# Universidad Politécnica de Cartagena


**Escuela Técnica Superior de Ingeniería de Telecomunicación**

**PRÁCTICAS DE TRANSMISIÓN DE DATOS**

## Práctica 4: Codificación de canal

Profesor:

Javier Vales Alonso

## 1. INTRODUCCIÓN

En esta práctica se estudiarán qué son y por qué son necesarios los esquemas de codificación de canal. Se presentarán algunos ejemplos simples de codificadores de canal, como el de control de paridad, repetición o Hamming.

## 2. OBJETIVOS

Teóricos:

- Entender la relación entre capacidad y tasa de transmisión.
- Presentar esquemas de codificación de canal básicos

Prácticos:

- Implementar un codificador de canal por repetición
- Implementar un codificador de canal por control de paridad simple
- Implementar un codificador de canal por control de paridad múltiple Hamming (7,4)

## 3. DESARROLLO DE LA PRÁCTICA

La práctica se desarrollará en dos sesiones y se divide en las siguientes partes:

1. Primero veremos un ejemplo donde se verá la diferencia entre tasa de transmisión, tasa de transmisión efectiva y capacidad, para ello usaremos una codificación de canal por repetición.
2. A continuación crearemos en matlab la función **paridad.m** que nos servirá para crear codificadores de canal.
3. Después implementaremos un esquema de codificación de canal por comprobación simple de la paridad.
4. Por último, implementaremos un esquema de codificación de canal por comprobación *múltiple* de la paridad, de tipo Hamming(7,4).

Para la realización de esta práctica puede serle de utilidad disponer y llamar a la función **canal.m** creada en la práctica 5 (si no la tiene puede usar alternativamente la función **canalBS.m** facilitada en el aula virtual). También necesitará la función **ascii.m** facilitada en prácticas anteriores.

### *3.1.* CAPACIDAD VERSUS TASA

Consideraremos (a modo de ejemplo) a lo largo de esta práctica el Canal Binario Simétrico, tal y como se definió en la práctica anterior, con $p$=0.9.

**Verificación:** *Calcule la capacidad de este canal con la función que implemento en la práctica anterior. Debería obtener 0.531 bits/símbolo.*

Para este canal, supongamos (también a modo de ejemplo) que la tasa de transmisión es de 1 Ksímbolo/seg. En este caso tenemos tres caracterizaciones aparentemente contradictorias:

- Tasa de transmisión: 1000 bits/s (ya que un símbolo es un dígito binario).
- Capacidad: 531 bits/s
- Tasa de transmisión efectiva (es decir, símbolos transmitidos sin error): 900 bits/s.

***¿Cuál es la correcta?*** Claramente podemos descartar la primera, ya que está claro que habrá símbolos con error entre los recibidos. Sin embargo, ***¿por qué la capacidad es de 531 bits/s si el canal es capaz de transmitir a 900 bits/s sin errores?*** El quid es ***¿cómo sabemos en el receptor qué bits se han alterado?*** Para poderlo saber necesitamos intercalar entre nuestros datos redundancia que permita saber si el mensaje tiene errores, si es posible corregirlos y sino descartar los datos. **El teorema de Shannon indica que del canal con tasa 1000 bits/s como mínimo vamos a necesitar 469 bits/s como redundancia para llegar a alcanzar una transmisión libre de errores máxima de 531 bits/s.** Vamos a ver esto a través de varios ejemplos.

## *3.2.* CANAL SIN CODIFICACIÓN

Consideremos la transmisión sin ninguna codificación. Los bits de la fuente se transmitirán tal cual. Vamos a transmitir la cadena `'Hola mundo'` codificada en ASCII (recuerde que tiene disponible de prácticas anteriores la función `ascii.m`). Para emular la transmisión puede usar su función de canal `canal.m` o usar la función que se le proporciona en aula virtual `canalBS.m`

**Verificación:** *Realice el siguiente experimento:*

Emule la transmisión de la cadena `'Hola mundo'` 1000 veces. La transmisión se considera correcta si tiene a la salida la misma cadena. Calcule el ratio de transmisiones correctas ($r$). Calcule la tasa de transmisión real: $r \times 1000$ *bits/s.* Para ello vamos a partir del siguiente código, que debe completar donde se indica:

```
N = 1000;
f = ascii('hola mundo');

exito = 0;
for i=1:N
    tx = % RELLENAR, TX ES EL FLUJO A TRANSMITIR
    rx = canalBS(tx, 0.9); % SE OBTIENE EL FLUJO EN RECEPCION
    if strcmp(rx,tx) % SE COMPRUEBA SI LA TX FUE CORRECTA
       exito = exito + 1;
    end
end

r = exito / N
tasa = % CALCULAR LA TASA
```

