Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Yongbum Park (20117)
Jose Rodrigo Barrera Garcia (20807)

Lab2.1 - Esquemas de detección y corrección de errores

(CC 3067) Redes
Catedrático: Jorge Andrés Yass Coy

Indice

Descripción

En esta práctica, se abordó el tema de esquemas de detección y corrección de errores, centrándonos en dos técnicas ampliamente utilizadas: el Código de Hamming para la corrección de errores y el CRC-32 para la detección de errores. El objetivo principal fue comprender y aplicar estas técnicas en el contexto de transmisión de datos entre dos sistemas: un emisor y un receptor.

El uso de lenguajes de programación diferentes, Python y Java, para implementar los códigos de emisor y receptor permitió una comprensión más profunda de las técnicas de detección y corrección de errores y resaltó la portabilidad y flexibilidad de estos algoritmos.

Durante la práctica, se llevaron a cabo una serie de experimentos para evaluar la efectividad de los esquemas de detección y corrección de errores utilizando diferentes conjuntos de datos y niveles de ruido simulados. Se analizaron los resultados obtenidos y se compararon las tasas de detección y corrección de errores para el Código de Hamming y el CRC-32.

Algoritmos utilizados

Corrección de errores

- Código de Hamming

El algoritmo de Hamming es una técnica de detección y corrección de errores en transmisiones digitales. Se utiliza para asegurar que los datos enviados lleguen de manera confiable al destinatario, incluso si ocurren errores durante la transmisión.

El algoritmo agrega bits redundantes a los datos originales antes de enviarlos. Estos bits redundantes se calculan de tal manera que al llegar al destino, se puedan detectar y corregir errores que puedan haber ocurrido durante la transmisión.

El proceso de codificación de Hamming implica tres pasos principales:

1. Cálculo de bits redundantes: Se determina el número de bits redundantes necesarios según la cantidad de datos a transmitir.

2. Colocación de bits redundantes: Se ubican los bits redundantes en posiciones específicas dentro de la secuencia de datos.

3. Cálculo de bits de paridad: Se calculan los bits de paridad en función de los bits de datos y los bits redundantes colocados previamente.

Cuando los datos llegan al destinatario, se recalculan los bits de paridad. Si se detectan discrepancias entre los bits originales y los bits recalculados, el algoritmo puede identificar la posición del error y corregirlo automáticamente. Si no se detectan discrepancias, significa que la transmisión fue exitosa y los datos están libres de errores.

Detección de errores

- CRC-32

Es un algoritmo de detección de errores que se utiliza en redes digitales y dispositivos de almacenamiento para detectar cambios accidentales en los datos. Este tipo de verificación es particularmente efectiva para detectar errores ocasionados por ruido en los canales de transmisión.

Su paso comienza desde el emisor donde este recibe la trama y según la cantidad de bits con el cual trabajar se agregan esa misma cantidad de ceros a la trama luego para operarlo con polinomial del bit con el que se está trabajando, en este caso se trabaja con la lógica del XOR.

Una vez esto se envía para el receptor que se encarga también de resolverlo y si todo sale bien los bits que se agregaron a la trama darían todos ceros mostrando que efectivamente que se recibió correctamente de lo contrario se llegaría a mostrar un bit de 1 o más en alguna parte del código.

