

Universidad del Valle de Guatemala

Facultad de ingeniería

Redes

Catedrático: Carlos Boix



Laboratorio 2 - Esquemas de detección y corrección

Christian Echeverría 221441

Gabriel Alberto Paz González 221088

Primera parte:

CRC32

Mensaje Correcto:

Emisor

```
=====
  EMISOR CRC-32 (IEEE802.3)
=====
Parametrizacion: poly=0x04C11DB7, init=0xFFFFFFFF, xorout=0xFFFFFFFF, reflejado=No
Entrada esperada: cadena binaria (solo '0' y '1').

Ingrese mensaje binario (solo 0/1): 1010101001100110

-----
✅ CRC32 generado correctamente
Longitud mensaje: 16 bits
CRC32 (binario): 1100 0001 1111 1110 0010 0011 0100 0100
CRC32 (hex):      0xC1FE2344
Trama (msg||CRC) [copiar/pegar]: 101010100110011011000001111111100010001101000100
Overhead CRC:     32 bits (66.7% de la trama)
-----
Sugerencia: copie la línea "Trama " y péguela en el RECEPTOR.
```

Receptor

```
=====
  RECEPTOR CRC-32 (IEEE802.3)
=====
Parametrizacion: poly=0x04C11DB7, init=0xFFFFFFFF, xorout=0xFFFFFFFF, reflejado=No

Ingrese trama (mensaje||CRC32, 32 bits al final): 101010100110011011000001111111100010001101000100

-----
Longitud mensaje: 16 bits
CRC recibido      : 1100 0001 1111 1110 0010 0011 0100 0100
CRC calculado     : 1100 0001 1111 1110 0010 0011 0100 0100

✅ Resultado: No se detectaron errores.
Mensaje original: 10101010 01100110
Nota: CRC32 detecta todos los errores de 1 bit y todas las rafagas de hasta 32 bits; no corrige errores.
```

Nota: Si que detecta que ningun bit fue cambiado.

Mensaje con bit cambiado:

Emisor

101010100110011011000001111111100010001101000101 (ultimo bit cambiado)

Receptor

```
RECEPTOR CRC-32 (IEEE802.3)
Parametrizacion: poly=0x04C11DB7, init=0xFFFFFFFF, xorout=0xFFFFFFFF, reflejado=No
Ingrese trama (mensaje||CRC32, 32 bits al final): 101010100110011011000001111111100010001101000101

Longitud mensaje: 16 bits
CRC recibido   : 1100 0001 1111 1110 0010 0011 0100 0101
CRC calculado  : 1100 0001 1111 1110 0010 0011 0100 0100

✗ Resultado: Se detectaron errores. La verificacion no coincide.
Distancia de Hamming (CRC): 1 bit(s) diferentes
Accion recomendada: descartar la trama.
Recordatorio: CRC32 es un algoritmo de deteccion (no corrige). Una coincidencia falsa es extremadamente improbable para cambios aleatorios.
```

Nota: el bit si fue detectado

Mensaje con 2 bits cambiados:

Emisor

101010100110011011000001111111100010001101000111 (último y penúltimos bits cambiados)

Receptor

```
RECEPTOR CRC-32 (IEEE802.3)
Parametrizacion: poly=0x04C11DB7, init=0xFFFFFFFF, xorout=0xFFFFFFFF, reflejado=No
Ingrese trama (mensaje||CRC32, 32 bits al final): 101010100110011011000001111111100010001101000111

Longitud mensaje: 16 bits
CRC recibido   : 1100 0001 1111 1110 0010 0011 0100 0111
CRC calculado  : 1100 0001 1111 1110 0010 0011 0100 0100

✗ Resultado: Se detectaron errores. La verificacion no coincide.
Distancia de Hamming (CRC): 2 bit(s) diferentes
Accion recomendada: descartar la trama.
Recordatorio: CRC32 es un algoritmo de deteccion (no corrige). Una coincidencia falsa es extremadamente improbable para cambios aleatorios.
```

