Laboratorio 2 – Parte 1
Esquemas de Detección y Corrección de Errores

Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 - Redes
Autores: Diederich Solis - 22952

Jorge Lopez - Carné2
Fecha: 24 de julio de 2025

Descripción del laboratorio

En este laboratorio se implementó un esquema de **corrección de errores** basado en el algoritmo de **Hamming**. El objetivo fue codificar tramas de datos binarios para detectar y corregir errores durante la transmisión, utilizando distintos lenguajes de programación para el emisor y el receptor.

Algoritmo de Corrección: Código de Hamming

Se utilizó el algoritmo de Hamming para (n, m) que cumple $m + r + 1 \leq 2^r$, donde:

- m: número de bits de datos.
- r: número de bits de paridad necesarios.
- $n = m + r$: longitud de la trama codificada.

Los bits de paridad se colocan en las posiciones que son potencias de 2 (1, 2, 4, 8, etc.) y se calculan de forma que cada uno verifica una combinación específica de bits, permitiendo detectar la posición del error.

Lenguajes utilizados

- Emisor: **C++**
- Receptor: **Python**

Pruebas realizadas

Escenarios sin errores

Trama original	Trama codificada	Resultado receptor
1011	0110011	Trama válida
1001	0011001	Trama válida
0110	1100110	Trama válida

Cuadro 1: Pruebas sin errores

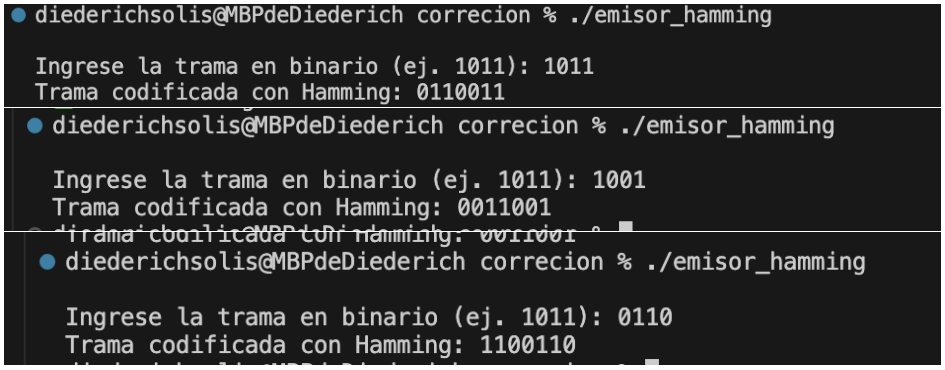

Figura 1: Evidencia de pruebas sin errores

Escenarios con un error

Original	Codificada	Bit alterado	Resultado receptor
1011	0110011	bit 4 $\rightarrow$ 1	Error en posición 4, corregido
1001	0011001	bit 7 $\rightarrow$ 0	Error en posición 7, corregido
0110	1100110	bit 2 $\rightarrow$ 0	Error en posición 2, corregido

Cuadro 2: Pruebas con un error

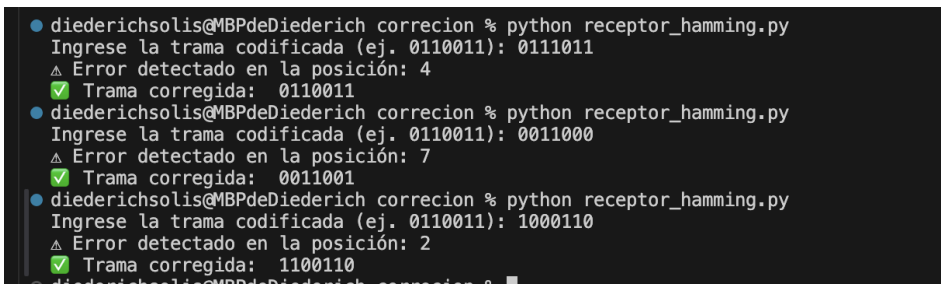

Figura 2: Evidencia de pruebas con un solo error

Escenarios con dos errores

Trama original	Codificada	Bits alterados	Resultado receptor
1011	0110011	bits 2, 5	El receptor detectó un error en la posición 7 y corrigió la trama, pero el resultado no coincidió con la original. Corrección incorrecta.
1001	0011001	bits 1, 6	El receptor detectó un error en la posición 7 y corrigió la trama, pero la salida no coincidió con la trama original. Trama corregida erróneamente.
0110	1100110	bits 4, 7	El receptor detectó error en la posición 3 y corrigió, pero la trama resultante fue inválida. Ejemplo de síndrome falso.

Cuadro 3: Pruebas con dos errores

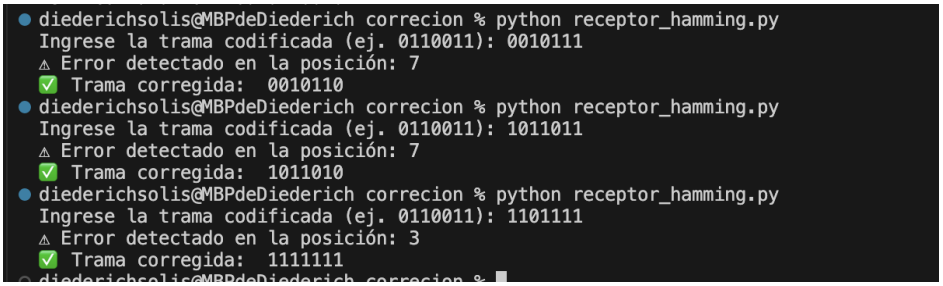

Figura 3: Evidencia de pruebas con dos errores y corrección errónea

Nota sobre errores múltiples: En todos los casos presentados con dos errores, el receptor detectó un error y aplicó una corrección. Sin embargo, el resultado final no coincidió con la trama original, demostrando una debilidad del algoritmo de Hamming estándar. Este algoritmo solo garantiza la detección y corrección de **un error** de bit. Cuando hay más de un error, el síndrome calculado puede coincidir con una posición válida, pero incorrecta, lo que lleva a una corrección errónea que no puede ser detectada automáticamente.

Análisis del algoritmo

Ventajas:

- Permite detectar y corregir errores de 1 bit en cualquier posición.
- Tiene poca redundancia (solo r bits adicionales).
- Es fácil de implementar y rápido.

Desventajas:

- No puede detectar o corregir todos los errores múltiples.
- Puede devolver una corrección incorrecta si hay más de un error (síndrome falso).

¿Se puede manipular la trama para que pase desapercibido el error?
Sí. Si dos errores afectan los bits de tal forma que el síndrome resultante corresponde a una posición válida, el sistema corregirá mal sin saberlo. Esta es una limitación conocida del código de Hamming.

Conclusiones

El código de Hamming resulta eficiente para tramas pequeñas y transmisión confiable donde los errores únicos son más probables. Su implementación en C++ y Python permite su uso en sistemas heterogéneos. No obstante, su confiabilidad disminuye cuando se producen errores múltiples.