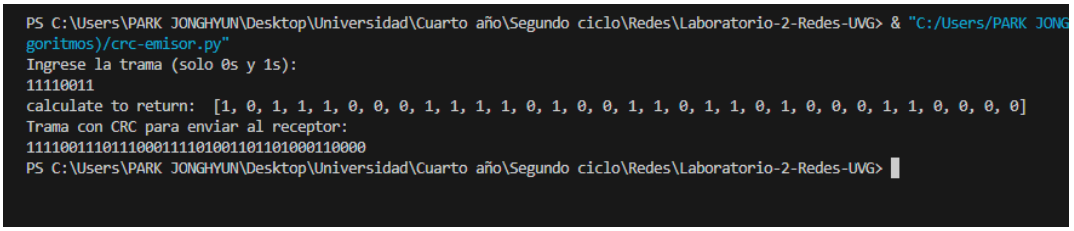
Resultados

Tramas distintas sin manipular

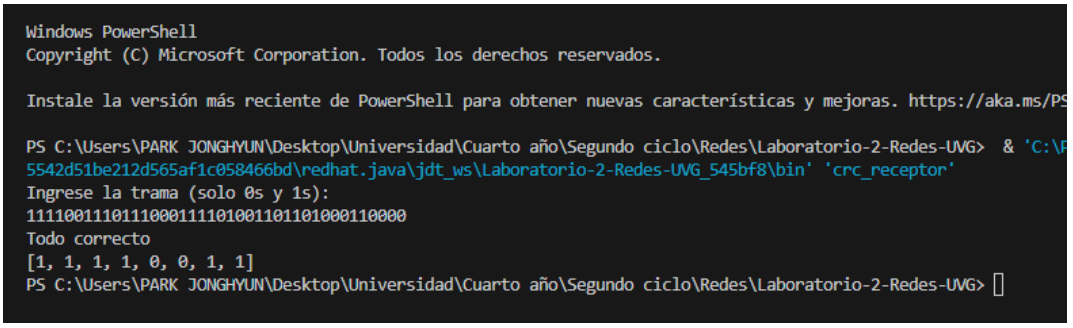
- **11110011**

CRC-32

Emisor

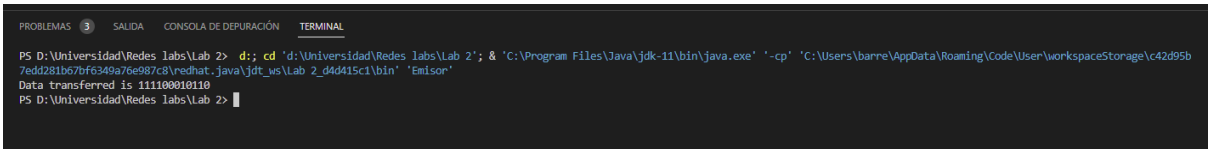


Receptor



Hamming

Emisor



Receptor

- **1011001001**

CRC-32

Emisor

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG> & "C:/Users/PARK JO
goritmos)/crc-emisor.py"
Ingrese la trama (solo 0s y 1s):
1011001001
calculate to return:  [0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1]
Trama con CRC para enviar al receptor:
10110010010110111101011101110010110001111
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Receptor

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>  & 'C:\Program Files\Jav
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'
Ingrese la trama (solo 0s y 1s):
10110010010110111101011101110010110001111
Todo correcto
[1, 0, 1, 1, 0, 0, 1, 0, 0, 1]
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Hamming

Emisor

```
PROBLEMAS 3   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS D:\Universidad\Redes labs\Lab 2>  d:; cd 'd:\Universidad\Redes labs\Lab 2'; & 'C:\Program Files\Java\jdk-11\bin\java.exe' '-cp' 'C:\Users\barre\AppData\Roaming\Code\User\workspaceStorage\c42d95b
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'
Data transferred is 10110011001101
PS D:\Universidad\Redes labs\Lab 2>
```

Receptor

```
PROBLEMAS 3   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS D:\Universidad\Redes labs\Lab 2> & C:/Users/barre/AppData/Local/Programs/Python/Python310/python.exe "d:/Universidad/Redes labs/Lab 2/Laboratorio-2-Redes-UVG/Rodrigo (Algoritmos)/Receptor.py"
Ingrese la trama recibida (trama + bit de paridad modificado manualmente): 10110011001101
No se detectaron errores en la trama recibida.
PS D:\Universidad\Redes labs\Lab 2>
```

- **10000001**

CRC-32

Emisor

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG> & "C:/Users/PARK J
goritmos)/crc-emisor.py"
Ingrese la trama (solo 0s y 1s):
10000001
calculate to return:  [0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1]
Trama con CRC para enviar al receptor:
10000001010011010000010111001100010111111
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Receptor

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>  & 'C:\Pro
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'
Ingrese la trama (solo 0s y 1s):
10000001010011010000010111001100010111111
Todo correcto
[1, 0, 0, 0, 0, 0, 0, 1]
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Hamming

Emisor

```
PROBLEMAS 3   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS D:\Universidad\Redes labs\Lab 2>  d:; cd 'd:\Universidad\Redes labs\Lab 2'; & 'C:\Program Files\Java\jdk-11\bin\java.exe' '-cp' 'C:\Users\barre\AppData\Roaming\Code\User\workspaceStorage\c42d95b
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'
Data transferred is 100010001111
PS D:\Universidad\Redes labs\Lab 2>
```

