4. Corrección de errores y control de enlace

Contenido

a. Tipos de errores
b. Detección y corrección de errores
c. Control de flujo
d. Control de errores

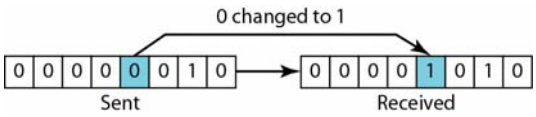
a. Tipos de errores

Los datos se puede corromper durante la transmisión.

Algunas aplicaciones requieren que los errores sean detectados y corregidos

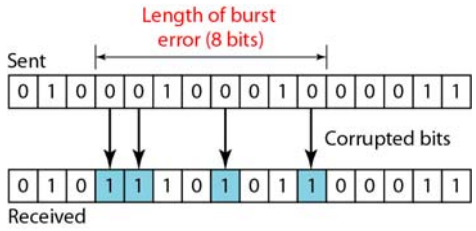
En errores de un solo bit, solo 1 bit en la unidad de datos ha sido cambiado

Error de un solo bit



Una ráfaga de errores significa que dos ó más bits en la unidad de datos han sido cambiados

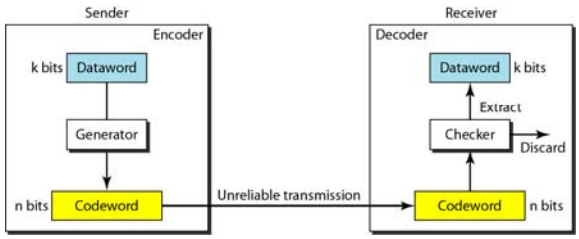
Ráfaga de errores de longitud 8



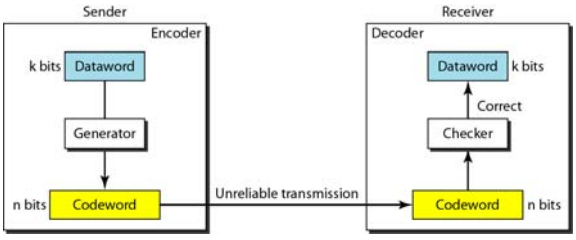
Para detectar ó corregir errores, necesitamos enviar extra (redundantes) bits con los datos

b. Detección y corrección de errores

Proceso de detección de errores en un código de bloques



Estructura de un codificador/decodificador en la corrección de errores



c. Control de flujo

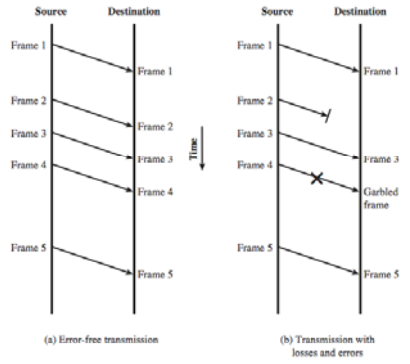
Control de flujo

Asegurar que la entidad emisora no sobrecarge a la entidad receptor previniento la saturación de su buffer influenciada por:

- Tiempo de transmisión
  - Tiempo que se toma para emitir a todos los bit al medio
- Tiempo de propagación
  - Tiempo que le toma a un bit atravezar el enlace

Asumir que no hay errores, ni retardos variables.

Modelos de transmisión de tramas sin/con errores

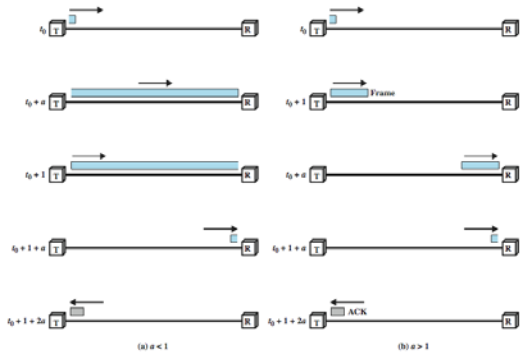


Stop and Wait

- La estación fuente transmite una trama
- La estación destino recibe la trama y responde con una confirmación (ACK)
- La estación fuente espera por la confirmación (ACK) antes de envair la proxima trama
- La estación destino puede parar el flujo al no enviar la confirmación (ACK)

