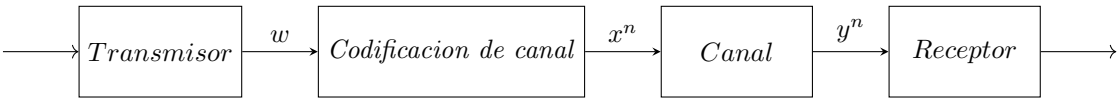
Apuntes de códigos bloque lineales y cíclicos. FEC y ARQ

Eugenio Cano Muñoz
G31 - ETSIT UPM

Curso 2023 - 2024

1 Códigos de fuente



1.1 Repaso de Álgebra binaria

Tenemos un alfabeto formado por $\{0, 1\}$ y las siguientes operaciones $\oplus$ (suma) y $\odot$ (multiplicación):

$\oplus$	0	1
0	0	1
1	1	1

$\odot$	0	1
0	0	1
1	0	1

1.1.1 Cuerpo de Galois:

Es un **grupo** - Sea g un conjunto de elementos:

1. Tiene una operación binaria
2. Esta asociación binaria es asociativa

$$a \circledast (b \circledast c) = (a \circledast b) \circledast c \tag{1}$$

3. Existe un elemento neutro tal que

$$\begin{aligned} \exists e \in G; \exists a \in G \\ a \circledast e = e \circledast a = a \end{aligned} \tag{2}$$

4. Todo elemento de un grupo tiene su inverso

$$\begin{aligned} \forall a \in G; \exists a' \in G \\ a \circledast a' \in e \end{aligned} \tag{3}$$

G es un **grupo conmutativo** si:

1. G es un grupo
2. se cumple la propiedad conmutativa:

$$a \circledast b = b \circledast a \tag{4}$$

F es un **grupo con dos operaciones** $\oplus$ y $\odot$ si:

1. F es un grupo conmutativo bajo $\cdot$
2. El 4^{o} elemento $\neq e$ del F es un grupo conmutativo bajo $\cdot$

3. La multiplicación des tristributiva bajo la condición:

$$a \cdot (b + c) = a \cdot b + a \cdot c \tag{5}$$

Si el alfabeto es finito $\Rightarrow$ Cuerpo de Galois ($GF(n)$), siendo n la longitud del alfabeto

1. El conjunto $\forall n$ tomando $(v_1, v_2, ..., v_n), v_i \in GF(2)$ junto con las operaciones:

$$\begin{aligned} + : \vec{u} + \vec{v} = \vec{w}; w_i = u_i + v_i \\ \cdot := \lambda \cdot \vec{u} = \vec{w}; w_i = \lambda b_i \end{aligned} \tag{6}$$

   es un espacio vectorial sobre sobre $GF(2)$

2. $\mathcal{S} \subseteq V_n$ es un subespacio vectorial si:
   (a) $\varnothing \in \mathcal{S}$
   (b) $\forall \vec{u}, \vec{v} \in \mathcal{S}, (\vec{u} + \vec{v}) \in \mathcal{S}$
   (c) $(\lambda \cdot \vec{u}) \in \mathcal{S}$

3. Dato $\vec{v_1}, \vec{v_2}, ..., \vec{v_n}$ se dice que son linealmente independientes si $\forall c_i \Rightarrow c_1\vec{v_1} + c_2\vec{v_2} + .. + c_n\vec{v_n} \neq 0$ excepto $c_1 = c_2 = ... = c_n = 0$

4. El conjunto $\{\vec{g_1}, \vec{g_2}, ..., \vec{g_n}\}$ genera V_n si $\forall v \in V_n \Rightarrow \vec{v} = \sum_{i=1}^{n} c_i\vec{g_i}$

5. En todo $V_n, \exists\{\vec{g_1}, \vec{g_2}, ..., \vec{g_n}\}$ linealmente independientes que generan V_n (Su cardinal es n)

6. Si $K < n; \{\vec{v_1}, \vec{v_2}, ...\vec{v_k}\} \subseteq V_n$ es un subespacio vectorial de dimensión K

7. Se define el operador del producto interior $\vec{u} \cdot \vec{v} = u_1v_1 + u_2v_2 + ... + u_nv_n$

8. Se dice que $\vec{u}$ y $\vec{v}$ son ortogonales si $\vec{u} \cdot \vec{v} = 0$

9. Sea $\mathcal{S}$ un subespacio vectorial de dimensión $k \subseteq V_n$. Sea $\mathcal{S}_d$ el conjunto de vectores $\subseteq V_n / \forall \vec{u} \in \mathcal{S}; \forall \vec{v} \in \mathcal{S}_d \Rightarrow \vec{u} \cdot \vec{v} = 0$ $\mathcal{S} \equiv$ Espacio dual de $\mathcal{S}$ dimensión $(n \cdot k)$

1.2 Códigos lineales

Código: Transformación de una palabra k bits a n bits.

$$\mathcal{C}(n,k) \tag{7}$$

1. Códigos
   (a) lineales
      i. Bloque
         A. Cíclicos
         B. No cíclicos
         A. Sistemáticos
         B. No sistemáticos
      ii. Convolucionales
   (b) No lineales

Código bloque: Los datos se dividen en bloques binarios de tamaño k, llamados bits de mensaje. Cada bloque puede representar uno de los 2^k mensajes distintos. El codificador transforma la palabra de k bits a otra de n bits. Los bits $r = (n - k)$ se denominan **bits reduntantes** o **bits de paridad**, ya que no aportan nueva información. El **radio de redundancia del código** es la relación de bits redundantes a bits de mensaje y se puede obtener de la siguiente forma $(n - k)/k$. La **tasa del código** es k/n, que se puede explicar como la porción del código que aporta información.

Código lineal: Son una clase de códigos que pueden ser caracterizados por la notación (n, k) ya mencionada. El codificador transforma un bloque de k bits en uno de n bits (aunque en este texto siempre se hable de bits, también es posible utilizar otro tipo de alfabetos). Los diferentes 2^k mensajes se denominan las **k-tuplas**, y los 2^n mensajes se denominan las **n-tuplas**. El proceso de codificación asigna una k-tupla a cada una de las n-tuplas de forma biyectiva. Esta asignación es lineal para códigos lineales.

