Universidad del Valle de Guatemala
Estefania Elvira 20725
Jose Antonio Cayetano Molina 20211
Jose Pablo Monzon 20309

Laboratorio 2.1

Esquemas de detección y corrección de errores

Descripción
En esta práctica se pusieron a prueba dos grandes familias de algoritmos para manejar errores, los de detección y de corrección, para analizar sus ventajas y desventajas así como sus distintas implementaciones.

En el caso de los algoritmos de detección se implementaron:

Algoritmo Fletcher Checksum:

Este algoritmo parte de la base de un checksum en donde se dividen los bloques de un mensaje entre bloques de 8, 16 o 32 bits. Cabe recalcar que si uno de los bloques no llega a la cantidad de bits especificada, se agregan ceros como padding. Luego, esos bloques se expresan como números y se suman todos. La suma módulo de base del bloque luego se agregan al mensaje inicial al final para que el receptor pueda verificar si está bien el mensaje.

Y el algoritmo de corrección se escogió el:

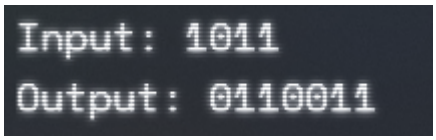
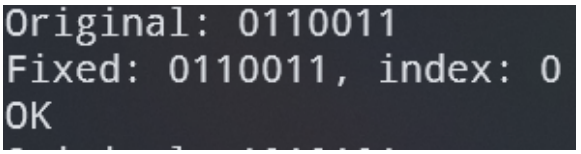
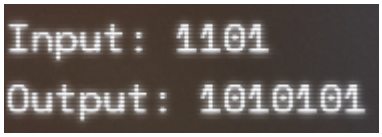
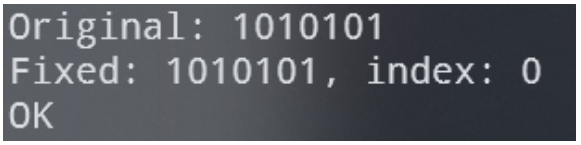
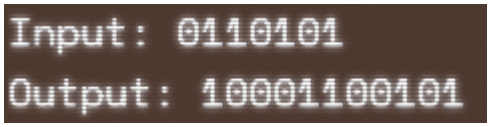
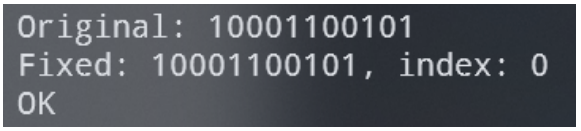
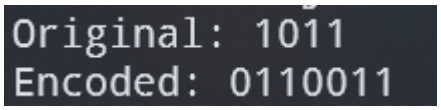
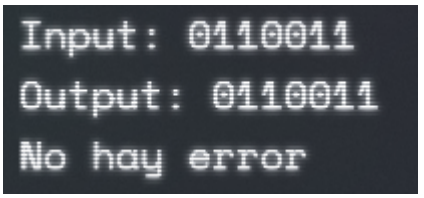
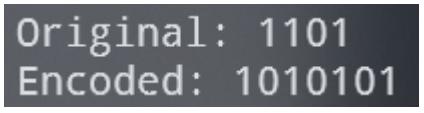
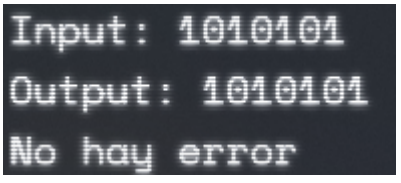
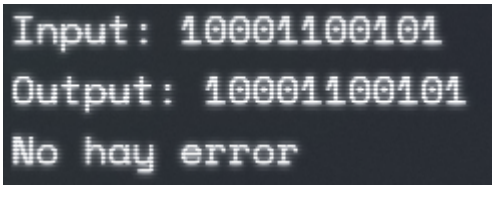
Algoritmo de Hamming

Que consiste en agregar múltiples bits de paridad en posiciones representativas de modo que al recibir el mensaje se puede verificar fácilmente si coinciden con estos bits de modo que se puede determinar y corregir en caso sea necesario.

