Universidad del Valle de Guatemala
Redes de Computadoras
MSc. Jorge Yass
Sección 20

Laboratorio 2

Esquemas de detección y corrección de errores

Diego Lemus, 21469
Fabián Juárez, 21440

Guatemala, 11 de julio 2024

PREPARACIÓN

Repositorio: https://github.com/TheDeloz-v2/RC-LAB2

Se definió que el emisor utilizase el lenguaje de programación Javascript y que el receptor utilizase Python. El emisor accede a un archivo de texto que contiene mensajes con bloques de longitud de 4 u 8 bits separados por un espacio en blanco; esto se hizo así para mantener una estructura consistente y entendible a primera vista. Se decidieron utilizar los algoritmos de Hamming y Fletcher.

Hamming

El código de Hamming (7,4) es un tipo de código de detección y corrección de errores utilizado en la transmisión y almacenamiento de datos. Este algoritmo se centra en detectar y corregir errores de bit único.

Principios básicos:

- Codificación: Se toma un bloque de 4 bits de datos y se codifica en un bloque de 7 bits agregando 3 bits de paridad.
- Bits de paridad: Estos bits se calculan de tal manera que cada bit de paridad supervisa un subconjunto específico de los bits de datos.
    - Paridad P1 supervisa los bits 1, 3, 5, 7
    - Paridad P2 supervisa los bits 2, 3, 6, 7
    - Paridad P3 supervisa los bits 4, 5, 6, 7

Fletcher-16

Este es un algoritmo de verificación de integridad que produce un checksum de 16 bits a partir de un conjunto de datos. Se utiliza para detectar errores en la transmisión de datos.

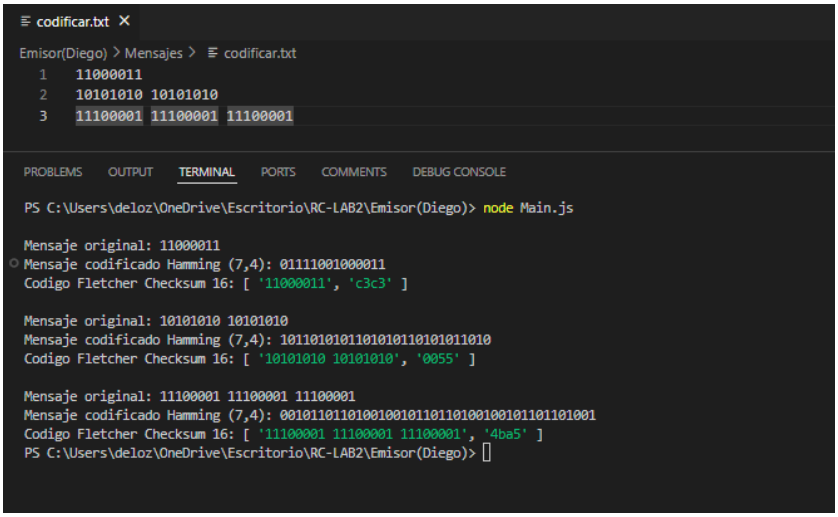
Principios básicos:

- Checksum: El algoritmo genera dos sumas de verificación de 8 bits, que se combinan para formar un checksum de 16 bits.
- Sumas parciales: Se calculan dos sumas, S1 y S2. S1 es la suma de todos los bytes de datos y S2 es la suma acumulativa de los valores de S1.