Nota: si detecto los dos bits cambiados

Hamming

Mensaje correcto

Emisor

```
=====
  EMISOR HAMMING (SEC)
=====
Codigo Hamming (SEC: Single-Error-Correcting), paridad par.
Corrige 1 bit y detecta 2 bits de error sin correccion.
Entrada esperada: cadena binaria (solo '0' y '1').

Ingrese mensaje binario (solo 0/1): 1010101

-----
✅ Trama Hamming generada correctamente
Longitud mensaje (m): 7 bits
Bits de paridad (r): 4 (posiciones 1, 2, 4, 8)
Longitud total (n): 11 bits
Overhead Hamming: 4 bits (36.4% de la trama)
Trama (copiar/pegar): 11110100101
-----
Sugerencia: copie la linea "Trama " y péguela en el RECEPTOR.
```

Receptor

```
=====
  RECEPTOR HAMMING (SEC)
=====
Codigo Hamming (SEC: corrige 1 bit, detecta 2 sin correccion). Paridad par.

Ingrese trama Hamming (emisor→receptor, sin espacios): 11110100101

-----
Longitud total (n): 11 bits   Paridades (r): 4   Mensaje (m): 7 bits
Trama (agrupada 8b): 11110100 101

✅ Resultado: No se detectaron errores.
Mensaje original: 1010101
Nota: Hamming (SEC) corrige 1 bit, detecta ≥2 (sin correccion).
```

Mensaje con un bit cambiado:

Emisor

11110100100 (ultimo bit cambiado)

Receptor

```
=====
RECEPTOR HAMMING (SEC)
=====
Codigo Hamming (SEC: corrige 1 bit, detecta 2 sin correccion). Paridad par.

Ingrese trama Hamming (emisor→receptor, sin espacios): 11110100100

-----
Longitud total (n): 11 bits   Paridades (r): 4   Mensaje (m): 7 bits
Trama (agrupada 8b): 11110100 100

❌ Verificacion fallida.
Sindrome (bin): 1011 Posicion indicada: 11 (1-indexada)
✅ Correccion aplicada: se invirtio el bit en la posicion 11 (dato).
Trama corregida: 11110100 101
Mensaje corregido: 1010101
Distancia de Hamming respecto a la trama recibida: 1 bit.
=====
```

Mensaje con dos bits cambiados

Emisor

11110100110 (ultimo y penultimo bits cambiados)

Receptor

```
=====
RECEPTOR HAMMING (SEC)
=====
Codigo Hamming (SEC: corrige 1 bit, detecta 2 sin correccion). Paridad par.

Ingrese trama Hamming (emisor→receptor, sin espacios): 11110100110

-----
Longitud total (n): 11 bits   Paridades (r): 4   Mensaje (m): 7 bits
Trama (agrupada 8b): 11110100 110

❌ Verificacion fallida.
Sindrome (bin): 0001 Posicion indicada: 1 (1-indexada)
✅ Correccion aplicada: se invirtio el bit en la posicion 1 (paridad).
Trama corregida: 01110100 110
Mensaje corregido: 1010110
Distancia de Hamming respecto a la trama recibida: 1 bit.
=====
```

¿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.

Hamming:

Sí. Mostramos un caso indetectable con flip en posiciones 1, 2 y 3 de la trama (síndrome 0), lo que cambia el mensaje sin que el receptor lo note. Esto se debe a que el Hamming SEC tiene $d=3$, y existen ternas de columnas cuya XOR es 0.

Mensaje original:

11110100101

Mensaje con las 3 primeras posiciones flipadas:

00010100101

Receptor

```
=====
RECEPTOR HAMMING (SEC)
=====
Codigo Hamming (SEC: corrige 1 bit, detecta 2 sin correccion). Paridad par.

Ingrese trama Hamming (emisor→receptor, sin espacios): 00010100101

=====
Longitud total (n): 11 bits   Paridades (r): 4   Mensaje (m): 7 bits
Trama (agrupada 8b): 00010100 101

✅ Resultado: No se detectaron errores.
Mensaje original: 0010101
Nota: Hamming (SEC) corrige 1 bit, detecta ≥2 (sin correccion).
=====
```

Si se puede truquear

CRC32:

En canal aleatorio, la probabilidad de no detección es $\sim 1/2^{32}$ para errores fuera de sus garantías; en la práctica, no lo observamos. Sin embargo, un atacante que recalcula el CRC tras modificar el payload pasa: CRC no es autenticación. Lo demostramos generando una trama válida para otro mensaje con el emisor.

```

=====
  EMISOR CRC-32 (IEEE802.3)
=====
Parametrizacion: poly=0x04C11DB7, init=0xFFFFFFFF, xorout=0xFFFFFFFF, reflejado=No
Entrada esperada: cadena binaria (solo '0' y '1').

Ingrese mensaje binario (solo 0/1): 1010101001100110

-----
✅ CRC32 generado correctamente
Longitud mensaje: 16 bits
CRC32 (binario): 1100 0001 1111 1110 0010 0011 0100 0100
CRC32 (hex):      0xC1FE2344
Trama (msg||CRC) [copiar/pegar]: 1010101001100110110000001111111100010001101000100
Overhead CRC:     32 bits (66.7% de la trama)
-----
Sugerencia: copie la línea "Trama " y péguela en el RECEPTOR.

```

```

=====
  RECEPTOR CRC-32 (IEEE802.3)
=====
Parametrizacion: poly=0x04C11DB7, init=0xFFFFFFFF, xorout=0xFFFFFFFF, reflejado=No

Ingrese trama (mensaje||CRC32, 32 bits al final): 1010101001100110110000001111111100010001101000100

-----
Longitud mensaje: 16 bits
CRC recibido   : 1100 0001 1111 1110 0010 0011 0100 0100
CRC calculado  : 1100 0001 1111 1110 0010 0011 0100 0100

✅ Resultado: No se detectaron errores.
Mensaje original: 10101010 01100110
Nota: CRC32 detecta todos los errores de 1 bit y todas las rafagas de hasta 32 bits; no corrige errores.

```

Cambiamos un bit de este mensaje saliente y lo volvemos a enviar al CRC y después la salida de eso lo volvemos a pasar al Receptor y si lo aceptara

```

=====
  RECEPTOR CRC-32 (IEEE802.3)
=====
Parametrizacion: poly=0x04C11DB7, init=0xFFFFFFFF, xorout=0xFFFFFFFF, reflejado=No

Ingrese trama (mensaje||CRC32, 32 bits al final): 101010100110011011000000111111110001000110100010100111100001110100011111100110011

-----
Longitud mensaje: 48 bits
CRC recibido   : 0011 1100 0011 1010 0011 1111 0011 0011
CRC calculado  : 0011 1100 0011 1010 0011 1111 0011 0011

✅ Resultado: No se detectaron errores.
Mensaje original: 10101010 01100110 11000001 11111110 00100011 01000101
Nota: CRC32 detecta todos los errores de 1 bit y todas las rafagas de hasta 32 bits; no corrige errores.

```

CRC-32 detecta errores aleatorios con probabilidad de fallo $\sim 1/2^{32}$, pero no auténtica: si un atacante modifica el mensaje y recalcula el CRC, el receptor lo acepta. Lo demostramos generando una trama válida para otro mensaje (B) y el receptor la aceptó.

Ventajas y desventajas

Hamming (SEC)

Ventajas

- Corrige 1 bit de error automáticamente.
- Overhead pequeño y predecible: $r \geq m+1$ bits con $2^r \geq m+r+1$.
- Implementación simple y muy rápida.

Desventajas

- Con ≥ 2 bits, puede fallar e incluso miscorregir (silencioso).
- Existen patrones de 3 bits indetectables (p. ej. 1,2,3).
- Requiere bloques con tamaños compatibles (ajustar $m, r, m+r$).

Cuándo usarlo

- Cuando necesitas corrección (SEC) y el canal no es muy ruidoso; payload corto/medio.

CRC-32

Ventajas

- Altísima capacidad de detección: 1-bit, 2-bits y ráfagas ≤ 32 bits garantizadas; prob. de fallo aleatorio $\sim 1/2^{32}$.
- Overhead fijo de 32 bits (relativamente bajo para mensajes largos).
- Muy eficiente (soporte HW/bitwise rápido) y sirve para cualquier longitud de mensaje.

Desventajas

- No corrige (solo detecta).
- No autentica: un atacante que recalcula el CRC tras modificar el payload lo hace pasar.
- En mensajes muy cortos, 32 bits de overhead puede ser alto.

Cuándo usarlo

- Cuando necesitas detectar con fiabilidad en tramas medianas/largas y combinar con reintentos/ARQ.

