

# *LABORATORIO*

## *NO. 2*

Esquemas de detección y corrección: Parte 1

*Marco Ramirez 21032 | Josué Morales 21116*

*[Company Name] | [Company Address]*

## Tabla de contenidos

<b>Corrección de errores.....</b>	<b>2</b>
<b>Algoritmo de Hamming .....</b>	<b>2</b>
Pruebas .....	2
<b>Detección de errores.....</b>	<b>4</b>
<b>Fletcher checksum.....</b>	<b>4</b>
Pruebas .....	4
<b>Discusión.....</b>	<b>6</b>
<b>Conclusiones .....</b>	<b>7</b>
<b>Bibliografía .....</b>	<b>8</b>

## Corrección de errores

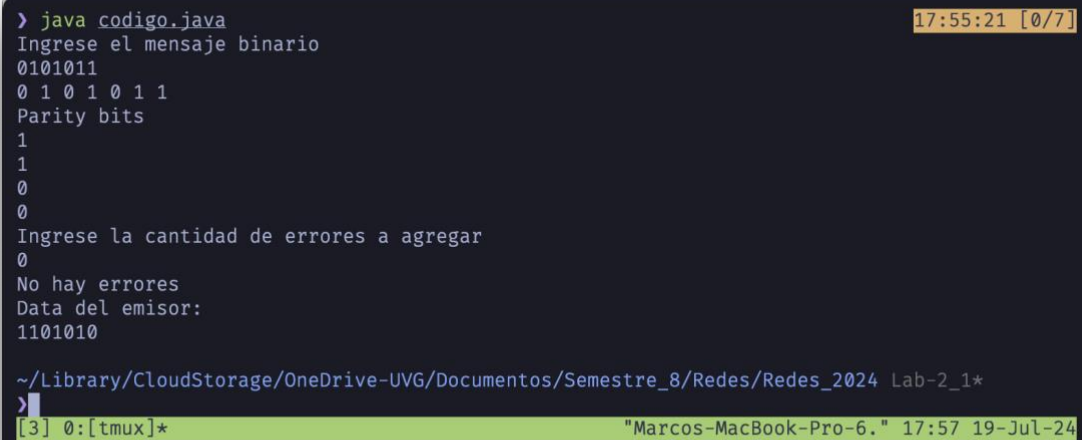
### Algoritmo de Hamming

Es una técnica de corrección de errores diseñada para asegurar la exactitud de los datos durante su transmisión o almacenamiento. Este código identifica y corrige errores que pueden surgir en el proceso de enviar o almacenar información desde el remitente hasta el receptor. (GeeksforGeeks, 2024)

### Pruebas

(GeekforGeeks, Hamming code Implementation in Java , 2020)

- 0101011



```
> java codigo.java
Ingrese el mensaje binario
0101011
0 1 0 1 0 1 1
Parity bits
1
1
0
0
Ingrese la cantidad de errores a agregar
0
No hay errores
Data del emisor:
1101010

~/Library/CloudStorage/OneDrive-UVG/Documentos/Semestre_8/Redes/Redes_2024 Lab-2_1*
>
[3] 0:[tmux]* "Marcos-MacBook-Pro-6." 17:57 19-Jul-24
```

- 000110

```
> java codigo.java
Ingrese el mensaje binario
000110
0 0 0 1 1 0
Parity bits
0
1
1
1
Ingrese la cantidad de errores a agregar
0
No hay errores
Data del emisor:
011000

~/Library/CloudStorage/OneDrive-UVG/Documentos/Semestre_8/Redes/Redes_2024 Lab-2_1*
>
[3] 0:zsh* "Marcos-MacBook-Pro-6." 18:00 19-Jul-24
```

- 1000111

```
> java codigo.java
Ingrese el mensaje binario
1000111
1 0 0 0 1 1 1
Parity bits
1
1
0
1
Ingrese la cantidad de errores a agregar
0
No hay errores
Data del emisor:
1110001

~/Library/CloudStorage/OneDrive-UVG/Documentos/Semestre_8/Redes/Redes_2024 Lab-2_1* 12s
>
[3] 0:zsh* "Marcos-MacBook-Pro-6." 18:01 19-Jul-24
```

## Detección de errores

### Fletcher checksum

Es un método de detección de errores utilizado por los protocolos de capa superior y se considera más fiable que el LRC, el VRC y el CRC. Este método emplea un Generador de Checksum en el lado del emisor y un Verificador de Checksum en el lado del receptor.

