

## Laboratorio 2 Reporte

### Pruebas

#### Sin errores:

<pre>PS C:\Users\garci\OneDrive\Documents\Tercer semestre U\IALab4\Lab2Redes&gt; &amp; C:\Users\garci\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/garci/OneDrive/Documentos/Tercer semestre U/IALab4/Lab2Redes/emitter.py" Ingrese la trama original en binario (ej. 1011001): 1011 Trama codificada (Hamming): 0110011 PS C:\Users\garci\OneDrive\Documents\Tercer semestre U\IALab4\Lab2Redes&gt;</pre>	<pre>PS C:\Users\garci\OneDrive\Documents\Tercer semestre U\IALab4\Lab2Redes&gt; node receiver.js Ingrese la trama codificada (Hamming): 0110011 No se detectaron errores. Trama recibida (corregida si aplica): 0110011 Datos extraídos: 1011 PS C:\Users\garci\OneDrive\Documents\Tercer semestre U\IALab4\Lab2Redes&gt;</pre>
<pre>Ingrese la trama original en binario (ej. 1011001): 0110011 Trama codificada (Hamming): 0100110011 PS C:\Users\garci\OneDrive\Documents\Tercer semestre U\IALab4\Lab2Redes&gt;</pre>	<pre>Ingrese la trama codificada (Hamming): 0100110011 No se detectaron errores. Trama recibida (corregida si aplica): 0100110011 Datos extraídos: 0110011 PS C:\Users\garci\OneDrive\Documents\Tercer semestre U\IALab4\Lab2Redes&gt;</pre>
<pre>Ingrese la trama original en binario (ej. 1011001): 01101110 Trama codificada (Hamming): 110011011110 PS C:\Users\garci\OneDrive\Documents\Tercer semestre U\IALab4\Lab2Redes&gt;</pre>	<pre>Ingrese la trama codificada (Hamming): 110011011110 No se detectaron errores. Trama recibida (corregida si aplica): 110011011110 Datos extraídos: 01101110</pre>

#### Un error:

**Original** 1011 → 0110011

Mutar bit 3 (cód. recibido = 0100011)

<pre>/Programs/Python/Python313/python.exe "c:/Users/garci/OneDrive/Documentos/Tercer semestre U/IALab4/Lab2Redes/emitter.py" Ingrese la trama original en binario (ej. 1011001): 1011 Trama codificada (Hamming): 0110011</pre>	<pre>Ingrese la trama codificada (Hamming): 0100011 Error detectado en posición 3. Corrigiendo... Trama recibida (corregida si aplica): 0110011 Datos extraídos: 1011</pre>
--	---

**Original** 1101001 → 01101011001

Mutar bit 5 (cód. recibido = 01100011001)

<pre>/Programs/Python/Python313/python.exe "c:/Users/garci/OneDrive/Documentos/Tercer semestre U/IALab4/Lab2Redes/emitter.py" Ingrese la trama original en binario (ej. 1011001): 1101001 Trama codificada (Hamming): 01101011001</pre>	<pre>Ingrese la trama codificada (Hamming): 01100011001 Error detectado en posición 5. Corrigiendo... Trama recibida (corregida si aplica): 01101011001 Datos extraídos: 1101001</pre>
---	--

**Original** 01101110 → 110011011110

Mutar bit 2 (cód. recibido = 100011011110)

<pre>/Programs/Python/Python313/python.exe "c:/Users/garci/OneDrive/Documentos/Tercer semestre U/IALab4/Lab2Redes/emitter.py" Ingrese la trama original en binario (ej. 1011001): 01101110 Trama codificada (Hamming): 110011011110</pre>	<pre>Ingrese la trama codificada (Hamming): 100011011110 Error detectado en posición 2. Corrigiendo... Trama recibida (corregida si aplica): 110011011110 Datos extraídos: 01101110</pre>
---	---