1.2.1 Matriz generadora:

Como las palabras código que forman un bloque lineal, es un subespacio vectorial de dimensión k del espacio vectorial de dimensión n ($k < n$), utilizar un conjunto de vectores generadores para "expandir" el subespacio. La combinación linealmente independiente más pequeña que expande el subespacio se denomina la **base generadora** del subespacio, y el número de vectores en esta basa es la dimensión del subespacio (k). En general, definimos la matriz generadora como la siguiente matriz de tamaño $k \times n$:

$$\vec{v} = (v_0, v_1, ...v_{n-1}); \vec{u} = (u_0, u_1, ..., u_{n-1})$$
$$\vec{v} = \vec{u} \begin{pmatrix} g_{00} & g_{01} & ... & g_{0,n-1} \\ g_{10} & g_{11} & ... & g_{1,n-1} \\ ... & ... & ... & ... \\ g_{k-1,0} & g_{k-1,1} & ... & g_{k-1,n-1} \end{pmatrix} \tag{8}$$

En esta ecuación, podemos ver que el vector $\vec{u}$ son las palabras código del subespacio vectorial n y $\vec{v}$ son las palabras del espacio vectorial k. La operación anterior también se puede representar como:

$$\vec{v} = \vec{u}G \tag{9}$$

Como podemos ver, la matriz G, está formada por los vectores código, que se establecen como vectores en filas.

$$G = \begin{pmatrix} \vec{v_0} \\ \vec{v_1} \\ ... \\ \vec{v_{k-1}} \end{pmatrix} \tag{10}$$

1.2.2 Estructuración sistemática:

Los códigos sistemáticos es una relación entre un vector mensaje k-dimensional y un una palabra código n-dimensional, de tal forma que parte de la secuencia generada coincida con los bits k del mensaje, los bits $n - k$ son bits de paridad. Los bloques de códificación sistemática tienen una matriz generadora de la siguiente forma:

$$G_s(p|I_c) = \left(\begin{array}{cccc|cccc} p_{0,0} & p_{0,1} & ... & p_{0,n-k-1} & 1 & 0 & ... & 0 \\ p_{1,0} & p_{1,1} & ... & p_{1,n-k-1} & 0 & 1 & ... & 0 \\ ... & ... & ... & ... & ... & ... & ... & ... \\ p_{k-1,0} & p_{k-1,1} & ... & p_{k-1,n-k-1} & 0 & 0 & ... & 1 \end{array} \right) \tag{11}$$

En este caso, p es la porción de paridad de la matriz generadora e I_c es la matriz identidad $k \times k$. En la Estructuración sistematica no hace falta guardar la parte de la matriz iddentidad, podemos entonces entender que código $\mathcal{C}(n, k)$ como $\mathcal{C}(k + paridad, k)$

Otra forma válida de establecer la matriz generadora sistemática G_s, podría ser para que los bits del mensaje ocupasen la parte izquierda y los bits de paridad la parte derecha, generando la matriz $G(I_c|p)$

1.2.3 Matriz de comprobación de paridad (H):

La matriz de comprobación de paridad H nos permitirá decodificar los vectores recibidos. Todas las matrices generadoras G tienen una matriz H de tamaño $(n-k)\times n$, de tal forma que las filas de G sean ortogonales a las filas de H, osea que $GH^T = 0$. Partiendo de las estructuración sistemática, podemos general la matriz de paridad de una forma sencilla, estableciendo una matriz identidad de tamaño $(n-k)$ en la parte izquierda y la submatriz de paridad transpuesta p^t que obtenemos de la matriz generadora G, dicho de otra forma:

$$G_s = (p|I_k) \rightarrow H(I_{n-k}|p^t) \tag{12}$$

por lo que la matriz de comprobación de paridad transpuesta H^T resultaría de la siguiente forma

$$H^T = \left(\begin{array}{c} I_{n-k} \\ \hline \\ P \end{array} \right) \tag{13}$$

Por lo tanto, si tenemos una cadena codificada $\vec{r}$ que ha pasado por el canal, podemos saber si esta pertenece al código si al multiplicarla por la matriz de comprobación de paridad transpuesta resulta en un vector $\vec{0}$, en caso contrario, la palabra ha sido alterada por el canal y la palabra no pertenece al código:

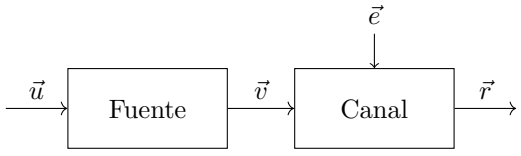
$$\vec{r} \rightarrow \vec{r} \cdot H^t = \left\{ \begin{array}{l} = 0 \Rightarrow \vec{r} \in \mathcal{C} \\ \neq 0 \Rightarrow \vec{r} \notin \mathcal{C} \end{array} \right\} \tag{14}$$

1.2.4 Síndrome:

Si $\vec{r}$ es el vector resultante de la transmisión por un canal del vector $\vec{u}$, podemos describir el vector recibido como:

$$\vec{r} = \vec{v} + \vec{e} \tag{15}$$

O con notación de bloques:



En donde que vector error $\vec{e} = (e_0, e_1, ...e_n)$ se define de la siguiente forma:

$$e_i = \left\{ \begin{array}{l} 0 \Rightarrow \text{No afecta bit } i \\ 1 \Rightarrow \text{Error en bit } i \end{array} \right\} \tag{16}$$

$\vec{e}$ es un vector de error introducido por el canal. Extisten un total de $2^n - 1$ Posibles errores en un espacio de 2^n. Para detectarlos, definimos el síndrome de $\vec{r}$:

$$\vec{S} = \vec{r}H^t = \left\{ \begin{array}{l} = 0(\vec{r} \in \mathcal{C}) \\ \neq 0(\vec{r} \notin \mathcal{C}) \end{array} \right\} \tag{17}$$

El síndrome es el resultado de la comprobación de paridad realizada a $\vec{r}$ para determinar si es un miembtro válido del conjunto de palabras código. Si $\vec{r}$ es un miembro, el síndrome S es igual a cero, y si contiene errores detectables, tiene cualquier otro valor que muestra el patrón de error. Posterior mente el codificador intentará corregir el error o pedirá una segunda transmisión. Combinando las ecuaciones anteriores:

$$\vec{S} = \vec{r}H^t = (\vec{v} + \vec{e})H^t = (\vec{v}H^t)|_{=0} + \vec{e}H^t = \vec{e}H^t \tag{18}$$

Una característica importante de los códigos bloque lineales es que la relación entre los patrones de errores corregibles y el síndrome es uno a uno..