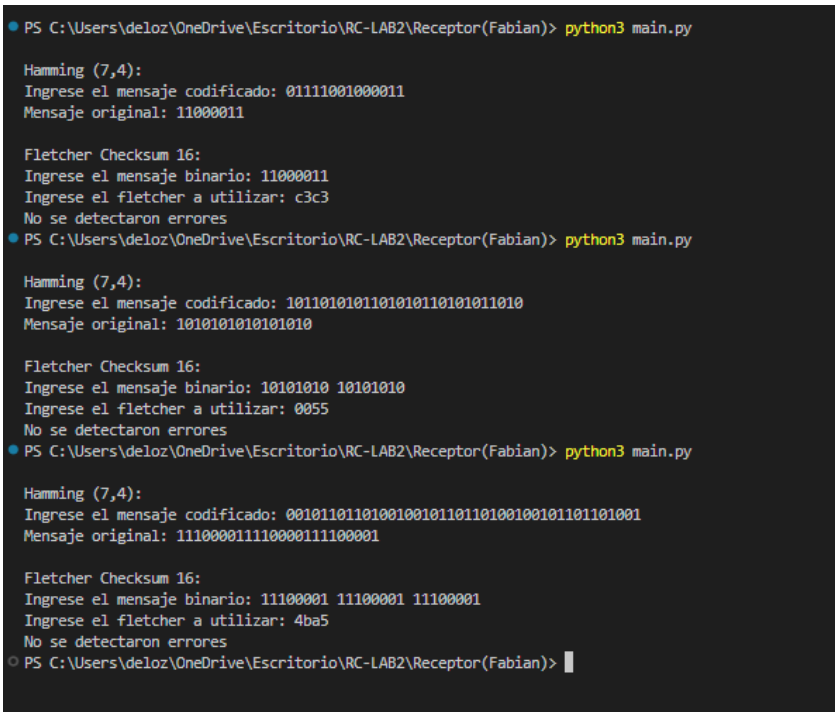
ESCENARIOS DE PRUEBA

- Prueba 1: Mensajes íntegros (ningún bit sufrió un cambio).

Emisor

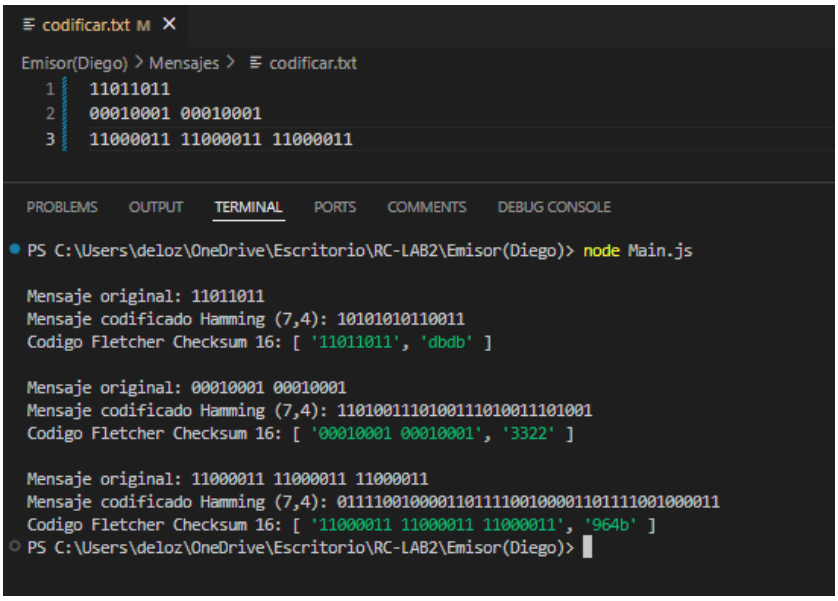


Receptor



- Prueba 2: Mensajes con un bit cambiado.

