Universidad del Valle de Guatemala
Facultad de Ingeniería
Redes

Esquemas de detección y corrección

Parte 01 y Parte 02

Carnet/Autores:
22473, Madeline Nahomy Castro Morales
22716, Aroldo Xavier López Osoy

Catedrático:
Juan Carlos Canteo

Sección 30

Fecha:
17/08/2025

Escenario de Pruebas Realizadas

Sin errores

Mensajes Utilizados:

1. 1011
2. 11010110
3. 1010101011110000

Respuesta del Emisor Hamming

```
PS C:\Users\ncast\OneDrive\Documentos\Universidad\Redes\Lab02-Redes\PT1\emisor> java -cp out app.Main
=== EMISOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (codificacion de errores)
  2) CRC-32 (deteccion de errores)
Selecciona algoritmo [1/2]: 1
Ingresa el mensaje en binario (solo 0/1): 1011
Hamming: especifica el tamano de bloque (n) del codigo (p.ej., 7, 12, 15): 4
Trama codificada Hamming: 1110000011101110
```

```
PS C:\Users\ncast\OneDrive\Documentos\Universidad\Redes\Lab02-Redes\PT1\emisor> java -cp out app.Main
=== EMISOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (codificacion de errores)
  2) CRC-32 (deteccion de errores)
Selecciona algoritmo [1/2]: 1
Ingresa el mensaje en binario (solo 0/1): 11010110
Hamming: especifica el tamano de bloque (n) del codigo (p.ej., 7, 12, 15): 8
Trama codificada Hamming: 1010101011001100
```

```
=== EMISOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (codificacion de errores)
  2) CRC-32 (deteccion de errores)
Selecciona algoritmo [1/2]: 1
Ingresa el mensaje en binario (solo 0/1): 1010101011110000
Hamming: especifica el tamano de bloque (n) del codigo (p.ej., 7, 12, 15): 12
Trama codificada Hamming: 111101001010111111100000
```

Respuesta del Receptor Hamming

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 1
Ingresa la trama en binario (solo 0/1): 1110000011101110
Hamming: especifica el tamaño de bloque (n) del código (p.ej., 7, 12, 15): 4
Resultado: No se detectaron errores.
Mensaje original (datos sin bits de paridad): 1011
```

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 1
Ingresa la trama en binario (solo 0/1): 1010101011001100
Hamming: especifica el tamaño de bloque (n) del código (p.ej., 7, 12, 15): 8
Resultado: No se detectaron errores.
Mensaje original (datos sin bits de paridad): 11010110
```

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 1
Ingresa la trama en binario (solo 0/1): 111101001010111111100000
Hamming: especifica el tamaño de bloque (n) del código (p.ej., 7, 12, 15): 12
Resultado: No se detectaron errores.
Mensaje original (datos sin bits de paridad): 1010101011110000
```

Respuesta del Emisor CRC32

```
=== EMISOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (codificacion de errores)
  2) CRC-32 (deteccion de errores)
Selecciona algoritmo [1/2]: 2
Ingresa el mensaje en binario (solo 0/1): 1011
Trama con CRC-32: 10110010101101001011110010110110001
```

```
=== EMISOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (codificacion de errores)
  2) CRC-32 (deteccion de errores)
Selecciona algoritmo [1/2]: 2
Ingresa el mensaje en binario (solo 0/1): 11010110
Trama con CRC-32: 11010110000010110001110100000110010110 11
```

```
=== EMISOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (codificacion de errores)
  2) CRC-32 (deteccion de errores)
Selecciona algoritmo [1/2]: 2
Ingresa el mensaje en binario (solo 0/1): 1010101011110000
Trama con CRC-32: 10101010111100000000000100101101110011101110 1010
```

Respuesta del Receptor CRC32

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 2
Ingresa la trama en binario (solo 0/1): 10110010101101001011110010110110001
Resultado: No se detectaron errores (CRC correcto).
Mensaje original (sin los 32 bits de CRC): 1011
```

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 2
Ingresa la trama en binario (solo 0/1): 11010110000010110001110100000110010110 11
Resultado: No se detectaron errores (CRC correcto).
Mensaje original (sin los 32 bits de CRC): 11010110
```

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 2
Ingresa la trama en binario (solo 0/1): 10101010111100000000000100101101110011101110 1010
Resultado: No se detectaron errores (CRC correcto).
Mensaje original (sin los 32 bits de CRC): 1010101011110000
```

Con 1 error

Mensajes Utilizados:

1. 1011
2. 11010110
3. 1010101011110000

Cambio de 1 bits Hamming (Mismo recibido que sin errores):

a. Bit cambiado: (n = 4)

1110 0000 1110 1110

1110 0000 1010 1110

b. Bit cambiado: (n = 8)

10101010 11001100

10101010 11101100

c. Bit cambiado: (n = 12)

