UNIVERSIDAD DEL VALLE DE GUATEMALA

Redes, sección 20

LAB # 2

Diego Linares - 221256
Joaquin Campos – 22155

Guatemala, Julio 2025

Mensajes usados (cadenas o tramas)

001101

10111000111

1100110

Corrección

Para correccion de errores se decidio implementar Hamming

Para esta parte se implementaron dos programas: un emisor en Java y un receptor en Python, cumpliendo el requisito de utilizar lenguajes distintos para cada lado.

Emisor (Java)

- Solicita un mensaje binario de 7 bits desde la consola.
- Coloca los bits de datos en las posiciones correspondientes de una trama de 11 bits.
- Calcula los 4 bits de paridad (P1, P2, P4, P8) mediante XOR. El XOR creo que no necesito explicarlo de por qué pero es porque por ejemplo si hubiesen una cantidad par de 1`s pues entonces bit de paridad seria 0, si hubiese una cantidad impar pues bit de paridad 1.
- Imprime la trama codificada completa de 11 bits.

Receptor (Python)

- Solicita una trama binaria de 11 bits desde la consola.
- Convierte la cadena ingresada a una lista de enteros para facilitar el procesamiento.
- Calcula el síndrome sumando las paridades y determinando si existe error y en qué posición.
- Si el síndrome es distinto de 0, corrige el bit correspondiente y muestra la trama corregida y los datos originales.

Funcionamiento del algoritmo

El código Hamming (11,7) consiste en:

- Tomar 7 bits de datos originales y agregar 4 bits de paridad.

- Colocar los bits de paridad en las posiciones potencias de 2 (1, 2, 4, 8) y los datos en las demás.
- Calcular cada bit de paridad para cubrir distintas combinaciones de posiciones.
- En el receptor, recalcular las paridades sobre la trama recibida para obtener el síndrome, un número que indica exactamente la posición del bit erróneo (si hay uno).
- Si el síndrome es 0, la trama está correcta; si no, se corrige el bit en esa posición.

Prueba realizada primero el de Java.

Posición	1	2	3	4	5	6	7	8	9	10	11
Tipo	P1	P2	D1	P4	D2	D3	D4	P8	D5	D6	D7

P = bit de paridad
D = bit de datos

Mensaje original:

1 1 0 0 1 1 0

Lo colocamos en las posiciones D1, D2, ..., D7:

Posición	1	2	3	4	5	6	7	8	9	10	11
Valor	P	P	1	P	1	0	0	P	1	1	0

Calcular paridades

Las fórmulas que usa el código Java con lo de los XOR que habia mencionado yo anteriormente.

P1 (posición 1): XOR de posiciones 3,5,7,9,11

P1 = D1 ^ D2 ^ D4 ^ D5 ^ D7

= 1 ^ 1 ^ 0 ^ 1 ^ 0

= 1

Y asi.... Una vez que terminamos de calcular todas las posiciones de de las posiciones de paridad pues

Con los bits de paridad calculados (P1=1, P2=0, P4=1, P8=0) la trama queda así:

Posición	1	2	3	4	5	6	7	8	9	10	11
Valor	1	0	1	1	1	0	0	0	1	1	0

Ahora para python...

Tomamos la convertida y le cambiamos el bit de la posicion 11 para ver si lo identifica.

1	2	3	4	5	6	7	8	9	10	11
1	0	1	1	1	0	0	0	1	1	1

Este receptor calcula las paridades y pues devuelve Síndrome: 1011 (binario) = 11 (decimal)

El receptor hace un flip al bit 11: 1 → 0. esto lo hace con otro xor al 1.
La trama corregida vuelve a ser la que envió el emisor:

10111000110

Y ya de ahi pues extraemos las posiciones de los datos

Posición	**3**	**5**	**6**	**7**	**9**	**10**	**11**
Valor	1	1	0	0	1	1	0

1100110

Y coincide con el mensaje original.

