

**UNIVERSIDAD DEL VALLE DE GUATEMALA**

CC3067 - Redes

Sección 20

Ing. Jorge Yass



## Laboratorio No. 2

Javier Alejandro Ramírez Heredia - 21600

Mario Andres Cristales Cardona - 21631

**GUATEMALA, 24 de julio de 2024**

## Objetivos

En esta práctica, se implementaron dos algoritmos, uno para corrección de errores y otro para detección de errores. Los emisores se programaron en Python y los receptores en JavaScript. Los algoritmos utilizados fueron:

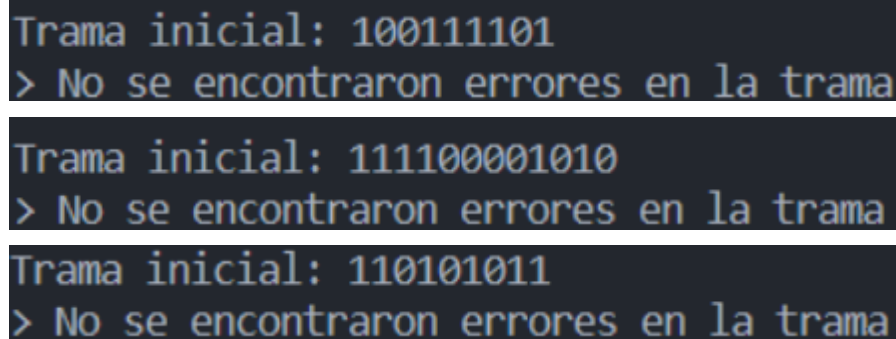
**CRC-32:** Este algoritmo de detección de errores utiliza la división polinomial de una secuencia de bits. En este caso, se empleó un polinomio de grado 32. El emisor divide el mensaje original por este polinomio y envía el resultado al receptor. El receptor realiza la misma operación y, según el residuo obtenido, detecta errores en el mensaje recibido.

**Hamming (7-4):** Este algoritmo de corrección de errores utiliza bits de paridad y potencias de 2. En la variante 7-4, el emisor empaqueta los mensajes en grupos de 4 bits, aplica el algoritmo de codificación y genera tramas de 7 bits. El receptor descompone la trama en sub-tramas de 7 bits para detectar y corregir errores.

## Pruebas

- **CRC-32**

- **Pruebas sin errores:**



```
Trama inicial: 100111101
> No se encontraron errores en la trama

Trama inicial: 111100001010
> No se encontraron errores en la trama

Trama inicial: 110101011
> No se encontraron errores en la trama
```

- Prueba 1:

- Trama = '100111101'
    - Polinomio = '10101'

- Prueba 2:

- Trama = '111100001010'
    - Polinomio = '10011'

- Prueba 3:

- Trama = '110101011'
    - Polinomio = '1001'

- **Pruebas con errores:**

```
Trama inicial: 11010110101011010000000000000000
> Se encontraron errores en la trama
> La trama se descarta
```

```
Trama inicial: 10100111011
> Se encontraron errores en la trama
> La trama se descarta
```

```
Trama inicial: 10100111011
> Se encontraron errores en la trama
> La trama se descarta
```

- Prueba 1:

- Trama = '11010110101011010000000000000000'
- Polinomio = '100000100110000010001110110110111'

- Prueba 2:

- Trama = '10100111011'
- Polinomio = '1101'

- Prueba 3:

- Trama = '10100111011'
- Polinomio = '1101'

- **Hamming**

- **Pruebas sin errores:**

```
Trama inicial: 0110011
> No se encontraron errores en la trama

Trama inicial: 1001100
> No se encontraron errores en la trama

Trama inicial: 1010010
> No se encontraron errores en la trama
```

- Prueba 1:

- Trama = '0110011'

- Prueba 2:

- Trama = '1001100'

- Prueba 3:

- Trama = '1010010'

