Corrección de errores

Fundamentos de Sistemas de Comunicación

Facultad de Ingeniería y Tecnología

25 de mayo de 2022

Co-funded by the
Erasmus+ Programme
of the European Union

Contenido

Codificación

Codificar es representar un mensaje para servir algún propósito en la comunicación.

Codificación

Codificar es representar un mensaje para servir algún propósito en la comunicación.

Ya vimos un ejemplo: codificación de Gray para mensajes binarios sobre canales M-arios.

Codificación

Codificar es representar un mensaje para servir algún propósito en la comunicación.

Ya vimos un ejemplo: codificación de Gray para mensajes binarios sobre canales M-arios.

Tres própositos principales:

Codificación

Codificar es representar un mensaje para servir algún propósito en la comunicación.

Ya vimos un ejemplo: codificación de Gray para mensajes binarios sobre canales M-arios.

Tres própositos principales:

1. Corrección de errores (codificación de canal).

Codificación

Codificar es representar un mensaje para servir algún propósito en la comunicación.

Ya vimos un ejemplo: codificación de Gray para mensajes binarios sobre canales M-arios.

Tres própositos principales:

1. Corrección de errores (codificación de canal).
2. Minimizar el largo del mensaje (codificación de fuente).

Codificación

Codificar es representar un mensaje para servir algún propósito en la comunicación.

Ya vimos un ejemplo: codificación de Gray para mensajes binarios sobre canales M-arios.

Tres própositos principales:

1. Corrección de errores (codificación de canal).
2. Minimizar el largo del mensaje (codificación de fuente).
3. Garantizar la autenticidad, privacidad, e integridad del mensaje (encriptación).

Objetivo: Detectar y corregir errores en la comunicación.

Objetivo: Detectar y corregir errores en la comunicación.

Dos arquitecturas de sistema posibles: detectar errores y solicitar re-transmisión (ARQ), o realizar la corrección directamente en el receptor (FEC).

Objetivo: Detectar y corregir errores en la comunicación.

Dos arquitecturas de sistema posibles: detectar errores y solicitar re-transmisión (ARQ), o realizar la corrección directamente en el receptor (FEC).

Estrategia: codificación de canal

En lugar de representar el mensaje a transmitir directamente, elijo una representación alternativa que permita detectar, o detectar y corregir, errores. Esta representación alternativa es la que es la que efectivamente se transmite.

Ejemplo: códigos de redundancia (1/2)

Quiero transmitir un solo bit '0' o '1'. En el sistema de comunicación el mensaje '0' se representa como '00' (dos ceros!), y el '1' como '11'.

Ejemplo: códigos de redundancia (1/2)

Quiero transmitir un solo bit '0' o '1'. En el sistema de comunicación el mensaje '0' se representa como '00' (dos ceros!), y el '1' como '11'.

Si en recepción obtengo un '00' o un '11', asumo que me envíaron un '0' o un '1' respectivamente. Si recibo un '01' o un '10', sé que hubo (al menos) un error.

Ejemplo: códigos de redundancia (1/2)

Quiero transmitir un solo bit '0' o '1'. En el sistema de comunicación el mensaje '0' se representa como '00' (dos ceros!), y el '1' como '11'.

Si en recepción obtengo un '00' o un '11', asumo que me envíaron un '0' o un '1' respectivamente. Si recibo un '01' o un '10', sé que hubo (al menos) un error.

Si BER $= \alpha$, la probabilidad de que ocurra un error que no detecto es α^2.

Ejemplo: códigos de redundancia (1/2)

Quiero transmitir un solo bit '0' o '1'. En el sistema de comunicación el mensaje '0' se representa como '00' (dos ceros!), y el '1' como '11'.

Si en recepción obtengo un '00' o un '11', asumo que me envíaron un '0' o un '1' respectivamente. Si recibo un '01' o un '10', sé que hubo (al menos) un error.

Si BER $= \alpha$, la probabilidad de que ocurra un error que no detecto es α^2.

El costo es la reducción de la tasa efectiva de transmisión a la mitad.

Ejemplo: códigos de redundancia (2/2)

En el sistema anterior, los errores se pueden detectar, pero no corregir. Incluyamos más redundancia: el mensaje '0' se representa como '000' (tres ceros!), y el '1' como '111'.

Ejemplo: códigos de redundancia (2/2)

En el sistema anterior, los errores se pueden detectar, pero no corregir. Incluyamos más redundancia: el mensaje '0' se representa como '000' (tres ceros!), y el '1' como '111'.