Emisor

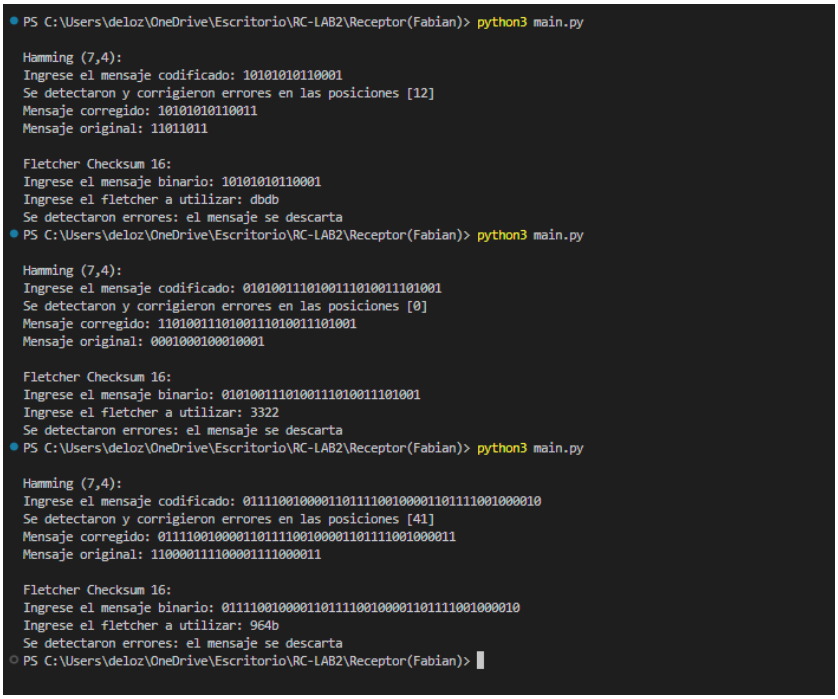


Receptor

11011011 : 10101010110001

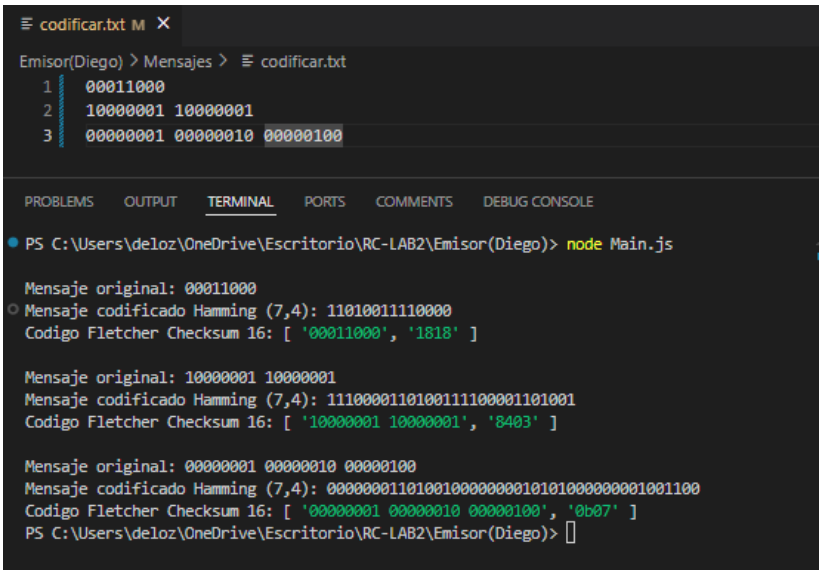
00010001 00010001 : 01010011101001110100111010010

11000011 11000011 11000011 : 011110010000110111100100001101111001000010



- Prueba 3: Mensajes con dos o más bits cambiados.

