Codificación de Fuente y Codificación de Canal.

Este módulo es una introducción muy breve a algunos conceptos sobre codificación de canal y codificación de fuente. El objetivo básico es introducir algunos de los principios más importantes a nivel conceptual de estos dos tipos de codificaciones. La necesidad de la codificación de fuente y la codificación de canal ya se introdujo en el módulo 4. Recuerde que la codificación de fuente consistía en intentar extraer toda la información redundante en la señal para reducir el número total de bits a transmitir. Esta extracción de información redundante o compresión de datos es especialmente importante cuando queremos transmitir señales de audio o vídeo. En cambio, la codificación de canal consistía en introducir redundancia a la señal, de forma controlada, con relaciones matemáticas precisas para proteger la señal ante posibles errores.

En este breve capítulo analizaremos algunas de las técnicas básicas que nos permiten hacer la codificación de fuente y la codificación de canal. El estudio es sólo introductorio y se profundizará en otras asignaturas optativas de comunicaciones. Todos los conceptos introducidos son muy importantes pero simples de comprender.

En cuanto a la parte de codificación de fuente sólo estudiaremos el concepto de entropía y el concepto de códigos de longitud variable:

7.1. Información asociada a un mensaje.
7.2 Entropía
7.3 Códigos de longitud variable.

En cuanto a la codificación de canal introduciremos los siguientes conceptos

7.4. Redundancia estructurada: Conceptos básicos
7.5. Estrategias para el control de error. ARQ full-duplex y half-duplex.

(Nota: Los contenidos de esta última parte proceden de la introducción del capítulo de códigos de bloque que se imparte en la asignatura de comunicaciones avanzadas. En esta asignatura de carácter optativo se pueden estudiar con mayor profundidad los conceptos de codificación que se introducen en éste módulo.)

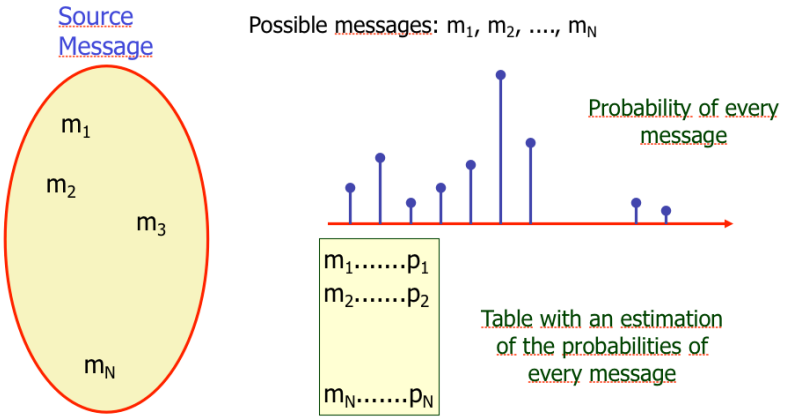
Introducción a la codificación de fuente.

7.1 Información asociada a un mensaje

Una fuente de mensajes se caracteriza por el número de mensajes que puede producir y por la probabilidad con que se produce cada mensaje. En la figura adjunta se representa la idea básica de la caracterización de una fuente de mensajes en la que se definen los diferentes mensajes m_k y sus probabilidades asociadas p_k .

En la práctica, es difícil conocer a priori la probabilidad con que se produce cada uno de los mensajes. Sin embargo, siempre se puede realizar un estudio en el que se determinen las frecuencias de aparición de cada mensaje y estimar su probabilidad a partir de estos resultados.

Es importante observar que SI TODOS LOS MENSAJES A TRANSMITIR TIENEN LA MISMA PROBABILIDAD LA CODIFICACIÓN ÓPTIMA SERÁ EMPLEAR CÓDIGOS CONVENCIONALES DE LONGITUD FIJA Y QUE NO SE PUEDE OBTENER NINGUNA COMPRESIÓN EFICIENTE DE LOS DATOS. Sin embargo, la mayoría de mensajes que queremos transmitir a través de un canal no tienen la misma probabilidad. Hay mensajes que tienen una probabilidad más alta que otros. Piense por ejemplo en que queremos enviar un texto escrito. En este caso está claro que hay algunos mensajes (las vocales) que tienen mucha más probabilidad de que las consonantes, incluso, dentro de las consonantes hay mensajes con mucha más probabilidad que otros.



Una vez especificada o estimada la estadística de la fuente, nos tendremos que plantear establecer una medida de la cantidad de información asociada a cada mensaje. La medida debe verificar ciertos requisitos que parecen naturales y que, generalmente, están asociados al concepto cualitativo de información. Estos postulados son los siguientes:

1) La información asociada a un mensaje es positiva. Es lógico suponer que siempre que se produce uno de los posibles mensajes, éste aporta algún tipo de información. No tiene sentido que se produzcan pérdidas de información si recibimos nuevos datos.

2) Los mensajes con poca probabilidad aportan más información que los mensajes con mucha probabilidad. Los mensajes que se producen frecuentemente nos aportan muy poca información porque son esperados de antemano. Así, cuando se nos informa de que hay circulación lenta en los accesos de entrada a Barcelona entre las 8 y las 9 de la mañana, estamos ante un mensaje que nos aporta muy poca información. En cambio, la noticia de una gran nevada en Barcelona aparecerá siempre con grandes titulares en los periódicos ya que, por su baja probabilidad, nos aportará mucha información.

3) La información que aportan dos mensajes simultáneos estadísticamente independientes es la misma que la que aportan cuando se producen separadamente. Esto significa que la información que nos aportan dos eventos independientes que se producen simultáneamente coincide con la suma de las informaciones que nos aportarían si se produjeran separadamente. La información que nos aporta que Nadal haya ganado el Roland Garros y que el Bilbao haya perdido contra el Español

es la misma tanto si nos dan los dos mensajes a la vez como si nos los dan por separado.