- Este procedimiento trabaja bien para la transmisión de pocas tramas de tamaño grande
- Stop and wait es inadecuado si un bloque grande de datos se parte en muchas tramas pequeñas

Factor ¨a¨ de utilización del enlace en la transmisión Stop and Wait



Ventanas deslizantes para control de flujo

- Permiten la transmisión de múltiples tramas numeradas en tránsito
  - El receptor tiene un buffer de longitud W
  - El transmisor envia hasta W frames sin ACK
- ACK incluye al número de la próxima trama esperada
- El número de secuencia esta limitado por el tamaño del campo (K)
  - Las tramas están numeradas como módulo 2^k
  - Brindan un maximo tamaño de ventana de: $2^k - 1$
- El receptor puede confirmar tramas ACK sin perminir mas transmsiones (Receive Not Ready)
- El receptor debe enviar un ACK normal para reasumir

d. Control de errores

Control de errores

1. Detección y corrección de errores tales como:
   - Tramas perdidas
   - Tramas dañadas
- Las técnicas comunes usan:
  - Detección de errores
  - Confirmación (ACK) positiva
  - Retransmisión después del término de un temporizador
  - Confirmación negativa y retransmisión

Automatic Repeat Request (ARQ)

Es el nombre colectivo para mecanismos de control de errores tales como:

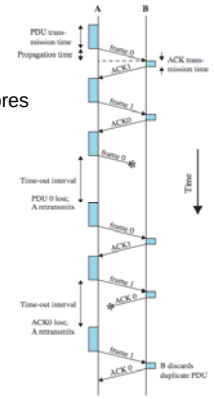
- stop and wait
- go back N
- Rechazo selectivo (retransmisión selectiva)

Stop and Wait

- La fuente transmite una sola trama
  - Espera por la confirmación ACK
- Si recibe la trama dañada, la descarta
  - El transmisor alcanza el limite de su temporizador
  - Si no llega un ACK dentro del límite, retransmite
- Si hay una ACK dañado, el transmisor no lo reconocerá
  - El transmisor retransmitirá
  - El receptor tendrá dos copias de la trama
  - Se emplea numeración alterna ACK0 / ACK1

Stop and Wait

- Ejemplo con ambos tipos de errores
- Pros y cons
  - simple
  - ineficiente



Go Back N

- Basado en ventana corrediza
  - Si no hay error, confirmar ACK como es usual
- Usar la ventana para controlar el número de tramas pendientes
- Si hay error, responder con una trama de rechazo
  - Descartar aquella trama y todas las tramas siguientes hasta que se reciba una trama sin error correctamente
  - El transmisor debe regresar y retransmitir aquella trama y las tramas subsecuentes

Go Back N - Handling

- Trama dañada
  - Error en la trama *i.* El receptor rechaza la trama *i*
  - El transmisor retransmite las tramas desde la trama *i*
- Trama perdida
  - La trama *i* la sido perdida, entonces
    - El transmisor envia la trama *i+1.* El receptor halla a la trama *i+1* fuera de secuencia y rechaza a la trama *i*
    - Ó el transmisor, activa su temporizador y envia una confirmación ACK con el bit P puesto para que el receptor responda con un ACK *i*
  - El transmisor entonces transmitre las tramas desde la trama *i*

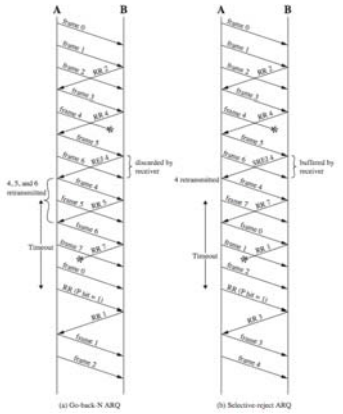
Go Back N - Handling

- ACK dañado
  - El receptor recibe la trama *i*, envía ACK (*i+1*) el cual se pierde
  - Los ACKs son acumulativos, de tal manera que el próximo ACK (*i+n*) podría llegar antes que el transmisor active su temporizador sobre la trama I
  - Si el transmisor activa su temporizador, este enviará un ACK con el bit P activado
  - Se puede repetir un número de veces antes que el procedimiento de reinicio sea iniciado
- Trama de rechazo REJ dañada
  - La trama de rechazo por una trama dañada se pierde
  - Manejada por la última trama cuando el trasmisor activa su temporizador