Emisor

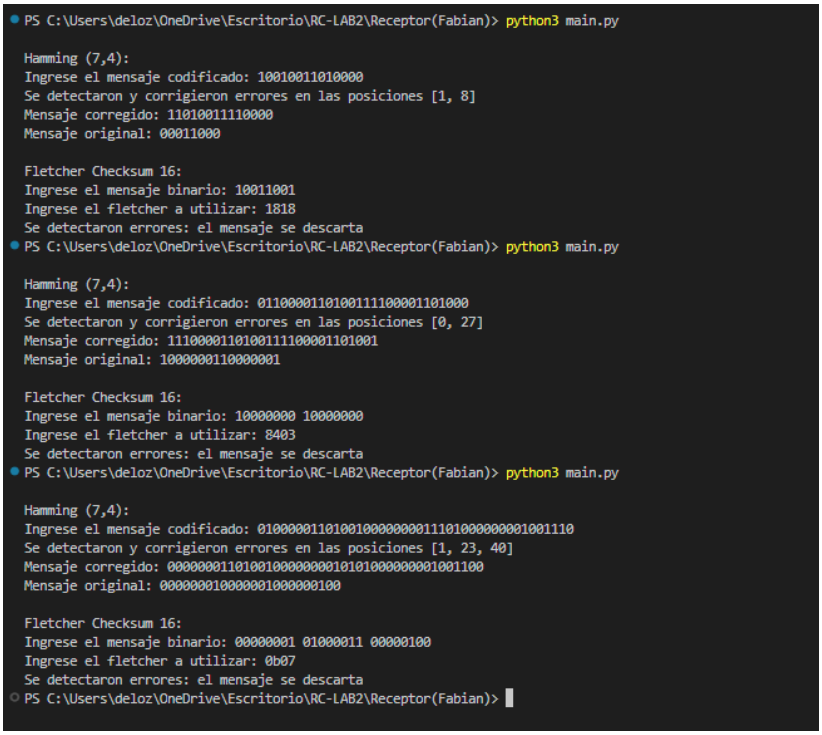


Receptor

00011000 : 10010011010000

10000001 10000001 : 01100001101001111100001101000

00000001 00000010 00000100 : 01000001101001000000001110100000000010011110



PREGUNTAS

1. **Primer Algoritmo (Hamming)**

    a. *¿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.*

    El algoritmo de Hamming está diseñado para detectar y corregir errores de un solo bit. Pero a pesar de ello, puede detectar pero no corregir errores en dos bits, y no puede detectar errores en tres bits o más. Por lo qué, sí es posible manipular los bits de tal manera que el algoritmo de Hamming no sea capaz de detectar el error, esto pasaría en el caso de poseer múltiples errores.

    El mensaje original ingresado fue 10101011 y al salir codificado por Hamming es: 10110100110011. Posteriormente modificamos 3 bits el primer, quinto y duodécimo bits y al visualizar el programa únicamente indica que se modificó el tercero y duodécimo bit por lo que aquí presenta que no es posible detectar el error que se metió en la entrada.

```
Mensaje original: 10101011
Mensaje codificado Hamming (7,4): 10110100110011
Codigo Fletcher Checksum 16: [ '10101011', 'abab' ]
PS C:\Users\domot\Desktop\UVG\UVG2024\SegundoSemestre\Redes\RC-L
"c:/Users/domot/Desktop/UVG/UVG2024/SegundoSemestre/Redes/RC-LAB

Hamming (7,4):
Ingrese el mensaje codificado: 00111100110001
Se detectaron y corrigieron errores en las posiciones [3, 12]
Mensaje corregido: 00101100110011
Mensaje original: 11101011
```

    b. *En base a las pruebas que realizó, ¿qué ventajas y desventajas posee cada algoritmo con respecto a los otros dos? Tome en cuenta complejidad, velocidad, redundancia (overhead), etc.*

    Ventajas:

    - Corrección de un solo bit: Puede detectar y corregir errores de un solo bit, lo que aumenta la confiabilidad de los datos transmitidos.

- Detección de dos bits: Puede detectar errores en dos bits, aunque no puede corregirlos.

- Baja redundancia: Añade un número relativamente pequeño de bits de paridad en comparación con otros algoritmos.

Desventajas:

- Limitado a errores simples: No puede corregir errores en más de un bit y no puede detectar errores en tres bits o más.

- Complejidad de implementación: Es más complejo de implementar que un simple código de paridad.

- Overhead relativo: Aunque bajo, la redundancia puede ser mayor que la de Fletcher checksum en algunos casos, especialmente cuando se requieren códigos más largos.

2. Segundo Algoritmo (Fletcher checksum)

a. *¿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.*

El algoritmo de Fletcher checksum utiliza dos sumas: la suma de los bytes de datos y la suma acumulativa de esas sumas parciales. Estas dos sumas se calculan de manera acumulativa, lo que hace que cualquier cambio en los datos afecte ambas sumas. Por lo tanto, para que un error no sea detectado, necesitaríamos encontrar una manipulación de los datos que no cambie ninguna de las dos sumas. Esto es extremadamente improbable debido a la manera en que las sumas se calculan y se acumulan.

b. *En base a las pruebas que realizó, ¿qué ventajas y desventajas posee cada algoritmo con respecto a los otros dos? Tome en cuenta complejidad, velocidad, redundancia (overhead), etc.*

Ventajas:

- Baja redundancia: Fletcher checksum, como el Fletcher-16, añade solo 16 bits de checksum, lo que genera poca sobrecarga.

- Velocidad: Es relativamente rápido de calcular y verificar debido a su simplicidad aritmética.

- Mejor detección de errores que simple paridad: Puede detectar ciertos errores que un simple checksum de paridad no detectaría, ya que considera tanto la suma de los valores de los bytes como la suma acumulativa.

Desventajas:

- Vulnerabilidad a manipulaciones específicas: Es susceptible a ciertos errores no detectables si los datos se manipulan de manera que las sumas no cambien.

- No corrige errores: Solo puede detectar errores, no corregirlos.

- Menor capacidad de detección de errores múltiples: Aunque mejor que un simple checksum, Fletcher checksum no es tan robusto en la detección de múltiples errores comparado con códigos más avanzados.