Además hay que añadir dos requisitos a la matriz de comprobación de paridad:

1. Ninguna columna de la matriz de comprobación de paridad puede contener todo ceros, sino un error en la palabra código correspondiente no podría afectar al síndrome y sería indetectable

2. Todas las columnas de la matriz de comprobación deben ser distintas. Si dos columnas de la matriz fuesen idénticas, no se podrían percibir errores en las posiciones de las palabras código

En el caso en el que el producto $\vec{r}H^t = \vec{0}$, podemos tener dos posibilidades, que $\vec{e} = \vec{0}$, es decir, que no haya un error o $\vec{e} \neq \vec{0}$ o que haya un error indeterminado, es por eso que se desea evitar que $\exists \vec{e} \in \mathcal{C}, \forall \mathcal{C} \Rightarrow \vec{v}$ tenga muchos 1s, si la probabilidad de error es menor que 0,5 ($p_e < 0.5$)

1.2.5 Corrección de errores y tabla estándar:

A partir del síndrome no solo podemos detectar el patrón de error, sino que podemos corregirlo, ya que existe una correspondencia bit a bit. Si representamos todas las 2^n n-tuplas que representan los vectores posiblemente recividos en una matriz, llamada la **tabla estándar**, tal que la primera fila contenga todas las palabras código empezando por la palabra código $\vec{0}$ y la primera columna contenga todos los posibles patrones de error corregibles. En esta tabla el resto de filas reciben el nombre de *cogrupo*. La primera columna de la fila recibe el nombre de *líder de cogrupo*, seguida de las palabras alteradas por dicho patrón de error, quedando dicha tabla de la siguiente forma:

$$\left\{ \begin{array}{ccccccc} \vec{v_0} & \vec{v_1} & ... & \vec{v_i} & ... & v_{2^{(\vec{k-1})}} \\ \vec{e_1} & \vec{e_1}+\vec{v_1} & ... & \vec{e_1}+\vec{v_i} & ... & \vec{e_1}+v_{2^{(\vec{k-1})}} \\ ... & ... & ... & ... & ... & ... \\ e_{\vec{n-k}} & e_{\vec{n-k-1}}+\vec{v_1} & ... & e_{2^{\vec{n-k-1}}}+\vec{e_1} & ... & e_{\vec{n-k-1}}+v_{2^{\vec{k-1}}} \end{array} \right\} \tag{19}$$

Como se ha mencionado antes, la creación de la matriz se realiza de la siguiente forma:

1ª Fila: códigos definidos dentro del código:

$$\vec{v_i} \in \mathcal{C} \left\{ \begin{array}{c} \vec{v_0} = \vec{0} \\ \vec{v_i} \end{array} \right. \tag{20}$$

Resto de filas: la primera columna es un vector de error y las siguientes son el código más el error:

$$\vec{v_i} \notin \mathcal{C} \left\{ \begin{array}{c} \vec{e_1} \Rightarrow 1^{\text{a}} \text{ columna} \\ \vec{e_j}+\vec{v_i} \Rightarrow \text{Demás columnas} \end{array} \right. \tag{21}$$

El término cogrupo en este caso se utiliza para denominar un grupo de números que comparten una característica común, en este caso, se puede apreciar a primera vista que cada miembro del cogrupo tienen el mismo grupo. Este síndrome lo utilizamos para detectar el patrón de error. Podemos demostrar lo dicho anteriormente de la siguiente forma:

$$\left\{ \begin{array}{l} \vec{v_i}+\vec{e_j} \Rightarrow \vec{s_i} = (\vec{v_i}+\vec{e_j})H^t = \vec{e_j}H^t \\ \vec{v_k}+\vec{e_j} \Rightarrow \vec{s_k} = (\vec{v_i}+\vec{e_j})H^t = \vec{e_j}H^t \end{array} \right. \Rightarrow \text{mismo síndrome} \tag{22}$$

Partiendo de esto, con el vector síndrome, podemos calcular el error del código, detectarlo y corregirlo de la siguiente forma:

1. Calculamos el síndrome de $\vec{r}$ utilizando la ecuación $\vec{s} = \vec{r}H$.
2. Identificar el líder del cogrupo con el mismo síndrome $\vec{s}$. Este patrón de error será considerada la perturbación introducida por el canal.
3. La palabra código corregida se identifica como $\vec{v} = \vec{r} + \vec{e}$ (En álgebra binaria la resta es idéntica a la suma).

La tabla estandar contiene todas las 2^n entradas en el espacio de n-tuplas, sin nada repetido. Al principio puede parecer que esta herramienta se limita a bloques código pequeños, pero la tabla estandar nos permite la visualización de problemas de rendimiento hasta en códigos grandes, como compromisos entre corrección de erreores y detección.

El número de coeficientes α de peso i (α_i) se puede calcular de la siguiente forma:

$$\alpha_i = \begin{pmatrix} n \\ i \end{pmatrix} \tag{23}$$

Estos coeficientes rellenarán la tabla estándar hasta completarla. Como ya se expuso antes, la tabla tiene una **capacidad** de 2^n entradas, con 2^k columnas y 2^{n-k} filas. En un **código perfecto**, los líderes de cogrupo deben ser todos los vectores menores a un determinado peso y ninguno más.

A pesar de que existen metodos de corrección, se tienen que cumplir ciertas condiciones para poder corregir un código.

1.2.6 Pesos y distancias de hamming:

Dado una palabra código $\vec{v}$ procedente de un código lineal $\mathcal{C}(n,k)$ podemos definir distintos parámetros que nos servirán a la hora de determinar las condiciones de corrección de un código.

El **Peso de Hamming de v ($\omega(\vec{v})$)** es el número de elementos no nulos en $\vec{v}$ (o dicho de otra forma el número de unos dentro de la palabra código).

La **Distancia de Hamming entre $\vec{v}\times\vec{w}$ ($d(\vec{v},\vec{w})$)** es el número de bits que diferencian las palabras código $\vec{v}$ y $\vec{w}$. Cabe recalcar que en este caso también se cumple la desigualdad triangular:

$$d(\vec{v},\vec{w}) + d(\vec{w},\vec{x}) \geq d(\vec{v},\vec{x}) \tag{24}$$

De lo anterior podemos apreciar que la distancia de Hamming entre dos vectores es igual al peso de la suma de los vectores, ya que de esta forma, los elementos idénticos se anularán

$$d(\vec{v},\vec{w}) = \omega(\vec{v}+\vec{w}) \tag{25}$$

Considere el conjunto de distancias entre todas las palabras código en el espacio vectorial. La **distancia mínima de un código** d_{min} es la mínima distancia de hamming en un código, esta distancia representa la parte más débil frente a errores del código, y por eso caracteriza la robustez del código.

$$d_{min} = min\{d(\vec{v},\vec{w}); \vec{v},\vec{w} \in \mathcal{C}; \vec{v} \neq \vec{w}\} = min\{\omega(\vec{v}+\vec{w}); \vec{v},\vec{w} \in \mathcal{C}; \vec{v} \neq \vec{w})\} \tag{26}$$