**¡¡¡Habrá llegado a la triste y penosa conclusión de que su canal no transmite nada!!!**

Podría argumentar que el mensaje es "muy largo" y por eso el experimento arroja este resultado, pero no olvide que en una situación real no se sabe que mensaje se está transmitiendo y por tanto en el extremo receptor ni siquiera podríamos saber si el mensaje es correcto o no. Lo que estamos haciendo "manualmente" es la función de un protocolo de enlace o de datos. En esos casos al transmitir una (trama o) paquete (en general bastante más largos que el mensaje `'Hola mundo'`) si éste tiene un error se tira. Por tanto, no parece que la longitud del mensaje sea excesiva.

## *3.3.* CODIFICACIÓN DE CANAL CON REPETICIÓN *R*

Esta claro que vamos a necesitar agregar redundancia a los mensajes para poder transmitir algo. Esto es lo que hace un codificador de canal. El más simple es repetir un símbolo ***R*** veces. Por ejemplo, si ***R=3*** para el '0' se transmite '000' y para el '1' se transmite '111'.

**EJERCICIO: Codifique la función repeticion**

```
function [nuevoflujo] = repeticion(flujo, R)
% Repite R veces cada bit del flujo y lo devuelve en nuevo flujo

end
```

**Verificación:** *Realice el siguiente experimento:*

Repita el experimento del punto anterior para ***R=3:2:11***. Se considerará que el símbolo recibido es el que tenga el voto mayoritario. Calcule la tasa de transmisión real: $r \times 1000/R$ *bits/s.* Ahora sí conseguirá transmitir algo, ¿para qué ***R*** alcanza la máxima tasa de transmisión real? ¿Cuál es ésta? Observe que para ***R*** mayores mejora la *r,* pero la excesiva redundancia baja la tasa de transmisión real. Para ello amplíe el código siguiente:

```
N = 1000;
f = ascii('hola mundo');

for R= % Falta, las configuraciones para las que se hace el experimento
      exito = 0;
      for i=1:N
            tx = repeticion(f, R);
            rx = canalBS(tx, 0.9); % SE OBTIENE EL FLUJO EN RECEPCION

            % FALTA
            rxcorregido = % FALTA
            % FALTA

            if strcmp(rxcorregido,tx) % TX CORRECTA?
                  exito = exito + 1;
             end
      end

      r = exito / N
      tasa = % CALCULAR LA TASA
end
```

El tipo que esquema que hemos probado se denomina Código de Corrección de Errores (CCE) ya que es capaz de corregir en el extremo receptor un mensaje donde hay ciertos errores. En este caso es capaz de corregir cambios hasta en `floor`((R-1)/2) símbolos.

Otro modo de emplear la repetición es como Código de Detección de Errores (CDE), en este caso se detectará un error si hay cambios hasta en R-1 símbolos.

**Verificación:** *Realice el siguiente experimento:*

Repita el experimento del punto anterior para ***R=2:2:10***. Si se detecta un error se repite la solicitud del símbolo hasta que no se detecten errores. Se calculará *t* como el ratio de transmisiones por símbolo. Por ejemplo, si se han transmitido 100 símbolos y para ello ha habido que hacer 125 transmisiones (25 de ellas retrasmisiones) *t*=1.25. Asimismo, se calculará *r* como antes (**note que si cambian *R* símbolos el error no se detecta y hay que descartar todo el mensaje**). Calcule la tasa de transmisión real: *r×1000/(R×t) bits/s* ¿Para qué ***R*** alcanza la máxima tasa de transmisión real? ¿Cuál es ésta?

```
N = 1000;
f = ascii('hola mundo');

for R= % Falta, las configuraciones para las que se hace el experimento
      exito = 0;
      transmisiones = 0;
      mensajes = 0;

      for i=1:N
            tx = repeticion(f, R);

            mensajes = mensajes + 1;
            while true
                  transmisiones = transmisiones + 1;
                  rx = canalBS(tx, 0.9); % SE OBTIENE RX

                  % FALTA
                  errordetectado = % Completar
                  % FALTA

                  if ~errordetectado
                        break;
                  end
            end

            if strcmp(rx,tx) % TX CORRECTA?
                  exito = exito + 1;
             end
      end

      r = exito / N
      t = transmisiones / mensajes
      tasa = % CALCULAR LA TASA
end
```

Por último, puede realizar una combinación de CCE y CDE.