Ahora puedo detectar hasta dos errores, y corregir hasta uno: p.ej. si recibo el '010' asumo que se quiso envíar el '000' que corresponde al bit '0'.

La nueva tasa de error (incorregibles) es $3\alpha^2 + \alpha^3$ y la tasa de errores no detectados es α^3.

¡Pero la tasa de bits efectiva se reduce a un tercio!

Códigos de paridad

Una versión más eficiente de los códigos de redundancia.

Una versión más eficiente de los códigos de redundancia.

Ejemplo: código de Hamming (7,4)

Msj.	Código	Msj.	Código	Msj.	Código	Msj.	Código
0000	000<u>0</u>0<u>00</u>	0100	100<u>1</u>1<u>00</u>	1000	111<u>0</u>0<u>00</u>	1100	011<u>1</u>1<u>00</u>
0001	110<u>1</u>0<u>01</u>	0101	010<u>0</u>1<u>01</u>	1001	001<u>1</u>0<u>01</u>	1101	101<u>0</u>1<u>01</u>
0010	010<u>1</u>0<u>10</u>	0110	110<u>0</u>1<u>10</u>	1010	101<u>1</u>0<u>10</u>	1110	001<u>0</u>1<u>10</u>
0011	100<u>0</u>0<u>11</u>	0111	000<u>1</u>1<u>11</u>	1011	011<u>0</u>0<u>11</u>	1111	111<u>1</u>1<u>11</u>

Una versión más eficiente de los códigos de redundancia.
Ejemplo: código de Hamming (7,4)

Msj.	Código	Msj.	Código	Msj.	Código	Msj.	Código
0000	0000000	0100	1001100	1000	1110000	1100	0111100
0001	1101001	0101	0100101	1001	0011001	1101	1010101
0010	0101010	0110	1100110	1010	1011010	1110	0010110
0011	1000011	0111	0001111	1011	0110011	1111	1111111

Cuadro 1: Notar que cada 4 bits de mensaje original se envían 7. De todas las palabras de código de 7 bits (128), solo algunas (16) son válidas. Los bits 1, 2, y 4 son de paridad de los bits de mensaje (1,2,4), (1,3,4), y (2,3,4) respectivamente.

Una versión más eficiente de los códigos de redundancia.
Ejemplo: código de Hamming (7,4)

Msj.	Código	Msj.	Código	Msj.	Código	Msj.	Código
0000	0000000	0100	1001100	1000	1110000	1100	0111100
0001	1101001	0101	0100101	1001	0011001	1101	1010101
0010	0101010	0110	1100110	1010	1011010	1110	0010110
0011	1000011	0111	0001111	1011	0110011	1111	1111111

Cuadro 1: Notar que cada 4 bits de mensaje original se envían 7. De todas las palabras de código de 7 bits (128), solo algunas (16) son válidas. Los bits 1, 2, y 4 son de paridad de los bits de mensaje (1,2,4), (1,3,4), y (2,3,4) respectivamente.

Este código permite corregir un error de bit por 'palabra' de código con 'solo' 75 % de redundancia.

Caracterización de códigos: tasa de código

Sea un código que codifica mensajes de largo k en palabras de código de largo n, lo que se denomina un código de bloque (n, k).

Tasa de código: cuantifica la redundancia (eficiencia) del código utilizado

$$R_c = k/n$$

Caracterización de códigos: tasa de código

Sea un código que codifica mensajes de largo k en palabras de código de largo n, lo que se denomina un código de bloque (n, k).

Tasa de código: cuantifica la redundancia (eficiencia) del código utilizado

$$R_c = k/n$$

Ejemplo: Hamming(7,4) tiene $R_c = 4/7$.

Ejemplo: El primer código de redundancia visto hoy tiene $R_c = 1/2$.

Caracterización de códigos: distancia de Hamming

La distancia entre dos palabras de código se define como el número de símbolos distintos entre ambas.

Ejemplo: 011 y 110 tienen $d = 2$.

Caracterización de códigos: distancia de Hamming

La distancia entre dos palabras de código se define como el número de símbolos distintos entre ambas.

Ejemplo: 011 y 110 tienen $d = 2$.

Distancia de Hamming de un código

Es la mínima distancia entre dos palabras válidas de código.

Caracterización de códigos: distancia de Hamming

La distancia entre dos palabras de código se define como el número de símbolos distintos entre ambas.

Ejemplo: 011 y 110 tienen $d = 2$.