- **Pruebas con errores:**

```
Trama inicial: 0100011
> Se encontraron errores en el bit: 5
Trama correcta: 0110011

Trama inicial: 1001110
> Se encontraron errores en el bit: 2
Trama correcta: 1001100

Trama inicial: 0010010
> Se encontraron errores en el bit: 7
Trama correcta: 1010010
```

- Prueba 1:
  - Trama = '0100011'
- Prueba 2:
  - Trama = '1001110'
- Prueba 3:
  - Trama = '0010010'

- **CRC-32**

- **Pruebas con error cambiando 2 bits:**

```
Trama inicial: 100101001
> Se encontraron errores en la trama
> La trama se descarta
```

```
Trama inicial: 110100001110
> Se encontraron errores en la trama
> La trama se descarta
```

- Prueba 1:
  - Trama = '100101001'
  - Polinomio = '10101'
- Prueba 2:
  - Trama = '110100001110'
  - Polinomio = '10011'

- **Hamming**

- **Pruebas con error cambiando 2 bits:**

```
Trama inicial: [ '0111011', '0100011' ]
> Se encontraron errores en el bit: 5
> Se encontraron errores en el bit: 8
> trama correcta: 0111010 0110011

Trama inicial: [ '1001000', '1001110' ]
> Se encontraron errores en el bit: 2
> Se encontraron errores en el bit: 13
> trama correcta: 1101000 1001100
```

- Prueba 1:
  - 0111011 0100011
- Prueba 2:
  - 1001000 1001110

- **CRC-32**

- **Pruebas sin error cambiando 2 bits:**

```
Trama inicial: 00100111011
> No se encontraron errores en la trama

Trama inicial: 101100001000
> No se encontraron errores en la trama
```

- Prueba 1:
  - Trama = 00100111000
  - Polinomio = '1101'
- Prueba 2:
  - Trama = 101100001000
  - Polinomio = '10101'

- **Hamming**
  - Pruebas sin error cambiando 2 bits:

```
Trama inicial: [ '1010010' ]
> No se detectaron errores en la trama

Trama inicial: [ '1100110' ]
> No se detectaron errores en la trama
```

- Prueba 1:
  - 1010010
- Prueba 2:
  - 1100110

## Discusión

### Análisis de Resultados y Algoritmos

#### Hamming (7-4)

- **Tramas Correctas:** Las tramas '0110011', '1001100', y '1010010' fueron correctamente identificadas como sin errores mediante la función “process\_hamming”, que verifica bits de paridad en las posiciones 0, 1 y 2.
- **Tramas con Error:** Las tramas '0100011', '1001110', y '0010010' fueron correctamente identificadas con errores. Por ejemplo, en '0100011', el bit de paridad en la posición 0 debería ser 1 para mantener la paridad par, pero hay tres bits 1, indicando un error.

#### CRC-32

- **Tramas Correctas:** Las tramas '111100001010' con polinomio '10011', '100111101' con polinomio '10101', y '110101011' con polinomio '1001' se procesaron correctamente. La función “process\_trama” realiza la conversión a bits y el cálculo del CRC, asegurando la exactitud de las tramas.
- **Tramas con Error:** Tramas como '10100111011' con polinomio '1101', y '11010110101011010000000000000000' con polinomio '100000100110000010001110110110111' fueron identificadas con errores mediante el cálculo del CRC, que resultó en valores no nulos, indicando la presencia de errores.

## Casos donde no se detectaron errores

### CRC-32

- **Prueba 1:** La trama '00100111000' con polinomio '1101' no detectó errores. El CRC resultante fue [0, 0, 0, 1, 0, 1, 1, 1], indicando falsamente que no había errores.
- **Prueba 2:** La trama '101100001000' con polinomio '10101' también pasó sin errores, con un CRC resultante de [0, 0, 0, 0, 0, 0, 0, 0].

### Hamming (7-4)

- El algoritmo Hamming (7-4) solo detecta y corrige un error por trama de 7 bits. Si hay múltiples errores o si ocurren en posiciones específicas, el algoritmo puede fallar en detectar o corregir los errores.