Podemos formular estos postulados mediante las siguientes ecuaciones:

$$I(m_k) \geq 0 \quad (1)$$

$$I(m_i) > I(m_j) \quad \text{si} \quad p_i < p_j \quad (2)$$

$$I(m_i, m_j) = I(m_i) + I(m_j) \quad \text{si} \quad P(m_i, m_j) = P(m_i) \cdot P(m_j) \quad (3)$$

Según teorema fundamental del cálculo, únicamente existe una función que verifique las tres propiedades simultáneamente. De esta manera, la medida cuantitativa de la información es totalmente axiomática, ya que se encuentra directamente a partir de las propiedades que queremos que verifique.

Así pues, la información asociada al mensaje m_k estará determinada por la única función que verifica las tres propiedades anteriores:

$$I(m_k) = -\log_b p_k \quad (4)$$

Es fácil comprobar que la definición anterior verifica las propiedades para cualquier base b del logaritmo. Cuando la base del logaritmo es 2, la unidad de información se denomina bit. Si la base del logaritmo es el número de Euler e la unidad de información se mide en natos.

Consideramos como ejemplo la información que proporciona una moneda al caer. Hay dos posibles mensajes (cara y cruz), ambos con la misma probabilidad (1/2). La información asociada al mensaje cara será:

$$I(cara) = -\log_2\left(\frac{1}{2}\right) = 1\ bit$$

Análogamente, si consideramos una fuente que puede transmitir ocho mensajes equiprobables, la información asociada a cada uno de estos mensajes depende de:

$$I(m_i) = -\log_2\left(\frac{1}{8}\right) = \log_2(2^3) = 3\ bits$$

De acuerdo con estos resultados, para estimar la cantidad de información ligada a uno de los mensajes de la fuente es necesario establecer, en primer lugar, las características estadísticas de la fuente de mensajes. Estas características se pueden estimar de manera experimental dejando que la fuente genere mensajes y aproximando su probabilidad para la frecuencia relativa de estos. En general, si la estimación se realiza sobre una muestra suficientemente alta, este procedimiento proporciona resultados satisfactorios.

7.2. Entropía de una fuente de mensajes

Conociendo la información asociada a cada mensaje, podemos determinar la información media que proporciona la fuente. Para ello, debemos ponderar la información de cada mensaje por la probabilidad de que se produzca:

$$H = \sum_{k=1}^{N} p_k \cdot I(m_k) = -\sum_{k=1}^{N} p_k \cdot \log_2 p_k \quad (5)$$

Esta función se conoce con el nombre de entropía y establece un límite al número de bits medio con el que se podrán codificar los mensajes de una fuente. Ningún codificador no podrá obtener códigos con un número de bits medio inferior a la entropía de la fuente. El nombre de entropía de la función anterior se debe a que su fórmula se parece mucho a la entropía termodinámica de L. Boltzman. Recientemente, algunas teorías físicas tienen relacionado ambas entropías (JD Bekenstein)

7.3 Códigos de longitud variable. Conceptos básicos

La idea básica de los códigos de longitud variable es asignar palabras código de longitudes diferentes en función de la probabilidad de los mensajes. Los mensajes más probables se codificarán con palabras con un número menor de bits que los mensajes menos probables.

Considere como ejemplo los mensajes de la fuente que se resumen en la tabla, en la que los códigos asignados a cada mensaje se han elegido para que cumplan una serie de restricciones que detallaremos más adelante.

mensajes	probabilidad	Código
m1	0.4	0
m2	0.3	10
m3	0.1	1100
m4	0.1	1101
m5	0.1	1110

Podemos calcular el número medio de bits que utiliza el código propuesto ponderando el número de bits que se utiliza para cada mensaje con la probabilidad de que se produzca el mensaje.

$$N_{medio} = \sum_{k=1}^{N} p_k \cdot N_k = 1 \cdot 0,4 + 2 \cdot 0,3 + 4 \cdot 0,3 = 2,2 \text{ bits}$$

El número de bits medio debe ser superior a la entropía. En efecto, la entropía representa la información media de la fuente, por lo que cualquier código práctico que utilizamos tendrá un número de bits media mayor o igual que la entropía. En nuestro caso,

$$H = -\sum_{k=1}^{N} p_k \cdot \log_2 p_k = 2,04 \text{ bits}$$

Problema.

Determine utilizando la tabla 2 la secuencia de bits con que quedaría codificada la secuencia de mensajes siguiente: m3 m5 m1 m1 m3 m2 m1 m1

Solución

Para obtener la secuencia de bits simplemente debemos sustituir cada mensaje por el código correspondiente. En nuestro caso obtenemos:

1100 1110 0 0 1100 10 0 0

Introducción a la Codificación de Canal.

1. Redundancia estructurada: conceptos básicos

Introducir redundancia en un código es añadir un conjunto de bits que posteriormente nos permitan verificar que el grupo de bits que recibimos (al que llamaremos *palabra* o *palabra código*) concuerda con la transmitida. Se dice que la redundancia se introduce de forma estructurada porque los bits adicionales se determinan utilizando procedimientos bien definidos y conocidos tanto por el receptor como por el transmisor.

1.1. Ejemplo 1. Códigos de paridad simple

El ejemplo más sencillo de introducción de redundancia estructurada son los códigos de paridad simple. En la figura 1 se muestra cómo se determina la paridad simple para diferentes palabras código. En nuestro ejemplo, las palabras código tienen originalmente una longitud de 6 bits, por lo que después de añadir la redundancia tendremos palabras con una longitud de 7 bits. Para añadir el bit de paridad se utiliza el criterio de que el número total de unos de cualquier palabra sea par. Así, si una palabra ya tiene originalmente un número par de unos, el bit de paridad tomará el valor cero. En cambio, si en la palabra original el número de unos es impar, el bit de paridad tomará el valor uno. De este modo, después de introducir el bit de redundancia podemos garantizar que todas las palabras código tienen un número par de unos. El código descrito se denomina de *paridad par*. También podría definirse un código de paridad impar, en el que el número de unos de la palabra codificada siempre es impar.