Distancia de Hamming de un código

Es la mínima distancia entre dos palabras válidas de código.

Ejemplo: el código de redundancia visto hoy en el que solo '000' y '111' eran palabras válidas, tiene $d_{min} = 3$. El código de Hamming (7,4) también tiene $d_{min} = 3$.

Caracterización de códigos: distancia de Hamming

La distancia entre dos palabras de código se define como el número de símbolos distintos entre ambas.

Ejemplo: 011 y 110 tienen $d = 2$.

Distancia de Hamming de un código

Es la mínima distancia entre dos palabras válidas de código.

Ejemplo: el código de redundancia visto hoy en el que solo '000' y '111' eran palabras válidas, tiene $d_{min} = 3$. El código de Hamming (7,4) también tiene $d_{min} = 3$.

Se puede probar que:

- Para detectar l errores, es necesario $d_{min} \geq l + 1$

Caracterización de códigos: distancia de Hamming

La distancia entre dos palabras de código se define como el número de símbolos distintos entre ambas.

Ejemplo: 011 y 110 tienen $d = 2$.

Distancia de Hamming de un código

Es la mínima distancia entre dos palabras válidas de código.

Ejemplo: el código de redundancia visto hoy en el que solo '000' y '111' eran palabras válidas, tiene $d_{min} = 3$. El código de Hamming (7,4) también tiene $d_{min} = 3$.

Se puede probar que:

- Para detectar l errores, es necesario $d_{min} \geq l + 1$
- Para corregir t errores, es necesario $d_{min} \geq 2t + 1$

Caracterización de códigos: distancia de Hamming

La distancia entre dos palabras de código se define como el número de símbolos distintos entre ambas.

Ejemplo: 011 y 110 tienen $d = 2$.

Distancia de Hamming de un código

Es la mínima distancia entre dos palabras válidas de código.

Ejemplo: el código de redundancia visto hoy en el que solo '000' y '111' eran palabras válidas, tiene $d_{min} = 3$. El código de Hamming (7,4) también tiene $d_{min} = 3$.

Se puede probar que:

- Para detectar l errores, es necesario $d_{min} \geq l + 1$
- Para corregir t errores, es necesario $d_{min} \geq 2t + 1$
- Siempre $d_{min} \leq n - k + 1$

Otras familias de códigos de redundancia

- Cyclic Redundancy Check (CRC)
- Convolutional codes (caso especial: turbo codes)

FEC: forward error-correction. El Rx intenta detectar y corregir errores sin participación del Tx. No requiere hardware adicional, pero los códigos necesarios son más complejos (y menos eficientes).

Arquitecturas de corrección de errores: ARQ y FEC

FEC: forward error-correction. El Rx intenta detectar y corregir errores sin participación del Tx. No requiere hardware adicional, pero los códigos necesarios son más complejos (y menos eficientes).

ARQ: automatic repeat request. El sistema solo busca detectar errores y cuando los encuentra, solicita la retransmisión de la parte corrupta del mensaje.
Requiere un canal de comunicación bi-direccional y lógica en Tx para la retransmisión.

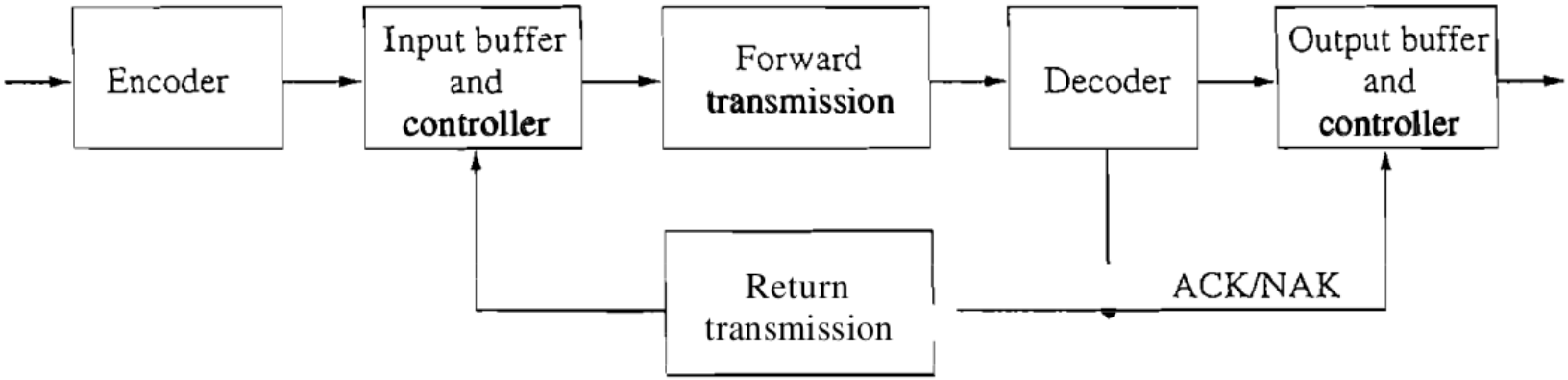


Figure 13.1–6 ARQ system

Performance FEC (1/2)