Para calcular la distancia mínima, tenemos que tener en cuenta que las palabras código son cualquiera de las palabras generadas a partir de las palabras código, por lo que hay que calcular la distancia de hamming entre todas las palabras resultantes de $\vec{w_{ij}} = \vec{u_i} + \vec{v_j}$. Para evitar todo este trabajo, tenemos varios mecanismos para calcular esta distancia mínima:

1. $d_{min} = 0$ Una columna contiene todo ceros
2. $d_{min} = 1$ Dos columnas iguales
3. $d_{min} = 2$ Dos columnas suman 0
4. $d_{min} = n$ n columnas suman 0

1.2.7 Capacidad de detección:

La tarea de un decodificador óptimo es asignar la palabra código más probable al mensaje recibido por el canal (teniendo en cuenta que el canal va a insertar errores).

$$max\{p\} \begin{cases} \vec{e_0} = \vec{r} + \vec{v_0} \\ \vec{e_1} = \vec{r} + \vec{v_1} \\ \dots \\ \vec{e_{L-1}} = \vec{r} + \vec{v_{L-1}} \end{cases} \tag{27}$$

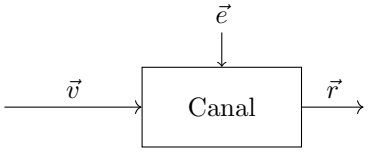
Debemos escoger $\vec{e_i}$ tal que:

$$min\{\omega(\vec{e_i})\} = min\{\omega(\vec{r} + \vec{v_i})\} = min\{d(\vec{v}, \vec{r})\} \tag{28}$$

Sabiendo como funciona el decodificador óptimo, podemos concluir que no siempre va a recuperar la palabra correcta. Llegado a una cierta distancia de Hamming, podemos encontrar otras palabras código válidas, por lo que puede que no tengamos errores detectables, sabiendo esto, podemos concluir que la máxima distancia de corrección S es:

$$S = d_{min} - 1 \tag{29}$$

La **Probabilidad de error** es la probabilidad de que ciertos bits hayan sido alterados y no se pueda recibir la palabra código original.



Para calcular las probabilidades de detección de errores, establecemos p como la probabilidad de que ocurra un cambio de bit en el canal $(1-p)$ como la de que no ocurra, a partir de esto pa partir de esto podemos establecer la probabilidad de error del código para un peso de Hamming *mayor* o igual que L, podemos seguir la siguiente ecuación:

$$P(\omega(e) \geq L) = \sum_{i=L}^{N} \binom{n}{i} p^i (1-p)^{n-i} \tag{30}$$

y la probabilidad de que el error sea *menor* que el peso de Hamming (incluyendo una transmisión correcta de la información):

$$P(\omega(e) < L) = \sum_{i=0}^{L-1} \left(\begin{array}{c} n \\ i \end{array} \right) p^i (1-p)^{n-i} \tag{31}$$

Distribución de pesos de palabras código A_i es el número de palabras código que pesan i. Con esto se puede hacer un histograma con el que representar la distribución de probabilidades.

La **probabilidad de error indetectable** es la probabilidad de que haya un error que no pueda ser detectado:

$$P_{ND} = \sum_{i=d_{min}}^{n} A_i p^i (1-p)^{n-i} \tag{32}$$

Si no se conoce la distribución A_i, para calcular P_{ND}, o calcularla es demasiado complicado, podemos obtener una cota superior:

$$P_{ND} = \sum_{i=d_{min}}^{n} A_i p^i (1-p)^{n-i} \leq \sum_{i=d_{min}}^{L-1} \left(\begin{array}{c} n \\ i \end{array} \right) p^i (1-p)^{n-i} \tag{33}$$

Si queremos hacer una aproximación mas burda de esta probabilidad siendo esta última mucha más rápida, todavía se puede realizar una aproximación algo más alta:

$$P_{ND} \leq \sum_{i=d_{min}}^{n} 2^k p^i (1-p)^{n-1} \tag{34}$$

La probabilidad de que haya una **transmisión sin errores** del código, equivale a la probabilidad de obtener una palabra de peso 0:

$$P(\omega(\vec{v}) = 0) = (1-p)^n \tag{35}$$

la probabilidad de tener **errores detectables** es el contrario de la probabilidad de tener errores indetectables $1 - P_{ND}$ además podemos escribirla como:

$$P(\text{Error detectable}) = \sum_{i=1}^{d_{min}-1 \text{ o } S} A_i p^i (1-p)^{n-i} \leq \sum_{i=0}^{d_{min}-1 \text{ o } S} \left(\begin{array}{c} n \\ i \end{array} \right) p^i (1-p)^{n-i} \tag{36}$$

Además, a la probabilidad de tener un error que pueda ser detectado o no se le llama **error residual**, que se rige por la fórmula:

$$P(\text{Error residual}) = \sum_{i=d_{min}}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) p^i (1-p)^{n-i} \tag{37}$$

La capacidad de corrección es cual es la distancia máxima de error que podemos corregir con un descodificador óptimo, la podemos denominar como t. Podemos calcular la **capacidad máxima de corrección (t_{max})** mediante la siguiente fórmula:

$$t_{max} = \lfloor \frac{d_{min} - 1}{2} \rfloor \tag{38}$$

La **probabilidad de error de corrección** es la probabilidad de que ocurra un error al corregir el vector $\vec{e_j} + \vec{v_i}$

$$P_{ec} = 1 - \sum_{i=0}^{\alpha} a_i p^i (1-p)^{n-1} \tag{39}$$

Es posible que no conozcamos todos los coeficientes α, por lo que podemos realizar una aproximación de la siguiente forma:

$$P_{ec} = 1 - \sum_{i=0}^{1} \left(\begin{array}{c} n \\ i \end{array} \right) p^i (1-p)^{n-i} = \sum_{i=2}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) p^0 (1-p)^{n-i} \tag{40}$$

1.2.8 Métodos mixtos:

Hasta ahora solo hemos visto la capacidad de corrección y la de detección de manera individual, pero ambas se pueden combinar, dando así lugar a los métodos mixtos

Dado un código $\mathcal{C}(n,k) = d_{min}$, con una s_{max} y una t_{max}, podemos alterar los valores de t y s siempre que estén dentro de su rango máximo y cumplan la siguiente ecuación:

$$d_{min} = s + t + 1; \tag{41}$$

Cuando errores de distancia t o menor occurren, el codigo es capaz de detectar y corregirlos. Cuando la distancia es mayor que te, pero menor que s, el código es capaz de detectar su presencia pero no corregirlos.