La redundancia se ha introducido siguiendo un procedimiento sistemático (el número total de unos debe ser par) que confiere una propiedad única a todas las palabras código. Esta propiedad es la que aprovecha el receptor para verificar que no se ha producido ningún error en la transmisión. En efecto, si la palabra recibida tiene un número par de unos, el receptor la dará por buena, mientras que, si el número de unos es impar, el receptor podrá asegurar que se ha producido algún error (al menos uno) durante la transmisión.

Este procedimiento resulta eficiente siempre que el número de errores que se produzca sea un número impar. En efecto, si se producen dos errores, la palabra código volverá a tener un número par de unos, por lo que no podrá detectarse como palabra incorrecta. Lo mismo ocurre cuando se produce cualquier número par de errores. Todos estos casos se corresponden con situaciones de errores que no pueden ser detectadas por nuestro código. En general, para cualquier código, siempre existirán patrones de errores que

no podrán ser detectados. Por lo tanto, se dice que el código de paridad simple permite detectar cualquier número impar de errores en una palabra código.

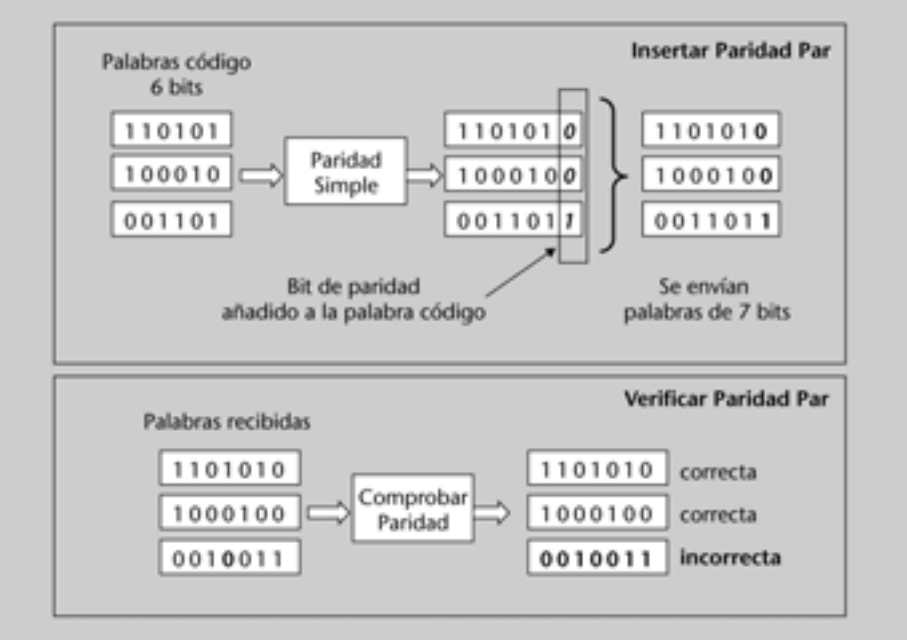


Figura 1. Códigos de paridad par: inserción y verificación

Problema 1

Supongamos que en un determinado sistema de comunicaciones la probabilidad, *p*, de cometer un error en un bit es $p = 10^{-4}$. Las palabras código originales son de 7 bits y se introduce un bit de paridad par adicional. Determinad la probabilidad de que en una palabra código se hayan producido errores que no puedan ser detectados.

Solución

La probabilidad de que se hayan producido errores que no puedan ser detectados por el código de paridad es:

$$P_{\text{error no detectado}} = P_{2\ \text{errores}} + P_{4\ \text{errores}} + P_{6\ \text{errores}} + P_{8\ \text{errores}}$$

Es decir, es la probabilidad de que se produzcan dos errores más la de que se produzcan 4, 6 u 8, ya que cualquier número impar de errores será detectado correctamente.

Para determinar la probabilidad de que se produzcan dos errores en la palabra código, debemos tener en cuenta todas las posibles situaciones en las que tendremos 2 bits erróneos en una palabra de 8 bits y ponderar cada una de estas situaciones por la probabilidad de que se produzca. Así:

$$p_{2\ errores} = \binom{8}{2} \cdot p^2 (1-p)^6$$

Donde el número combinatorio nos indica la totalidad de casos en los que se producen dos errores en una palabra de 8 bits, siendo p^2 la probabilidad de que se produzcan dos errores en estas posiciones y $(1-p)^6$ la probabilidad de que no se produzcan errores en el resto de las posiciones. Teniendo en cuenta la expresión anterior, la probabilidad de que se produzcan errores en una palabra y que no sean detectados por el código será:

$$p_{\text{error no detectado}} = \binom{8}{2} \cdot p^2 \cdot (1-p)^6 + \binom{8}{4} \cdot p^4 \cdot (1-p)^4 + \binom{8}{6} \cdot p^6 \cdot (1-p)^2 + \binom{8}{8} \cdot p^8$$

Recordad que...

... el número combinatorio $\binom{8}{2}$ representa todas las posibles combinaciones para tomar dos elementos en una palabra de 8 bits. El cálculo se realiza teniendo en cuenta la siguiente expresión:

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

Particularizando para el valor $p = 10^{-4}$, obtenemos:

$$p_{\text{error no detectado}} = \frac{8!}{2!6!} \cdot 10^{-8} \cdot (1-10^{-4})^6 + \frac{8!}{4!4!} \cdot 10^{-16} \cdot (1-10^{-4})^4 + \frac{8!}{6!2!} \cdot 10^{-24} \cdot (1-10^{-4})^2 + 10^{-32}$$
$$= 2{,}7983 \cdot 10^{-7}$$

Observad que únicamente el primer término de la suma es significativo.

