1 0 0 0 0	1 0 0 1 1
1 0 0 1 1	1
1 0 0 1 1	

– **Los códigos de redundancia cíclica consisten en añadir unos *bits de redundancia* al código, que se calculan a partir de la división del código por un polinomio generador.**

– **Parada estratégica: Recordemos cómo dividir en binario:**

Pasos 2, 3 y 4 de nuevo...

```
1 1 0 1 0 1 1 0 1 1 0 0 0 0   | 1 0 0 1 1
1 0 0 1 1                     |__________
-----------                     1 1
  1 0 0 1 1
  1 0 0 1 1
  -----------
    0 0 0 0 1
```

```
1 1 0 1 0 1 1 0 1 1 0 0 0 0   | 1 0 0 1 1
1 0 0 1 1                     |__________
-----------                     1 1 0
  1 0 0 1 1
  1 0 0 1 1
  -----------
    0 0 0 0 1
    0 0 0 0 0 0
    -----------
      0 0 0 1 0
```

Y así hasta que nos quede un resto de un número de bits inferior al del divisor.

```
1 1 0 1 0 1 1 0 1 1 0 0 0 0   | 1 0 0 1 1
1 0 0 1 1                     |___________
-------------                   1 1 0 0 0 0 1 0 1 0
  1 0 0 1 1
  1 0 0 1 1
  ---------
    0 0 0 0 1
      0 0 0 0 0
    -----------
          1 0 1 1 0
          1 0 0 1 1
          ---------
            0 1 0 1 0
            0 0 0 0 0
            ---------
              1 0 1 0 0
              1 0 0 1 1
              ---------
                0 1 1 1 0
                0 0 0 0 0
                ---------
                  1 1 1 0
```

Resto

- **La idea básica de los CRC es que se puede transformar de forma directa un polinomio de coeficientes en {0,1} en un código binario, únicamente indicando sus coeficientes a 0 o a 1.**

- Ejemplo: $\mathbf{p(x)= x^{12} + x}$

  - **En total es de grado 12 → tiene 13 coeficientes → 13 bits**

  - **Representación del polinomio en binario (con n=13 bits):**

1000000000010

- Ejemplo: $\mathbf{p(x)= x^{3} + x^{2}+ 1}$

  - **En total es de grado 3→ tiene 4 coeficientes → 4 bits**

  - **Representación del polinomio en binario (con n=13 bits):**

0000000001101

- **La idea básica de los CRC es que se puede transformar de forma directa un polinomio de coeficientes en {0,1} en un código binario, únicamente indicando sus coeficientes a 0 o a 1.**

- Ejemplo: **$p(x)= x^{12} + x$**

  - **En total es de grado 12 → tiene 13 coeficientes → 13 bits**

  - **Representación del polinomio en binario (con n=13 bits):**

    Coef. 12→ bit 12: x^{12}

    1000000000010

    Coef. 1→ bit 1: x^1

- Ejemplo: **$p(x)= x^3 + x^2 + 1$**

  - **En total es de grado 3→ tiene 4 coeficientes → 4 bits**

  - **Representación del polinomio en binario (con n=13 bits):**

    Coef. 0, 2 y 3→ bits 0, 2 y 3

    0000000001101

- **La inclusión de redundancia, en códigos CRC consiste en:**
  - **Seleccionar un polinomio G(x) para el divisor, de un orden dado k.**
  - A este polinomio **G(x)** se le denomina **polinomio generador**.
  - **Este polinomio es conocido por el emisor y el receptor.**
  - **Ejemplo general de polinomio generador de orden 3:**

$$G(x)= a_3x^3+a_2x^2+a_1x+a_0$$

  - **Si se desea transmitir un mensaje de *n bits*, la idea es añadir k bits de control de errores al final del mensaje.**
  - **Ejemplo: Para transmitir 8 bits con un polinomio generador de orden 4:**
    - **Bits a enviar: 10101001**
    - **Se enviarán 12 bits: 10101001????**

- La inclusión de redundancia, en códigos CRC consiste en:
  - Los bits de control de errores son el resto de dividir el polinomio, con *k* 0's al final, por el polinomio generador.
  - Los bits *extra* serán el resto de la división.
  - Ejemplo con polinomio generador de orden 4 $G(x)=x^4+1$ para transmitir 10101001:
  - Paso 1: Se le añaden *k* bits de 0's al final.
    - 101010010000
  - Paso 2: Se divide por el polinomio generador y se calcula el resto
  - $G(x)=x^4+1$ → Divisor= 10001
  - Dividir 101010010000 / 10001

– La inclusión de redundancia, en códigos CRC consiste en:
  – Dividir 101010010000 / 10001

```
1 0 1 0 1 0 0 1 0 0 0 0   | 1 0 0 0 1
1 0 0 0 1                 |__________
---------                   1 0 1 0 0 0 1 1
  0 1 0 0 0 0
    1 0 0 0 1
  -----------
    0 0 0 0 1 1 0 0 0
            1 0 0 0 1
    -----------------
            0 1 0 0 1 0
              1 0 0 0 1
            -----------
                0 0 1 1   <- Resto
```

  – Paso 3: Se sustituyen los últimos *k* bits por el resultado del resto
    – Mensaje 101010010000 → 101010010011
  – Paso 4: Se envía el mensaje final.