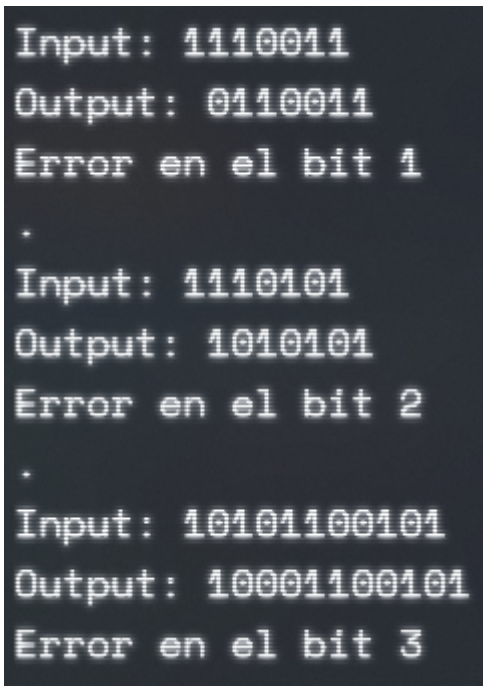
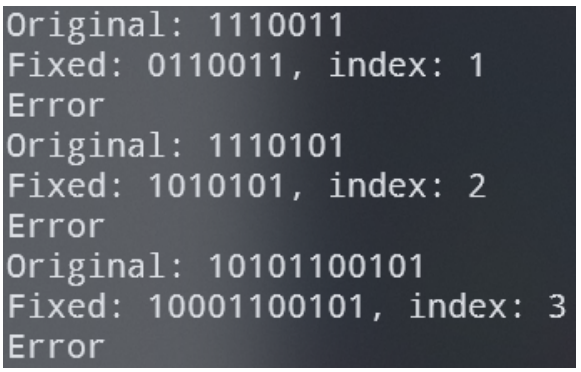
Resultados

Algoritmo de Hamming

Emisor-Receptor

Emisor Python	Receptor Javascript
	
	
	
Emisor Javascript	Emisor Python
	
	
	

1 Error

Python	Javascript
	

2 Errores

Python	Javascript

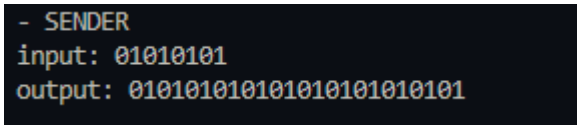
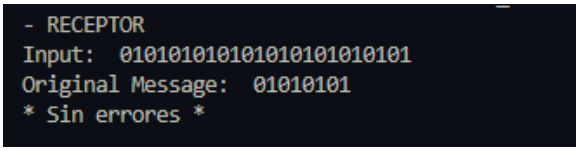
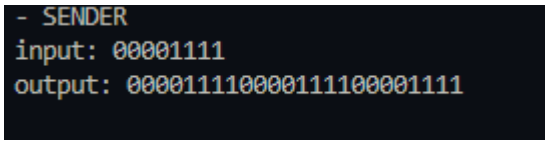
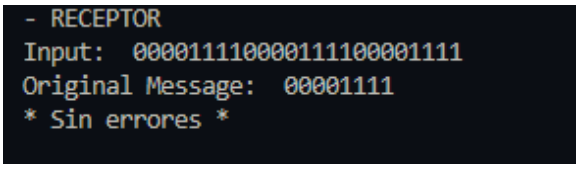
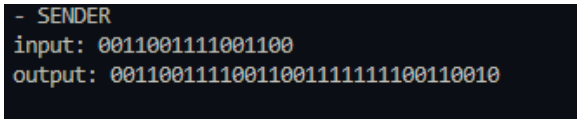
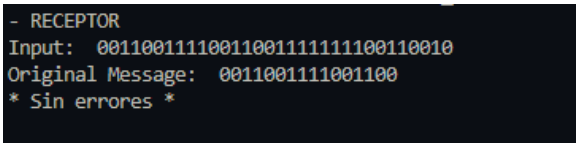
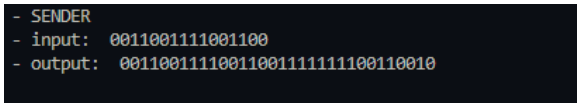
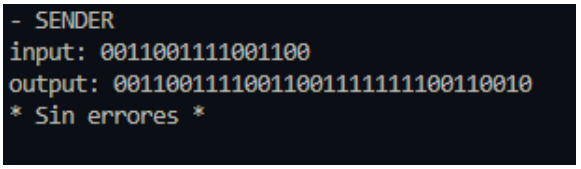
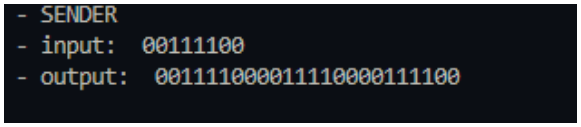
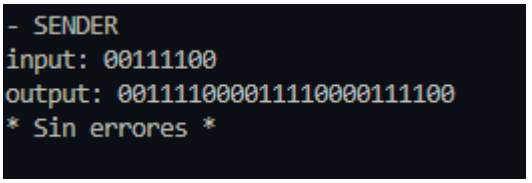
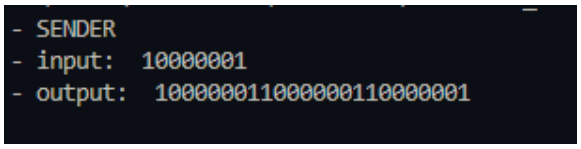
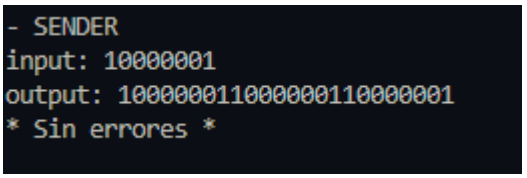
Original: 0110011 Input: 1111011 Output: 1111111 Error en el bit 5 . Original: 0110011 Input: 1110100 Output: 0110100 Error en el bit 1 . Original: 11001110101 Input: 10000100101 Output: 10001100101 Error en el bit 5	Original: 1111011 Fixed: 1111111, index: 5 Error Original: 1110100 Fixed: 0110100, index: 1 Error Original: 11001100101 Fixed: 10001100101, index: 2 Error