Sea un sist. de comunicación digital sin codificación de canal.
Supongamos que:

$$\text{BER} = \alpha_0$$

Performance FEC (1/2)

Sea un sist. de comunicación digital sin codificación de canal. Supongamos que:

$$\text{BER} = \alpha_0$$

Con codificación de canal, **si mantenemos la tasa de bits de mensaje**, es necesario **aumentar la tasa de bits totales enviados en el sistema**. Específicamente:

$$r_b^{mensaje} = r_b^{real} R_c$$

Performance FEC (1/2)

Sea un sist. de comunicación digital sin codificación de canal. Supongamos que:

$$\text{BER} = \alpha_0$$

Con codificación de canal, **si mantenemos la tasa de bits de mensaje**, es necesario **aumentar la tasa de bits totales enviados en el sistema**. Específicamente:

$$r_b^{mensaje} = r_b^{real} R_c$$

Una mayor tasa de bits real implica un **aumento** de la tasa de errores ($\alpha > \alpha_0$)...

Performance FEC (1/2)

Sea un sist. de comunicación digital sin codificación de canal. Supongamos que:

$$\text{BER} = \alpha_0$$

Con codificación de canal, **si mantenemos la tasa de bits de mensaje**, es necesario **aumentar la tasa de bits totales enviados en el sistema**. Específicamente:

$$r_b^{mensaje} = r_b^{real} R_c$$

Una mayor tasa de bits real implica un **aumento** de la tasa de errores ($\alpha > \alpha_0$)... pero algunos de ellos pueden corregirse.

Performance FEC (2/2)

Hasta t errores por palabra de código son corregibles

Hasta t errores por palabra de código son corregibles
La tasa de errores que 'pasan' es:

$$\text{p(t+1 o más errores en n)} = \sum_{k=t+1}^{n} C_k^n \alpha^k \approx C_{t+1}^n \alpha^{t+1}$$

Hasta t errores por palabra de código son corregibles
La tasa de errores que 'pasan' es:

$$\text{p(t+1 o más errores en n)} = \sum_{k=t+1}^{n} C_k^n \alpha^k \approx C_{t+1}^n \alpha^{t+1}$$

Normalizando por bit:

$$\text{BER} \approx \frac{t+1}{n} C_{t+1}^n \alpha^{t+1}$$

De donde se desprende:

Performance FEC

$$\text{BER} = C_t^{n-1} \alpha^{t+1}$$

Ejemplo: performance sistema FEC (1/3)

Sea un sist. digital binario óptimo sin codificación de canal.
Aceptemos (sin prueba) que:

$$\text{BER} = Q(\sqrt{2\gamma_b})$$

Con $\gamma_b = E_b/N_0$, E_b es la energía **por bit de mensaje** [Joule/bit] y N_0 es la densidad espectral de potencia del ruido [Watt/Hz].

Ejemplo: performance sistema FEC (1/3)

Sea un sist. digital binario óptimo sin codificación de canal.
Aceptemos (sin prueba) que:

$$\text{BER} = Q(\sqrt{2\gamma_b})$$

Con $\gamma_b = E_b/N_0$, E_b es la energía **por bit de mensaje** [Joule/bit] y N_0 es la densidad espectral de potencia del ruido [Watt/Hz].

Ejemplo: performance sistema FEC (1/3)

Sea un sist. digital binario óptimo sin codificación de canal. Aceptemos (sin prueba) que:

$$\text{BER} = Q(\sqrt{2\gamma_b})$$

Con $\gamma_b = E_b/N_0$, E_b es la energía **por bit de mensaje** [Joule/bit] y N_0 es la densidad espectral de potencia del ruido [Watt/Hz].

Con codificación de canal, **si mantenemos la tasa de bits de mensaje**, la probabilidad de error de bit **crece** a:

$$\alpha = Q(\sqrt{2R_c\gamma_b})$$

...porque invertimos menos energía por bit transmitido para el mismo mensaje!