**Verificación:** *Realice el siguiente experimento:*

Repita el experimento del punto anterior, pero corrigiendo el error si el voto mayoritario por lo menos duplica al minoritario. Puede partir del siguiente código:

```
N = 1000;
f = ascii('hola mundo');

for R= % Falta, las configuraciones para las que se hace el experimento
      exito = 0;
      transmisiones = 0;
      mensajes = 0;

      for i=1:N
            tx = repeticion(f, R);

            mensajes = mensajes + 1;
            while true
                  transmisiones = transmisiones + 1;
                  rx = canalBS(tx, 0.9); % SE OBTIENE RX

                  % FALTA
                  errornocorregible = % Completar
                  % FALTA

                  if ~errornocorregible
                        % FALTA
                        rxcorregido = % Completar si procede
                        % FALTA

                        break;
                  end
            end

            if strcmp(rxcorregido,tx) % TX CORRECTA?
                  exito = exito + 1;
             end
      end

      r = exito / N
      t = transmisiones / mensajes
      tasa = % CALCULAR LA TASA
end
```

¿Con qué codificación se queda de todas las anteriores? Razone si esta dependerá o no de la longitud del mensaje o del canal.

## 3.5. CODIFICACIÓN DE CANAL CON PARIDAD SIMPLE

El ejemplo anterior deja intuir que transmitir con tasa real cercana a la capacidad es una meta muy difícil. Los mejores códigos conocidos son los Turbo, descubiertos en la década de los 90, del tipo CCE, y muy usados hoy en día, por ejemplo, en comunicaciones celulares y espaciales, y ya muy cerca del límite de Shannon, ver: https://es.wikipedia.org/wiki/Turbo_código

En lo que resta de práctica vamos a implementar dos tipos de codificaciones adicionales basadas en calculo de paridad simple o múltiple.

**EJERCICIO: Codifique la función paridad**

```
function [valor, bloquep] = paridad(bloque)
% Calcula la paridad de un bloque y devuelve 1 si el numero de '1's es
% impar o 0 si es par. Devuelve también el bloque con un bit añadido al
% final para que el numero de unos sea par

end
```

**Verificación:** *Realice el siguiente experimento (se sugiere que reutilice los códigos de ejemplo anteriores)*

Divida su flujo de bits en bloques de tamaño **B=*2,4,8,16,32***, calcule su paridad, transmítalo y calcule su paridad de nuevo en destino. Si hay un error, se procederá como antes. Calcule la tasa de transmisión real: $r \times 1000 \times B/((B+1) \times t)$ *bits/s* ¿Para qué ***B*** alcanza la máxima tasa de transmisión real? ¿Cuál es ésta? ¿Con qué código se quedaría de los vistos hasta ahora?

## 3.5. CODIFICACIÓN DE CANAL CON PARIDAD MÚLTIPLE: HAMMING (7,4)

El código Hamming (7,4) genera 3 bits de redundancia por cada 4 bits de datos. Es un CCE que corrige cualquier cambio de 1 bit, pero que no posee propiedades detectoras. Consideraremos bloques de tamaño B=4 para la codificación (aunque puede extenderse a cualquier tamaño de bloque). Denotaremos D1, D2, D3 y D4 a los bit de datos y P1, P2 y P3 a los bits de la redundancia.

Se transmitirá la siguiente secuencia: 'P1 P2 D1 P3 D2 D3 D4', donde:

P1 = D1 + D2 + D4
P2 = D1 + D3 + D4
P3 = D2 + D3 + D4

La operación + es la 'o exclusiva' que se comportará igual que la función de paridad implementada anteriormente. En recepción se calculan las paridades P1, P2 y P3. Si todas son 0 no hay error (o no se puede detectar). Si hay alguna no nula, la posición con error es:

error = P1+2*P2+4*P3

Se calcula esa posición y se corrige.

**Verificación** *(se sugiere que reutilice los códigos de ejemplo anteriores)***:** Siguiendo el esquema anterior, calcule la tasa de transmisión real para un código Hamming (7,4): $r \times 1000 \times 4/7$ *bits/s* ¿Con qué código se quedaría de los vistos en la práctica?