1.2. Ejemplo 2. Códigos rectangulares

Los códigos rectangulares, también conocidos como *códigos de producto*, son una variante directa de los códigos de paridad. Esta variante permite visualizar, de forma muy sencilla, una estrategia para la corrección de errores. Para aplicar un código rectangular, los bits del mensaje original deben organizarse en una matriz. En la figura 2 se muestra un mensaje original de 20 bits organizado en una matriz de 5 columnas y 4 filas. Los bits de redundancia se calculan como un código de paridad simple, aplicándolo primero a las filas y después a las columnas (o viceversa). En la figura se muestran todos los bits resultantes en los que la última columna y la última fila se corresponden a los bits de paridad. Los bits pueden transmitirse en el orden que se desee, siempre que el transmisor y el receptor se pongan de acuerdo. Así, podremos transmitir todas las filas, una detrás de otra. El receptor irá situando los bits recibidos en una matriz de 6 columnas por 5 filas, por lo que los bits de información y de redundancia estarán dispuestos en el mismo orden que en el transmisor.

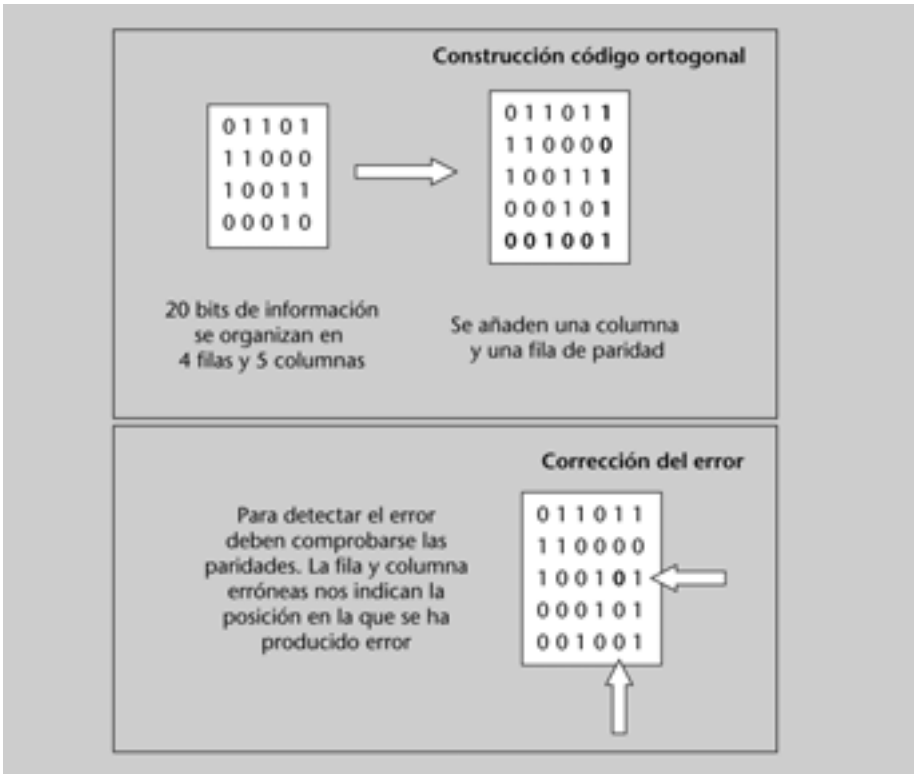


Figura 2. Construcción y corrección de errores en un código ortogonal

En esta misma figura se muestra cómo puede realizarse la corrección de un bit erróneo. El receptor realiza la comprobación de paridad de cada una de las filas, *detectando* como errónea la fila en la que se ha producido el error. Posteriormente, al realizar la comprobación de la paridad de cada una de las columnas, *detectará* la columna en la que se ha producido el mismo error. De esta forma, podemos aprovechar esta información para saber la fila y la columna en la que se ha producido el error. Es obvio que, si sabemos la posición en la que se ha producido el error, su corrección consiste simplemente en cambiar su valor.

El código propuesto sólo puede garantizar la **corrección de un error en un bit**. Es fácil pensar situaciones en las que se produce más de un error y en las que sería posible determinar las posiciones en las que se ha producido (siempre que la columna y la fila de los dos errores sean distintas). No obstante, si se producen en una misma fila o columna no será posible detectarlos. En estos casos, se dice que los errores superan las capacidades de corrección del código.

1.3. Tasa del código

Los códigos de paridad y los códigos ortogonales son muy simples, pero nos permiten ilustrar algunas de las definiciones sobre los códigos de protección de error que tienen validez general. Tomando como referencia la figura 3 podemos ver que, en general, a partir de un paquete de *k* bits de información (mensaje original) se añaden *r* bits de redundancia. Los bits de redundancia también reciben a menudo el nombre de *bits de paridad*. El número total de bits de cada palabra código es la suma de los bits de redundancia más los bits del mensaje original n = k + r. Un código con estas características se identifica como código (n, k). La *tasa de redundancia* de un código se define como el cociente entre los bits de redundancia y los bits totales (n – k/n) y proporciona una idea del porcentaje de bits de redundancia que contiene un código. Análogamente, la *tasa de un código* se define como el cociente entre el número de bits del mensaje y el número de bits totales de una palabra código (k/n). La tasa del código nos da una idea de la relación entre la información útil y la información total que contienen los mensajes. Así por ejemplo, un código con una tasa 2/5 nos indica que contiene dos bits de información útil por cada 5 bits que recibimos. En la figura 3 se representan de forma esquemática estos conceptos.