Rechazo selectivo

- También llamado retransmisión selectiva
- Solo las tramas rechazadas son retrasmitidas
- Las tramas subsecuentes, son aceptadas por el receptor y almacenadas en su buffer
- Minimiza las retransmisiones
- El receptor debe mantener un suficiente buffer grande
- Tiene un lógica más compleja en el transmisor
- Por esto es menos usado ampliamente
- Muy util en enlaces satelitales con grandes retardos de propagación

Go Back N vs Rechazo selectivo



Control de errores

Para detectar errores existen diversos métodos dependiendo su aplicación del tipo de errores que se presenten en la línea.

Existen errores aleatorios de un bit o dos bits y errores que se presentan en ráfagas.

Control de errores

- **PARIDAD**
- **CRC**
- **VRC**
- **LRC**
- **CHECKSUM**

Control de errores por paridad

Este método detecta un solo errores en transmisiones asíncronas y protocolos orientado al byte.

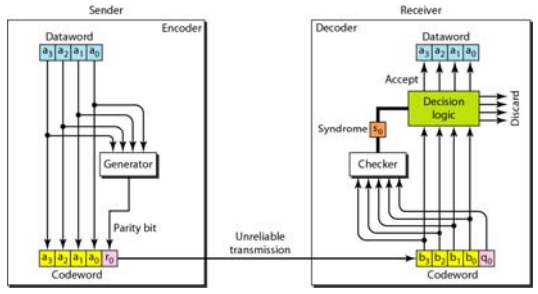
Con este esquema se añade un bit extra, el bit de paridad, a cada carácter antes de ser transmitido.

El receptor realiza una operación inversa y si:

- el resultado es el mismo, se asume que no ha habido errores.
- Si es diferente se asume que ha habido un error.

Sin embargo si hay dos errores, este carácter pasar con estos errores sin detectar.

Codificador y decodificador para código de paridad



Un código de paridad simple puede detectar errores de números impares

Checksum

Cuando se transmiten bloques de caracteres, hay una probabilidad incrementada que un carácter y de aquí el bloque contenga un error.

En este método se colocan los caracteres en un bloque de dos dimensiones. A cada carácter se le adiciona un bit de paridad con el método de control de errores por paridad.

Adicionalmente se le añade un bit de paridad por cada posición de bits a través de todos los caracteres.

Es decir se genera un carácter adicional, en el cual su bit iésimo es el bit de paridad para los iésimos bits de los caracteres.

Esto puede ser expresado con una operación de OR-EX.

Checksum

Entonces el bit de paridad al final de cada carácter es el bit de paridad de fila y es:

$$R_j = b_{1j} \oplus b_{2j} \oplus .... b_{nj}$$

Donde:

R_j = bit de paridad del carácter jiésimo

B_{ij} = bit iésimo del carácter jesímo

n = número de bits en un carácter

Los bits de paridad generados al final de cada carácter se conocen como Control de Redundancia vertical (Vertical Redundancy Check – VRC).

Checksum

Para generar el carácter de chequeo de paridad tenemos la fórmula:

$$Ci = b_{i1} \oplus b_{i2} \oplus ... \oplus b_{in}$$

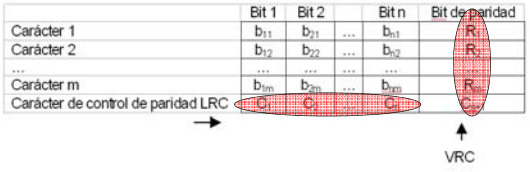
Donde:

C_i = bit iésimo del carácter de control de paridad

m = número de caracteres en un trama

Este carácter se denomina Control de Redundancia longitudinal (Longitudinal Redundancy Check – LRC)

Checksum



Control de redundancia cíclico - CRC

Una técnica más poderosa de control de errores es el control ciclico de redundancia (Cyclic Redundancy Check – CRC).

