UNIVERSIDAD DEL VALLE DE GUATEMALA

Redes

Sección 20

“Laboratorio 2 Parte 1”

ESTEBAN ALDANA GUERRA 20591
JESSICA PAMELA ORTIZ IXCOT 20192

GUATEMALA, Agosto 2023

Descripción de la práctica

El Laboratorio consiste en implementar y probar algoritmos de detección y corrección de errores en la transmisión de datos. Se trabajó en parejas de estudiantes, donde cada uno implementó un algoritmo distinto, uno de detección y otro de corrección de errores.

Los objetivos principales del laboratorio son:

- Comprender en profundidad algoritmos de detección y corrección de errores
- Implementar y probar estas técnicas con diversos lenguajes de programación
- Evaluar y comparar la efectividad de cada algoritmo
- Aplicar conceptos clave de Redes de Computadoras

Para poder realizar la parte de detección de errores se decidió usar el algoritmo CRC-32

El algoritmo CRC (Cyclic Redundancy Check) es un método popular y potente para detectar errores en la transmisión de datos. Está diseñado para detectar errores comunes, como los errores de bits individuales y los errores de ráfagas

De este mismo sale una sub rama que es el CRC-32 que es una variante específica que genera una suma de comprobación de 32 bits. Este se basa en la división polinomial, donde el mensaje original se divide por un polinomio generador de 32 bits. El resto de esta división (de 32 bits) se apena al mensaje original y se transmite junto con los datos.

Al igual que suele utilizarse en protocolos como Ethernet, PNG, gzip, Bzip2, PDF, MPEG-2, etc.

El código Hamming(7,4) es un esquema de corrección de errores que agrega 3 bits de paridad a 4 bits de datos, creando un mensaje de 7 bits. Los bits de paridad se calculan en función de los bits de datos y se utilizan para detectar y corregir un solo error de bit en el mensaje transmitido.

Los bits de paridad son calculados a partir de diferentes combinaciones de los bits de datos: Paridad 1 se calcula a partir de los bits en las posiciones 1, 3, 5, 7; Paridad 2 a partir de las posiciones 2, 3, 6, 7; y Paridad 3 a partir de las posiciones 4, 5, 6, 7. La estructura final del mensaje codificado es [P1, P2, D1, P3, D2, D3, D4].

El código Hamming(7,4) es eficaz para corregir errores únicos en la transmisión de datos, pero tiene limitaciones en cuanto a la detección de múltiples errores y la eficiencia del ancho de banda, ya que aumenta el tamaño del mensaje en un 75%.

Resultados

Algoritmo de detección de errores:

Trama 1:

Mensaje: "Hola Mundo"

01001000 01101111 01101100 01100001 00100000 01001101 01110101 01101110 01100100 01101111

Mensaje devuelto por el emisor

01001000 01101111 01101100 01100001 00100000 01001101 01110101 01101110 01100100 01101111af4a2d48

Respuesta del Receptor

```
Ingrese el texto a enviar: 01001000 01101111 01101100 01100001 00100000 01001101 01110101 01101110 01100100 011011
Texto con CRC guardado en output.txt
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1> node crc32_receptor.js
Trama recibida: 01001000 01101111 01101100 01100001 00100000 01001101 01110101 01101110 01100100 01101111af4a2d48
No se detectaron errores.
El mensaje decodificado es: Hola Mundo
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1>
```

Mensaje modificado para generar la detección de error (1 bit) :

01001000 01101111 01101100 01100001 00100000 01001101 01110101 01101110 01100100 01101110af4a2d48

Respuesta del Receptor

```
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1> node crc32_receptor.js
Trama recibida: 01001000 01101111 01101100 01100001 00100000 01001101 01110101 01101110 01100100 01101110af4a2d48
Se detectó un error en la data recibida. Error: 77073096
La trama se descarta por detectar errores.
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1>
```

Mensaje modificado para generar la detección de error (2 bit) :

01001000 11101111 01101100 01100001 00100000 01001101 01110101 01101110 01100100 01101110af4a2d48

Respuesta del Receptor

```
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1> node crc32_receptor.js
Trama recibida: 01001000 11101111 01101100 01100001 00100000 01001101 01110101 01101110 01100100 01101110af4a2d48
Se detectó un error en la data recibida. Error: 6b20a883
La trama se descarta por detectar errores.
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1>
```

Trama 2:

Mensaje: "Redes Neuronales"

01010010 01100101 01100100 01100101 01110011 00100000 01001110 01100101
01110101 01110010 01101111 01101110 01100001 01101100 01100101 01110011

Mensaje devuelto por el emisor

01010010 01100101 01100100 01100101 01110011 00100000 01001110 01100101
01110101 01110010 01101111 01101110 01100001 01101100 01100101
01110011399b7daf