Los valores de t y s más utilizados son más utilizados son:

$$\text{Detección pura} \Rightarrow \left\{ \begin{array}{c} s = d_{min} - 1 \\ t = 0 \end{array} \right. \tag{42}$$

$$\text{Corrección pura máxima} \Rightarrow \left\{ \begin{array}{c} s = d_{min} - t - 1 \\ t = \lfloor \frac{d_{min}-1}{2} \rfloor \end{array} \right. \tag{43}$$

Debo copiar la parte de $w(e)$

Teorema: En un grupo todos los componentes son distintos, la demostración es:

$$\vec{v_j} + \vec{e_i} = \vec{v_k} + \vec{e_i} \Rightarrow \vec{v_j} = \vec{v_k} \Rightarrow \text{Imposible} \tag{44}$$

Teorema: En distintos grupos, las componentes son distintas, la demostración es:

$$\vec{v_j} + \vec{e_i} = \vec{v_k} + \vec{e_m} \Rightarrow el = e_m + (\vec{v_i} + \vec{v_k}) = e_i + v_m = \text{Imposible} \tag{45}$$

1.2.9 Códigos Hamming:

Los códigos hamming sirven para generar códigos lineales de forma sistemática, utilizando una base generadora mayor que 3 .El código Hamming tendrá las siguientes características:

$$\mathcal{C}(n,k) \Rightarrow \begin{cases} n = 2^m - 1 \\ k = 2^m - m - 1 \\ n - k = m \\ d_{min} = 3 \end{cases} \tag{46}$$

Para generar el código partimos de la matriz de paridad H, que consta de todas las m-tuplas no lunas como columnas, quedando de la siguiente forma:

$$H = (I_{n-k}|p^t)||| H = (I_m|Q) \tag{47}$$

Siendo Q la sumbatriz que consta de $2^m - m - 1$ columnas, las cuales son las m-tuplas de peso 2 o más. El orden de las columnas no afecta a la propiedad de la distancia, ni a la disribución de peso. En el método sistemático, la matriz G resulta de la siguiente forma:

$$G = (Q^T|I_{2^m-m-1}) \tag{48}$$

1.3 Códigos cíclicos:

Los códigos cíclicos son una subcategoría bastante importante dentro de los códigos bloques lineales, ya que su implementación es bastante sencilla utilizando registros de desplazamiento, además de poder ser descritos mediante notación polinomial.

Para definir los códigos cíclicos, primero debemos definir la **operación de rotación cíclica**, sea:

$$\begin{aligned} \vec{v} &= (v_0, v_1, ..., v_{n-1}) \\ \vec{v^{(1)}} &= (v_{n-1}, v_0, ..., v_{n-2}) \\ \vec{v^{(i)}} &= (v_{n-i}, v_{n-i+1}, ..., v_{n-1}, v_0, ..., v_{n-i-1}) \end{aligned} \tag{49}$$

Definida esta operación, cualquier código $\mathcal{C}(n,k)$ es cíclico si y solo si cualquier rotación cíclica de un vector $\vec{v}$ perteneciente a $\mathcal{C}$ es también un vector del código $\mathcal{C}$

$$\exists \vec{v} \in \mathcal{C} \Leftrightarrow \vec{v}^{(i)} \in \mathcal{C} \tag{50}$$

1.3.1 Notación polinomial:

Los componentes de una palabra $\vec{v} = (v_0, v_1, ..., v_i)$ pueden ser tratados como los coeficientes de un polinomio $V(x)$, de la siguiente manera:

$$\vec{v} = (v_0, v_1, ..., v_{n-1}) \Rightarrow V(x) = v_0 + v_1 x + v_2 x^2 + ... + v_{n-1} x^{n-1} \tag{51}$$

Se puede pensar que la función polinómica $V(x)$ es una plantilla para los dígitos de la palabra código $\vec{v}$. La presencia o no de cada término en el polinomio indica si el componente es distinto de cero.

Al expresar las palabras códigos en la fórma polinómica, podemos obtener otras palabras código dividiendo $x^i V(x)$ entre $x^n + 1$. El resto de esta división es también una palabra código:

$$x^i V(x) = q(x)(x^n + 1) + V^{(i)}(x) \Rightarrow V^{(i)}(x) \equiv resto\left(\frac{x^i V(x)}{x^n + 1}\right) \tag{52}$$

La ecuación anterior también podemos interpretarla como el siguiente módulo:

$$V^{(i)}(x) = x^i V(x) mod(x^n + 1) \tag{53}$$

1.3.2 Polinomio código de grado mínimo y polinomio generador:

Todos los códigos cíclicos $\mathcal{C}(n,k)$ contienen un polinomio de grado mínimo, en el que siempre $g_0 = 1$, además este es único, ya que un desplazamiento a la derecha provocaría un incremento del grado del polinomio.

Partiendo del código cíclico $\mathcal{C}(n,k)$ siendo $g(x)$ su polinomio de grado mínimo, las palabras códigos son todos los múltiplos de $g(x)$

$$\forall V(x)/V(x) = a(x)g(x) \Rightarrow V(x) \in \mathcal{C}\text{con grado} \leq (n-1) \in \mathcal{C} \tag{54}$$

Podemos llamar a este polinomio *polinomio generador* ya que tiene una función parecida a la matriz generadora de los códigos lineales. Ya que este polinomio debe ser de grado mínimo, $g_0 = 1$ y además $g_p = 1$, donde se debe cumplir la relación.

$$p = n - k \tag{55}$$

Además, de este polinomio generador $g(x)$ sabemos que debe ser un factor de $x^n + 1$, el número de factores posibles f que marca el número de códigos que pueden ser generados por este polinomio con la siguiente fórmula $2^f - 2$

Se dice que $v(x)$ es una palabra código válida del subespacio $\mathcal{S}$ si y solo si $g(x)$ divide a $v(x)$ sin resto.

$$v(x) mod(g(x)) = 0 \tag{56}$$

1.3.3 Codificación sistemática:

En la forma sistemática, los bits de mensajes se utilizan como parte de la palabra. Podemos pensar que estamos desplazando el mensaje a los k espacios de la parte derecha y añadiendo $n - k$ bits de paridad en la parte izquierda. El vector de la palabra código quedaría de la siguiente forma:

$$\vec{v} = (p_0, p_1, ..., p_{n-k-1}, m_0, m_1, ..., m_{k-1}) \tag{57}$$

siendo p los bits de paridad y m los bits mensaje.