Dado una mensaje o trama de k bits de longitud, el transmisor genera una secuencia de n bits, conocido como la secuencia de control de trama (Frame Check Sequence – FCS), de tal manera que la trama resultante, consistente en k + n bits es exactamente divisible por algún número predeterminado.

Control de redundancia cíclico - CRC

El receptor divide la trama entrante por el mismo número predeterminado y si no hay residuo, se asume que la trama a llegado sin error.

Este procedimiento puede ser presentado en varias maneras, es decir con arimética de modulo 2, polinomios y compuertas OR-EX con registros de desplazamiento.

Control de redundancia cíclico - CRC

Trabajaremos con números binarios y aritmética de módulo 2.

La aritmética de módulo 2 utiliza la suma binaria sin llevar, como una operación de puertas del tipo OR-EX.

Por ejemplo:

```
  1 1 1 1                1 1 0 0 1
+ 1 0 1 0              x       1 1
  0 1 0 1                1 1 0 0 1
                       1 1 0 0 1
                       1 0 1 0 1 1
```

Control de redundancia cíclico - CRC

Ahora definimos:

T = trama de (k+n) bits a ser transmitida con n < k

M = mensaje de k bits. Son los primeros bits de la trama T

F = Secuencia de Control de Trama (Frame Check Secuence – FCS). Son los últimos bits de la trama T

P = patrón de n+1 bits. Este el divisor predeterminado mencionado anteriormente

Control de redundancia cíclico - CRC

Se desea que la división entre la trama T a ser transmitida y el patrón de n + 1 bits no tenga residuo.

Entonces:

$$T = 2^n M + F \qquad (1)$$

Es decir que multiplicando M por 2^n, tenemos el efecto de desplazar n bits a la izquierda y completar los bits corridos con 0's.

El añadir F nos da la concatenación de M y F, el cual resulta en T.

Control de redundancia cíclico - CRC

Ahora deseamos que T sea exactamente divisible por P. Si dividimos 2^nM entre P.

$$\frac{2^nM}{P} = Q + \frac{R}{P} \qquad (2)$$

Aquí tenemos un cociente y un residuo.

Debido a que la división es binaria el residuo es siempre un bit menor que el divisor.

Usamos este residuo como nuestro FCS.

Entonces se tiene:

$$T = 2^nM + R \qquad (3)$$

Control de redundancia cíclico - CRC

Ahora demostraremos que R satisface nuestra condición consideremos:

$$\frac{T}{P} = \frac{2^nM + R}{P} = \frac{2^nM}{P} + \frac{R}{P} \qquad (4)$$

Sustituyendo la ecuación (2) en la (4) tenemos que:

$$\frac{T}{P} = Q + \frac{R}{P} + \frac{R}{P} \qquad (5)$$

Control de redundancia cíclico - CRC

Se sabe, que todo número binario añadido a si mismo en módulo 2 es igual a cero. Entonces:

$$\frac{T}{P} = Q + \frac{R + R}{P} = Q$$

Así se demuestra que no hay residuo, de allí que T es divisible exactamente por P.

De esta manera se genera el FCS.

Se divide 2^nM por P y se usa el residuo como el FCS.

En la recepción, el receptor dividirá a T entre P y si no se obtiene residuo, es señal que la trama se ha recibido sin errores.

Ejemplo de Control de redundancia cíclico

Dados:

Mensaje M = 1010001101 (10 bits)
Patrón P = 110101 (6 bits)
FCS (R) = a ser calculado (5 bits)

El mensaje se multiplica por 2^5, resultando:

101000110100000 = 2^nM = 2^5M

Ejemplo de Control de redundancia cíclico

Este producto 2^5M es dividido por P:

```
2^5M    101000110100000 | 110101        P
        110101            1101010110  <- Q
        111011
        110101
          111010
          110101
            111110
            110101
              101100
              110101
                110010
                110101
                  1110  <- R
```

Ejemplo de Control de redundancia cíclico

El residuo es añadido al valor de 2^nM para dar:

T = 101000110101110,

el cual es transmitido.

