

**UNIVERSIDAD DEL VALLE DE GUATEMALA**

*CC3067 - Informe de Laboratorio*

Sección 11

Ing. Miguel Novella



## Esquema de detección y corrección de errores

- Luis Pedro Gonzalez Aldana 20008
- Axel Leonardo López 20768

**GUATEMALA, 04 de Agosto de 2023**

# Descripción de la práctica

La práctica consiste en probar algunas técnicas para la detección y corrección de tramos de bits en la red. Probando sus capacidades y limitaciones en distintas situaciones. Para ejemplificar estos procesos, seleccionamos el algoritmo *Hamming* para la corrección de errores, y el algoritmo de *CRC-1* para la detección de errores. Para probar limitaciones y capacidades se evaluaron tres tramas de la función emisión a receptor sin error; tres tramas con un bit cambiado, con el error detectado; tres tramas con más de un bit cambiado; y una trama modificada pero indetectable.

## Hamming

Es un método para detectar y corregir errores en los datos transmitidos a través de un medio ruidoso, como un canal de comunicación. Funciona añadiendo "bits de paridad" a los datos que se van a transmitir. Estos bits de paridad son calculados de tal forma que se pueden usar para detectar y corregir un número limitado de errores en los datos transmitidos.

Proceso:

- Paso 1 - Preparar la información a enviar:  
Primero se convierte el mensaje en una cadena de bits.
- Paso 2 - Agregar "bits de paridad": El código de Hamming añade algunos bits adicionales al mensaje. Estos bits adicionales se llaman "bits de paridad", y su trabajo es ayudar a detectar y corregir los errores que puedan ocurrir durante la transmisión del mensaje.
- Paso 3 - Calcular los bits de paridad:  
Para cada bit de paridad, se toma ciertos bits de tu mensaje original y se calcula si hay un número par o impar de 1's. Si hay un número par de 1's, entonces el bit de paridad es 0. Si hay un número impar de 1's, entonces el bit de paridad es 1.
- Paso 4 - Enviar el mensaje:  
Una vez que calculado todos los bits de paridad, se insertan en el mensaje original en las posiciones que corresponden a las potencias de 2 (1, 2, 4, 8, etc.). Luego se envía todo el paquete: el mensaje original más los bits de paridad.
- Paso 5 - Verificar y corregir el mensaje:  
Cuando se recibe el mensaje, se vuelven a calcular los bits de paridad. Si los bits de paridad calculados coinciden con los bits de paridad recibidos, se sabe que el mensaje llegó correctamente. Pero si hay una discrepancia, entonces sabes que hay un error. Además se puede usar los bits de paridad para determinar exactamente dónde está el error, y corregirlo.

# CRC-1

Es una técnica usada para detectar errores en la transmisión de datos en redes y otras formas de comunicación digital. El CRC-1 se refiere a la versión más simple de este algoritmo, que se basa en operaciones con números binarios.

Proceso:

- Paso 1 - Generar el Polinomio Generador:  
Este es un número binario que se utilizará para "dividir" la trama. En el caso del CRC-1, este polinomio generador es simplemente "11" (o 3 en decimal).
- Paso 2 - Preparar los datos para la transmisión:  
Lo primero es añadir un '0' al final de nuestra cadena de bits.
- Paso 3 - Dividir los datos con el Polinomio Generador:  
Se divide la cadena de bits entre el polinomio generador, utilizando la división binaria. En la división binaria, en lugar de restar, como haríamos en la división normal, se hace una operación XOR'.
- Paso 4: Agregar el resto a los datos:  
Se agrega el resto de la división a la cadena de bits original (sin el '0' añadido), reemplazando el '0' añadido en el Paso 2.
- Paso 5 - Enviar los datos:  
Ahora se envía la cadena de bits. Cuando llega al destinatario, éste realizará la misma división del Paso 3. Si no hay errores de transmisión, el destinatario debería obtener el mismo resto. Si el resto que obtiene el destinatario es diferente, entonces sabrán que ha ocurrido un error de transmisión.

## Resultados

Los bits volteados están en negrilla.