Receptor

```
PROBLEMAS 3   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS D:\Universidad\Redes labs\Lab 2> & C:/Users/barre/AppData/Local/Programs/Python/Python310/python.exe "d:/Universidad/Redes labs/Lab 2/Laboratorio-2-Redes-UVG/Rodrigo (Algoritmos)/Receptor.py"
Ingrese la trama recibida (trama + bit de paridad modificado manualmente): 100010001111
No se detectaron errores en la trama recibida.
PS D:\Universidad\Redes labs\Lab 2>
```

Tramas distintas donde se modifique manualmente un bit

- **1001001**

CRC-32

Emisor

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG> & "C:/Users/PARK
goritmos)/crc-emisor.py"
Ingrese la trama (solo 0s y 1s):
1001001
calculate to return:  [1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0]
Trama con CRC para enviar al receptor:
10010011000100101100000000011101000
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Receptor
original: 10010011000100101100000000011101000
modificado: 10010011000100101100100000011101000

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>  & 'C:\Program F
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'
Ingrese la trama (solo 0s y 1s):
10010011000100101100100000011101000
problema! trama con problema
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Hamming

Emisor

```
PROBLEMAS 3   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL

PS D:\Universidad\Redes labs\Lab 2>  d:; cd 'd:\Universidad\Redes labs\Lab 2'; & 'C:\Program Files\Java\jdk-11\bin\java.exe' '-cp' 'C:\Users\barre\AppData\Roaming\Code\User\workspaceStorage\c42d95b
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'
Data transferred is 10011001111
PS D:\Universidad\Redes labs\Lab 2>
```

Receptor
Original: 10011001111
Modificado: 10011011111

```
PS D:\Universidad\Redes labs\Lab 2> & C:/Users/barre/AppData/Local/Programs/Python/Python310/python.exe
Ingrese la trama recibida (trama + bit de paridad modificado manualmente): 10011011111
Se detectó un error en la trama recibida en la posición: 7 desde la izquierda.
Trama corregida sin el error: 10011001111
PS D:\Universidad\Redes labs\Lab 2>
```

- **1010101**

CRC-32

Emisor

```
Ingrese la trama (solo 0s y 1s):
1001001
calculate to return:  [1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0]
Trama con CRC para enviar al receptor:
10010011000100101100000000011101000
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Receptor
original: 10010011000100101100000000011101000
modificado: 10010011000100101100010000011101000

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>  & 'C:\Progr
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'
Ingrese la trama (solo 0s y 1s):
10010011000100101100010000011101000
problema! trama con problema
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Hamming

Emisor

```
PS D:\Universidad\Redes labs\Lab 2>  d:; cd 'd:\Universidad\Redes labs\Lab 2'; & '
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'
Data transferred is 10100101111
PS D:\Universidad\Redes labs\Lab 2>
```

Receptor
Original: 10100101111
Modificado: 10100101011

```
PS D:\Universidad\Redes labs\Lab 2> & C:/Users/barre/AppData/Local/Programs/Python/Python310/python.exe '
Ingrese la trama recibida (trama + bit de paridad modificado manualmente): 10100101011
Se detectó un error en la trama recibida en la posición: 9 desde la izquierda.
Trama corregida sin el error: 10100101111
PS D:\Universidad\Redes labs\Lab 2>
```

- **110101**

CRC-32

Emisor

```
Ingrese la trama (solo 0s y 1s):
110101
calculate to return:  [1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1]
Trama con CRC para enviar al receptor:
11010111001110101101000101011010010001
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Receptor
original: 11010111001110101101000101011010010001
modificado: 11010111001110101101000101011010010101

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>  & 'C:\Program Fi
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'
Ingrese la trama (solo 0s y 1s):
11010111001110101101000101011010010101
problema! trama con problema
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Hamming

Emisor

```
PS D:\Universidad\Redes labs\Lab 2>  d:; cd 'd:\Universidad\Redes labs\Lab 2'; & 'C
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'
Data transferred is 1100101110
PS D:\Universidad\Redes labs\Lab 2>
```

