Laboratorio 2 – Parte 1

Santiago Pereira – 22318

Nancy Mazariegos – 22513

Ejemplo de implementación de algoritmo de Hamming:

- Trama: 0110011100110010
    - Algoritmo: Código de Hamming Extendido (16,11) — Paridad par.

Emisor: implementado en C++	**Receptor: implementado en Python**
Input: mensaje a enviar: 10110011001	Input: trama generada por el emisor: 0110011100110010
Se insertan bits de paridad en posiciones de potencia de 2: p1, p2, p4, p8	Se leen los bits de paridad: p1, p2, p4, p8 y el bit de paridad general (posición 0)
Se calcula un bit de paridad global al inicio para extender la capacidad a detección de 2 errores	Se calcula el síndrome usando los bits de paridad recibidos y recalculados
Output: trama con paridad extendida: 0110011100110010	Si síndrome ≠ 0 y bit global correcto: se corrige 1 bit Si síndrome ≠ 0 y bit global incorrecto: se detectan 2 errores → trama descartada

ESCENARIOS DE PRUEBA Y RESULTADOS

CASO 1: SIN ERRORES

Prueba 1.1

- Mensaje original: 10110101001
- Trama generada: 1110011101010011
- Bits modificados: Ninguno
- Resultado del receptor:

Trama sin errores detectados.
Mensaje original extraído: 10110101001

Prueba 1.2

- Mensaje original: 11111111111
- Trama generada: 1111111111111111
- Bits modificados: Ninguno
- Resultado del receptor:

Trama sin errores detectados.
Mensaje original extraído: 11111111111

Prueba 1.3

- Mensaje original: 00000000000
- Trama generada: 0000000000000000
- Bits modificados: Ninguno
- Resultado del receptor:

Trama sin errores detectados.
Mensaje original extraído: 00000000000

CASO 2: UN ERROR

Prueba 2.1

- Mensaje original: 10110101001
- Trama generada: 1110011101010011
- Trama modificada: 1100011101010011
- Bits modificados: Posición 3 (1→0)
- Resultado del receptor:
  Error de 1 bit detectado en posición 3. Corrigiendo...
  Mensaje original extraído: 10110101001

Prueba 2.2

- Mensaje original: 11111111111
- Trama generada: 1111111111111111
- Trama modificada: 1111011111111111
- Bits modificados: Posición 5 (1→0)
- Resultado del receptor:

Error de 1 bit detectado en posición 5. Corrigiendo...
Mensaje original extraído: 11111111111

Prueba 2.3

- Mensaje original: 00000000000
- Trama generada: 0000000000000000
- Trama modificada: 0000010000000000
- Bits modificados: Posición 6 (0→1)
- Resultado del receptor:

Error de 1 bit detectado en posición 6. Corrigiendo...
Mensaje original extraído: 00000000000

CASO 3: MÚLTIPLES ERRORES

Prueba 3.1

- Mensaje original: 10110101001
- Trama generada: 1110011101010011
- Trama modificada: 0110011101010010
- Bits modificados: Posición 1 (1→0), Posición 16 (1→0)
- Resultado del receptor:

Se detectaron 2 errores. Trama descartada.

Prueba 3.2

- Mensaje original: 11111111111
- Trama generada: 1111111111111111
- Trama modificada: 1011111111111101
- Bits modificados: Posición 2 (1→0), Posición 15 (1→0)
- Resultado del receptor:

Se detectaron 2 errores. Trama descartada.

Prueba 3.3

- Mensaje original: 00000000000
- Trama generada: 0000000000000000
- Trama modificada: 0001000000010000
- Bits modificados: Posición 4 (0→1), Posición 11 (0→1)
- Resultado del receptor:

Se detectaron 2 errores. Trama descartada.

Preguntas:

- **¿Es posible manipular los bits de tal forma que el algoritmo seleccionado no sea capaz de detectar el error?** *Sí, sí es posible manipular los bits de una trama para que el algoritmo de Hamming no detecte el error, pero esto solo ocurre cuando se cambian dos bits al mismo tiempo.* **¿Por qué sí o por qué no?** *Si se cambian dos bits de datos, los bits de paridad pueden coincidir de forma que el error parezca invisible*

o que apunten a una posición equivocada. Esto se debe a que el algoritmo calcula un "síndrome" que suma posiciones donde hay inconsistencias, y con dos errores, ese síndrome puede confundirse con el de un solo error diferente. **En caso afirmativo, demuéstrelo con su implementación.** *Si suponemos que la trama "correcta" que se generó el emisor fue: 0110011100110010, luego la cambiamos para que genere algún error: 0100111100110010 (aquí cambiamos la posición 3 y 5 para que de error). Lo que podría suceder al ingresar esto, tenemos dos opciones:*

- *Detectar solo 1 error incorrectamente,*
- *Corregir el bit equivocado, o simplemente decir que no hay error (si el síndrome da 0 por casualidad)*

Pero con el bit de paridad extendido (el número total de unos cambia), sí se detecta que hubo más de un error, y por eso nuestra implementación dice: "Se detectaron 2 errores. Trama descartada."

Conclusiones:

- El código de Hamming permite detectar y corregir errores de 1 bit de forma confiable.
- Al extenderlo con un bit de paridad general, también se pueden detectar errores de 2 bits, aunque no corregirlos.
- El sistema es útil para mejorar la confiabilidad de las transmisiones de datos, evitando que se usen datos dañados.
- La práctica ayudó a entender cómo funciona la detección y corrección de errores a nivel de bits, usando lógica binaria y posiciones de potencia de 2.

Link al repositorio:

https://github.com/nancygmm/lab2-redes.git