## Hamming

Tres tramos correctos

**Tabla 1: Tramas sin ruido usando hamming**

Entrada	Salida	Evidencia emisor	Evidencia receptor
1001010	01110011010	<div><div>Ingrese la trama: 1001010</div><div>Submit</div><div>Respuesta: 01110011010</div></div>	<div><pre>48 if __name__ == "__main__": 49     # Introducir un error para probar la corrección de errores 50     #transmitted_message = interfer_and_modify_trama(transmitted_message, [5]) 51 52     error_position, received_message = ham_receptor("01110011010") 53     if error_position == 0: 54         print("No se encontraron errores. Trama recibida:", received_message) 55     else: 56         print("Se encontró un error y se corrigió. Trama recibida:", received_message) 57</pre></div>

Ingrese la trama:

1110110 11101100110

Respuesta: 11101100110

Ingrese la trama:

1101011 00101010011

Respuesta: 00101010011

```
48 if __name__ == "__main__":
49     # Introducir un error para probar la corrección de errores
50     #transmitted_message = interfer_and_modify_trama(transmitted_message, [5])
51
52     error_position, received_message = ham_receptor("11101100110")
53     if error_position == 0:
54         print("No se encontraron errores. Trama recibida:", received_message)
55     else:
56         print(f"Se encontró un error y se corrigió. Trama recibida:", received_message)
57
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

PS C:\Users\lp109\OneDrive\Documents\0lps\01 U\0l.a\U\000Octavo semestre\Redes\Lab2> & C:/Users/lp109/OneDrive/Documents/0lps/01 U/0l.a/U/000Octavo semestre/Redes/Lab2/python\_version/hamming.py  
No se encontraron errores. Trama recibida: 01110011010  
PS C:\Users\lp109\OneDrive\Documents\0lps\01 U\0l.a\U\000Octavo semestre\Redes\Lab2> & C:/Users/lp109/OneDrive/Documents/0lps/01 U/0l.a/U/000Octavo semestre/Redes/Lab2/python\_version/hamming.py  
No se encontraron errores. Trama recibida: 11101100110  
PS C:\Users\lp109\OneDrive\Documents\0lps\01 U\0l.a\U\000Octavo semestre\Redes\Lab2> █

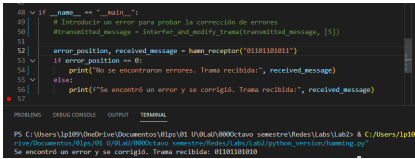
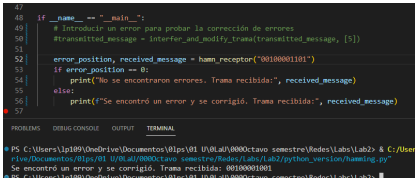
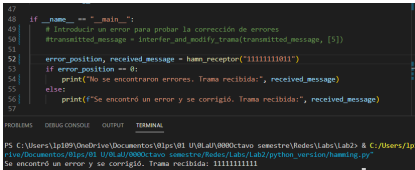
```
47 if __name__ == "__main__":
48     # Introducir un error para probar la corrección de errores
49     #transmitted_message = interfer_and_modify_trama(transmitted_message, [5])
50
51     error_position, received_message = ham_receptor("00101010011")
52     if error_position == 0:
53         print("No se encontraron errores. Trama recibida:", received_message)
54     else:
55         print(f"Se encontró un error y se corrigió. Trama recibida:", received_message)
56
57
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

PS C:\Users\lp109\OneDrive\Documents\0lps\01 U\0l.a\U\000Octavo semestre\Redes\Lab2> & C:/Users/lp109/OneDrive/Documents/0lps/01 U/0l.a/U/000Octavo semestre/Redes/Lab2/python\_version/hamming.py  
Se encontró un error y se corrigió. Trama recibida: 00101010011  
PS C:\Users\lp109\OneDrive\Documents\0lps\01 U\0l.a\U\000Octavo semestre\Redes\Lab2> █

## Tres tramos con un bit volteado

**Tabla 2: Tramas con un bit volteado usando hamming**

Entrada	Salida	Salida mod.	Evidencia emisor	Evidencia receptor
			<div><div>Ingrese la trama: <input type="text" value="1110010"/></div><div><input type="button" value="Submit"/></div><div>Respuesta: 01101101010</div></div>	
1110010	01101101010	01101101011		
			<div><div>Ingrese la trama: <input type="text" value="1000001"/></div><div><input type="button" value="Submit"/></div><div>Respuesta: 00100001001</div></div>	
1000001	00100001001	00100001101		
			<div><div>Ingrese la trama: <input type="text" value="1111111"/></div><div><input type="button" value="Submit"/></div><div>Respuesta: 11111111111</div></div>	
1111111	11111111111	11111111011		