Segunda parte:

Pruebas de emisor receptor

```
=====
RECEPTOR por CAPAS (CRC32 / HAMMING)
=====
Servidor RECEPTOR escuchando en 0.0.0.0:50007

↘ Conexión de 127.0.0.1:34772
-----
Algoritmo: CRC32
len(frame): 152 bits
-----

----- CRC32 -----
Datos (bits): 01001000 01101111 01101100 01100001 00100000 01100011 01101111 01101101 01101111 00100000 011
00101 01110011 01110100 01100001 01110011
CRC recibido : 0001 0111 1111 0101 0111 1001 0010 0110
CRC calculado: 0001 0111 1111 0101 0111 1001 0010 0110
✅ Resultado: No se detectaron errores.
Mensaje (ASCII): Hola como estas
Nota: CRC32 detecta todos los errores de 1 bit y todas las ráfagas de hasta 32 bits; no corrige.
□
```

```
=====
EMISOR por CAPAS (CRC32 / HAMMING)
=====
Mensaje a enviar (texto ASCII): Hola como estas
Algoritmo [1=CRC32, 2=HAMMING]: 1
Probabilidad de error por bit (ej. 0.00, 0.01, 0.05): 0,01
IP receptor [enter=127.0.0.1]:
Puerto receptor [enter=50007]:

----- RESUMEN EMISOR -----
Texto original: Hola como estas
ASCII→Bits: 01001000 01101111 01101100 01100001 00100000 01100011 01101111 01101101 01101111 00100000 01100101 01110011 01
110100 01100001 01110011
CRC32 (bin): 0001 0111 1111 0101 0111 1001 0010 0110
CRC32 (hex): 0x17F57926
Trama sin ruido: 01001000 01101111 01101100 01100001 00100000 01100011 01101111 01101101 01101111 00100000 01100101 01110011
01110100 01100001 01110011 00010111 11110101 01111001 00100110
Trama con ruido: 01001000 01101111 01101100 01100001 00100000 01100011 01101111 01101101 01101111 00100000 01100101 01110011
01110100 01100001 01110011 00010111 11110101 01111001 00100110
Destino socket: 127.0.0.1:50007

✅ Trama enviada por socket.
```