Versión 2 de corrección de errores:

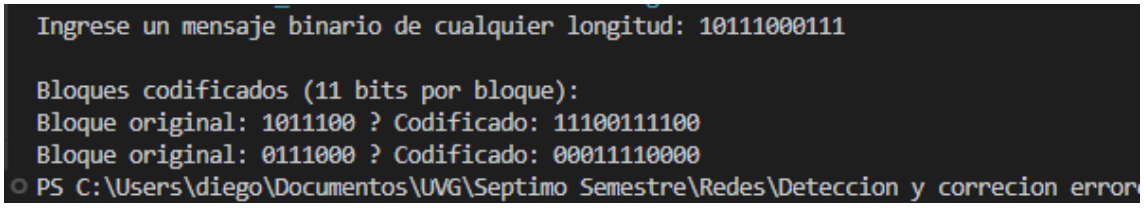
Para extender la funcionalidad del código Hamming (11,7) y cumplir con el requisito de aceptar cadenas de cualquier longitud, se modificó el emisor para que dividiera el mensaje binario ingresado en bloques de 7 bits. Si el mensaje ingresado tenía una longitud menor a 7, se rellenaba con ceros a la derecha; si era mayor, se dividía en múltiples bloques de 7 bits y se rellenaba el último si era necesario. Cada bloque se codifica por separado utilizando Hamming (11,7), generando una trama de 11 bits por bloque. En el receptor, se incorporó un mecanismo para procesar múltiples

bloques codificados. Se solicita al usuario cuántos bloques desea ingresar y se analiza cada uno de manera individual, detectando y corrigiendo errores si es necesario. Luego, se extraen los 7 bits de datos originales de cada bloque corregido y se reconstruye el mensaje completo concatenando todos los datos. Esta adaptación permite que el sistema funcione con mensajes de cualquier longitud, sin perder la capacidad de detección y corrección por bloque, lo cual se refleja claramente en los resultados mostrados bloque a bloque, seguidos del mensaje completo reconstruido.

Entonces ajaa ya ahora podríamos aceptar cadenas más largas o cortas para la prueba así que vamos con...

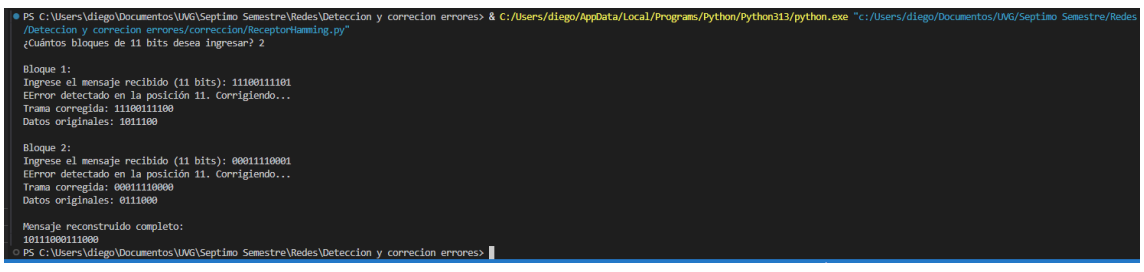
10111000111

El emisor.



El receptor:

Ahora con un error:

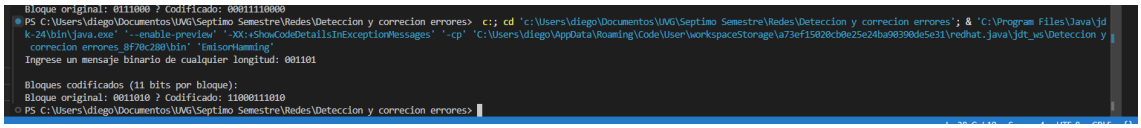


Notar que cambie solo el ultimo bit a cada cadena y pues funciono, lo detecta y lo corrige.