No detectable

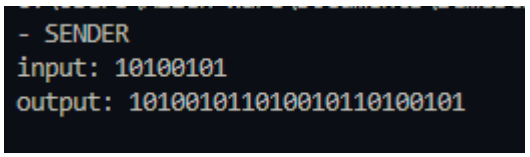
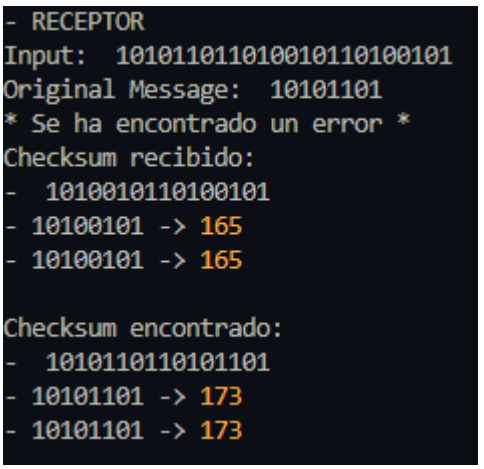
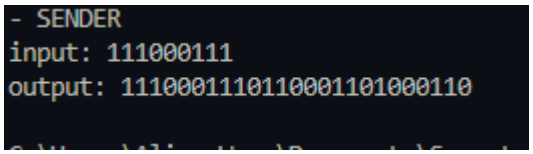
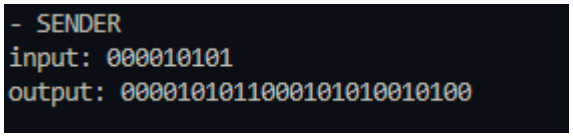
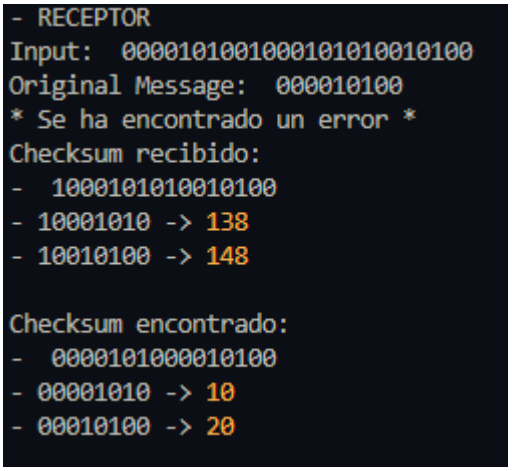
Python	Javascript
Original: 11001110101 Input: 10001100101 Output: 10001100101 No hay error	Original: 10001100101 Fixed: 10001100101, index: 0 OK