La codificación sistematica nos permite generar códigos a partir de un $g(x)$ consiste en:

1. Multiplicar x^{n-k} por $u(x)$
2. Calcular $b(x)$ que es el resto de dividir $x^{n-k}u(x)$ entre $g(x)$
3. Concatenar $b(x)$ y $u(x)$

1.3.4 Matriz generadora:

Debido a que los códigos cíclicos forman parte de los códigos bloque lineales, tamién disponen de matrices generadoras, una forma de generarla podría ser partiendo de un código cíclico $\mathcal{C}(n,k)$ y $g(x) = g_0 + g_1x + ... + g_{n-k}x^n$, el conjunto de k polinomios $\{g(x), xg(x), ..., x^{k-1}g(x)\}$ forman la base del código, las n-tuplas se disponen de la siguiente manera

$$G = \begin{pmatrix} g_0 & g_1 & g_2 & ... & g_{n-k} & 0 & 0 & ... & 0 \\ 0 & g_0 & g_1 & ... & g_{n-k-1} & g_{n-k} & 0 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & g_0 & g_1 & g_2 & ... & g_{n-k} \end{pmatrix} \tag{58}$$

A pesar de todo, esta matriz no es sistemática, si quisiésemos realizar el proceso de esta forma, podríamos:

1. Diagonalizar la matriz para obtener $G_s(p|I_c)$ (por ejemplo con Gauss-Jordan)
2. Debido a que $\mathcal{C}(n,k)$ es un código cíclico y dispone de un polinomio generador $g(x)$, podemos calcular la codificación sistemática de las palabras de peso 1

1.3.5 Matriz comprobadora de paridad:

Al igual que con la matriz anterior, tenemos dos fórmulas:

1. Partimos del polinomio generador $g_0(x)$, obtenemos su matriz generadora sistemática y finalmente la de comprobación de paridad H

$$g_0(x) \Rightarrow G \Rightarrow G_s = (P|I_k) \Rightarrow H = (I_{n-k}|p_t) \tag{59}$$

2. Como $g(x)$ es factor de $(x^n + 1) = g(x)h(x)$, siendo $h(x) = h_0 + h_1x + h_2x^2 + ... + h_kh^k$.

   Si $g(x)$ tiene grado $n-k$, se puede demostrar que el polinomio recíproco de $h(x) = x^k h(x^{-}1)$ es también factor de $x^n + 1$, por lo que $h(x)$ generará un código cíclico $\mathcal{C}(n, n-k)$. La matriz generadora de este código será:

$$H = \begin{pmatrix} h_k & h_{k-1} & h_{k-2}2 & ... & h_0 & 0 & 0 & ... & 0 \\ 0 & h_k & h_{k-1} & h_{k-2} & ... & h_0 & 0 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & h_k & h_{k-1} & h_{k-2} & ... & h_0 \end{pmatrix} \tag{60}$$

   En este caso H genera el código dual de $\mathcal{C}(n,k)$

Además, podemos generar el código dual de un código cíclico $C(n,k)$ mediante el polinomio generador $g(x)$ a partir de la siguiente ecuación:

$$x^k h(x^-1) \tag{61}$$

Siendo $h(x) = \frac{x^n+1}{g(x)}$

1.3.6 Codificador basado en $g(x)$:

Existen dos maneras de implementar un codificador basado en el polinomio generador $g(x)$

1. Implementando un codificador lineal a partir de su matriz generadora G
2. Codificar siguiendo los pasos de la codificación sistemática.

$$v(x) = b(x) + n^{n-k}u(x) \tag{62}$$

El circuito generador es el siguiente:

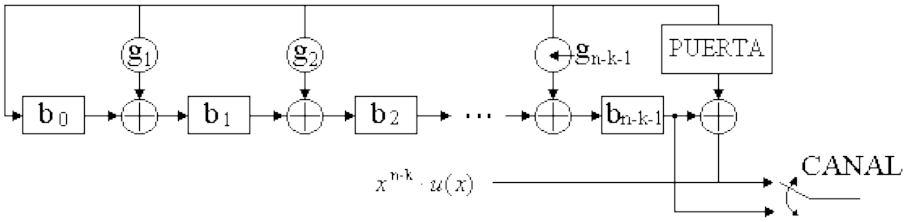


Figure 1: Circuito generador

Preferiblemente utilizamos un codificador basado en $g(x)$ cuando tenemos $n-k$ etapas.

1.3.7 Codificador basado en $h(x)$:

Este codificador parte del polinomio de comprobación de paridad $h(x)$, las ecuaciones diferenciales quedan de la siguiente forma;

$$v_{n-k-j} = \sum_{i=0}^{k-1} h_i v_{n-i-j} \quad 1 \leq j \leq n-k \tag{63}$$

El circuito generador basado en $h(x)$ es el siguiente:

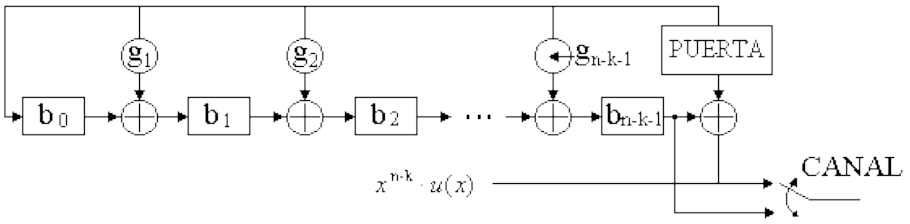
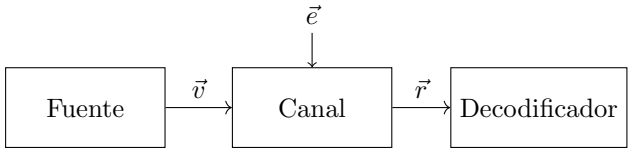


Figure 2: Circuito generador

1.3.8 Síndrome

Al igual que en los códigos lineales, también podemos calcular el síndrome de un vector recibido a través de un canal para ver si ha sido alterado.