## Tres tramos con un dos bits volteados

**Tabla 3: Tramas con dos bits volteados usando hamming**

Entrada	Salida	Salida mod.	Evidencie emisor	Evidencia receptor
---------	--------	-------------	------------------	--------------------

11110100

1111010010

10101010

1010

01

Ingrese la trama:

10101010

Submit

Respuesta:

111101001010

11100010

01111100

0111110100

1

00101

001

Ingrese la trama:

111000101

Submit

Respuesta:

0111110000101

10001100

0010000

0010000011

0

011000

011

Ingrese la trama:

100011000

Submit

Respuesta:

0010000011000

```
27 if __name__ == "__main__":
28     error_position, received_message = ham_receptor("11110100101")
29     if error_position == 0:
30         print("No se encontraron errores. Trama recibida:", received_message)
31     elif received_message == "Error detectado pero no se pudo corregir":
32         print("Se encontró un error, pero no se pudo corregir. Trama recibida: (received_message)")
33     else:
34         print("Se encontró un error y se corrigió. Trama recibida:", received_message)
```

PROBLEMS

DEBUG CONSOLE

OUTPUT

TERMINAL

PS C:\Users\lp189\OneDrive\Documents\01ps\01 U\01al\0000ctavo semestre\Redes\Lab2 & C:\Users\lp189\OneDrive\Documents\01ps\01 U\01al\0000ctavo semestre\Redes\Lab2\python\_version\hamming.py

Se encontró un error y se corrigió. Trama recibida: 11110100101

```
27 if __name__ == "__main__":
28     error_position, received_message = ham_receptor("01111100001")
29     if error_position == 0:
30         print("No se encontraron errores. Trama recibida:", received_message)
31     elif received_message == "Error detectado pero no se pudo corregir":
32         print("Se encontró un error, pero no se pudo corregir. Trama recibida: (received_message)")
33     else:
34         print("Se encontró un error y se corrigió. Trama recibida:", received_message)
```

PROBLEMS

DEBUG CONSOLE

OUTPUT

TERMINAL

PS C:\Users\lp189\OneDrive\Documents\01ps\01 U\01al\0000ctavo semestre\Redes\Lab2 & C:\Users\lp189\OneDrive\Documents\01ps\01 U\01al\0000ctavo semestre\Redes\Lab2\python\_version\hamming.py

Se encontró un error y se corrigió. Trama recibida: 01111100001

```
27 if __name__ == "__main__":
28     error_position, received_message = ham_receptor("0010000011000")
29     if error_position == 0:
30         print("No se encontraron errores. Trama recibida:", received_message)
31     elif received_message == "Error detectado pero no se pudo corregir":
32         print("Se encontró un error, pero no se pudo corregir. Trama recibida: (received_message)")
33     else:
34         print("Se encontró un error y se corrigió. Trama recibida:", received_message)
```

PROBLEMS

DEBUG CONSOLE

OUTPUT

TERMINAL

PS C:\Users\lp189\OneDrive\Documents\01ps\01 U\01al\0000ctavo semestre\Redes\Lab2 & C:\Users\lp189\OneDrive\Documents\01ps\01 U\01al\0000ctavo semestre\Redes\Lab2\python\_version\hamming.py

Se encontró un error y se corrigió. Trama recibida: 0010000011000

Un tramo modificado con malicia

Tabla 4: Trama modificada a conveniencia

Entrada	Salida	Salida mod.	Evidencie emisor	Evidencia receptor
011111111	000111111111111	1111111111111	<div><div>Ingrese la trama:</div><div>0111111111</div><div>Submit</div></div> <div><div>Respuesta:</div><div>000111111111111</div></div>	<div><pre>27 if __name__ == "__main__": 28     error_position, received_message = ham_receptor("1111111111111") 29     if error_position == 0: 30         print("No se encontraron errores. Trama recibida:", received_message) 31     elif received_message == "Error detectado pero no se pudo corregir": 32         print("Se encontró un error, pero no se pudo corregir. Trama recibida: (received_message)") 33     else: 34         print("Se encontró un error y se corrigió. Trama recibida:", received_message)</pre></div> <div><div>PROBLEMS</div><div>DEBUG CONSOLE</div><div>OUTPUT</div><div>TERMINAL</div></div> <div><div>PS C:\Users\lp189\OneDrive\Documents\01ps\01 U\01al\0000ctavo semestre\Redes\Lab2 &amp; C:\Users\lp189\OneDrive\Documents\01ps\01 U\01al\0000ctavo semestre\Redes\Lab2\python_version\hamming.py</div><div>No se encontraron errores. Trama recibida: 1111111111111</div></div>