Respuesta del Receptor

```
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1> node crc32_receptor.js
Trama recibida: 01010010 01100101 01100100 01100101 01110011 00100000 01001110 01100101 01110101 01110010 01101111 01101110 01100001 01101100 01100101 01110011399b7daf
No se detectaron errores.
El mensaje decodificado es: Redes Neuronales
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1>
```

Mensaje modificado para generar la detección de error (1 bit):

01010010 01100101 01100100 01100101 01110011 00100000 01001110 01100101
01110101 01110010 01101111 01101110 01100001 01101100 01100101
01110010399b7daf

Respuesta del Receptor

```
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1> node crc32_receptor.js
Trama recibida: 01010010 01100101 01100100 01100101 01110011 00100000 01001110 01100101 01110101 01110010 01101111 01101110 01100001 01101100 01100101 01110010399b7daf

Se detectó un error en la data recibida. Error: -133a839d
La trama se descarta por detectar errores.
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1>
```

Mensaje modificado para generar la detección de error (2 bits):

01010010 01100101 01100100 01100101 01110011 00100000 01001110 01100101
01110101 01110010 01101111 01101110 01100001 01101100 01100100
01110010399b7daf

Trama 3:

Mensaje: "Messi GOAT"

01001101 01100101 01110011 01110011 01101001 00100000 01000111 01001111 01000001 01010100

Mensaje devuelto por el emisor

01001101 01100101 01110011 01110011 01101001 00100000 01000111 01001111 01000001 010101008e14f9e2

Respuesta del Receptor

```
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1> node crc32_receptor.js
Trama recibida: 01001101 01100101 01110011 01110011 01101001 00100000 01000111 01001111 01000001 010101008e14f9e2
No se detectaron errores.
El mensaje decodificado es: Messi GOAT
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1>
```

Mensaje modificado para generar la detección de error (1 bit):

01001101 01100101 01110011 01110011 01101001 00100000 01000111 01001111 01000001 0101010**1**8e14f9e2

Respuesta del Receptor

```
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1> node crc32_receptor.js
Trama recibida: 01001101 01100101 01110011 01110011 01101001 00100000 01000111 01001111 01000001 010101018e14f9e2
Se detectó un error en la data recibida. Error: 77073096
La trama se descarta por detectar errores.
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1>
```

Mensaje modificado para generar la detección de error (2 bits):

01001101 01100100 01110011 01110011 01101001 00100000 01000111 01001111 01000001 0101010**1**8e14f9e2

Respuesta del Receptor

```
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1> node crc32_receptor.js
Trama recibida: 01001101 01100100 01110011 01110011 01101001 00100000 01000111 01001111 01000001 010101018e14f9e2
Se detectó un error en la data recibida. Error: -7f349368
La trama se descarta por detectar errores.
PS C:\Users\esteb\OneDrive\Documents\GitHub\Laboratorio2-Parte1>
```

Caso donde la trama cambia pero no se detecta el error

Para el algoritmo CRC-32

En la mayoría de los casos, la única manera de que un error no sea detectado por CRC-32 es si el error ocurrió de tal manera que coincidentemente resultó en el mismo valor de checksum, lo cual es altamente improbable dado que CRC-32 utiliza un polinomio de 32 bits, lo que significa que hay más de 4 mil millones de posibles valores de checksum. Por lo tanto no se pudo realizar el cambio de bits para que el error no sea detectado.

Corrector:

1. Utilizar mínimo tres tramas distintas sin manipular en el lado del receptor. Se debe evidenciar que el receptor no detecta errores y muestra la trama.

data = "0110" # Ejemplo 1
data = "1010" # Ejemplo 2
data = "1001" # Ejemplo 3

receptor:
1

```
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1> node hamming_receptor.js
Codeword Hamming: 1100110
Datos decodificados: 0110
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1>
```

2

```
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1> node hamming_receptor.js
Codeword Hamming: 1011010
Datos decodificados: 1010
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1>
```

3

```
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1> node hamming_receptor.js
Codeword Hamming: 0011001
Datos decodificados: 1001
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1>
```

2. Utilizar mínimo otras tres tramas distintas donde se modifique manualmente un bit al momento de pasarlas al receptor. Se debe evidenciar que el algoritmo detecta/errores los errores

data = "0000" # Ejemplo 4
data = "1111" # Ejemplo 5
data = "1100" # Ejemplo 6

receptor:
1.

```
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1> node hamming_receptor.js
Codeword Hamming: 0000001
Error encontrado en la posición 7
Datos decodificados: 0000
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1>
```

2

```
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1> node hamming_receptor.js
Codeword Hamming: 0111111
Error encontrado en la posición 4
Datos decodificados: 1111
```

3

```
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1> node hamming_receptor.js
Codeword Hamming: 1111100
Error encontrado en la posición 4
Datos decodificados: 1100
```