Si no hay errores, el receptor recibirá a T intacto.

Ejemplo de Control de redundancia cíclico

La trama recibida es dividida por P:

```
101000110101110 | 110101
110101                  110101
 111011
 110101
   111010
   110101
     111110
     110101
       101111
       110101
         110101
         110101
           00
```

Control de redundancia cíclico - CRC

Debido a que no hay residuo, se asume que se ha recibido una trama sin errores.

Existen 4 versiones de CRC ampliamente utilizadas:

Name	*Polynomial*	*Application*
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

Códigos lineales en bloque

Un código de bloque requiere que un mensaje sea particionado en bloques de bits.

Un palabra de datos, (data word) es un bloque tiene k bits.

Luego el número de palabras de datos (data words) es: 2^k

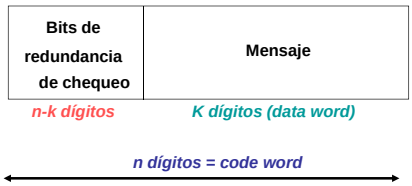
Las palabras de datos se codifican dentro de palabras de código (code words).

Cada codeword es un bloque de **n** bits.

El número posible de palabras de código (code words), es 2^n, pero sólo se emplean 2^k, las cuales serán transmitidas.

Los n-k bits adicionales son denominados ó referidos como bits de chequeo de paridad

Palabra de código en forma sistemática



Velocidad de código

La velocidad de código r_c, esta definida por:

$$r_c = \frac{k}{n}$$

La notación del código es (n, k).

Por ejemplo un código que contiene una palabra de datos de 4 bits en una palabra de código de 7 bits será un código:

(k = 4, n =7,)

Su velocidad de código será: $r_c = 4/7$

Código de repetición

Un código de repetición muestra algunas de las propiedades de los códigos de bloque.

En un código de repetición cada bit es una palabra de datos:

K = 1

Para una codificación de redundancia **n**, la salida del codificador será de **n** bits idénticos al bit de entrada.

Considerando el caso de **n** = 3

Si se transmite 1, la salida es la palabra de código 111

Si se transmite 0 la palabra de código será 000

Distancia de Hamming

La distancia de Hamming entre dos palabras de código es el número de posiciones en las cuales estas palabras difieren.

Cuanto menor sea la distancia entre dos palabras de código, el código será mejor.

En este código la distancia mínima es 2.

Dataword	Modulo-2 addition of dataword	Codeword
000	0	0000
001	1	0011
010	1	0101
011	0	0110
100	1	1001
101	0	1010
110	0	1100
111	1	1111

Matrices empleadas en códigos

Una palabra de datos de k bits, se denota por un vector fila **d**.

Por ejemplo la sexta palabra de datos en la tabla es vector fila **d6** = [101] y su palabra de código es el vector fila **c6** = [1010]

Dataword	Modulo-2 addition of dataword	Codeword
000	0	0000
001	1	0011
010	1	0101
011	0	0110
100	1	1001
101	0	1010
110	0	1100
111	1	1111

Matrices empleadas en códigos

En general, una palabra de código **c** se genera multiplicando a la palabra de datos **d** por una matriz generadora **G**, donde:

c = **dG**

Un ejemplo de matriz generadora para el código (7,4) es:

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

Generación de una palabra de código

La primeras 4 columnas forman una matriz identidad.

Esta submatriz permite que los primeros 4 bits de la palabra de código, correspondan a la palabra de datos.

Los bits restantes generan los bits de paridad desde los bits de datos.

Generación de una palabra de código

Como ejemplo generemos la palabra de código desde la palabra de datos [1010].

$$C = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Submatriz P y su transpuesta

Las 3 últimas columnas forma la submatriz **P** (Parity bits):

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

Donde tomando su transpuesta, tenemos a P^T:

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

Matriz de chequeo de paridad H

Luego formamos a la matriz H incluyendo la matriz identidad a la matriz P^T.