Como ya se vió en los códigos lineales, el síndrome de un vector que no ha sido alterado por el canal es $\vec{s} = \vec{0}$. Adaptando este vector síndrome a la notación polinomial propia de los códigos síndrome, en este caso tendremos el *polinomio síndrome*.

$$r(x) = b(x)g(x) + s(x) \text{ siendo } s(x) = 0 \text{ si } \vec{r} \in \mathcal{C}(n,k) \tag{64}$$

De esta ecuación, podemos deducir que el polinomio síndrome es el resto o módulo de la palabra recibida y el polinomio generador.

$$s(x) = (r(x))mod(g(x)) \tag{65}$$

Debido a la naturaleza cíclica de los códigos, el síndrome desplazado $s^(i)(x)$ puede ser calculado de la siguiente forma:

$$(x^i g(x))mod(g(x)) \text{ ya que } (g^(i)(x))mod(g(x)) \tag{66}$$

1.3.9 Tabla estándar:

La tabla estándar también se aplica de la misma forma que en los códigos lineales, adaptándolo a la notación polinomial, sabemos que:

$$\begin{cases} r(x) = a(x)g(x) + e(x) \\ r(x) = b(x)g(x) + s(x) \end{cases} \tag{67}$$

Igualando ambas ecuaciones y despejando el error $e(x)$:

$$e(x) = (a(x) + b(x))g(x) + s(x) \tag{68}$$

El diagrama siguiente muestra el proceso de la corrección mediante la tabla estándar y el síndrome:

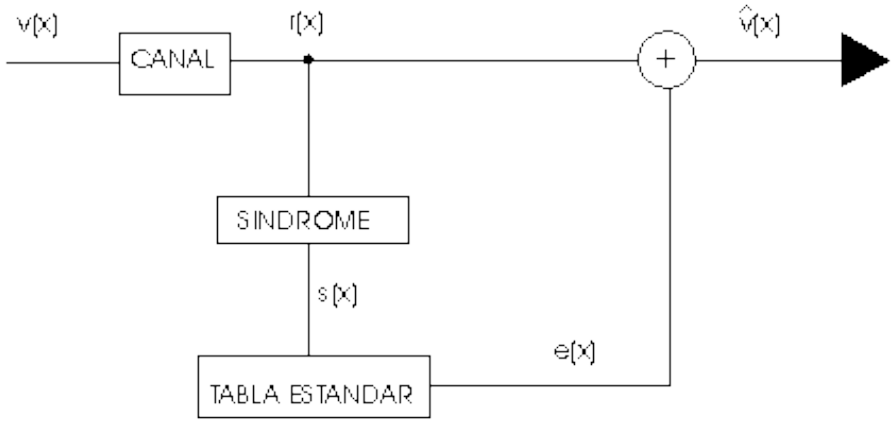


Figure 3: Circuito generador

1.3.10 Propiedades de detección de errores:

Las características más interesantes de los códigos cíclicos aparecen a la hora de detectar errores, ya que son muy potentes en este ámbito. Si se cumplen ciertas características puedo detectar errores aunque no se cumpla la propiedad $s = d_{min} - 1$, algunas son:

- Si $g(x)$ es distinto del polinomio 1 se *detectan todos los errores simples.*
- Si $g(x)$ es múltiplo de $(1 + x)$ se *detectan todos los errores impares.*
- Si $g(x)$ tiene factor a un polinomio primitivo, se detectan todos los errores dobles. Los polinomios cumplen las siguientes características

$$\begin{array}{rccc} \text{factor de} & (x^n + 1) & n = 2^r - 1 \\ \text{no es factor de} & (x^m + 1) & m < n \end{array} \tag{69}$$

1.3.11 Errores ráfaga:

Los errores ráfaga son un tipo de error producido por un canal que altera los bits dentro de una parte de la transmisión, quedando el vector error de la siguiente forma:

$$\vec{e} = (0, 0, ..., 1, {}^{1}\!/_{0}, {}^{1}\!/_{0}, ..., {}^{1}\!/_{0}, {}^{1}\!/_{0}, 1, 0, 0, ..., 0) \tag{70}$$

Podemos representar matemáticamente la ráfaga de errores de la siguiente forma:

$$e(x) = x^i B(x) \quad 0 < i \leq n-1 \tag{71}$$

Debemos saber que los errores que pertenezcan al código son indetectable, ya que una palabra que una palabra que pertenece al código es múltiplo de $g(x)$. Teniendo esto en cuenta, podemos deducir que todas las ráfagas de longitud menor o igual a $n-k$ son detectadas. Llamamos a la fracción de ráfagas indetectables al cociente del número ráfagas indetectables (1) dentro del número de ráfagas totales (2^{n-k-1}).

$$ri = \frac{1}{2^{n-k+1}} \tag{72}$$

1.3.12 Corrección en serie:

Dado el síndrome $s(x)$ de una palabra código $r(x)$ de un código cíclico $\mathcal{C}(n,k)$, podemos corregir errores con la tabla estándar, como si fuese un código bloque estándar o realizar una corrección en serie, en la que realizamos la conversión registro a registro, ahorrando costes en el decodificador. Este proceso en serie consiste en:

1. Obtener $\vec{e_{n-1}}$ de $\vec{s_{n-1}}$
2. Calcular el bit código de la operación $\vec{v_{n-1}} = \vec{r_{n-1}} + \vec{e_{n-1}}$
3. rotar $r(x)$ para obtener $r^{(1)}(x)$

1.3.13 Códigos de Hamming cíclicos:

Existe una manera de crear códigos Hamming cíclicos siempre que $m \geq 3$ (recordando que $m = n-k$)

$$\begin{cases} n = 2^{n-1} \\ k = 2^m - m - 1 = H(I_{n-k}|p^t) \\ d_{min} = 3 \end{cases} \tag{73}$$

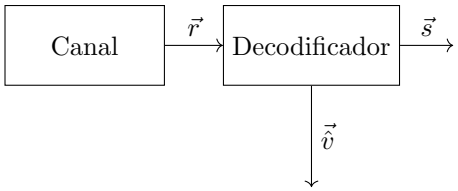
En las que se establecen todas las n-tuplas como columnas dentro de la matriz H

1.3.14 Códigos cíclicos recortados:

Dado un código cíclico $\mathcal{C}(n,k)$, podemos generar un código cíclico recortado $\mathcal{C}(n-l,k-l)$. Estos códigos tienen las mismas propiedades que el código original, se construyen con los mismos circuitos y NO son cíclicos. El *mecanismo de construcción* se realiza eliminando las l últimas palabras del código.

1.4 ARQ y análisis de prestaciones

Como ya hemos visto antes, los vectores recibidos a través del canal no tienen que ser los mismos que los enviados por el codificador, ya que el canal puede introducir ciertos errores dentro de la palabra código que deben ser detectados en corrección. Es por esto que diseñamos los códigos bloque y, dentro de estos, los cíclicos. Cuando la palabra código recibida es correcta, el decodificador simplemente la decodifica, pero en caso contrario tenemos algunos mecanismos para obtener la palabra correcta deseada. En esta sección se ven dos mecanismos cuando el síndrome $\vec{s}$ no es nulo en recepción, la ARQ y la FEQ



El protocolo **FEC (Forward Error Correction)** corrige los errores en recepción mediante un circuito de detección, en cambio el protocolo **AQR (Automatic Repeat Request)** manda tramas de control dependiendo de si el mensaje se recibió o no correctamente. En caso de recepción aparentemente correcta ($\vec{s}=\vec{0}$) se responde con una trama de control ACK, en caso contrario ($\vec{s}\neq\vec{0}$), se responde con una trama $NACK$.