## CRC-1

Tres tramos correctos

Tabla 5: Tramas sin ruido usando CRC-1

Entrada	Salida	Evidencia emisor	Evidencia receptor
1001	01001	<div><div>CRC-1</div><div>Ingrese la trama:</div><div>1001</div><div>Submit</div></div> <div><div>Respuesta:</div><div>01001</div></div>	<div><pre>27 if __name__ == "__main__": 28       print("Verificación de paridad: " + str(check_parity_bit("01001")))</pre></div> <div><div>PROBLEMS</div><div>DEBUG CONSOLE</div><div>OUTPUT</div><div>TERMINAL</div></div> <div><div>PS C:\Users\lp189\OneDrive\Documents\01ps\01 U\01al\0000ctavo semestre\Redes\Lab2 &amp; C:\Users\lp189\OneDrive\Documents\01ps\01 U\01al\0000ctavo semestre\Redes\Lab2\python_version\parity.py</div><div>Verificación de paridad: True</div></div>

Ingrese la trama:

Submit

Respuesta: 110011

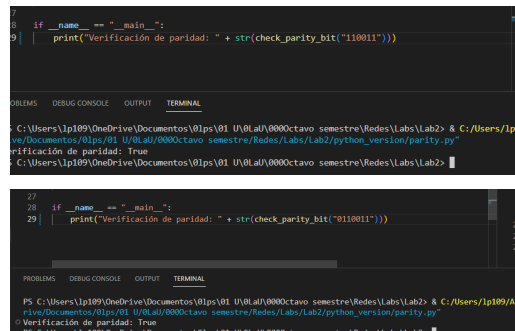
10011    110011

Ingrese la trama:

Submit

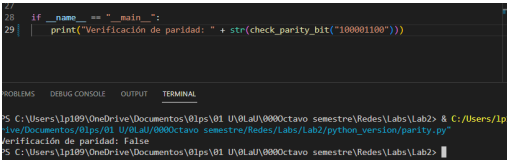
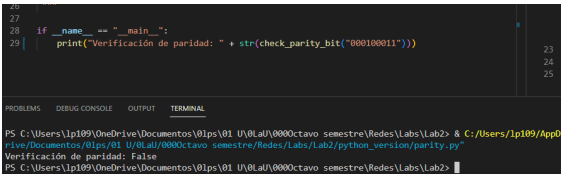

Respuesta: 0110011

110011    0110011



Tres tramos con un bit volteado

Tabla 6: Tramas con un bit volteado usando CRC-1

Entrada	Salida	Salida mod.	Evidencie emisor	Evidencia receptor
CRC-1				
			Ingrese la trama: <input type="text" value="00001101"/>	
			<input type="button" value="Submit"/>	
			Respuesta: 100001101	
00001101	1101	100001100		
CRC-1				
			Ingrese la trama: <input type="text" value="00101011"/>	
			<input type="button" value="Submit"/>	
			Respuesta: 00010011	
00101011	1011	000100011		
CRC-1				
			Ingrese la trama: <input type="text" value="10111111"/>	
			<input type="button" value="Submit"/>	
			Respuesta: 01011111	
10111111	11	01011101		

Tres tramos con un dos bits volteados

Tabla 7: Tramas con dos bits volteados usando CRC-1

Entrada	Salida	Salida mod.	Evidencie emisor	Evidencia receptor
10010010	Ingrese la trama: <input type="text" value="10010010"/>		Ingrese la trama: <input type="text" value="10010010"/>	
	Submit		Submit	
	Respuesta: 1100100100		Respuesta: 1100100100	
	0 00100 0010			

CRC-1

Ingrese la trama: 0101111

Submit

Respuesta: 10101111

0101111 111 0

0101101 101 11

Ingrese la trama: 0101101

Submit

Respuesta: 00101101

```

26
27
28 if __name__ == "__main__":
29 | print("Verificación de paridad: " + str(check_parity_bit("10111110")))