111101001010 111111100000

111101001010 111101100000

Cambio de 1 bits CRC32 (Mismo recibido que sin errores):

a. Bit cambiado:

10110010101101001011110010110110000 1

10110010101101001011111010110110000 1

b. Bit cambiado:

11010110000010110001110100001100101101 1

11010110000010110001110100100110010110 11

c. Bit cambiado:

10101010111100000000001001011011100111011101010

10101010111100000000010100101101110011101110101 0

Respuesta del Receptor Hamming (Corrige errores)

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 1
Ingresa la trama en binario (solo 0/1): 1110000010101110
Hamming: especifica el tamaño de bloque (n) del código (p.ej., 7, 12, 15): 4
Resultado: Se detectaron y corrigieron errores.
Posiciones corregidas (posicion dentro de cada bloque de longitud n, base 1):
  Bloque 3, bit 2
Mensaje corregido (datos sin bits de paridad): 1011
```

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 1
Ingresa la trama en binario (solo 0/1): 1010101011101100
Hamming: especifica el tamaño de bloque (n) del código (p.ej., 7, 12, 15): 8
Resultado: Se detectaron y corrigieron errores.
Posiciones corregidas (posición dentro de cada bloque de longitud n, base 1):
  Bloque 2, bit 3
Mensaje corregido (datos sin bits de paridad): 11010110
```

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 1
Ingresa la trama en binario (solo 0/1): 111101001010111101100000
Hamming: especifica el tamaño de bloque (n) del código (p.ej., 7, 12, 15): 12
Resultado: Se detectaron y corrigieron errores.
Posiciones corregidas (posición dentro de cada bloque de longitud n, base 1):
  Bloque 2, bit 5
Mensaje corregido (datos sin bits de paridad): 1010101011110000
```

Respuesta del Receptor CRC32 (Solo los detecta)

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 2
Ingresa la trama en binario (solo 0/1): 10110010101101001011111010110110000 1
Resultado: Se detectaron errores: trama descartada (CRC inválido).
```

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 2
Ingresa la trama en binario (solo 0/1): 11010110000010110001110100100110010110 11
Resultado: Se detectaron errores: trama descartada (CRC inválido).
```

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 2
Ingresa la trama en binario (solo 0/1): 10101010111100000000010100101101110011101110101 0
Resultado: Se detectaron errores: trama descartada (CRC inválido).
```

Con 2 o más bits de error

Mensajes Utilizados:

1. 1011
2. 11010110
3. 1010101011110000

Cambio de 2 bits Hamming (Mismo recibido que sin errores):

d. Bit cambiado: (n = 4)

1110 0000 1110 1110

1100 0100 1110 1100

e. Bit cambiado: (n = 8)

10101010 11001100

10111010 10001100

f. Bit cambiado: (n = 12)

111101001010 111111100000

111100001010 111111000000

Cambio de 2 bits CRC32 (Mismo recibido que sin errores):

d. Bit cambiado:

10110010101101001011110010110110000 1

10110010101001001011110110110100000 1

e. Bit cambiado:

11010110000010110001110100000110010110 11

11010110001010110001110100100110010110 11

f. Bit cambiado:

10101010111100000000000100101101110011101110 1010

10101010111100000100001100101111110010 1011101010

Respuesta del Receptor Hamming (Corrige errores)

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 1
Ingresa la trama en binario (solo 0/1): 1100010011101100
Hamming: especifica el tamaño de bloque (n) del código (p.ej., 7, 12, 15): 4
Resultado: Se detectaron y corrigieron errores.
Posiciones corregidas (posición dentro de cada bloque de longitud n, base 1):
  Bloque 1, bit 3
  Bloque 2, bit 2
  Bloque 4, bit 3
Mensaje corregido (datos sin bits de paridad): 1011
```

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 1
Ingresa la trama en binario (solo 0/1): 1011101010001100
Hamming: especifica el tamaño de bloque (n) del código (p.ej., 7, 12, 15): 8
Resultado: Se detectaron y corrigieron errores.
Posiciones corregidas (posición dentro de cada bloque de longitud n, base 1):
  Bloque 1, bit 4
  Bloque 2, bit 2
Mensaje corregido (datos sin bits de paridad): 11010110
```

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 1
Ingresa la trama en binario (solo 0/1): 111100001010111111000000
Hamming: especifica el tamaño de bloque (n) del código (p.ej., 7, 12, 15): 12
Resultado: Se detectaron y corrigieron errores.
Posiciones corregidas (posición dentro de cada bloque de longitud n, base 1):
  Bloque 1, bit 6
  Bloque 2, bit 7
Mensaje corregido (datos sin bits de paridad): 1010101011110000
```

Respuesta del Receptor CRC32 (Solo los detecta)

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 2
Ingresa la trama en binario (solo 0/1): 10110010101001001011110110110100000l
Resultado: Se detectaron errores: trama descartada (CRC inválido).
```