Ahora dos errores y vamos a pribar una cadena de longitud menor a 7

001101

Emisor:

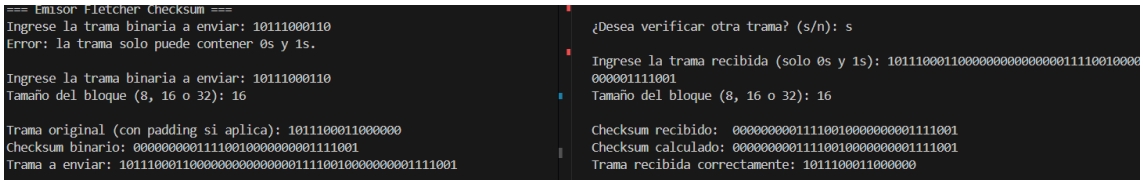


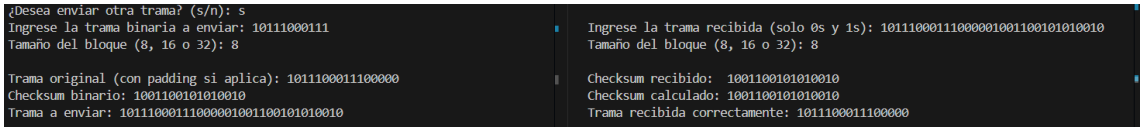
Receptor:

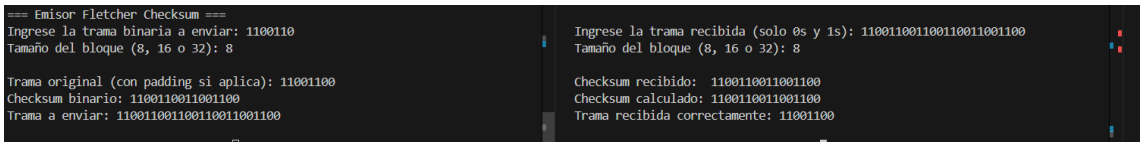
Notar que acá ya no es capaz de detectar dos errores porque Hamming no está hecho para ello.

Pruebas de detección:

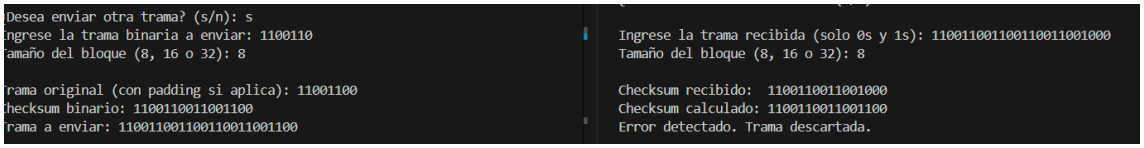
- Sin errores:

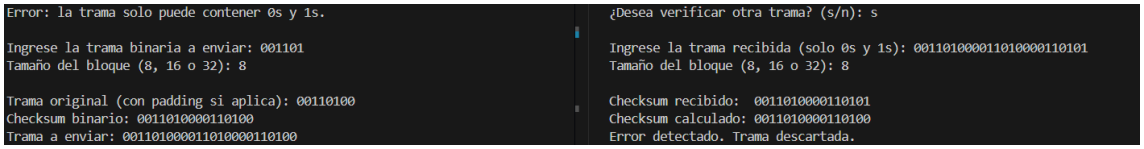


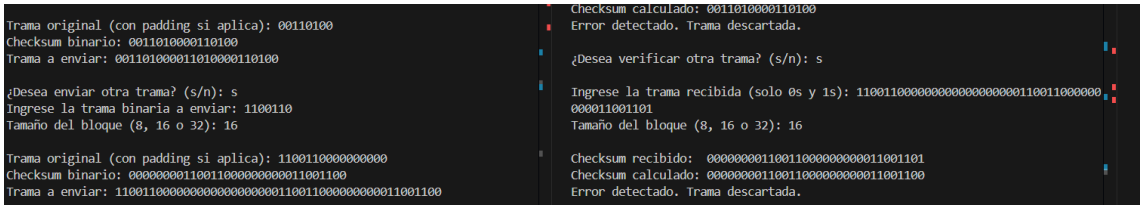




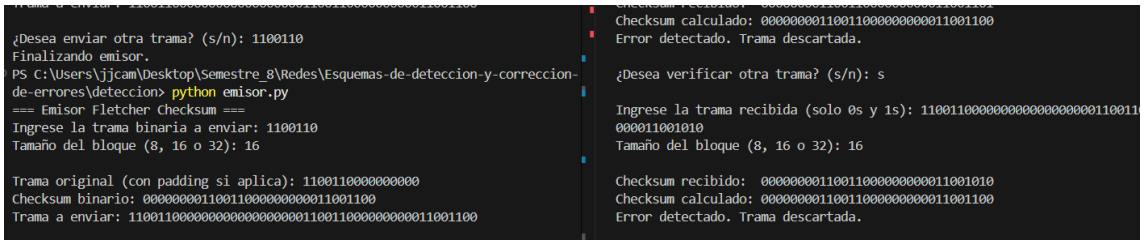
- Un error:

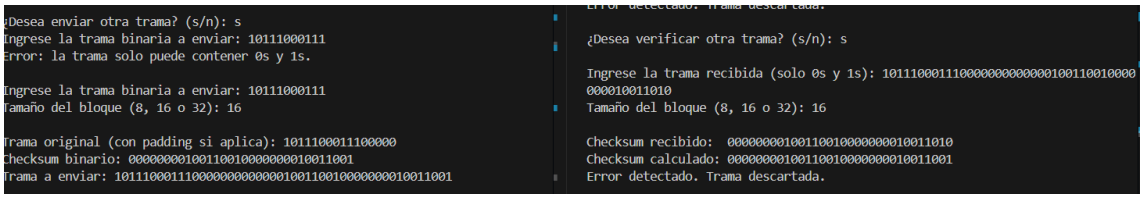


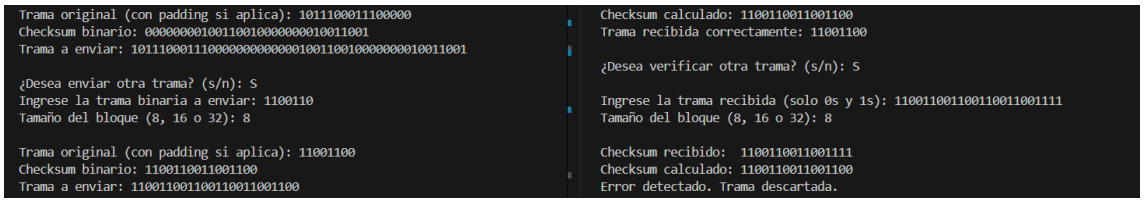




- Dos+errores:







- ¿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?

Sí, es posible manipular los bits de una trama de tal forma que el algoritmo Fletcher Checksum no detecte el error.

Esto se debe a que Fletcher calcula la validez de una trama a partir de dos sumas acumulativas:

sum1: la suma de todos los bloques de datos.

sum2: la suma acumulada de sum1 a lo largo de la secuencia.

- ¿qué ventajas y desventajas posee cada algoritmo con respecto a los otros dos?

Ventajas del algoritmo Fletcher Checksum

Mejor detección que la paridad o suma simple

Fácil de implementar: Solo requiere sumas y operaciones módulo.

Flexible: Permite usar bloques de 8, 16 o 32 bits según se necesite.

Desventajas del algoritmo

No corrige errores, solo los detecta.

Puede fallar si los errores mantienen las sumas sum1 y sum2

Más overhead que la paridad (agrega dos bloques de checksum).

Sensible al orden de los bloques, lo que complica su análisis en ciertos casos.