Receptor
Original: 1100101110
Modificado: 0100101110

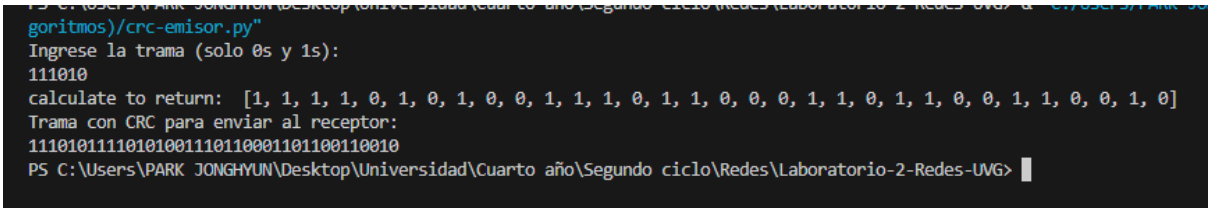
```
PS D:\Universidad\Redes labs\Lab 2> & C:/Users/barre/AppData/Local/Programs/Python/Python310/python.exe
Ingrese la trama recibida (trama + bit de paridad modificado manualmente): 0100101110
Se detectó un error en la trama recibida en la posición: 1 desde la izquierda.
Trama corregida sin el error: 1100101110
PS D:\Universidad\Redes labs\Lab 2>
```

Tramas distintas donde se modifique manualmente por los menos dos bits

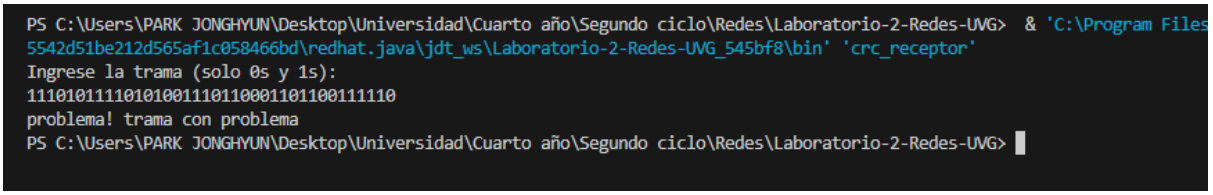
- **111010**

CRC-32

Emisor

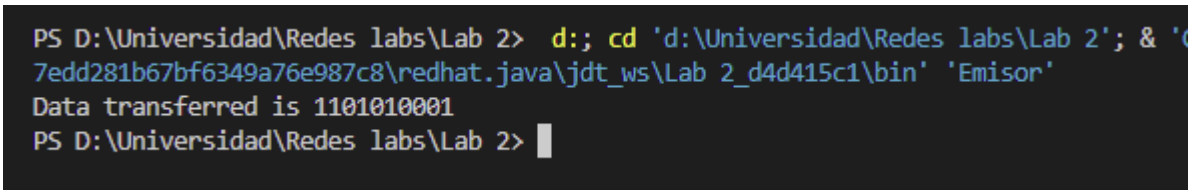


Receptor
original: 11101011110101001110110001101100110010
modificado: 11101011110101001110110001101100111110

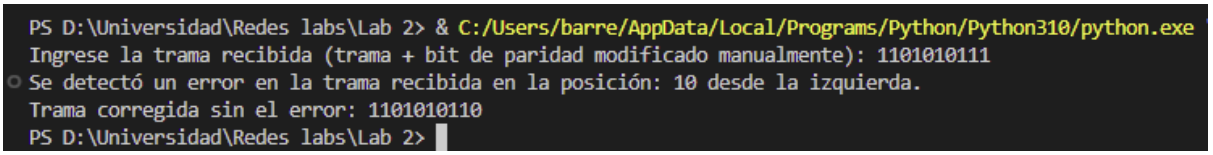


Hamming

Emisor



Receptor
Original: 1101010001
Modificado: 1101010111



- **1010101011**

CRC-32

Emisor

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG> & "C:/Users/PARK JONGHYUN/Ap
goritmos)/crc-emisor.py"
Ingrese la trama (solo 0s y 1s):
1010101011
calculate to return:  [1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0]
Trama con CRC para enviar al receptor:
1010101011110100101011110000000011000100000
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Receptor
original: 1010101011110100101011110000000011000100000
modificado:1010101011110100101011110000000011000111000

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>  & 'C:\Program Files\Java\jd
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'
Ingrese la trama (solo 0s y 1s):
1010101011110100101011110000000011000111000
problema! trama con problema
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Hamming

Emisor

```
PS D:\Universidad\Redes labs\Lab 2>  d:; cd 'd:\Universidad\Redes labs\Lab 2';
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'
Enter data:
1010101011
Data transferred is 10101011010101
PS D:\Universidad\Redes labs\Lab 2>
```

Receptor
Original: 10101011010101
Modificado: 10101011011111

```
PS D:\Universidad\Redes labs\Lab 2> & C:/Users/barre/AppData/Local/Programs/Python/Python310/python.exe
Ingrese la trama recibida (trama + bit de paridad modificado manualmente): 10101011011111
Se detectó un error en la trama recibida en la posición: 9 desde la izquierda.
Trama corregida sin el error: 10101011111111
PS D:\Universidad\Redes labs\Lab 2>
```