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 2
Ingresa la trama en binario (solo 0/1): 11010110001010110001110100100110010110 11
Resultado: Se detectaron errores: trama descartada (CRC inválido).
```

```
=== RECEPTOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (corrección de errores)
  2) CRC-32 (detección de errores)
Selecciona algoritmo [1/2]: 2
Ingresa la trama en binario (solo 0/1): 10101010111100000100001100101111110010101110 1010
Resultado: Se detectaron errores: trama descartada (CRC inválido).
```

Preguntas Realizadas

A. ¿Es posible manipular los bits de tal forma que el algoritmo seleccionado no sea capaz de detectar el error? ¿Por qué sí o por qué no? En caso afirmativo, demuestrelo con su implementación.

a. Hamming

Si existen escenarios en los cuales el código de Hamming no puede detectar ciertos errores, esto se da particularmente cuando dos errores existen dentro del mismo bloque. En estos casos, el sistema no detecta que hay un problema y peor, corregirlo de forma incorrecta (cambiando un bit que no era el erróneo). Esto produce un mensaje equivocado sin ningún tipo de advertencia. A este fallo se le conoce como **miscorrección silenciosa** y es una de las principales limitaciones de Hamming.

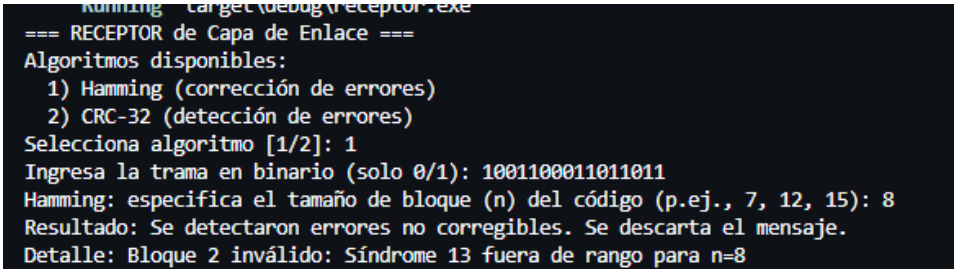
i. Demostración:

En el emisor se codificó este mensaje:

```
=== EMISOR de Capa de Enlace ===
Algoritmos disponibles:
  1) Hamming (codificacion de errores)
  2) CRC-32 (deteccion de errores)
Selecciona algoritmo [1/2]: 1
Ingresa el mensaje en binario (solo 0/1): 01000001
Hamming: especifica el tamano de bloque (n) del codigo (p.ej., 7, 12, 15): 8
Trama codificada Hamming: 1001100011010010
```

Se obtuvo este mensaje: 10011000 11010010 y se modificaron 2 bits en el mismo bloque 10011000 11011011

Y el receptor obtuvo este resultado:



Donde el receptor detectó un síndrome fuera del rango válido (13 para n=8), lo que indica la presencia de múltiples errores. Como era de esperarse, el algoritmo no pudo corregir el mensaje, confirmando que Hamming puede detectar errores múltiples pero solo corregir uno. Esta situación representa un riesgo potencial de miscorrección o descarte cuando ocurren errores severos.

b. CRC32

Es casi imposible que un error pase desapercibido usando CRC32, a menos que el patrón del error coincida exactamente con una combinación específica (muy improbable) que produzca el mismo residuo. Por eso, se considera un algoritmo muy seguro y confiable para detección, aunque no puede corregir errores como Hamming.

B. En base a las pruebas que realizó, ¿qué ventajas y desventajas posee cada algoritmo con respecto a los otros dos? Tome en cuenta complejidad, velocidad, redundancia (overhead), etc.

a. Hamming

i. Ventajas:

1. Puede detectar y corregir un solo error por bloque, lo cual permite mantener una alta tasa de entrega incluso en canales con errores moderados. Esto lo hace particularmente útil cuando no hay posibilidad de retransmisión.
2. La operación de verificación y localización del error es muy rápida, ya que se basa en el cálculo del síndrome mediante XORs simples.

3. Al poder ajustarse el tamaño del bloque n (por ejemplo, 7 o 15 bits), se puede adaptar la relación entre overhead y robustez a errores según las condiciones del canal.

ii. Desventajas:

1. Para lograr la corrección, se deben añadir múltiples bits de paridad. Por ejemplo, en Hamming(7,4) el overhead es del 75%.
2. Si ocurren dos o más errores en el mismo bloque, no solo no se corrigen, sino que pueden interpretarse como válidos, corrompiendo silenciosamente el mensaje.

b. CRC32

i. Ventajas:

1. CRC32 es capaz de detectar con una probabilidad extremadamente alta todos los errores de 1, 2, 3 bits, muchos de hasta 4 bits y casi todos los errores con burst menores a 32 bits. Tiene una tasa de colisión bajísima para datos reales.

2. A pesar de ser de 32 bits, el CRC representa un overhead constante, lo que lo hace atractivo para mensajes largos (por ejemplo, en 64 bits el overhead es solo del 50%).

ii. Desventajas:

1. En ausencia de retransmisión (ARQ), descarta el mensaje si se detecta cualquier error, lo cual reduce drásticamente la tasa de entrega a medida que crece la BER o el tamaño del mensaje.
2. A mayor longitud de la trama (mensaje + CRC), mayor probabilidad de que al menos un bit se dañe y todo el paquete sea descartado.

Descripción de la práctica y metodología utilizada

Se implementó una comunicación emisor–receptor sobre sockets TCP para evaluar esquemas de detección y corrección de errores en una arquitectura por capas.

a. **Emisor:** Java.
b. **Receptor:** Rust (modo servidor).
c. **Algoritmos:**
    i. **Corrección:** Hamming SEC (Single-Error Correction) con n = 7 y n = 15.
    ii. **Detección:** CRC-32 “puro” (polinomio 0x04C11DB7, sin reflect/init/xor).
    iii. **Canal:** se modeló como BSC (Binary Symmetric Channel) aplicando ruido en el emisor: cada bit se voltea con probabilidad p=BER.
d. **Arquitectura por capas implementada**
    i. **Aplicación (emisor):** solicita el texto y el algoritmo.
    ii. **Aplicación (receptor):** muestra el mensaje recibido o reporta error/no-corregible.
    iii. **Presentación:** ASCII de 8 bits por carácter (codificación/decodificación).
    iv. **Enlace (emisor):**
        1. **Hamming SEC:** codifica los bits de datos en bloques de longitud n y envía además el pad de ceros usado para completar el último bloque.
        2. **CRC-32 puro:** concatena el remanente de 32 bits al final del mensaje.
        3. **Enlace (receptor):** verifica integridad (CRC) o detecta/corrige (Hamming) y entrega a Presentación solo los bits de datos (removiendo el pad).

v. **Ruido (emisor):** bit-flip independiente con probabilidad BER (incluye bits de paridad/CRC).

vi. **Transmisión:** TCP cliente (emisor) → servidor (receptor). Protocolo textual de tres líneas:

```
ALGO=<CRC32|HAMMING>
PARAM=<clave=valor;...>
BITS=<trama_en_binario>
```

e. Metodología experimental

i. Parrilla de parámetros

1. **BER:** {0.0, 0.01, 0.02}
2. **Tamaño de mensaje (bytes):** {1, 8} → 8 o 64 bits de datos.
3. **Hamming (n):** {7, 15}
4. **Algoritmo:** {HAMMING, CRC32}

f. Procedimiento automatizado

i. Se desarrolló un script Python que:

1. Arranca el receptor Rust (servidor TCP en 0.0.0.0:9000).
2. Para cada combinación de la parrilla se realizó en su ejecución:
    a. Genera un mensaje aleatorio de M bytes (A–Z).
    b. Invoca un emisor headless (clase Java app.Bench) con argumentos: texto, algoritmo, parámetros (p. ej., n), BER, host y puerto.
    c. El emisor codifica (Hamming/CRC), aplica ruido y envía la trama.

d. El script lee y parsea la salida del receptor (líneas "Hamming: … Mensaje: …" o "CRC válido. Mensaje: …" o "descartado/no corregibles").
e. Registra una fila CSV con: mensaje_original, mensaje_recibido, valido, errores_corregidos, algoritmo, ber.

ii. **Genera un CSV por combinación con nombre auto-descriptivo (p. ej., resultados_algo=Hamming_n=7_msg=8B_ber=0.01.csv).**

g. **Métricas y cálculo**

i. **Tasa de entrega:** #válidos / #pruebas por CSV.

ii. **Tasa de corrección (Hamming):** fracción de pruebas donde el receptor reportó "errores corregidos".

iii. **Goodput:** bits útiles entregados / bits transmitidos.

1. **Para CRC-32:** bits transmitidos = $M \cdot 8+32$
2. **Para Hamming(n):** bits transmitidos = $\lceil M \cdot 8/m \rceil \cdot n$, con $m=n-r$ y r paridades (SEC).
3. **Los gráficos se generaron con Python (pandas + matplotlib) agregando promedios por BER, algoritmo y tamaño de mensaje.**

h. **Validaciones**

i. **BER=0:** goodput coincide con teoría (p. ej., Hamming(7,4) = 4/7, Hamming(15,11) = 11/15, CRC-32 = $M \cdot 8/(M \cdot 8+32)$).