Ejemplo: performance de sistema FEC (2/3)

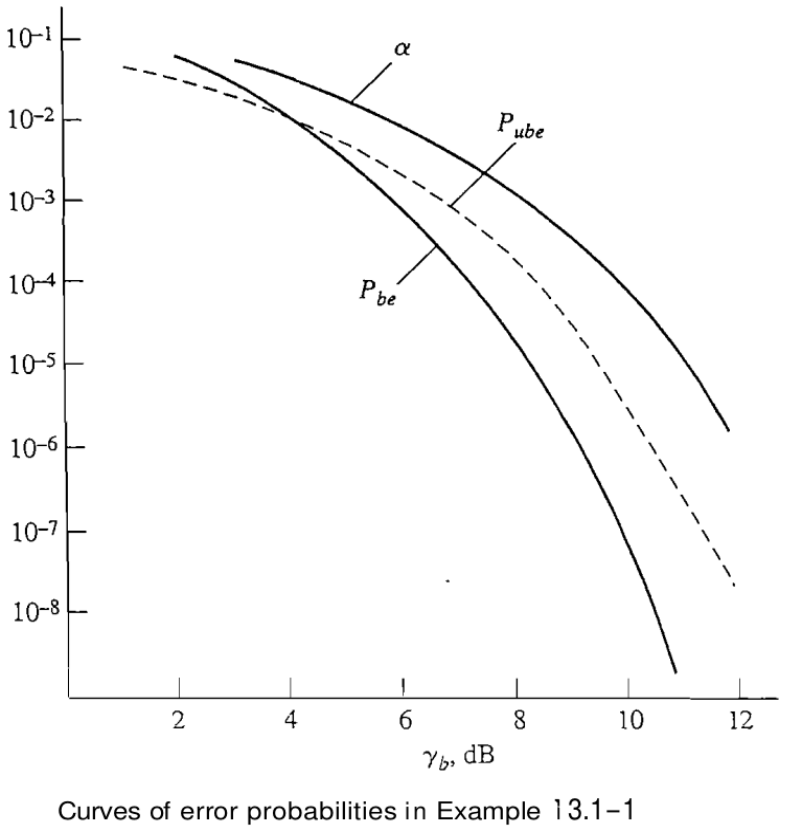
Se utiliza un código (15,11) con $d_{min} = 3$, en un sistema de corrección de errores tipo FEC. Si $S_R = 3dBm$, $N_0 = 10^{-7} W/Hz$, y la tasa de bits de mensaje debe ser $r_b = 2000 bps$, halle la tasa de errores del sistema con y sin corrección de errores. Para el sistema con corrección de errores, calcule la nueva tasa de bits necesaria en transmisión, y el aumento en el ancho de banda del sistema si no se cambia el sistema de comunicación.

Ejemplo: performance de sistema FEC (2/3)

Se utiliza un código (15,11) con $d_{min} = 3$, en un sistema de corrección de errores tipo FEC. Si $S_R = 3dBm$, $N_0 = 10^{-7} W/Hz$, y la tasa de bits de mensaje debe ser $r_b = 2000 bps$, halle la tasa de errores del sistema con y sin corrección de errores. Para el sistema con corrección de errores, calcule la nueva tasa de bits necesaria en transmisión, y el aumento en el ancho de banda del sistema si no se cambia el sistema de comunicación.

Soluciones: 4×10^{-6} (sin), 10^{-7} (con), nueva tasa es $2727 bps$: 40 % mayor (¡y el ancho de banda lo mismo!).

Ejemplo: performance de sistema FEC (3/3)



Curves of error probabilities in Example 13.1–1

Figura 1: Performance de un sistema FEC con código de bloque $(15, 11)$ $(t = 1)$.

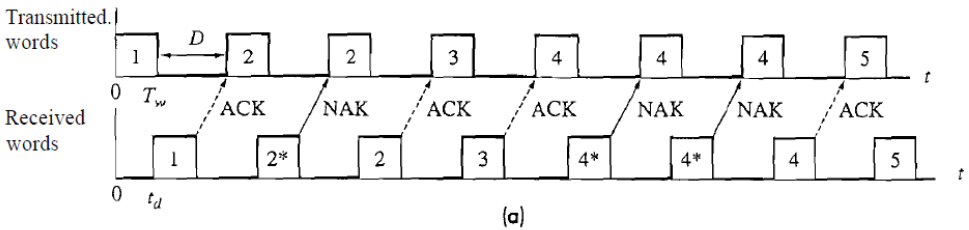
Figura 13.1-3. *Communication Systems.* Carlson, Crilly, y Rutledge. 4ta Ed.