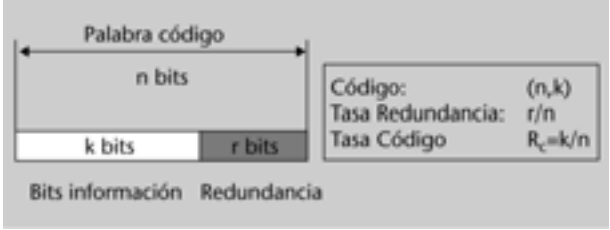


Figura 3. Definiciones básicas de redundancia y tasa de código

La tasa del código nos da una idea del coste que tiene proteger la información de los posibles errores de canal. En efecto, al añadir los bits de redundancia deberemos transmitir más bits que los estrictamente necesarios. Esto significa que, si deseamos realizar la transmisión en tiempo real, deberemos transmitir un total de n = k + r bits en el mismo tiempo que antes transmitíamos *k* bits, lo que supone que necesitaremos un mayor ancho de banda. El aumento del ancho de banda debido a la introducción de la redundancia está directamente relacionado con la tasa de bits. En efecto, si antes debíamos transmitir *k* bits en un tiempo T y ahora debemos transmitir *n*, la duración de un bit pasa de ser T/k a T/n, por lo que el ancho de banda aumenta aproximadamente en un factor n/k. Así pues, el aumento del ancho de banda coincide con el inverso de la tasa del código, que también se conoce como *factor de expansión*.

La *tasa de un código* es la relación entre el número de bits de información y el número de bits totales que contiene:

$$R_c = k / n$$

El inverso de la tasa de un código da una idea de cómo el uso del código aumenta el ancho de banda de la señal y se denomina *factor de expansión* de un código:

$$B = \frac{W_{coding}}{W_{no\ coding}} = \frac{1}{R_c} = \frac{n}{k}$$

Veremos cómo se aplican todos estos conceptos en el siguiente problema para los códigos de paridad y los códigos ortogonales.

Problema 2

Determinad la notación del código y las tasas de código y redundancia para los ejemplos 1 y 2 presentados en este apartado.

Solución

Para el código de paridad simple, hemos visto que para cada seis bits de información útil (k = 6) añadíamos un bit de paridad (r = 1). El número total de bits de una palabra código es de n = 7. Teniendo en cuenta estos valores, el código puede designarse como (7,6) con una tasa de redundancia de 1/7 y una tasa de código $R_c = 6/7$.

Para el código ortogonal, hemos visto que la información útil se organizaba en una matriz de 4 filas por 5 columnas; por tanto, cada palabra código (o bloque de información) tiene un total de 20 bits útiles (k = 20). Al calcular las paridades por filas y por columnas se añaden 10 bits de información (r = 10; 4 filas + 6 columnas). Con estos resultados obtenemos que el código debe denotarse como (30,20), su tasa de redundancia es 10/30 = 1/3 y la tasa del código es de $R_c = 20/30 = 2/3$.

2. Estrategias para el control de errores

La introducción de redundancia de forma estructurada en una palabra código permite que el receptor pueda detectar o corregir, según el caso, la presencia de errores. En este apartado veremos las posibles estrategias de control de error que pueden utilizarse en función de las características del canal de comunicaciones o de la aplicación, es decir, intentaremos justificar en qué tipo de situaciones resulta más conveniente utilizar técnicas de detección de errores o técnicas de corrección.

Como primera consideración, cabe aclarar que *detectar* la presencia de un error consiste en analizar la palabra código recibida y determinar que no cumple las reglas mediante las que se ha añadido la redundancia. Por tanto, sabemos que la palabra es incorrecta pero no conocemos en qué posición se ha producido el error.

En cambio, la *corrección* de errores implica no sólo determinar que la palabra código es incorrecta, sino también conocer las posiciones en las que se han producido los errores para poder proceder a corregirlos.

A partir de estos razonamientos, parece lógico que la corrección de errores será siempre más compleja que la detección, donde entendemos por *compleja* que requiere un mayor número de bits de redundancia y que aumenta la complejidad computacional para realizar la codificación y la decodificación. La necesidad de un mayor número de bits de redundancia también puede interpretarse como un aumento del ancho de banda de la señal que se va a transmitir.

Deben distinguirse, pues, dos estrategias principales para controlar los errores en el canal. La primera se basa en utilizar códigos que permitan que el receptor *detecte* los errores que se han producido en el canal solicitando la retransmisión de aquellas palabras que se han recibido de forma incorrecta. Esta técnica se conoce con el acrónimo inglés ARQ (*automatic repeat request* o también *automatic retransmission query*), que indica que el receptor pide la retransmisión del mensaje original cuando detecta que se ha producido un error.

Esta estrategia tiene como principales ventajas:

- Los códigos de detección de errores son fáciles de calcular y detectar.
- La redundancia añadida al mensaje es menor que para un código de corrección.
- El aumento del ancho de banda debido a la introducción de la redundancia también es menor.

Los principales inconvenientes son:

- Algunas aplicaciones, como en la transmisión de audio y vídeo, no se aceptan retardos en la recepción de los paquetes, ya que estos retardos podrían interrumpir el flujo continuo del reproductor.

- En muchas aplicaciones, el canal de retorno no está disponible.

Las técnicas **ARQ** tienen diversas variantes y se utilizan con profusión en la descarga de archivos por Internet (TCP/IP).

La otra alternativa se conoce con el nombre de **FEC** (*forward error correction*) y utiliza códigos de corrección de errores que permiten que el receptor pueda restaurar por sí solo la información original en el caso de que se hayan producido errores (siempre que los errores estén dentro de las capacidades correctoras del código).