ii. **La ejecución se realizó en entorno Linux/WSL; receptor Rust compilado con cargo build, emisor Java con *javac -d out …* .**

Resultados

Canal BSC (bit-flip con probabilidad p= BER). Para cada combinación de algoritmo, tamaño de mensaje y BER. En Hamming se evaluaron n=7 y n=15 (SEC). En CRC-32 se usó la variante pura (polinomio 0x04C11DB7).

a. **Métricas:**

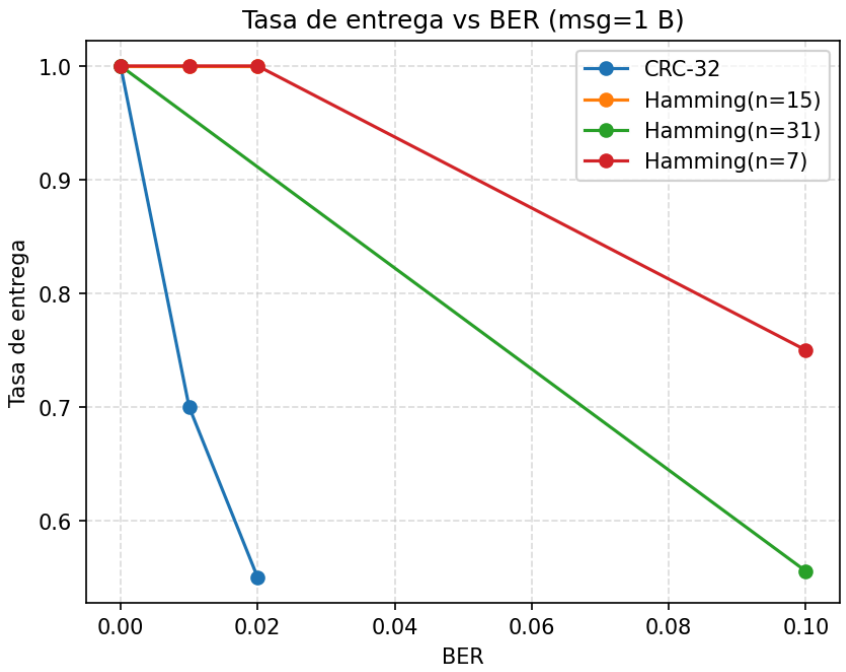
   i. **Tasa de entrega:** fracción de pruebas en las que el receptor devolvió el mensaje correcto.

   ii. **Tasa de corrección (Hamming):** fracción de pruebas en las que el receptor reportó "errores corregidos".

   iii. **Goodput:** bits útiles entregados / bits transmitidos.

b. **Hallazgos:**

   i. **Tasa de Entrega vs. BER msg1B:**

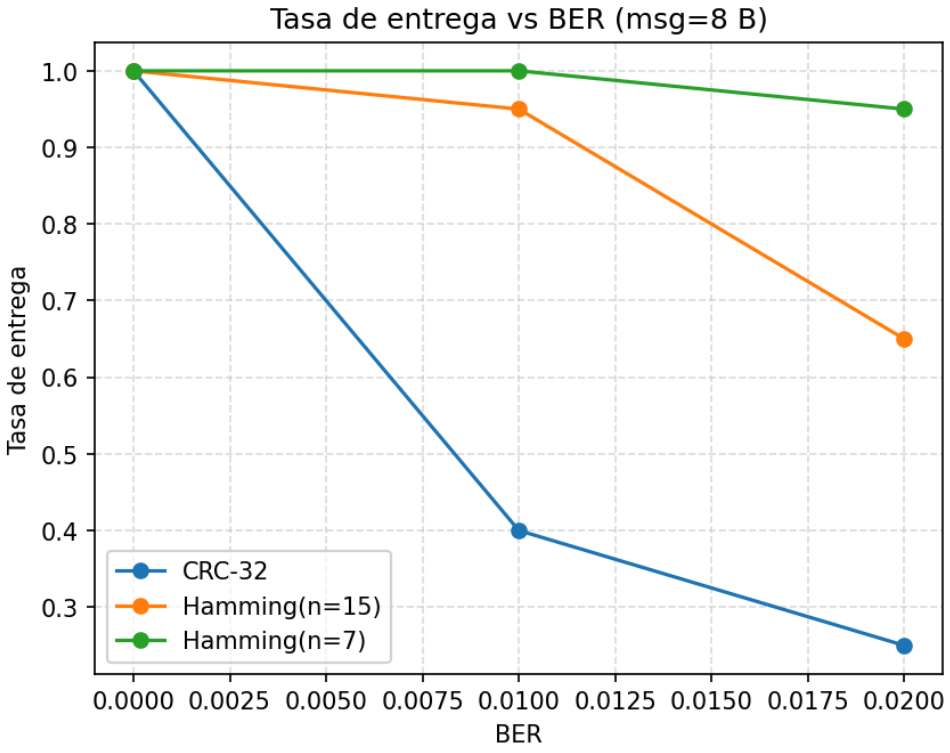


CRC-32 cae de forma marcada al subir la BER: con 1 B (40 bits totales) basta un solo error para descartar.

Hamming(n=7) y Hamming(n=15): mantienen altas tasas de entrega hasta BER=0.02; con 1 B hay pocos bloques y el evento "≥2 errores en el mismo bloque" es raro, por lo que SEC funciona bien.

Entre ambos Hamming, las diferencias son pequeñas para 1 B: con un único bloque, n=15 suele aguantar tanto como n=7 en este rango de BER.

ii. **Tasa de Entrega vs. BER msg8B:**

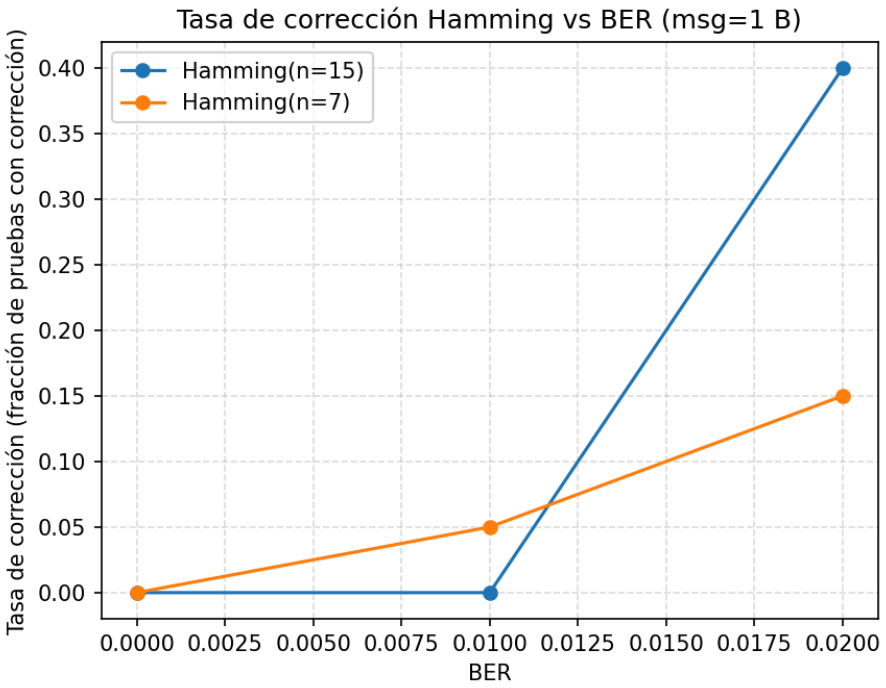


CRC-32: se degrada con fuerza al pasar de BER=0.01 a 0.02 (96 bits totales a validar → mayor prob. de al menos 1 error).

Hamming(n=7) y Hamming(n=15): conservan tasas de entrega altas a BER=0.01 y razonables a 0.02.

Para BER=0.02, n=7 suele entregar algo más que n=15, porque sus bloques son más cortos y la probabilidad de "≥2 errores en el mismo bloque" (no corregible por SEC) es menor.

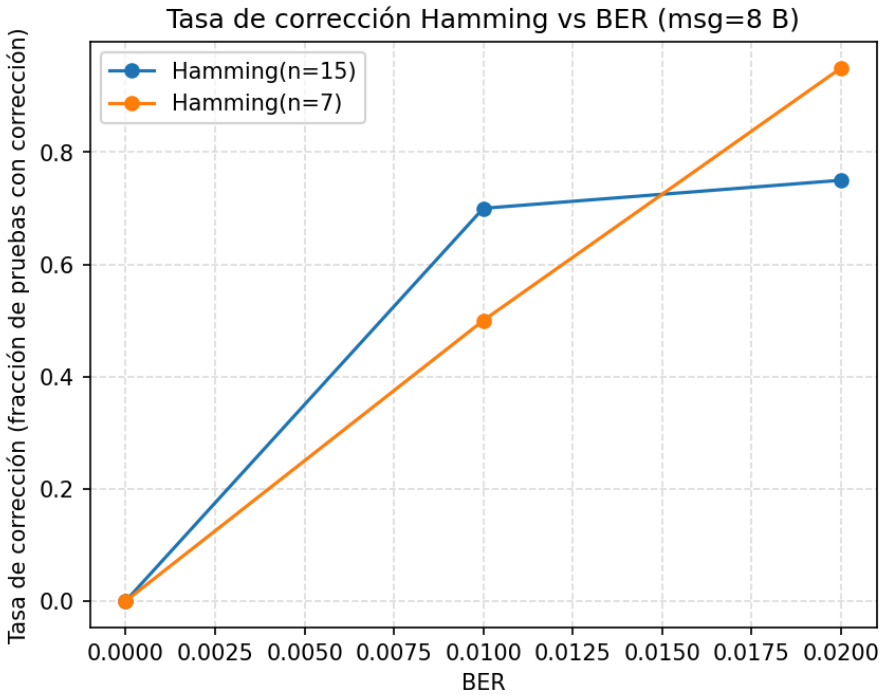