Algoritmo Fletcher Checksum

Sin errores

Emisor - Python	Receptor - Javascript
	
	
	
Emisor - Javascript	**Receptor - Python**
	
	
	

1 error:

Emisor - Python	Receptor - Javascript
	
	
	
Emisor - Javascript	**Receptor - Python**

```
Input:  10101101101001011010010 1
Original Message:  10101101
* Se ha encontrado un error *
Checksum recibido:
- 1010010110100101
- 10100101 -> 165
- 10100101 -> 165

Checksum encontrado:
- 1010110110101101
- 10101101 -> 173
- 10101101 -> 173
```

```
- SENDER
- input:  10100101
- output:  10100101101001011010010 1
```

```
Input:  11100011111100011010001 10
Original Message:  111000111
* Se ha encontrado un error *
Checksum recibido:
- 1110001101000110
- 11100011 -> 227
- 01000110 -> 70

Checksum encontrado:
- 0110001101000110
- 01100011 -> 99
- 01000110 -> 70
```

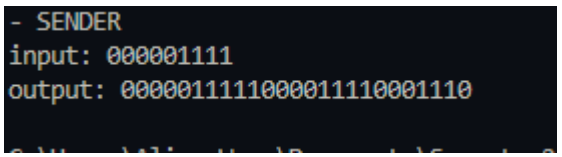
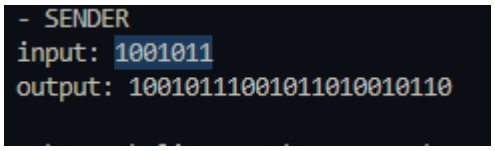
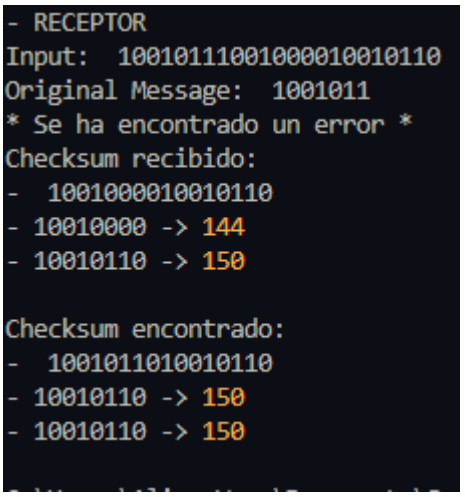
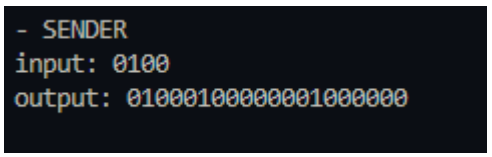
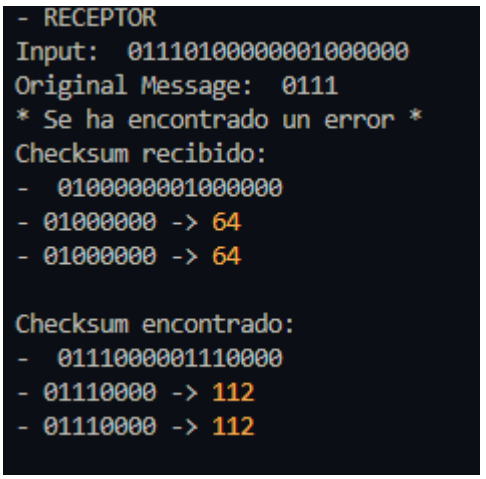
```
- SENDER
- input:  111000111
- output:  11100011101100011010001 10
```

```
Input:  00001010010001010100101 00
Original Message:  000010100
* Se ha encontrado un error *
Checksum recibido:
- 1000101010010100
- 10001010 -> 138
- 10010100 -> 148

Checksum encontrado:
- 0000101000010100
- 00001010 -> 10
- 00010100 -> 20
```

```
- SENDER
- input:  000010101
- output:  00001010110001010100101 00
```

2 Errores

Emisor - Python	Receptor - Javascript
	
	
	