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

El número de filas en H es igual al número de bits de paridad n-k, y el número de columnas es n,

Así la matriz de chequeo de paridad es de (n-k, n).

Una propiedad fundamental de las matrices de código es el producto:

$$GH^T = 0$$

Matriz de chequeo de paridad H

Cuando se recibe una palabra de código, esta puede ser verificada si es correcta al multiplicarla por $\mathbf{H^T}$.

El producto $\mathbf{cH^T}$ debe ser igual a $\mathbf{0}$.

Esto es una consecuencia desde que $\mathbf{c} = \mathbf{dG}$.

Si a ambos términos los multiplicamos por $\mathbf{H^T}$, tendremos: $\mathbf{cH^T} = \mathbf{dGH^T}$ lo cual es igual a $\mathbf{0}$

Si el resultado no es cero, es que se ha recibido un error.

En términos generales, el producto $c\mathbf{H^T}$ proporciona lo que es conocido como un síndrome, y esta denotado por $\mathbf{s}$:

$$\mathbf{s} = c\mathbf{H^T}$$

Síndrome s

En general, si se recibe una palabra de código $\mathbf{c_R}$ y la palabra de código transmitida es $\mathbf{c_T}$ y el vector error es $\mathbf{e}$, usando la adición módulo 2 tenemos:

$$\mathbf{c_R} = \mathbf{c_T} + \mathbf{e}$$

Sustituyendo $\mathbf{c}$ en la ecuación $\mathbf{s} = \mathbf{cH^T}$ tenemos:

$\mathbf{s} = (\mathbf{c_T} + \mathbf{e})\ \mathbf{H^T} = \mathbf{c_T}\ \mathbf{H^T} + \mathbf{e}\ \mathbf{H^T}$ pero $\mathbf{c_T}\ \mathbf{H^T} = 0$, entonces:

$$\mathbf{s} = \mathbf{eH^T}$$

Este resultado muestra que el síndrome depende sólo del error vector y es independiente de la palabra de código transmitida.

Como el vector error tiene n bits, son posibles 2^n vectores de error.

Síndrome s

El síndrome solo tiene n-k bits (determinado por el número de filas de la matriz H, lo cual nos da 2^{n-k} síndromes.

Como uno de estos es el síndrome cero, el número de errores que puede ser detectado es 2^{n-k} -1.

En la práctica los decodificadores se diseñan para corregir los errores de mayor probabilidad.

Ejemplo: error de un solo bit.

El síndrome recibido es comparado con los valores de una tabla de tabulación de los patrones de error conocidos, y se halla al error más probable. Esto se conoce como decodificación de máxima probabilidad.

Ejemplo de corrección de un solo error

Supongamos que se transmite la palabra de código [10100010]. Pero ocurre un error en el quinto bit (contando de la izquierda), de tal manera que la palabra de código recibida es [10101110]. Aplicando la ecuación: $\mathbf{s} = c\mathbf{H}^T$

$$s = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$s = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

El síndrome [100] corresponde a la quinta fila de la matriz H, indica que el quinto bit esta errado. Este bit se cambia y corrige el error.

Códigos cíclicos

Estos códigos son una subclase de los códigos de bloque. Su propiedad es que un desplazamiento cíclico de una palabra de código también es una palabra de código.

Ejemplo:

Una palabra de código es $\{c_1\ c_2\ c_3\ c_4\ c_5\ c_6\ c_7\}$, luego la palabra $\{c_2\ c_3\ c_4\ c_5\ c_6\ c_7\ c_1\}$ también es una palabra de código.

Los códigos cíclicos se implementan con registros de desplazamiento y se usan en transmisión satelital.

Como todo código es una invención, los códigos son nombrados por sus inventores.