iii. Tasa de Corrección Hamming vs. BER msg1B:



La tasa de corrección aumenta con la BER: aparecen más casos de "exactamente 1 error por bloque".

Con 1 B: las tasas de corrección siguen siendo moderadas (hasta ~20–30% con 0.02), coherente con que hay pocos bits en juego.

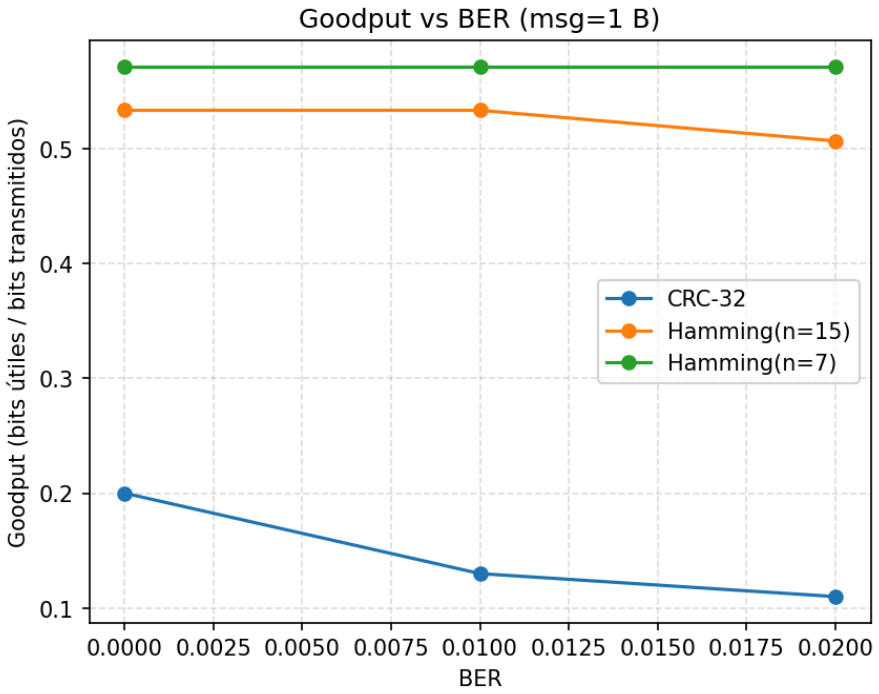
iv. Tasa de Corrección Hamming vs. BER msg8B:



La corrección es muy frecuente incluso a BER moderadas, porque el mensaje se parte en varios bloques; basta que uno tenga 1 error para que la corrida contabilice “hubo corrección”.

n=7 suele mostrar algo más de “hubo corrección” que n=15 simplemente porque hay más bloques por mensaje (m=4 vs m=11 datos por bloque).

v. **Goodput vs. BER msg1B:**

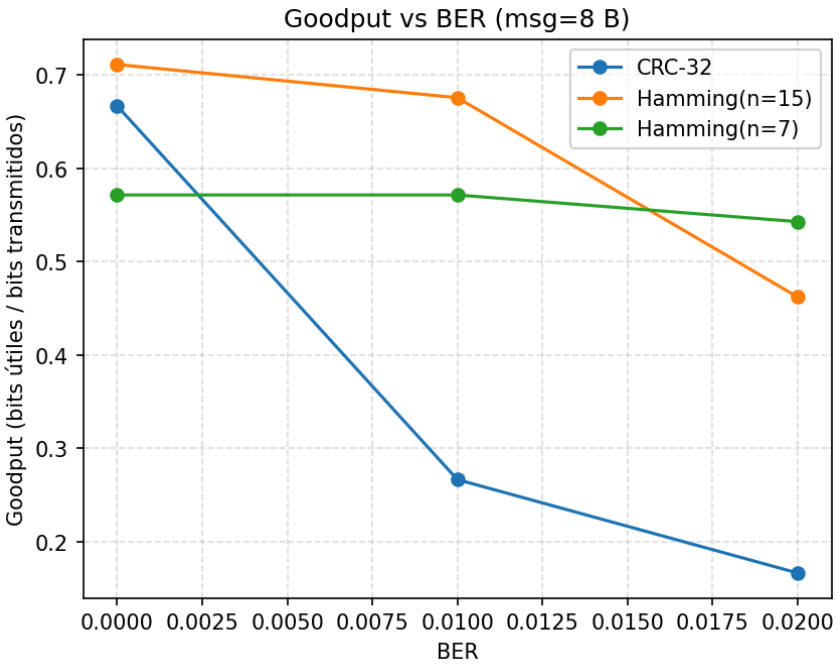


En BER=0 los valores coinciden con la teoría:

- CRC-32: 8/40=0.20
- Hamming(7,4): 8/14≈0.571
- Hamming(15,11): 8/15≈0.533

Con BER>0, el goodput cae según la tasa de entrega. En este escenario, Hamming(n=7) mantiene mejor goodput que CRC-32.

vi. Goodput vs. BER msg8B:



En BER=0:

- CRC-32: 64/96≈0.667
- Hamming(15,11): 64/90≈0.711 (mayor que CRC)
- Hamming(7,4): 64/112≈0.571

Con BER>0, CRC-32 se hunde por descartes; Hamming(n=15) mantiene el mejor compromiso entre entrega y overhead, por eso su goodput es el más alto en 8 B.

Discusión

¿Qué algoritmo tuvo mejor funcionamiento y por qué?

Depende del régimen:

a. **BER baja y mensajes medianos (8 B):** Hamming(n=15) logra el mejor goodput (menor overhead relativo que n=7 y mucha mayor entrega que CRC-32 cuando aparece algún error).
b. **BER moderada (0.01–0.02):** Hamming supera claramente a CRC-32 en tasa de entrega y por tanto en goodput, porque corrige 1 error por bloque en vez de descartar toda la trama.