- **La inclusión de redundancia, en códigos CRC consiste en:**
  - **En el receptor:**
    - Únicamente **hay que comprobar que el mensaje recibido es divisible por el polinomio generador**.
    - Si es así, el resto debe ser 0 y no hay errores.
    - Si el resto es distinto de 0, entonces hay errores en la transmisión.
  - **Ejemplo: Se recibe 101010010011 y G(x)= 10001**
    - Hay que dividir el mensaje de entrada entre el polinomio y ver si es 0.

– **La inclusión de redundancia, en códigos CRC consiste en:**
  – **Dividir 101010010011 / 10001**

```
1 0 1 0 1 0 0 1 0 0 1 1 | 1 0 0 0 1
1 0 0 0 1                 1 0 1 0 0 0 1 1
  0 1 0 0 0 0
    1 0 0 0 1
    0 0 0 0 1 1 0 0 1
            1 0 0 0 1
            0 1 0 0 0 0
              1 0 0 0 0
                0 0 0 0
```

Resto

  – **Como es divisible, no se han detectado errores**

– Ejemplo: Supongamos que se envía 101010010011, pero que por errores en la transmisión se cambia algún bit: 101110010011

– Receptor: Dividir 101110000011 / 10001

```
1 0 1 1 1 0 0 1 0 0 1 1     | 1 0 0 0 1
1 0 0 0 1                   |__________
-------------                 1 0 1 1 0 0 1 0
  0 1 1 0 0 0
    1 0 0 0 1
  -------------
      1 0 0 1 1
      1 0 0 0 1
  -------------
        0 0 1 0 0 0 1
            1 0 0 0 1
          -------------
              0 0 0 0 1
              0 0 0 0 0
          -------------
                0 0 0 1   <- Resto
```

– ¡Como no es divisible, se han detectado errores!

– Ejemplo: Supongamos que se envía 101010010011, pero que por errores en la transmisión se cambian algunos bits: 101110000011
  – Receptor: Dividir 101110000011 / 10001

```
1 0 1 1 1 0 0 0 0 0 1 1    | 1 0 0 0 1
1 0 0 0 1                  |______________
  0 1 1 0 0 0                1 0 1 1 0 0 1 1
    1 0 0 0 1
      1 0 0 1 0
      1 0 0 0 1
        0 0 1 1 0 0 1
            1 0 0 0 1
              1 0 0 0 1
              1 0 0 0 1
                0 0 0 0   <- Resto
```

– ¡Como es divisible, no se han detectado errores aunque los hubiese!

- **El número de bits erróneos que se pueden detectar depende de:**

  - **El tamaño del mensaje original.**
  - **El orden del polinomio generador.**
  - **El propio polinomio generador (hay mejores y peores) → Hay que buscar un buen polinomio, adecuado a nuestras necesidades.**

- **Ejemplos de CRC's estándar:**

  - **CRC-12:** $G(x)= x^{12} + x^{11} + x^{3} + x^{2} + x + 1$
  - **CRC-16:** $G(x)= x^{16} + x^{15} + x^{2} + 1$
  - **CRC-CCITT:** $G(x)= x^{16} + x^{12} + x^{5} + 1$

  - **Capacidades de detección de CRC-CCITT:**
    - 100% errores simples y dobles
    - 100% errores en un número impar de bits
    - 100% errores en ráfagas de igual a o menos de 16 bits
    - 99.997% errores de ráfagas de 17 bits
    - 99.998% de errores en ráfagas de 18 o más bits

- Ejemplo: Supongamos que se envía 10110110. Si utilizamos $G(x)= x^3+1$,
  - ¿Cuántos bits hay que añadir al mensaje?
  - ¿Cuál sería el mensaje final enviado?
  - Si el receptor recibiese 1011011010, ¿sería un código válido?
  - Si el receptor recibiese 10110110010, ¿sería un código válido? ¿Cuál sería la decisión del receptor (hay error/no hay error)?

Universidad de Granada

decsai.ugr.es

Teoría de la Información y la Codificación

Grado en Ingeniería Informática

Tema 4.- Información en canales con ruido.

DECSAI

Departamento de Ciencias de la Computación e Inteligencia Artificial