- **111111010**

CRC-32

Emisor

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG> & "C:/Users/PARK JONGHYUN/Algoritmos)/crc-emisor.py"
Ingrese la trama (solo 0s y 1s):
111111010
calculate to return:  [0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1]
Trama con CRC para enviar al receptor:
11111101000001011011101000011100111111001
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Receptor
original: 11111101000001011011101000011100111111001
modificado:11111101000001011011101000011100111111111

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>  & 'C:\Program Files\5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'
Ingrese la trama (solo 0s y 1s):
11111101000001011011101000011100111111111
problema! trama con problema
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Hamming

Emisor

```
PS D:\Universidad\Redes labs\Lab 2>  d:; cd 'd:\Universidad\Redes labs\Lab 2'; 7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'
Enter data:
111111010
Data transferred is 1111111010011
PS D:\Universidad\Redes labs\Lab 2>
```

Receptor
Original: 1111111010011
Modificado: 1101101010011

```
PS D:\Universidad\Redes labs\Lab 2> & C:/Users/barre/AppData/Local/Programs/Python/Python310/python.exe
Ingrese la trama recibida (trama + bit de paridad modificado manualmente): 1101101010011
Se detectó un error en la trama recibida en la posición: 11 desde la izquierda.
Trama corregida sin el error: 1101101010111
PS D:\Universidad\Redes labs\Lab 2>
```

Trama modificada especialmente para que el algoritmo del lado del receptor no sea capaz de detectar error

- **1010101011**

CRC-32

emisor

```
goritmos)/crc-emisor.py"
Ingrese la trama (solo 0s y 1s):
1010101011
calculate to return:  [1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0]
Trama con CRC para enviar al receptor:
10101010111010010101111000000011000100000
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

receptor

original: 10101010111010010101111000000011000100000

modificado:10101010110101000011111100100000000111110101

```
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>  & 'C:\Prog
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'
Ingrese la trama (solo 0s y 1s):
10101010110101000011111100100000000111110101
Todo correcto
[1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0]
PS C:\Users\PARK JONGHYUN\Desktop\Universidad\Cuarto año\Segundo ciclo\Redes\Laboratorio-2-Redes-UVG>
```

Hamming

Emisor

```
PS D:\Universidad\Redes labs\Lab 2>  d:; cd 'd:\Universidad\Redes labs\Lab 2';
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'
Enter data:
1010101011
Data transferred is 10101011010101
PS D:\Universidad\Redes labs\Lab 2>
```

Receptor

Original: 10101011010101

Modificado: 0010101101010100