c. **Entre n=7 y n=15, a BER más alta n=7:** suele entregar más (bloques cortos → menos casos de ≥2 errores no corregibles); a BER baja, n=15 aprovecha mejor el ancho de banda (menos redundancia).

¿Qué algoritmo es más flexible para aceptar mayores tasas de error y por qué?

Hamming (SEC) es más tolerante que CRC-32 en BER moderadas porque corrige errores de 1 bit. La longitud de bloque es clave: n pequeño (7) tolera mejor BER más altas; n intermedio (15) equilibra overhead y robustez cuando la BER es baja–media. CRC-32 tiene detección excelente (colisiones despreciables), pero sin ARQ su tasa de entrega cae rápido con el tamaño y la BER.

¿Cuándo conviene detección (CRC-32) en lugar de corrección (Hamming) y por qué?

a. Cuando hay ARQ (retransmisión) y la BER es baja → CRC-32 + ARQ minimiza redundancia y asegura integridad fuerte.
b. Cuando el retardo tolera reintentos y el canal ya es confiable (p. ej., cableado).

¿Cuándo conviene la corrección (Hamming-SEC)?

a. Cuando la retransmisión es costosa o imposible (latencias altas, unidireccional, enlaces con pérdida).
b. Cuando la BER es moderada y los errores son aislados → SEC corrige la mayoría.
c. Ajustando n: bloques cortos (n=7) para BER más alta; n=15 si se busca mejor eficiencia en BER baja–media.

Conclusiones

Los resultados muestran que:

a. **CRC-32 (puro):** tiene excelente poder de detección, pero su tasa de entrega cae rápido al aumentar la BER y/o la longitud del mensaje, porque cualquier bit errado invalida toda la trama (lo que es esperado para un detector diseñado con un polinomio de alta distancia mínima; v. g. el de IEEE 802.3).
b. **Hamming:** ofrece una clara ventaja de entrega y goodput en el rango de BER evaluado (0.01–0.02), ya que corrige 1 error por bloque.