La necesidad del canal de retorno impone una restricción muy importante a las estrategias de control ARQ. En efecto, pensemos por ejemplo en aplicaciones como la grabación de señales de vídeo o audio en soporte óptico o magnético. Una vez que la información ha sido grabada, si han aparecido errores en la señal registrada (generalmente debido a la degradación del soporte), es imposible pedir la regrabación de la información original. Por eso, en todos los sistemas de grabación de audio y vídeo se utilizan técnicas FEC que permiten la corrección de los errores en el propio decodificador. Los sistemas digitales de registro de señales de audio (CD-audio *compact disc audio*, DAT *digital audio tape*, minidisc, DVD-Audio *digital versatile disc-audio*, etc.) o de registro de vídeo (DVD-Vídeo) utilizan distintas variantes de códigos con estrategias FEC que se analizarán más adelante en este módulo. Otro ejemplo típico son los sistemas de difusión (televisión digital, audio digital) que tampoco permiten que el receptor resolicite la transmisión de las palabras recibidas de forma incorrecta. Así pues, la Televisión Digital Terrestre (DVB-T, *digital video broadcasting-terrestrial*) o la radio digital DAB (*digital audio broadcasting*) también utilizan códigos con estrategias FEC. Otro ejemplo típico es la difusión de contenidos audiovisuales por Internet (*streaming*). En este caso, el motivo principal para utilizar estrategias FEC es que la propia naturaleza de las señales audiovisuales requiere la reproducción continua sin que aparezcan las posibles interrupciones que puede introducir la solicitud de retransmisiones.

Elegir si en un determinado sistema de comunicaciones es más conveniente utilizar estrategias de corrección ARQ o FEC es un problema complejo que depende de muchos factores, entre los que destacamos, a modo de resumen, los siguientes:

Complejidad del codificador/decodificador. Algunos sistemas para la corrección de errores FEC pueden tener complejidades computacionales consi-

derables, por lo que sólo pueden ser utilizados en equipos terminales de coste elevado que puedan permitirse incorporar procesadores avanzados. En cambio, las estrategias ARQ tienen una implementación más simple que las FEC, por lo que pueden ser utilizados en equipos más económicos.

Aumento del ancho de banda. En general, para realizar la corrección de errores es necesario introducir más redundancia que para permitir su detección. Esto supone que el aumento de ancho de banda debido al código suele ser mayor cuando se utilizan estrategias FEC.

Probabilidad de error del canal. El número de errores que se producen en el sistema de comunicación (antes de su corrección) puede ser muy elevado, lo que hace que las técnicas ARQ resulten muy ineficientes (ver como ejemplo el Problema 3).

Contenido del mensaje. En general, cuando se transmiten archivos de datos podemos permitirnos retardos en los bloques para los que solicitamos la retransmisión. El receptor puede gestionar un *buffer* de memoria en el que se recompone el orden original de los bloques. En aplicaciones en las que intervienen señales de audio o vídeo, estos retardos no son admisibles, por lo que no suelen utilizarse técnicas de ARQ.

Características del canal de comunicaciones. Hemos visto que en los sistemas en los que no existe canal de retorno es necesario utilizar estrategias FEC. Los ejemplos más típicos son los sistemas de difusión audiovisual (como televisión y radio), el *streaming* de Internet y la grabación de datos en soportes ópticos o magnéticos. Cuando existe canal de retorno, pueden utilizarse distintas variantes de la estrategia ARQ.

Veremos dos ejemplos de cómo pueden incidir las características del canal en las estrategias utilizadas en las técnicas ARQ. En el primer caso, consideraremos un canal *full-duplex* y, en el segundo, un canal *half-duplex*. Recordamos estas definiciones en el recuadro de la derecha.

Un canal es *half-duplex…*

… cuando puede transmitir en los dos sentidos pero no de forma simultánea: primero debe realizarse en un sentido y luego en el otro. En un canal *full-duplex* se puede transmitir en ambos sentidos de forma simultánea.

2.1. Ejemplo 3. ARQ en canales *full-duplex*

En la figura 4 se muestra de forma esquemática una estrategia ARQ denominada *de repetición selectiva* (*selective repeat* ARQ) y que requiere un canal *full-duplex* (la comunicación puede realizarse en los dos sentidos de forma simultánea).

En este ejemplo, el transmisor va enviando continuamente los bloques del mensaje en el orden preestablecido y recibe el reconocimiento (ACK –*acknowledge*) del receptor. Cuando el ACK es negativo (NAK – *non-acknowledge*) el transmisor vuelve a repetir el bloque que se ha recibido de forma errónea. Si no aparecen errores en el canal, el retardo entre el transmisor y el receptor depende del tiempo de propagación del bloque. Si se producen errores (bloque número 4), el re-

ceptor deberá esperar a recibir este bloque de forma correcta si desea realizar la decodificación secuencial del mensaje. Observad que esta estrategia ARQ requiere que se realice una comunicación bilateral completa (*full-duplex*) entre el transmisor y el receptor. Es interesante comparar esta variante de ARQ con la que se introduce en el siguiente problema.

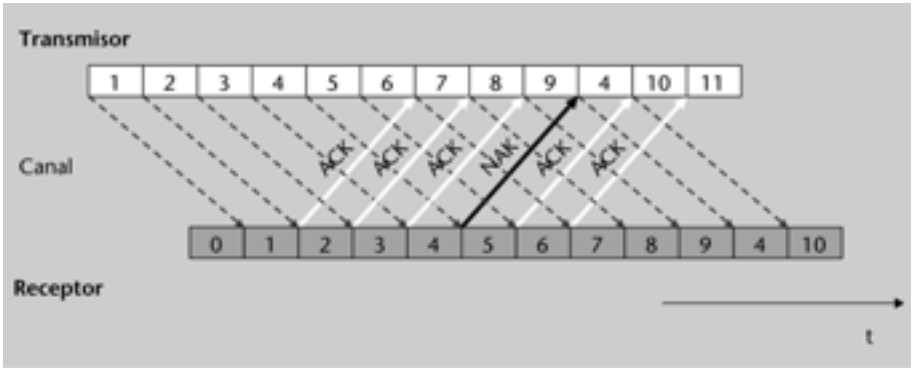


Figura 4. Ejemplo de una variante de estrategia ARQ (*selective repeat*). En este caso, se requiere disponer de un canal de comunicaciones *full-duplex*.