```
PS D:\Universidad\Redes labs\Lab 2> & C:/Users/barre/AppData/Local/Programs/Python/Python310/python.exe
Ingrese la trama recibida (trama + bit de paridad modificado manualmente): 0010101101010100
No se detectaron errores en la trama recibida.
PS D:\Universidad\Redes labs\Lab 2>
```

Discusión

Resultados de las pruebas sin manipulación:

- En esta primera fase de pruebas, se utilizó un conjunto de tres tramas distintas sin manipulación en el lado del receptor. Se pudo observar que el receptor no detectó ningún error y mostró correctamente las tramas recibidas. Esto valida la efectividad del Código de Hamming y CRC-32 para la transmisión de datos sin errores y resalta su capacidad para mantener la integridad de los datos transmitidos.

Resultados de las pruebas con un bit modificado:

- En la segunda fase de pruebas, se utilizaron otras tres tramas distintas donde se modifica manualmente un bit al momento de pasarlas al receptor. Se pudo evidenciar que el algoritmo de detección de errores (CRC-32) detectó los errores en todas las tramas modificadas, lo que indica que el receptor puede identificar con precisión la presencia de errores y descartar las tramas corruptas.

- Ahora bien, para el caso del algoritmo de hamming, debido a su capacidad de detectar un error y corregirlo, se concluye que fue todo un éxito obteniendo los resultados esperados, demostrando de esta manera que el algoritmo de Hamming tiene la capacidad de detectarlos y corregirlos.

Resultados de las pruebas con dos bits modificados:

- En la tercera fase de pruebas, se utilizaron tres tramas distintas donde se modificaron manualmente al menos dos bits al momento de pasarlas al receptor. Se pudo evidenciar que el algoritmo de corrección de errores (Código de Hamming) no fue capaz de detectar los errores, y esto se debe a que, el algoritmo de Hamming está diseñado para detectar y corregir un solo error en la trama de datos. Sin embargo, este algoritmo no puede detectar cuando hay dos errores simultáneos en la trama.

  La razón detrás de esto radica en la forma en que se calculan los bits de paridad y se colocan los bits redundantes en la trama. El algoritmo de Hamming está basado en la técnica de códigos de Hamming, que utiliza la posición de los bits de paridad para detectar y corregir errores.

  Cuando se introducen dos errores en la trama de datos, puede ocurrir una situación en la que los bits de paridad calculados para ambos errores se anulen entre sí, y el algoritmo no detectará ninguna discrepancia. Esto se debe a que los bits de paridad se calculan en función de las posiciones de los bits de datos y los bits redundantes. y corregir los errores en todas las tramas modificadas, y esto se debe a que

Justificación de la capacidad de detección y corrección:

- Es importante destacar en la discusión que los algoritmos de detección y corrección de errores no son infalibles y tienen limitaciones en cuanto a la cantidad de errores que pueden detectar y corregir. Si en alguna prueba el algoritmo no pudo detectar una trama con errores, se debe investigar las razones detrás de ello y explicar las posibles limitaciones del esquema utilizado.

Trama no detectada por el algoritmo del receptor:

- En la última fase de pruebas, se diseñó una trama modificada especialmente para que el algoritmo del lado del receptor no fuera capaz de detectar el error. En este caso, es crucial analizar y explicar las razones por las cuales el algoritmo no pudo detectar el error en esa trama particular, destacando las vulnerabilidades específicas del esquema utilizado y posibles mejoras o alternativas para fortalecer la detección de errores.

Comentario grupal

Ambos algoritmos, Hamming y CRC-32, son herramientas esenciales en el mundo de las transmisiones digitales y la detección de errores. Me impresiona cómo Hamming puede no solo detectar, sino también corregir errores en los datos transmitidos, lo que aumenta significativamente la confiabilidad de la comunicación. Es genial ver cómo estos códigos se utilizan en sistemas de almacenamiento y redes para garantizar que los datos lleguen intactos a su destino.

Por otro lado, el CRC-32 es asombroso en su simplicidad y eficacia para detectar errores en los datos. Aunque no puede corregirlos, proporciona una forma rápida y confiable de verificar la integridad de los datos sin necesidad de retransmitirlos. Es como un "sello de aprobación" para asegurarnos de que nuestros datos lleguen sin alteraciones.

En general, estos algoritmos son fundamentales para la transmisión segura de datos en el mundo digital actual. Aprendiendo sobre ellos, he obtenido una mayor apreciación de cómo la tecnología se esfuerza por brindar confiabilidad y precisión en nuestras comunicaciones y almacenamiento de información.

Conclusiones

- Esta actividad nos ha permitido comprender la importancia crítica de los esquemas de detección y corrección de errores en la transmisión de datos. Estos algoritmos juegan un papel fundamental en la garantía de la integridad y confiabilidad de la información en entornos propensos a errores y ruido.
- Si bien los esquemas de detección y corrección de errores son poderosos, también hemos identificado sus limitaciones. En algunos casos, los algoritmos no fueron capaces de detectar ciertos errores en este caso la última prueba realizada, lo que resalta la importancia de explorar soluciones adicionales y entender sus vulnerabilidades en contextos específicos.
- Los conceptos aprendidos en esta actividad son fundamentales para aplicaciones prácticas en diversas áreas, como las comunicaciones de red, almacenamiento de datos y sistemas críticos. La habilidad para detectar y corregir errores es esencial para mantener la integridad y seguridad de la información.

Citas / Referencias

- Verificación de redundancia cíclica - Wikipedia, la enciclopedia libre
- VRC, LRC Y CRC – MÉTODOS DE DETECCIÓN DE ERRORES - WordPress.com
- Comprobación de redundancia cíclica (CRC) Definición / explicación
- “Hamming Code in Java - Javatpoint.” *Www.javatpoint.com*,

  www.javatpoint.com/hamming-code-in-java.

- Greyat, Rudeus . “Implementación Del Código Hamming En Python – Barcelona

  Geeks.” *Barcelonageeks.com*,

  barcelonageeks.com/implementacion-del-codigo-hamming-en-python/. Accessed 3

  Aug. 2023.