Códigos cíclicos y códigos lineales

- Códigos cíclicos:
  - Códigos Hamming
  - Código BCH (Bose, Chaudhuri y Hocquenghen)
  - Códigos R-S (Reed-Solomon)
- Códigos lineales:
  - Códigos de convolución
    - Decodificación de códigos de convolución
- Intercalado: Interleaving
- Códigos Concatenados

Códigos cíclicos: Códigos Hamming

Se define que para un entero m mayor ó igual a 2, los valores de k y n están relacionados por:

$n = 2^m - 1$ y $k = n - m$

Conforme la velocidad de código $r_c = k/n$ se aproxima a 1, m aumenta, lo cual hace más eficiente. Sin embargo, este tipo de código solo se puede corregir un solo error. Algunas de las combinaciones permisibles son:

m	n	k
2	3	1
3	7	4
4	15	11
5	31	26
6	63	57
7	127	120

Código cíclico: BCH (Bose, Chaudhuri y Hocquenghen)

Estos códigos corrigen hasta t errores y m puede ser cualquier entero positivo. Entre los valores permisibles son:

$n = 2^m - 1$ y k mayor/igual = $(n - mt)$

Los enteros m y t son arbitrarios lo cual brinda flexibilidad al diseñador de códigos.

n	k	t
7	4	1
15	11	1
15	7	2
15	5	3
31	26	1
31	21	2
31	16	3
31	11	5
31	6	7

Códigos cíclicos: Códigos R-S (Reed-Solomon)

Se emplean con errores que ocurren en ráfagas, es decir los errores vienen agrupados.

Estos códigos en lugar de codificar directamente en bits, los bits primero son agrupados en símbolos.

Luego las palabras de datos y las palabras de código son codificadas en estos símbolos.

Lo errores que afectan a un grupo de bits es más probable que afecten a solo un solo símbolo, el cual puede ser corregido por el código R-S.

Corrección de errores hacia delante - FEC

La corrección de errores hacia adelante (en inglés, Forward Error Correction o FEC) es un tipo de mecanismo de corrección de errores que permite su corrección en el receptor sin retransmisión de la información original.

Se utiliza en sistemas sin retorno o sistemas en tiempo real donde no se puede esperar a la retransmisión para mostrar los datos.

Corrección de errores hacia delante - Funcionamiento

La posibilidad de corregir errores se consigue añadiendo al mensaje original unos bits de redundancia.

La fuente envía la secuencia de datos al codificador, encargado de añadir dichos bits de redundancia.

A la salida del codificador obtenemos la denominada palabra código.

Esta palabra código es enviada al receptor y éste, mediante el decodificador adecuado y aplicando los algoritmos de corrección de errores, obtendrá la secuencia de datos original.

Principales tipos de codificación FEC

- Códigos convolucionales

Códigos convolucionales

Los bits se van codificando tal y como van llegando al codificador. Cabe destacar que la codificación de uno de los bits está influenciada por la de sus predecesores.

La decodificación para este tipo de código es compleja y es necesaria una gran cantidad de memoria para estimar la secuencia de datos más probable para los bits recibidos.

En la actualidad se utiliza para decodificar este tipo de códigos algoritmo de Viterbi, por su gran eficiencia en el consumo de recursos.

Códigos lineales: Códigos de convolución

Un codificador de convolución consiste en:

- Un registro de desplazamiento que almacena temporalmente y permite desplazar los bits entrantes y
- Una circuitería de puertas OR-Ex que generará una salida codificada desde los bits que están almacenados en ese momento en el registro de desplazamiento.

En general, los k bits de datos puede ser desplazados en el registro a la vez, y se generarán n bits de código.

Ejemplo: k = 1 y n = 2, obteniéndose un código de r_c=1/2.

El registro se inicia cargado con bits 0s.