3.Utilizar mínimo otras tres tramas distintas donde se modifique manualmente por los menos dos bits al momento de pasarlas al receptor. Se debe evidenciar que el algoritmo detecta/corrige los errores. Si el algoritmo no es capaz de detectar alguna de las tramas con errores, justificar la razón en la discusión del reporte

data = "0011" # Ejemplo 7
data = "0101" # Ejemplo 8
data = "1011" # Ejemplo 9

receptor:
1

```
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1> node hamming_receptor.js
Codeword Hamming: 1000000
Error encontrado en la posición 4
Datos decodificados: 0000
```

2

```
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1> node hamming_receptor.js
Codeword Hamming: 0100110
Error encontrado en la posición 4
Datos decodificados: 0110
```

3

```
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1> node hamming_receptor.js
Codeword Hamming: 0000011
Error encontrado en la posición 4
Datos decodificados: 0011
```

4.Utilizar una trama modificada especialmente para que el algoritmo del lado del receptor no sea capaz de detectar error e indicar por qué la trama no fue detectada en la discusión del reporte.

output modificado especialmente para que no sea detectado: 1101001

```
PS D:\Program Files (x86)\GitHub\Laboratorio2-Parte1> node hamming_receptor.js
Codeword Hamming: 1101001
Datos decodificados: 0001
```

Discusión

En los tres casos el algoritmo CRC-32 funcionó de acuerdo a lo esperado, detectando eficientemente los errores introducidos manualmente en un bit de cada trama. Esto verifica su capacidad para detectar errores aleatorios en la transmisión.

No hubo casos donde no se pudiera detectar el error, ya que CRC-32 está diseñado para detectar este tipo de errores aleatorios en una o múltiples posiciones con muy alta probabilidad, siempre que la longitud del burst de errores sea menor o igual a 32 bits.

A través de las pruebas solicitadas también se evaluó la eficacia de la implementación del código Hamming. Con tramas sin manipular confirmaron que el algoritmo pudo recibir y procesar datos correctamente sin detectar errores inexistentes. Sin embargo, las pruebas con tramas modificadas manualmente revelaron algunos hallazgos interesantes:

Las tramas con un solo bit modificado fueron correctamente detectadas y corregidas. En cambio las tramas con dos bits modificados presentaron algunas inconsistencias en la detección de errores, lo que llevó a una discusión más profunda sobre las limitaciones.Se observó que no todos los errores de dos bits, e incluso algunos de un solo bit, fueron detectados. Esto se debe a la configuración particular de los bits de paridad en la implementación.

Comentario grupal sobre el tema (errores)

Este laboratorio nos permitió comprender en profundidad la problemática de los errores en la transmisión de datos y la importancia de los mecanismos para detectarlos y corregirlos. Quedó claro que en cualquier medio físico de transmisión existen fuentes de ruido que pueden alterar los datos, y si no se manejan adecuadamente estas perturbaciones se acumulan y degradan completamente la información.

En la prueba 4 de Hamming donde se implementó una entrada especial para que no diera error el cual fue '1111001' se tuvo que cambiar uno de los bits de datos que no estuviera involucrado en los cálculos de paridad en este caso si se cambia el bit D3(posición 6 en la trama codificada) no afectará los cálculos de paridad y, por lo tanto, no fue detectada(1111011).

Esta situación es una limitación de la codificación de Hamming (7,4) en general y de la implementación en especial, donde no todos los bits de datos están involucrados en todos los cálculos de paridad.

Conclusiones

Para el código de CRC-32, las pruebas realizadas verifican el correcto funcionamiento del algoritmo CRC-32 implementado y su capacidad para detectar eficientemente errores aleatorios en la transmisión mediante el cálculo y chequeo del checksum.

El código Hamming(7,4) es un método de corrección de errores que permite la detección y corrección de un solo error en una transmisión de datos. A pesar de su eficacia en la corrección de errores únicos, tiene limitaciones, como la incapacidad de manejar múltiples errores y una eficiencia reducida en términos de ancho de banda. Aun así, sigue siendo una herramienta útil en la comunicación digital donde la integridad de los datos es crítica.

Citas y Referencias

Wikipedia contributors. (2023, August 3). Cyclic redundancy check. In Wikipedia. Retrieved August 3, 2023, from https://en.wikipedia.org/wiki/Cyclic_redundancy_check

Jc-Mouse. (s. f.). CRC32 - Verificación de Redundancia Cíclica. Recuperado de https://www.jc-mouse.net/java/crc32-verificacion-de-redundancia-ciclica

Ross Williams. (n.d.). CRC_V3: 3rd Generation CRC Polynomials. Retrieved Month Day, Year, from http://www.ross.net/crc/download/crc_v3.txt

Invarato, R. (2017, August 12). Código de Hamming: Detección y Corrección de errores - Jarroba. Jarroba. https://jarroba.com/codigo-de-hamming-deteccion-y-correccion-de-errores/

Vista de DETECCIÓN Y CORRECCIÓN DE ERRORES MEDIANTE EL CÓDIGO HAMMING | Revista Vínculos. (n.d.). https://revistas.udistrital.edu.co/index.php/vinculos/article/view/4271/8749