2.2. Ejemplo 4. ARQ en canales *half-duplex*

En este ejemplo se ilustra el mecanismo de control de errores de una variante de ARQ denominada Parada y Espera (*Stop&Wait*). En la figura 5 se muestra el esquema general de los paquetes enviados, su propagación por el canal y la respuesta de reconocimiento del receptor. Este mecanismo consiste en que el transmisor envía un paquete de datos y espera a recibir la confirmación de que se han recibido correctamente. Durante el tiempo de espera no se transmite más información, por lo que el canal puede ser *half-duplex*, de manera que el transmisor, una vez enviado el paquete, pasa (conmuta) a modo de recepción para esperar el ACK/NAK del receptor.

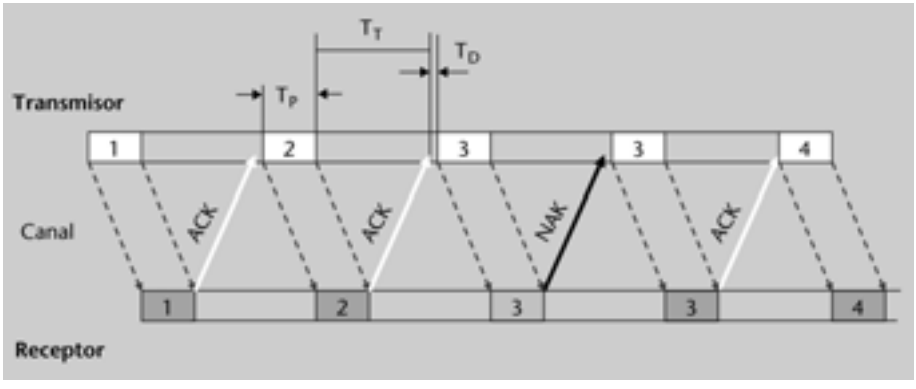


Figura 5. Esquema simplificado de la variante *Stop&Wait* para la estrategia de control de errores ARQ

En este ejemplo queremos ver los efectos que tienen en este esquema de control de errores algunos de los parámetros básicos como la distancia entre el emisor y el receptor y la probabilidad de error. Supongamos que el emisor envía paquetes de 210 bits (con los códigos de detección de error ya incluidos) a una velocidad de 100.000 bits/s, es decir, la duración de cada bit es de 10^{-5} s.

La distancia entre el transmisor y el receptor es de 300 km y podemos suponer, en primera aproximación, que el mensaje se propaga a la velocidad de la luz, que tomaremos como c = 3 10^8 m/s.

El paquete de ACK o NACK tiene una duración de 10^{-4} s, y el emisor responde con la repetición del paquete anterior o con un nuevo paquete sin retardo.

a) Determinaremos la tasa efectiva de transmisión de bits entre el emisor y el receptor, suponiendo que no se producen errores en los paquetes.

Para determinar la tasa efectiva de transmisión de bits entre el emisor y el receptor, debemos calcular el tiempo total que se invierte para enviar los 210 bits de cada paquete. Este tiempo debe incluir:

T_P = Tiempo para transmitir los 210 bits a 100 kbits/s
T_P = 210 bits x 10^{-5} = 2,10 10^{-3} s
T_T = Tiempo de ida y vuelta de un mensaje
T_T = 2x300x10^3(m)/3x10^8 (m/s) = 2 10^{-3} s
T_D = Duración del paquete ACK + retardos de respuesta
T_D = 10^{-4} s

Así, el tiempo total necesario para la transmisión de los 210 bits es T = 4,2 10^{-3} s, por lo que la tasa efectiva de transmisión de bits es:

$$R = 210 \text{ bits} / 4{,}2\ 10^{-3} \text{ s} = 50.000 \text{ bits/s}$$

Para nuestro ejemplo, la tasa de bits se ha reducido a la mitad debido a la estrategia de control de errores.

b) Determinaremos ahora la tasa efectiva de transmisión suponiendo que la probabilidad de que se produzca algún error en un paquete sea de 10^{-2}.

En este caso, debemos tener en cuenta que algunos paquetes de 210 bits se recibirán de forma incorrecta, por lo que volverán a transmitirse y el tiempo medio para la transmisión de un paquete aumenta.

Podemos calcular el tiempo medio para la recepción de un paquete mediante la siguiente ecuación:

$$T_m = T \cdot (1 - p) + 2 \cdot T \cdot p(1 - p) + 3 \cdot T \cdot p^2 \cdot (1 - p) + \ldots.$$

Donde T representa el tiempo total calculado en el apartado anterior. Observad que el primer término de esta ecuación es el tiempo total por la probabilidad de que no se cometa error. El segundo término es el tiempo que tardaríamos en transmitir un paquete y volverlo a retransmitir por la probabilidad de que la primera vez se reciba incorrectamente y la segunda se reciba

correctamente. El tercer término corresponde a haber recibido dos paquetes con error y el tercero correctamente, y así sucesivamente. Para determinar el valor de la progresión anterior, reordenamos los términos:

$$T_m = T \cdot (1-p) \cdot \left[1 + 2 \cdot p + 3 \cdot p^2 + \ldots\right] = T(1-p) \cdot \sum_{k=0}^{\infty} (k+1)p^k = T(1-p) \cdot S(p)$$

Donde hemos definido S(p) como:

$$S(p) = \sum_{k=0}^{\infty} (k+1)p^k = \frac{d}{dp}\left(\sum_{k=0}^{\infty} p^{k+1}\right) = \frac{d}{dp}\left(\frac{p}{1-p}\right) = \frac{1}{(1-p)^2}$$

En esta última línea, utilizamos varias propiedades matemáticas que merece la pena comentar. En la segunda igualdad nos damos cuenta de que los términos $(k+1)p^k$ pueden expresarse como la derivada respecto a p de p^{k+1}. Posteriormente, utilizamos la formula de una progresión geométrica convergente para sumar todos los términos (el primer término dividido por 1 menos la razón). Finalmente, derivamos respecto a *p* el resultado.