En el lado del emisor, el generador de checksum divide los datos en subunidades de  $n$  bits (generalmente de 16 bits), las suma usando el método de complemento a uno, y luego complementa el resultado. Este checksum complementado se añade al final de los datos originales y se envía al receptor. (GeekforGeeks, 2024)

### Pruebas

(yadav, 2020)

- 1010001

```
> python3 detectionFletcher.py
Ingrese la trama de datos en formato binario: 1010001
Ingrese el tamaño del bloque (8, 16 o 32): 8
Checksum calculado para datos originales: 1010001010100010
Trama de datos recibidos sin errores: 10100010
Checksum recibido: 1010001010100010
El checksum es válido.

~/OneDrive - UVG/Documentos/Semestre_8/Redes/Redes_2024 Lab-2_1* 48s
> █

[2] 0:zsh* "Marcos-MacBook-Pro-6." 17:29 19-Jul-24
```

- 1101001

```
> python3 detectionFletcher.py
Ingrese la trama de datos en formato binario: 1101001
Ingrese el tamaño del bloque (8, 16 o 32): 8
Checksum calculado para datos originales: 1101001011010010
Trama de datos recibidos sin errores: 11010010
Checksum recibido: 1101001011010010
El checksum es válido.

~/OneDrive - UVG/Documentos/Semestre_8/Redes/Redes_2024 Lab-2_1* 6s
> █

[2] 0:zsh* "Marcos-MacBook-Pro-6." 17:34 19-Jul-24
```

- 0011101

```
> python3 detectionFletcher.py
Ingrese la trama de datos en formato binario: 0011101
Ingrese el tamaño del bloque (8, 16 o 32): 8
Checksum calculado para datos originales: 11101000111010
Trama de datos recibidos sin errores: 00111010
Checksum recibido: 11101000111010
El checksum es válido.

~/OneDrive - UVG/Documentos/Semestre_8/Redes/Redes_2024 Lab-2_1*
> █

[2] 0:zsh* "Marcos-MacBook-Pro-6." 17:36 19-Jul-24
```

## Discusión

El análisis de los algoritmos de corrección y detección de errores, específicamente el código de Hamming y el checksum de Fletcher, tiene algunas características clave y limitaciones en la práctica.

Para el **algoritmo de Hamming** implementado en Java, observamos que la detección y corrección de errores están limitadas a un solo error por trama. Aunque el algoritmo es eficaz para corregir errores simples y detectar errores en una sola posición, no es capaz de identificar más de un error en la trama. Esto se debe a que el algoritmo se ajusta a una estructura específica que solo detecta el primer error. Una posible mejora sería la incorporación de recursividad en la función de corrección, lo que permitiría detectar y corregir múltiples errores. Sin embargo, se optó por mantener la estructura original para no desviarse de los estándares establecidos para el algoritmo.

Por otro lado, en el caso del **checksum de Fletcher** desarrollado en Python, se diseñó una función que generaba tramas de datos aleatorios para simular errores y verificar la capacidad del algoritmo para detectarlos. En los experimentos realizados con datos correctos, no se detectaron errores, pero el algoritmo mostró que podía identificar el primer error en las tramas alteradas. Similar al código de Hamming, el checksum de Fletcher también tiene la limitación de detectar solo el primer error.

## Conclusiones

- El código de Hamming y el checksum de Fletcher solo detectan y corrigen el primer error en una trama.
- Ambos métodos son eficaces para la corrección y detección de errores simples, pero limitados en escenarios con múltiples errores.
- El código de Hamming sigue estándares estrictos, lo que restringe su capacidad para detectar errores adicionales más allá del primero.
- El checksum de Fletcher, aunque flexible, también muestra limitaciones similares en la corrección de errores múltiples.



## **Bibliografía**

GeekforGeeks. (11 de junio de 2020). *Hamming code Implementation in Java* . Obtenido de

Geek for Geeks: <https://www.geeksforgeeks.org/hamming-code-implementation-in-java/>

GeekforGeeks. (7 de mayo de 2024). *Error Detection Code checksum*. Obtenido de Geek for

geeks: <https://www.geeksforgeeks.org/error-detection-code-checksum/>

GeeksforGeeks. (17 de Junio de 2024). *Hamming Code in Computer Network*. Obtenido de Geek

for Geeks: <https://www.geeksforgeeks.org/hamming-code-in-computer-network/>

yadav, C. (30 de junio de 2020). *Fletcher's Checksum* . Obtenido de tutorial's point:

<https://www.tutorialspoint.com/fletcher-s-checksum>