```

PROBLEMS

DEBUG CONSOLE

OUTPUT

TERMINAL

```

PS C:\Users\lp109\OneDrive\Documents\01ps\01 U\01al\0000octavo semestre\Redes\Labs\Lab2> & C:/Users/lp109/
rive/Documents/01ps/01 U/01al/0000octavo semestre/Redes/Labs/Lab2/python_version/parity.py
Verificación de paridad: True
PS C:\Users\lp109\OneDrive\Documents\01ps\01 U\01al\0000octavo semestre\Redes\Labs\Lab2>

```

```

7
8 if __name__ == "__main__":
9 | | print("Verificación de paridad: " + str(check_parity_bit("10101111")))

```

PROBLEMS

DEBUG CONSOLE

OUTPUT

TERMINAL


```

C:\Users\lp109\OneDrive\Documents\01ps\01 U\01al\0000octavo semestre\Redes\Labs\Lab2> & C:/Users/lp10
rive/Documents/01ps/01 U/01al/0000octavo semestre/Redes/Labs/Lab2/python_version/parity.py
Verificación de paridad: True
C:\Users\lp109\OneDrive\Documents\01ps\01 U\01al\0000octavo semestre\Redes\Labs\Lab2>

```

Un tramo modificado con malicia

Tabla 8: Trama modificada a conveniencia CRC-1

Entrada	Salida	Salida mod.	Evidencie emisor	Evidencia receptor
10000011	0 1100000110	1100001001	<div> <div>Ingrese la trama: 100000110</div> <div>Submit</div> </div> <div>Respuesta: 1100000110</div>	

# Discusión

Analizando los resultados de la práctica, y tomando en cuenta que para los dos algoritmos se realizaron las mismas pruebas: (Decodificación correcta, modificación de un bit, modificación de dos bits, modificación con malicia) podemos analizar los siguiente:

En el caso de Hamming, para la primera prueba, es una forma simple de poder codificar y decodificar los datos ya que al ser el caso de uso más optimista no se puede hacer un análisis más allá de la eficiencia de codificar y decodificar. Sin embargo, al ir a la siguiente prueba podemos ya comenzar a analizar sus verdaderas fortalezas y debilidades. Al detectar un error en la trama recibida el algoritmo es bastante eficaz y asertivo. Detecta que existe un error, en dónde ocurrió y lo corrige. Al realizar esta corrección el receptor puede leer el mensaje original y trabajar con él.

Para la tercera prueba se observó que el algoritmo comienza a fallar, debido a que solo puede corregir un error por mensaje, al tener dos o más llega a fallar, puede detectar el error o no, no sabe en qué bit existe el error y no puede corregirlos. Esta es la falla que tiene este algoritmo, por consiguiente al cambiar con malicia la trama esta falla estrepitosamente.

Por el lado de CRC-1 este algoritmo al ser uno solamente de detección no puede corregir los posibles errores recibidos. En el caso de las tramas recibidas correctamente solo debe de remover el primer bit de la trama y comenzar a utilizarla. Cuando existe un error, puede detectarlo, sin embargo no sabe cuál bit es erróneo ni puede corregirlo.

Para la tercera prueba el algoritmo puede detectar falsos positivos porque la suma de los números 1 's puede ser pares y tomarlo como correcto, si no es el caso detecta el error. Esto afecta de la misma manera la cuarta prueba y tiene los mismos efectos y/o resultados.

## Comentario grupal

Fue una buena experiencia para comprender lo complejo que a veces llegan a ser problemas que en un principio se ven simples, sobre todo en informática. Otro aprendizaje valioso es que existen muchas propuesta para la resolución de un problema, en el caso de la práctica, las diversas formas de detectar y/o corregir los errores al enviar datos de un lugar a otro.

## Conclusiones

- Las técnicas para para detección de errores son un buen punto de partida para preservar la consistencia de los datos transmitidos, pero hace falta implementar varios esquemas y más robustos para mejorar la confiabilidad.
- La poca confiabilidad en la transmisión de datos, según se puedan implementar débiles esquemas de detección/corrección de errores implica necesariamente aplicar técnicas de redundancia o mejorar el medio de transmisión.

## Referencias

- Pérez C. (2019) Error de verificación por redundancia cíclica: cómo solucionarlo. Extraído de: <https://www.muyinteresante.es/tecnologia/23119.html>
- Invarato R. (2016) Código de Hamming: Detección y Corrección de errores. Extraído de: <https://jarroba.com/codigo-de-hamming-deteccion-y-correccion-de-errores/>