Emisor - Javascript	Receptor - Python

Input: 00000001110000111110001110 Original Message: 00000011 * Se ha encontrado un error * Checksum recibido: - 1000011110001110 - 10000111 -> 135 - 10001110 -> 142 Checksum encontrado: - 1000000110000010 - 10000001 -> 129 - 10000010 -> 130	- SENDER - input: 000001111 - output: 000001111100001111001110
Input: 10010111001000010010110 Original Message: 1001011 * Se ha encontrado un error * Checksum recibido: - 1001000010010110 - 10010000 -> 144 - 10010110 -> 150 Checksum encontrado: - 1001011010010110 - 10010110 -> 150 - 10010110 -> 150	- SENDER - input: 1001011 - output: 10010111001011010010110
Input: 01110100000001000000 Original Message: 0111 * Se ha encontrado un error * Checksum recibido: - 0100000001000000 - 01000000 -> 64 - 01000000 -> 64 Checksum encontrado: - 0111000001110000 - 01110000 -> 112 - 01110000 -> 112	- SENDER - input: 0100 - output: 01000100000001000000

No detectable

Emisor - Javascript	Receptor - Python
- SENDER - input: 00000000 - output: 000000000000000000000000	Input: 10000000010000000000000000 Original Message: 100000000 * Sin errores *

Algoritmo CRC-32

Sin errores

Emisor - Python

```
Mensaje original: 110101
CRC calculado: 10000100101100010010101110101110
```

Receptor - Javascript

```
Emisor envía: 11010101000000111100010010110110110111
Mensaje original: 110101
CRC recibido: 01000000111100010010110110110111
CRC calculado: 01000000111100010010110110110111
No se detectaron errores.
```

```
Mensaje original: 101010
CRC calculado: 00001001101110010010011001011011
```

```
Emisor envía: 10101010101010111100110000100010000010
Mensaje original: 101010
CRC recibido: 10101010111100110000100010000010
CRC calculado: 10101010111100110000100010000010
No se detectaron errores.
```

```
Mensaje original: 11110000
CRC calculado: 01101111101111110001110110010001
```

```
Emisor envía: 11110000010111001110100101111110000111100
Mensaje original: 11110000
CRC recibido: 01011100111010010111110000111100
CRC calculado: 01011100111010010111110000111100
No se detectaron errores.
```

Emisor - Javascript

```
Mensaje original: 110101
CRC calculado: 01000000111100010010110110110111
```

Receptor - Python

```
Emisor envía: 11010110000100101100010010101110101110
Mensaje original: 110101
CRC recibido: 10000100101100010010101110101110
CRC calculado: 10000100101100010010101110101110
No se detectaron errores.
Mensaje original: 101010
CRC calculado: 00001001101110010010011001011011
```

```
Mensaje original: 101010
CRC calculado: 10101010111100110000100010000010
```

```
CRC recibido: 00001001101110010010011001011011
CRC calculado: 00001001101110010010011001011011
No se detectaron errores.
Mensaje original: 11110000
CRC calculado: 01101111101111110001110110010001
```

```
Mensaje original: 11110000
CRC calculado: 01011100111010010111110000111100
```

```
Emisor envía: 11110000011011110111111000111011001000 1
Mensaje original: 11110000
CRC recibido: 01101111101111110001110110010001
CRC calculado: 01101111101111110001110110010001
```

1 error:

Emisor - Python	Receptor - Javascript
Emisor envía: 0101011000010010110001001010111010110 Mensaje original: 010101	Mensaje original: 010101 CRC recibido: 0100000011110001001010110110111 CRC calculado: 1000101110101101111111000010010 Se detectó un error en el bit 1: la trama se descarta
Emisor envía: 0010100000100110111001001001100101101 1 Mensaje original: 001010	Mensaje original: 001010 CRC recibido: 1010101011110011000010010000010 CRC calculado: 0110000110101111110110100100111 Se detectó un error en el bit 1: la trama se descarta
Emisor envía: 0111000001101111101111110001110110010001 Mensaje original: 01110000	Mensaje original: 01110000 CRC recibido: 0101110011101001011110000111100 CRC calculado: 1001000001000011011111001010010 Se detectó un error en el bit 1: la trama se descarta
Emisor - Javascript	**Receptor - Python**
Emisor envía: 0101010100000011110001001011011011 0111 Mensaje original: 010101	Mensaje original: 010101 CRC recibido: 1000010010110001001010111010111 0 CRC calculado: 1011111111101111110000101101100110 Se detectó un error en el bit 1: la trama se descarta
Emisor envía: 0010101010101011110011000010001000001 0 Mensaje original: 001010	Mensaje original: 001010 CRC recibido: 0000100110111001001001100101101 1 CRC calculado: 0011001011010111000011010010011 Se detectó un error en el bit 1: la trama se descarta
Emisor envía: 0111000001011100111010010111110000111100 Mensaje original: 01110000	Mensaje original: 01110000 CRC recibido: 0110111110111111000111011001 0001 CRC calculado: 1000001000000111100111101011 0001 Se detectó un error en el bit 1: la trama se descarta