La entrada es R_b bits

Codificador convolucional de tasa $r_c = 1/2$

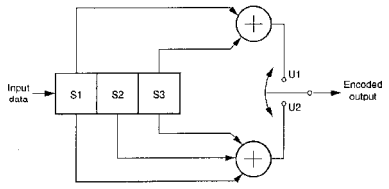


Diagrama de árbol del codificador $r_c = ½$

Entrada: 1001

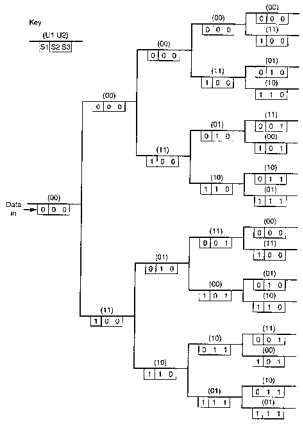


Diagrama de árbol del codificador $r_c = ½$

Entrada: 1001

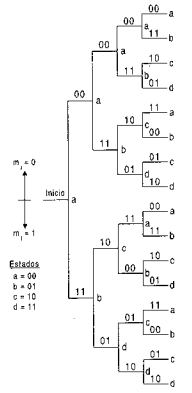


Figura B.3. Diagrama de árbol para el codificador (2, 1, 2)

Diagrama de rejilla (trellis) del codificador $r_c = ½$

Entrada: 1001

(n, k, L)
(2, 1, 2)

L = número de bits de estado

k = bits de datos

n = bits de código

——— = 0

..... = 1

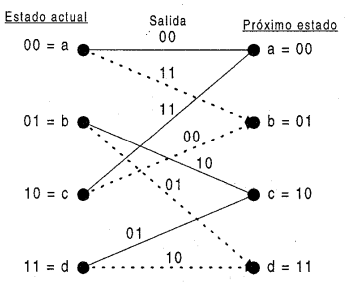
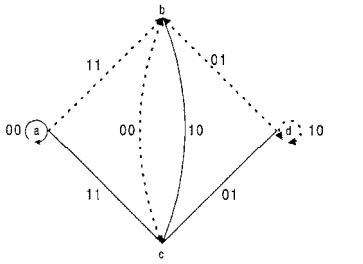


Diagrama de rejilla (trellis) del codificador $r_c = ½$ Entrada: 1001 (n, k, L) (2, 1, 2) L = número de bits de estado



Entrada	1	1	0	1	1	1	0	0	1	0	0	0	
Estado	a	b	d	c	b	d	d	c	a	b	c	a	a
Salida	11	01	01	00	01	10	01	11	11	10	11	00	

Decodificación

Hay tres tipos de decodificación:

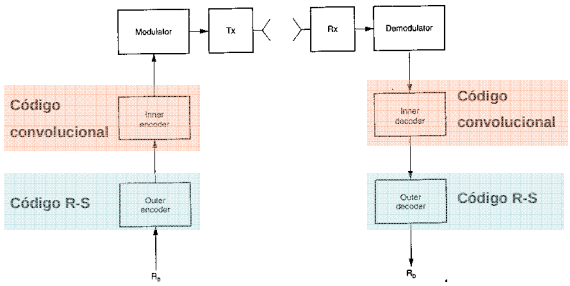
- Decodificación de Viterbi (máxima verosimilitud)
- Decodificación secuencial
- Descodificación de realimentación (feedback decoding)

Codigos concatenados para mejora de la performance

Los códigos clásicos de bloques y los códigos convolucionales se combinan frecuentemente en esquemas concatenados.

En estos esquemas, los codigos convolucionales hacen la corrección de errores aleatorios, mientras que el código de bloque (usualmente Reed Solomon) corrige los errores en ráfagas.

Códigos concatenados

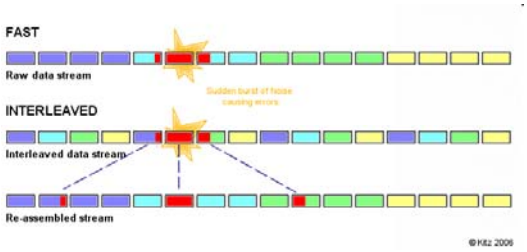


Intercalamiento (Interleaving)

La idea es cambiar el orden en la cual los bits son transmitidos de tal manera que las ráfagas de errores sean dispersados en varias palabras de código en vez de concentrarse en una sola palabra de código.

Se emplea conjuntamente con los codificadores de bloque y los codificadores de convolución.

Intercalamiento (Interleaving)



Generación del intercalamiento