- $P_{ube} = Q(\sqrt{2\gamma_b})$ probabilidad de error de bit sin usar codif. de canal.
- $\alpha = Q(\sqrt{22/15\gamma_b})$ probabilidad de error de bit usando codif. de canal (corregible)
- $P_{be} = 14\alpha^2$ prob. de error de bit no corregible con codif. de canal

Un sistema ARQ (solicitud de respuesta automática) solamente **detecta** errores en recepción, y en lugar de intentar corregirlos, solicita su reenvío. **Requiere canal bidireccional**.

Un sistema ARQ (solicitud de respuesta automática) solamente **detecta** errores en recepción, y en lugar de intentar corregirlos, solicita su reenvío. **Requiere canal bidireccional**.

Varias maneras de implementar:



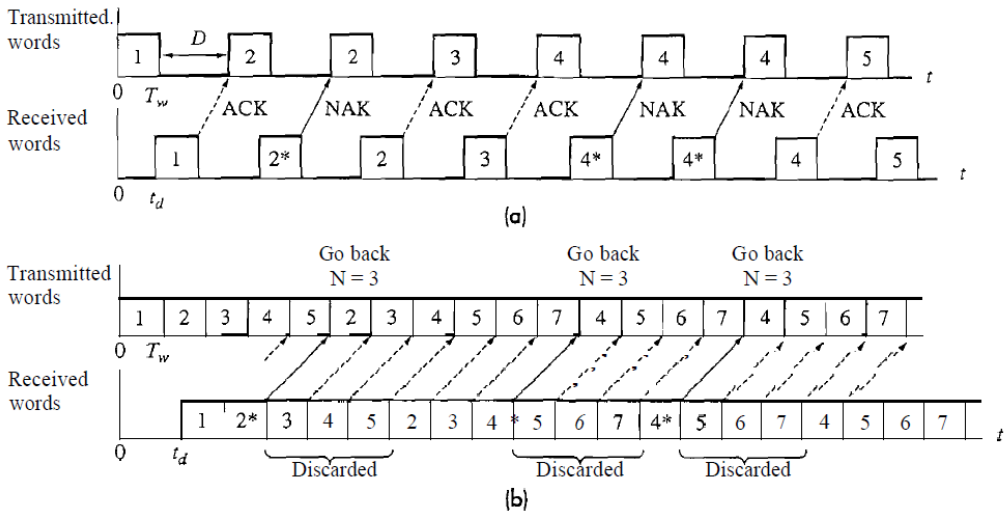
(a)

- Transmito y espero (stop-and-wait).

Un sistema ARQ (solicitud de respuesta automática) solamente **detecta** errores en recepción, y en lugar de intentar corregirlos, solicita su reenvío. **Requiere canal bidireccional**.

Varias maneras de implementar:

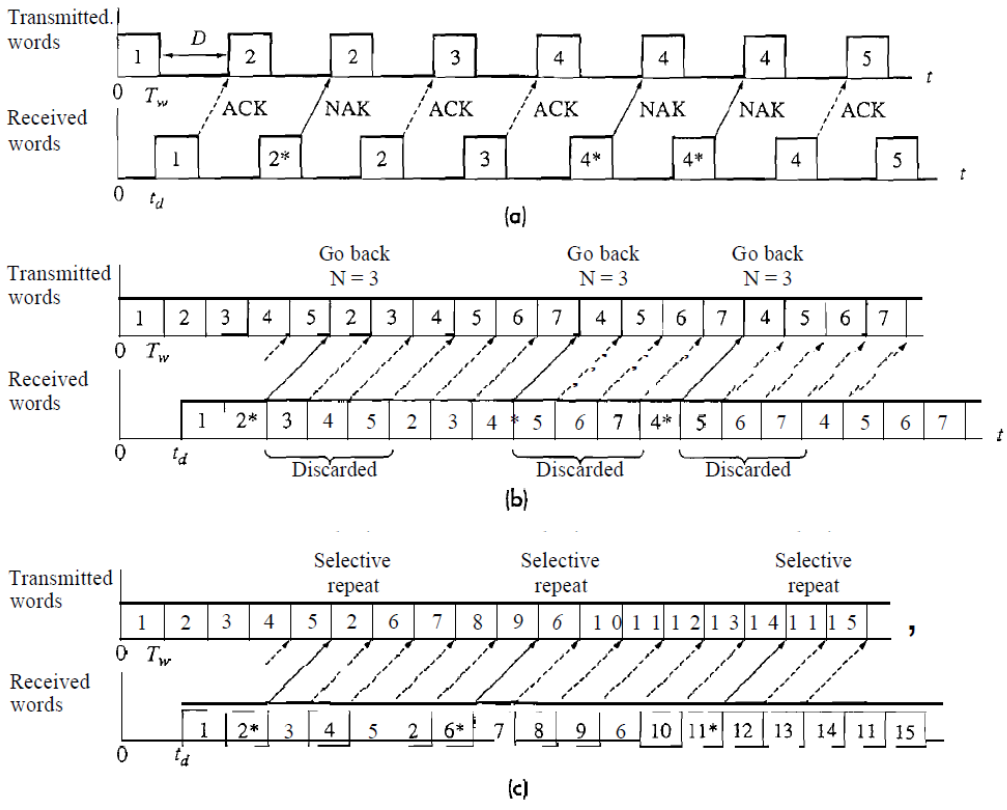
- Transmito y espero (stop-and-wait).
- Retrocede N (go-back-N).