2 Errores

Emisor - Python	Receptor - Javascript
Emisor envía: 0101011000010010110001001010111010110 Mensaje original: 010101	Mensaje original: 010101 CRC recibido: 0100000011110001001010110110111 CRC calculado: 1000101110101101111111110000010010 Se detectó un error en el bit 1: la trama se descarta
Emisor envía: 0010100000100110111001001001100101 1011 Mensaje original: 001010	Mensaje original: 001010 CRC recibido: 1010101011110011000010010000010 CRC calculado: 0110000110101111110110100100111 Se detectó un error en el bit 1: la trama se descarta
Emisor envía: 0111000001101111101111110001110110010001 Mensaje original: 01110000	Mensaje original: 01110000 CRC recibido: 0101110011101001011110000111100 CRC calculado: 1001000001000011011111010100010 Se detectó un error en el bit 1: la trama se descarta
Emisor - Javascript	**Receptor - Python**
Emisor envía: 0101010100000011110001001011011011011 1 Mensaje original: 010101	Mensaje original: 010101 CRC recibido: 1000010010110001001010111010111 0 CRC calculado: 1011111111101111100001011011001 10 Se detectó un error en el bit 1: la trama se descarta
Emisor envía: 0010101010101011110011000010001000001 0 Mensaje original: 001010	Mensaje original: 001010 CRC recibido: 0000100110111001001001100101 1011 CRC calculado: 0011001011010111000001101001001 1 Se detectó un error en el bit 1: la trama se descarta
Emisor envía: 0111000001011100111010010111110000111100 Mensaje original: 01110000	Mensaje original: 01110000 CRC recibido: 0110111110111111100011101100100 01 CRC calculado: 1000001000000111100111101011000 1 Se detectó un error en el bit 1: la trama se descarta

No detectable

Emisor - Python	Receptor - Python
Mensaje original: 010101 CRC calculado: 1011111111011111000010110110011 0	Emisor envía: 0101011011111111011111000010110110011 0 Mensaje original: 010101 CRC recibido: 1011111111011111000010110110011 0 CRC calculado: 1011111111011111000010110110011 0 No se detectaron errores.
Emisor- Javascript	**Receptor- Javascript**
Mensaje original: 010101 CRC calculado: 1000101110101101111111110000010010	Emisor envía: 0101011000101110101101111111100001 0010 Mensaje original: 010101 CRC recibido: 1000101110101101111111110000010010 CRC calculado: 1000101110101101111111110000010010 No se detectaron errores.

Discusión

Hamming

El algoritmo detecta correctamente los errores y es sumamente efectivo para detectar errores individuales, de un bit, sin embargo ya al cambiar más de un bit falla al detectar o detecta solamente uno de los errores, o incluso detecta un bit no original que modifica el mensaje de modo que se vuelva una configuración correcta, sin ser la original.

Si hubo un caso en el que no fue posible detectar errores, esto se debe a que como funciona el Hamming si se modifican los suficientes valores para hacer un una configuración correcta a pesar de que no sea el mensaje original lo marca como un byte válido.

El algoritmo de Fletcher Checksum detectó correctamente errores y es efectivo para ello. Lo que lo diferencia de otros checksums, es la utilización de 2 checksums. Esto ayuda a que el orden en el que entran las tramas afecte también al resultado, por lo que ahora la cantidad de posibles combinaciones de las dos sumas sería un número muy alto para un blocksize de 8. Por lo tanto, es muy poco probable que no se vaya a poder detectar un error, no obstante, no es imposible.