1.4.1 Prestaciones técnicas ARQ:

Teniendo en cuenta que contamos con una transmisión que cumple:

- Tráfico unidireccional $E \rightarrow R$
- Las tramas de control no tinenen errores
- $P_{DET_{ERROR}} = \sum_{i=1}^{S} \left(\begin{array}{c} n \\ i \end{array} \right) p^i (1-p)^{(n-1)}$
- Código bloque lineal $\mathcal{C}(n,k)$
- Siempre tenemos datos que transmitir

Podemos realizar un análisis de prestaciones técnicas de ARQ calculando la **cadencia eficaz**, que es la relación entre los bits de informaciín de usuario y el tiempo que está el canal ocupado.

$$C_{ef} = \frac{K}{T_{oc}} \tag{74}$$

1.4.2 Protocolo de parada y espera

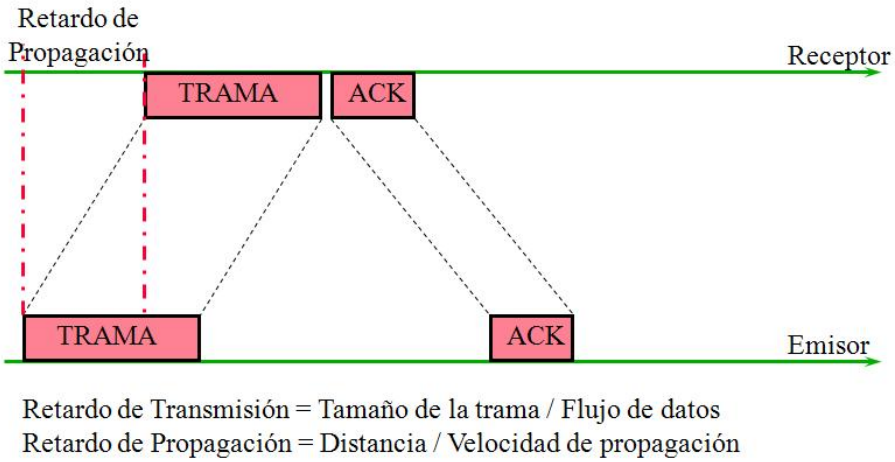


Figure 4: Protocolo de parada y espera (origen - Wikipedia)

El **régimen binario** $R[^{bits}/_{seg}]$ es la capacidad del canal bruta sin tener en cuenta las tramas del control, por lo que el tiempo que se tarda en transmitir los n bits es:

$$x_p = \frac{n}{R}[seg] \tag{75}$$

El **tiempo de asentimiento** T_{as} es el tiempo que tarda el receptor en recibir la información y que el asentimiento sea recibido por el bloque que lo ha enviado. Podemos calcular este tiempo sumando todos los retardos que aportan al canal:

- t_{prop} - Tiempo de propagación de los datos
- t_{procs} - Tiempo de porcesado del síndrome en recepción
- $t_{transcon} = {}^{1}/_{R}$ - Tiempo de transmisión de la trama de control (aproximando como si fuese de longitud 1)
- t_{propc} - Tiempo de propagación de la trama de control
- t_{procc} - Tiempo de procesado de la trama de control

Tenemos varias opciones para calcular la cadencia de bits:

1. $C_{ef} = \frac{\overline{K}}{\overline{t_{oc}}}$ - como la media de la información y la media del tiempo de ocupación del canal
2. $C_{ef} = \frac{K}{\overline{t_{oc}}}$ - Fijando el nº de bits de información entre la media de tiempo de ocupación de canal
3. $C_{ef} = \frac{\overline{K}}{t_{oc}}$ - Fijando el tiempo de ocupación del canal

La media del tiempo de ocupación $\overline{t_{oc}}$ se calcula de la siguiente forma

$$\overline{t_{oc}} = \sum_{i=1}^{\inf} t_{oc_i} p(i) = (x_p + T_{as}) \frac{1}{1 - P_{RTX}} \tag{76}$$

E insertando dentro de la cadencia de bits:

$$C_{ef} = (1 - P_{RTX}) \frac{k}{k + m + T_{as} R} R \tag{77}$$

Y finalmente, el **factor de rendimiento** del protocolo de parada y espera es:

$$\rho = \frac{C_{ef}}{R} = (1 - P_{RTX}) \frac{k}{k + m + T_{as} R} \tag{78}$$

1.4.3 Envío continuo - Full duplex y rechazo simple

En este caso, se envían todas las tramas y si se recibe un error, debemos tirar todas las tramas que lleguen hasta que se vuelva a mandar la que toca sin error.

Ahora la cadencia eficaz pasa a ser

$$C_{ef} = \frac{K}{\overline{T_{oc}}} = (1 - P_{RTX}) \frac{k}{k + m + T_{as} R P_{RTX}} R \tag{79}$$

y la eficacia del canal

$$\rho = \frac{C_{ef}}{R} = (1 - P_{RTX}) \frac{k}{k + m + T_{as} R P_{RTX}} \tag{80}$$

1.4.4 Envío continuo y rechazo selectivo

Este protoco consiste en enviar todos los datos hasta que se recibe un NACK como respuesta, entonces se termina de enviar el paquete que se estaba transmitiendo y se procede a enviar el paquete erróneos.

En este caso, la media del tiempo de ocupación $\overline{t_{oc}}$

$$\overline{t_{oc}} = \frac{k + m}{R} \frac{1}{1 - T_{RTX}} \tag{81}$$

Y la cadencia eficaz

$$C_{ef} = \frac{K}{\overline{T_{oc}}} = (1 - P_{RTX})\frac{k}{k+m}R \tag{82}$$

1.5 Técnicas hibridas

- **FEC + FEC:** Consiste en concatenar dos códigos lineares para que el código sea resistente a distintos tipos de errores, por lo que el código se compone de un primer decodificador llamado *codificador externo* y un segundo codificador llamado *codificador interno*, después de la transmisión del canal, el vector recibido pasa priemro por el decodificador interno y después por el externo. El código interno y externo no tienen por que ser del mismo tipo.

- **FEC + ARQ:** Consiste en combinar ambos protocolos para corregir errores pequeños y detectar además de notificar errores de mayor distancia, todo si la distancia el error d entra dentro de la distancia de corrección t o la distancia de detección d.