Con proxy emitiendo ruido para corrección de errores

```
Lab2-Redes
python part2/receiver/receiver.py --host 0.0.0.0 --port 50007
RECEPTOR por CAPAS (CRC32 / HAMMING)
Servidor RECEPTOR escuchando en 0.0.0.0:50007

Conexión de 127.0.0.1:34958
Algoritmo: CRC32
len(frame): 72 bits

CRC32
Datos (bits): 00001000 01111010 01101100 00100001 00100000
CRC recibido : 0101 1110 0001 1000 1011 1000 1010 1100
CRC calculado: 0101 1110 0001 1000 0010 1110 1010 1111
Resultado: Se detectaron errores. Verificación no coincide.
Distancia de Hamming (cnc): 6 bit(s)
Acción: descartar trama (CRC es de detección, no corrige).

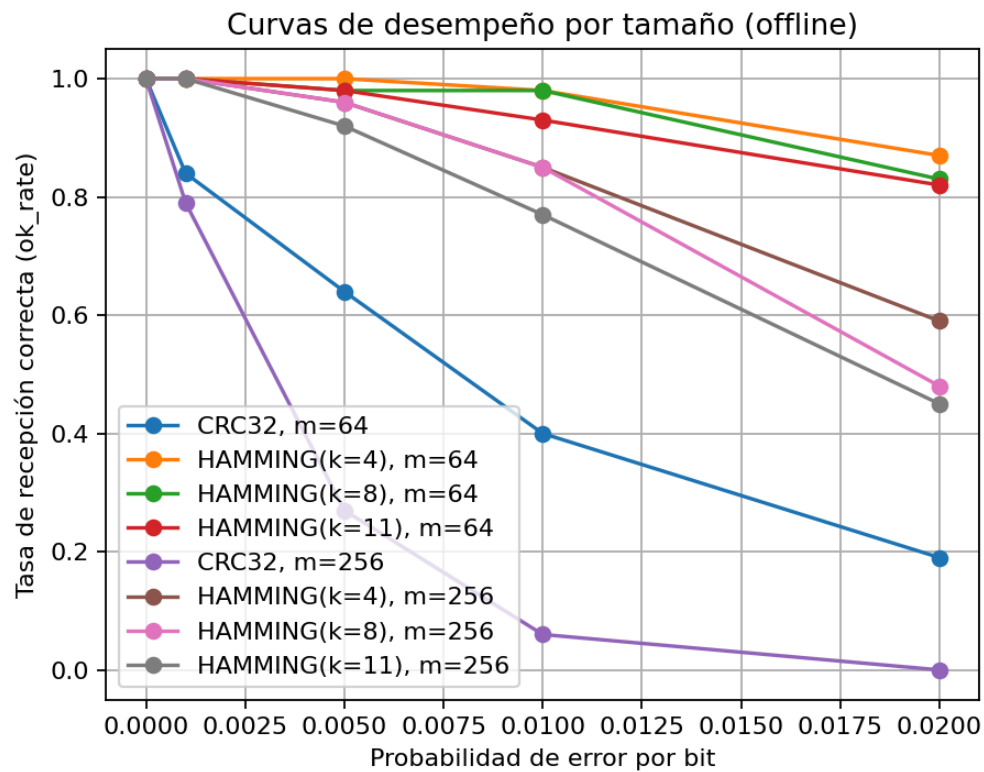
SIMULADOR DE CANAL (Pruvy con ruido)
Escuchando en 0.0.0.0:50006 - destino 127.0.0.1:50007, BER=0.02

EMISOR por CAPAS (CRC32 / HAMMING)
Mensaje a enviar (texto ASCII): Hola
Algoritmo [1=CRC32, 2=HAMMING]: 1
Probabilidad de error por bit (e): 0.00, 0.01, 0.05: 0.05
IP receptor (entero): 127.0.0.1
Puerto receptor (entero): 50007

RESUMEN EMISOR
Texto original: Hola
ASCII-Bits: 01001000 01101111 01101100 01100001 00100000
CRC32 (bin): 0100 1110 0001 1000 1011 1010 1010 1100
CRC32 (hex): 0x01350AC
Trama sin ruido: 01001000 01101111 01101100 01100001 00100000 01001110 00011000 10111010 10101100
Trama con ruido: 00001000 01111010 01101100 00100001 00100000 01011110 00011000 10111000 10101100
Destino socket: 127.0.0.1:50007

Trama enviada por socket.
```

Métricas de estrés en modo offline:



Resultados

Los datos obtenidos coinciden con la teoría y permiten comparar directamente el comportamiento de CRC-32 y del código Hamming SEC.

- CRC-32 (solo detección). La proporción de tramas aceptadas (ok) iguala la probabilidad de que no ocurra ningún flip en todo el paquete: $(1-p)^{(m+32)}$. Para $m=64$ bajamos de 0,91 a 0,38 y luego a casi 0 al pasar de $p=0,001$ a 0,01 y 0,05; para $m=256$ la caída es aún más pronunciada. Esto confirma que cualquier error, aunque sea mínimo, provoca el rechazo de la trama.
- Hamming SEC (corrección de 1 bit). A BER bajos (0,1 %–1 %) el ratio ok se mantiene por encima de 0,95 porque la mayoría de los errores afectan a un solo bit y se corrigen. El contador `corr_avg` crece con p , reflejando más correcciones efectivas. Al aumentar p o el tamaño de bloque ($k=8/11$) crece la probabilidad de que aparezcan dos o más errores en un mismo bloque, lo que genera `uncor_avg` > 0 y reduce ok; sin embargo, incluso a $p=0,05$ el código sigue aceptando un 34–48 % de tramas ($m=64$), superior al desempeño de CRC-32 en ese punto.

Conclusiones

CRC-32 ofrece una detección muy confiable, pero exige retransmisión cuando aparece cualquier error; su tasa de éxito decae conforme $(1-p)^{(m+32)}$. El código Hamming SEC, por el contrario, incrementa la tasa de recepción sin retransmisión en entornos con baja o moderada tasa de errores, gracias a la capacidad de corregir 1 bit por bloque; esta ventaja se pierde cuando la probabilidad de ≥ 2 errores por bloque se vuelve apreciable, especialmente en bloques largos.