El caso en donde el algoritmo no fue apto para detectar un error fue cuando se cambió tanto un bit de entrada, o mejor dicho se le agregó un bit que no estaba y uno de los bits que llevaba el checksum también cambió. Esto hizo que el algoritmo no detectara que hubo un error dado que el checksum ahora quedó casi inutil ya que fue cambiado. Aun así, como se dijo anteriormente, este evento es muy improbable por la cantidad de combinaciones posibles las cuales se pueden aumentar si se aumenta el blocksize a 16 o a 32 para tener un Fletcher-32 o Fletcher-64 respectivamente. No obstante, para

mensajes pequeños esto agrega mucha información adicional al payload en especial si se aumenta a un checksum combinado de 64 bits (Fletcher, J. G., 1982).

El algoritmo CRC-32 (Cyclic Redundancy Check) es un método empleado para detectar errores durante el proceso de transmisión de datos. Este algoritmo origina un valor de comprobación desde un conjunto de datos ingresados, que luego se emplea para confirmar la integridad de dichos datos tras su transmisión.

Para el caso de prueba donde la trama estaba intacta, se notó que la la comprobación CRC-32 llevada a cabo por el receptor coincide con el valor CRC proporcionado por el emisor, de igual manera para el caso de la trama con un error, pues en la mayoría de las circunstancias, si un único bit de error aparece en la trama de datos, el algoritmo CRC-32 tiene la capacidad para detectarlo. por lo que la comprobación CRC realizada por el receptor no concuerda con el valor CRC proporcionado por el emisor, señalando un error en la trama de datos, el algoritmo CRC-32; tiene la habilidad para detectar un elevado porcentaje de errores únicos de bit, así como errores de múltiples bits y errores en ráfaga.
Sin embargo, a pesar de la eficiencia del algoritmo CRC-32, hay escenarios en los que este no logra detectar los errores. Un ejemplo de esto es cuando los errores en la trama de datos resultan en una trama modificada que genera el mismo valor CRC que los datos originales. Esto resalta una debilidad del algoritmo CRC-32 y sirve de recordatorio de que, aunque es altamente efectivo en la detección de errores, no es infalible.

Comentario Grupal

En este laboratorio se abordó el tema fundamental de la comunicación por redes de manejo de errores en sistemas de

transmisión. Es importante tener en cuenta como funcionan todas estas tecnologías y algoritmos ya que en este mundo tan conectado es sumamente necesario utilizar conexiones y comunicación. Al conocer cómo funcionan los algoritmos de detección y corrección de errores puede dar una buena perspectiva y conocimiento sobre ciertos problemas que pueden ocurrir y cómo solucionarlos.

Conclusiones

El algoritmo de Fletcher Checksum, presenta una versatilidad indiscutible al tener 2 checksums en vez de uno. Esto logra que las combinaciones de números posibles entre las 2 checksums lleguen a cantidades sorprendentes, por lo que la posibilidad de no encontrar un error baja. Incluso, esto puede aumentarse al aumentar los tamaños de los bloques que se quieren analizar, sin embargo, esto causaría que haya más información adicional al payload lo cual generaría más carga de envío. En general, el Fletcher Checksum es un algoritmo útil para la detección de errores que pese a no ser perfecto, puede ser factible y confiable para ciertos casos.

En conclusión, el algoritmo CRC-32 es una herramienta inestimable para la detección de errores en la transmisión de datos. Posee un alto índice de éxito en la identificación de errores únicos de bit y errores en ráfaga, y en muchas situaciones puede detectar errores de múltiples bits. Sin embargo este no es perfecto ya que hay situaciones en las que puede fallar en la detección de errores

El algoritmo de Hamming demuestra ser un algoritmo robusto y completo para lograr identificar errores y corregirlos, este es un algoritmo bastante consistente al momento de corregir errores de bit individuales, sin embargo al momento de errores con múltiples bits invertidos las correcciones se vuelven mucho más complejas incluso pudiendo llegar a un punto en que se indique que la cadena es veraz, siendo en realidad una inversión de bits. Esta es complicada de

conseguir, pero no imposible. Sin embargo, este es un fuerte algoritmo y corrige errores de manera eficiente y sencilla

Citas y Referencias

- Yadav, C. (2020). Fletcher's Checksum. https://www.tutorialspoint.com/fletcher-s-checksum
-
- Fletcher, J. G. (1982). "An Arithmetic Checksum for Serial Transmissions". *IEEE Transactions on Communications*. COM-30 (1): 247–252.
-
- Stallings, W. (2010). Comunicaciones y Redes de Computadoras. (8ª ed.). Pearson.
-
- Tanenbaum, A. S. (2010). Computer Networks. (5ª ed.). Prentice Hall.