Sustituyendo este resultado en la expresión anterior, obtenemos:

$$T_m = \frac{T}{1-p}$$

Observad que, para la probabilidad de error que tenemos (10^{-2}), el tiempo total no se modifica de forma muy apreciable, dando lugar a una tasa efectiva de 49.500 bits/s. No obstante, la dependencia de T_m con el inverso de $(1 - p)$ nos dice que, si las tasas de error por paquete son importantes, el control de errores mediante estas técnicas puede resultar muy ineficiente.

Para finalizar, cabe comentar que el método *half-duplex* tiene una dependencia directa con la distancia entre los dos extremos de la comunicación. En efecto, el tiempo medio para recibir un paquete depende de modo directo del tiempo de transmisión, que a su vez depende de la distancia entre los terminales. Esto no era así para la estrategia ARQ del ejemplo 3, en el que la distancia sólo podía afectar al retardo global pero no a la velocidad de transmisión.

3. ¿Para qué sirven los códigos de protección de errores?

La respuesta a esta pregunta puede parecer un tanto inmediata aunque, si nos detenemos a reflexionar, veremos que no resulta tan evidente. En efecto, hemos visto que la estrategia para protegernos frente a los posibles errores consiste en introducir redundancia adicional en los mensajes que pretendemos transmitir. Esto significa que para protegernos debemos pagar un precio: transmitir más bits que los estrictamente necesarios. La consecuencia inmediata es que el ancho de banda del canal necesario para transmitir la información debe aumentar. En general, veremos que, si queremos proteger la información original frente a un gran número de errores, deberemos usar códigos con tasas bajas, que suponen un aumento considerable del ancho de banda. La tasa de un código es un indicativo del coste en ancho de banda que representa la introducción de este código

Si continuamos reflexionando sobre el resultado anterior, nos daremos cuenta de que al aumentar el ancho de banda se reduce el tiempo disponible para transmitir cada uno de los bits. Por lo tanto, la energía dedicada a la transmisión de un bit en el mensaje codificado es menor que la dedicada a transmitir un bit en la información sin codificar (suponemos que la potencia del transmisor se mantiene constante). Esto significa que la probabilidad de que se produzca un error en el mensaje codificado es mayor que en el mensaje sin codificar. La cuestión clave es si este aumento de la probabilidad de error queda compensado por la capacidad de corrección de errores del código.

La respuesta a la pregunta es que generalmente sí sale a cuenta, que obtenemos una ganancia neta, que la capacidad de corrección de los códigos de canal compensa la pérdida de energía en cada uno de los bits del mensaje codificado. La clave de la respuesta a esta pregunta puede encontrarse en la gráfica de la figura 6, en la que se compara la probabilidad de error que se obtiene con un mensaje sin codificar y con un mensaje codificado. Las probabilidades de error se representan en función de la relación E_b/N_0 que representa la relación entre la energía dedicada a un bit E_b y la densidad espectral de ruido N_0. Es evidente que, a medida que aumentamos la energía de los bits, la probabilidad de error decrece, ya que el mensaje se ve menos afectado por el ruido. La gráfica también indica que existe una región en la que el código de protección resulta rentable, ya que se obtienen probabilidades de error menores que con el mensaje sin codificar. En cambio, existe una zona en la que no resulta rentable codificar el mensaje, ya que la probabilidad de error con el código es mayor que sin el código. En esta región se dice que los errores superan las capacidades de corrección del código. Se producen tantos errores que la corrección no sale a cuenta.

En resumen, es importante comprender que existe un delicado compromiso entre varios factores implicados y relacionados: la redundancia introducida, el aumento del ancho de banda, la tasa del código y la potencia con la que se transmiten los mensajes. Todos estos factores inciden en la probabilidad de error final del sistema de comunicaciones, y su selección adecuada exige estudios rigurosos sobre las características de los canales y los códigos que deben utilizarse. La elección de un código u otro en un sistema de comunicaciones real exige profundos estudios y simulaciones de las condiciones en las que se realizarán las comunicaciones, las características del canal y los criterios de calidad que queremos obtener.

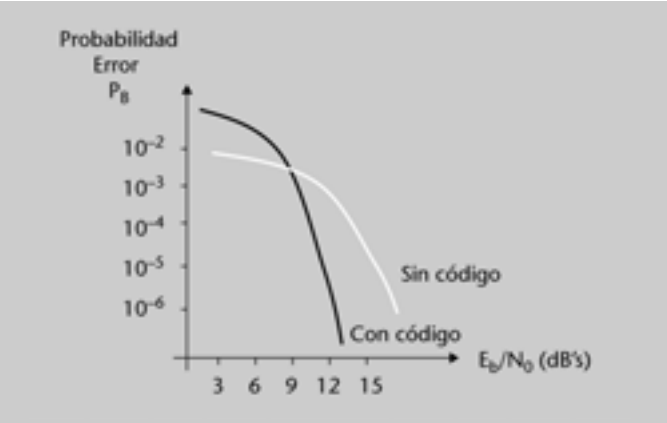


Figura 6. Ejemplo de la mejora en probabilidad de error en un sistema de comunicaciones con códigos correctores