Un sistema ARQ (solicitud de respuesta automática) solamente **detecta** errores en recepción, y en lugar de intentar corregirlos, solicita su reenvío. **Requiere canal bidireccional**.

Varias maneras de implementar:

- Transmito y espero (stop-and-wait).
- Retrocede N (go-back-N).
- Repetición selectiva (selective-repeat).



13.1-7 ARQ schemes. (a) Stop-and-wait; (b) go-back-N; (c) selective-repeat.

Communication Systems. Carlson, Crilly, Rutledge. 4ta ed.

Performance ARQ (1/2)

Un sistema ARQ que detecta hasta l errores opera como un sistema FEC con $t = l$, pero con una tasa de código efectiva aún menor debido a la retransmisión.

Performance ARQ (1/2)

Un sistema ARQ que detecta hasta l errores opera como un sistema FEC con $t = l$, pero con una tasa de código efectiva aún menor debido a la retransmisión.

Asumiendo un sistema de repetición selectiva sobre comunicación binaria óptima (solo se retransmiten las palabras con errores), la tasa efectiva de comunicación es:

$$R'_c = (1 - p)R_c = (1 - p)k/n$$

Donde p es la probabilidad de que ocurra algún error en cada palabra de código a la tasa **real** de comunicación. Si α es la BER del sistema, entonces $p \approx n.\alpha$ (asumiendo errores independientes).

Para un sistema ARQ que es capaz de detectar l errores en cada palabra de n bits, los errores "reales" suceden cuando tenemos $l+1$ o más errores en la misma palabra. Por el mismo argumento que en el caso FEC:

Performance ARQ

$$\text{BER} = \mathcal{C}_l^{n-1} \alpha^{l+1}$$

Ejemplo: performance sistema ARQ

Sea un sist. digital binario óptimo sin codificación de canal.
Aceptemos (sin prueba) que:

$$\text{BER} = Q(\sqrt{2\gamma_b})$$

Ejemplo: performance sistema ARQ

Sea un sist. digital binario óptimo sin codificación de canal. Aceptemos (sin prueba) que:

$$\text{BER} = Q(\sqrt{2\gamma_b})$$

Sea un sistema ARQ de repetición selectiva que utiliza un código (10,9) con $d_{min} = 2$. Determine γ_b necesario para obtener una BER de 10^{-5}. ¿Cuál es el ahorro energético que se logra respecto de un sistema con la misma tasa de error y tasa de bits efectiva sin utilizar corrección de errores?

Recordar: Si se mantiene la tasa de bits de mensaje (tasa "efectiva"), al usar codificación la tasa real de bits debe aumentarse en un factor R'_c, por lo que la probabilidad de error en un bit pasa a ser $\alpha' = Q(\sqrt{2R'_c\gamma_b})$

Ejemplo: performance sistema ARQ

Sea un sist. digital binario óptimo sin codificación de canal. Aceptemos (sin prueba) que:

$$\text{BER} = Q(\sqrt{2\gamma_b})$$

Sea un sistema ARQ de repetición selectiva que utiliza un código (10,9) con $d_{min} = 2$. Determine γ_b necesario para obtener una BER de 10^{-5}. ¿Cuál es el ahorro energético que se logra respecto de un sistema con la misma tasa de error y tasa de bits efectiva sin utilizar corrección de errores?