#### Dos errores o más:

Hamming sólo asegura corrección de un bit. Con dos o más, el receptor trata de corregir “un” error, pero el resultado queda incorrecto:

**Original** 1011 → 0110011

Mutar bits 2 y 5 → recibido 0010111

<pre>/Programs/Python/Python313/python.exe "c:/Users/garci/OneDrive/Documentos/Tercer semestre U/IALab4/Lab2Redes/emitter.py" Ingrese la trama original en binario (ej. 1011001): 1011 Trama codificada (Hamming): 0110011</pre>	<pre>Ingrese la trama codificada (Hamming): 0010111 Error detectado en posición 7. Corrigiendo... Trama recibida (corregida si aplica): 0010110 Datos extraídos: 1110</pre>
--	---

No logra corregirlo por la naturaleza del algoritmo de Hamming

Sofía García – 22210  
Julio García Salas - 22076

**Original** 1101001 → 01101011001

Mutar bits 3 y 4 → recibido 01011011001

```
/Programs/Python/Python313/python.exe "C:/Users/garci/OneDrive/Documentos/Tercer semestre U/IA/Lab4/Lab2R
edes/emitter.py"
Ingrese la trama original en binario (ej. 1011001): 1101001
Trama codificada (Hamming): 01101011001

Ingrese la trama codificada (Hamming): 01011011001
Error detectado en posición 7. Corrigiendo...
Trama recibida (corregida si aplica): 01101011001
Datos extraídos: 0100001
```

**Original** 01101110 → 11001101110

Mutar bits 5 y 6 → recibido 110000011110

```
/Programs/Python/Python313/python.exe "C:/Users/garci/OneDrive/Documentos/Tercer semestre U/IA/Lab4/Lab2R
edes/emitter.py"
Ingrese la trama original en binario (ej. 1011001): 01101110
Trama codificada (Hamming): 11001101110

Ingrese la trama codificada (Hamming): 110000011110
Error detectado en posición 3. Corrigiendo...
Trama recibida (corregida si aplica): 111000011110
Datos extraídos: 10001110
```

**¿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**

Es posible manipular los bits de tal forma que el algoritmo de Hamming no sea capaz de detectar un error múltiple, ya que el esquema de Hamming estándar está diseñado para corregir un solo bit erróneo y puede confundir dos o más errores con uno solo. En nuestra implementación, cuando alteramos dos bits en posiciones específicas de la trama codificada, el receptor calcula un síndrome distinto de cero, interpreta que hay un único error en otra posición y “corrige” ese bit, sin advertir la presencia de un segundo error. De este modo, el mensaje final queda corrupto sin que el protocolo lo detecte como fallo irremediable, demostrando que Hamming puro carece de un mecanismo de alarma para errores de ráfaga o múltiples.

## Comparación entre Hamming y los otros algoritmos

En base a las pruebas realizadas (que incluyeron únicamente el código de Hamming) y a la investigación bibliográfica de otros métodos, podemos identificar las siguientes ventajas y desventajas. El bit de paridad par ofrece la redundancia más baja (solo un bit extra) y una implementación ultrarrápida gracias a una sola operación XOR, pero solo detecta errores de paridad impar y no corrige ninguno, siendo incapaz de advertir dos errores concurrentes. Hamming añade un overhead logarítmico en bits de paridad para corregir automáticamente un único error y detectar cuándo hay al menos dos, aunque no puede diferenciar dos fallos de uno solo, y su complejidad aumenta con el número de rutas de paridad a verificar. CRC-32, basándonos en la literatura, inserta 32 bits de redundancia y emplea aritmética polinómica para detectar con alta fiabilidad ráfagas largas, dobles o triples errores, pero no corrige fallos y supone un mayor coste de procesamiento para el cálculo del residuo. En definitiva, la elección del método dependerá del tipo de canal, la tolerancia al overhead y la necesidad de corrección frente a detección pura.