    i. **Con 1 B, tanto n=7 como n=15:** mantienen entregas altas en 0.01–0.02; hay pocos bloques y es raro observar ≥2 errores en el mismo bloque.
    ii. **Con 8 B, n=7 tiende a entregar más que n=15 en BER más alta,** porque los bloques más cortos reducen la probabilidad de errores múltiples no corregibles dentro de un mismo bloque.
    iii. **En BER baja, n=15:** logra mejor goodput que n=7 gracias a su menor overhead (11/15 frente a 4/7).
    iv. **En goodput:**
        1. **Con BER = 0, mandan las tasas m/n:** Hamming(15,11) > CRC-32 (para 8 B) > Hamming(7,4) en nuestras pruebas.
        2. **Con BER > 0, manda la entrega:** Hamming mantiene el goodput mientras que CRC-32 cae por descartes.

c. **Recomendación por régimen:**
    i. Si hay ARQ y la BER es baja, CRC-32 + retransmisión es preferible (mínima redundancia, detección muy fuerte).
    ii. Si la retransmisión es costosa o la BER es moderada, Hamming (SEC) es mejor elección: n=15 para eficiencia en BER baja–media; n=7 cuando la BER crece.

Anexos

Repositorio de GitHub: **https://github.com/XavierLopez25/Lab02-Redes.git**

Citas y Referencias

Peterson, W. W., & Brown, D. T. (1961). Cyclic codes for error detection. *Proceedings of the IRE, 49*(1), 228–235. https://doi.org/10.1109/JRPROC.1961.287814

IEEE. (2018). *IEEE Standard for Ethernet (IEEE Std 802.3-2018)*. IEEE.

Koopman, P. (2002). 32-bit cyclic redundancy codes for Internet applications. In *Proceedings of the 2002 International Conference on Dependable Systems and Networks* (pp. 459–468). IEEE. https://doi.org/10.1109/DSN.2002.1028931

Koopman, P., & Chakravarty, T. (2004). Cyclic redundancy code (CRC) polynomial selection for embedded networks. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks* (pp. 145–154). IEEE. https://doi.org/10.1109/DSN.2004.1311885