Recordar: Si se mantiene la tasa de bits de mensaje (tasa "efectiva"), al usar codificación la tasa real de bits debe aumentarse en un factor R'_c, por lo que la probabilidad de error en un bit pasa a ser $\alpha' = Q(\sqrt{2R'_c\gamma_b})$

Soluciones: 7.3 vs 9.6 dB, ver ejemplo 13.1-2 del libro

Errores en ráfaga e *interleaving*

En muchos sistemas reales, los errores no son independientes: ocurren en ráfagas (*bursts*) por descargas eléctricas, etc.

Ejemplo:
$0101010100\underline{1111}00100011110 \rightarrow 0101010100\underline{0000}00100011110$

Los errores en ráfaga son casi imposibles de corregir directamente!

Errores en ráfaga e *interleaving*

En muchos sistemas reales, los errores no son independientes: ocurren en ráfagas (*bursts*) por descargas eléctricas, etc.

Ejemplo:
$0101010100\underline{1111}00100011110 \rightarrow 0101010100\underline{0000}00100011110$

Los errores en ráfaga son casi imposibles de corregir directamente!

Solución: *interleaving* /intercalado
En lugar de enviar las palabras de código en forma secuencial, se intercalan bit a bit.

Errores en ráfaga e *interleaving*

En muchos sistemas reales, los errores no son independientes: ocurren en ráfagas (*bursts*) por descargas eléctricas, etc.

Ejemplo:
$0101010100\underline{1111}00100011110 \rightarrow 0101010100\underline{0000}00100011110$

Los errores en ráfaga son casi imposibles de corregir directamente!

Solución: *interleaving* /intercalado
En lugar de enviar las palabras de código en forma secuencial, se intercalan bit a bit.

Ejemplo: consideramos 4 palabras de 5 bits
Tx serie $\rightarrow a_1 a_2 a_3 a_4 a_5 b_1 b_2 \underline{b_3 b_4 b_5 c_1} c_2 c_3 c_4 c_5 d_1 d_2 d_3 d_4 d_5$

Tres errores en una sola palabra!

Errores en ráfaga e *interleaving*

En muchos sistemas reales, los errores no son independientes: ocurren en ráfagas (*bursts*) por descargas eléctricas, etc.

Ejemplo:
$0101010100\underline{1111}00100011110 \rightarrow 0101010100\underline{0000}00100011110$

Los errores en ráfaga son casi imposibles de corregir directamente!

Solución: *interleaving* /intercalado
En lugar de enviar las palabras de código en forma secuencial, se intercalan bit a bit.

Ejemplo: consideramos 4 palabras de 5 bits
Tx serie $\rightarrow a_1 a_2 a_3 a_4 a_5 b_1 b_2 \underline{b_3 b_4 b_5 c_1} c_2 c_3 c_4 c_5 d_1 d_2 d_3 d_4 d_5$
Tres errores en una sola palabra!

intercalado $\rightarrow a_1 b_1 c_1 d_1 a_2 b_2 c_2 \underline{d_2 a_3 b_3 c_3} d_3 a_4 b_4 c_4 d_4$
Un solo bit errado por palabra!

- Las arquitecturas ARQ y FEC se pueden utilizar conjuntamente.

- Las arquitecturas ARQ y FEC se pueden utilizar conjuntamente.
- De la misma manera, sistemas de corrección de errores se pueden usar de forma anidada.

- Las arquitecturas ARQ y FEC se pueden utilizar conjuntamente.
- De la misma manera, sistemas de corrección de errores se pueden usar de forma anidada.
- ARQ requiere comunicación bidireccional. Típicamente usado en redes de computadoras.

- Las arquitecturas ARQ y FEC se pueden utilizar conjuntamente.
- De la misma manera, sistemas de corrección de errores se pueden usar de forma anidada.
- ARQ requiere comunicación bidireccional. Típicamente usado en redes de computadoras.
- Existe un compromiso entre capacidad de la codificación/decodificación y la complejidad del procesamiento (mejores códigos son más difíciles de implementar).

Límites a la corrección de errores

¿Es posible diseñar códigos para eliminar los errores de comunicación en cualquier sistema de comunicación?

Límites a la corrección de errores

¿Es posible diseñar códigos para eliminar los errores de comunicación en cualquier sistema de comunicación?

NO, existen límites fundamentales sobre la cantidad de información (tasa de bits) que se puede transmitir sin errores sobre un canal con especificaciones físicas dadas (capacidad del canal).