

Angel Higueros 20460

Fredy Velásquez 201011

Laboratorio 2.1

Esquemas de detección y corrección de errores

Descripción de la práctica

En el transcurso de esta actividad práctica, nos enfocamos en el estudio de estrategias destinadas a detectar y enmendar fallos en la transmisión de datos, se utilizaron dos estrategias para esto: el Código de Hamming, empleado para subsanar deslices, y el método CRC-32, destinado a la identificación de errores. Nuestro propósito consistió en asimilar y aplicar estas técnicas en el contexto de la comunicación de información entre dos sistemas interconectados: uno que actúa como emisor y el otro como receptor.

La diversidad en la elección de lenguajes de programación, en este caso, Python y Java, para llevar a cabo la implementación de los protocolos de emisión y recepción, posibilitó un estudio más profundo en los mecanismos de detección y corrección de fallos, al mismo tiempo que resaltó la versatilidad y la adaptabilidad de dichos algoritmos.

A lo largo de la actividad, se ejecutaron una serie de experimentos destinados a evaluar la eficacia de los esquemas de detección y corrección de errores. Esta evaluación se realizó en diversos conjuntos de datos y bajo distintos escenarios de simulación de interferencias. Los resultados obtenidos fueron sometidos a un análisis y se procedió a la comparación de los índices de efectividad en la detección y corrección de errores entre el Código de Hamming y el método CRC-32.

Resultados

Tramas distintas sin manipular

11110011

CRC-32

Emisor:

```
goritmos)/crc-emisor.py"
Ingrese la trama (solo 0s y 1s):
11110011
calculate to return: [1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0]
Trama con CRC para enviar al receptor:
1111001110111000111101001101101000110000
```

Receptor:

```
5542d51be212d565af1cd058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin    crc_receptor
Ingrese la trama (solo 0s y 1s):
111100110111000111101001101101000110000
Todo correcto
[1, 1, 1, 1, 0, 0, 1, 1]
```

Hamming

Emisor:

```
/edd281bb/bf6349a/6e98/c8\redhat.java\jdt_ws\Lab_2_d4d415c1\bin    Emisor
Data transferred is 111100010110
PS D:\Universidad\Redes_labs\Lab_2> █
```

Receptor:

```
D:\Universidad\Redes_labs\Lab_2> & C:/Users/barre/AppData/Local/Programs/Python/Python
Ingrese la trama recibida (trama + bit de paridad modificado manualmente): 111100010110
No se detectaron errores en la trama recibida.
PS D:\Universidad\Redes_labs\Lab_2> █
```

1011001001

CRC-32

Emisor:

```
goritmos)/crc-emisor.py
Ingrese la trama (solo 0s y 1s):
1011001001
calculate to return: [0, 1, 1, 0, 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:
1011001001011011101011100101100011111
```

Receptor:

```
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'  
Ingrese la trama (solo 0s y 1s):  
1011001001011011101011100101100011111  
Todo correcto  
[1, 0, 1, 1, 0, 0, 1, 0, 0, 1]
```

Hamming

Emisor:

```
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'  
Data transferred is 10110011001101  
PS D:\Universidad\Redes labs\Lab 2> █
```

Receptor:

```
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:

```
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, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1]  
Trama con CRC para enviar al receptor:  
1000000101001101000001011100110001011111
```

Receptor:

```
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'  
Ingrese la trama (solo 0s y 1s):  
1000000101001101000001011100110001011111  
Todo correcto  
[1, 0, 0, 0, 0, 0, 0, 1]
```

Hamming

Emisor:

```
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab_2_d4d415c1\bin' 'Emisor'  
Data transferred is 100010001111  
PS D:\Universidad\Redes_labs\Lab_2>
```

Receptor:

```
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:

```
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, 0, 1, 1, 1, 0, 1, 0, 0]  
Trama con CRC para enviar al receptor:  
100100110001001011000000011101000
```

Receptor:

original:100100110001001011000000011101000

modificado: 1001001100010010110010000011101000

```
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'  
Ingrese la trama (solo 0s y 1s):  
1001001100010010110010000011101000  
problema! trama con problema
```

Hamming

Emisor:

```
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab_2_d4d415c1\bin' 'Emisor'  
Data transferred is 10011001111
```

Receptor:

Original: 10011001111 Modificado: 10011011111

```
Ingrese la trama recibida (trama + bit de paridad modificado manualmente): 100110111111
Se detectó un error en la trama recibida en la posición: 7 desde la izquierda.
Trama corregida sin el error: 100110011111
PS D:\Universidad\Redes labs\Lab 2> 
```

1010101

CRC-32

Emisor:

```
Bonito hoy, CRC Emisor v0.1
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]
Trama con CRC para enviar al receptor:
100100110001001011000000011101000
```

Receptor:

original:1001001100010010110000000011101000

modificado: 1001001100010010110001000011101000

```
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'
Ingrese la trama (solo 0s y 1s):
1001001100010010110001000011101000
problema! trama con problema
```

Hamming

Emisor:

```
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'
Data transferred is 10100101111
```

Receptor:

Original: 10100101111

Modificado: 10100101011

```
Ingrese la trama recibida (trama + bit de paridad modificado manualmente): 101001010111
Se detectó un error en la trama recibida en la posición: 9 desde la izquierda.
Trama corregida sin el error: 10100101111
```

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, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1]  
Trama con CRC para enviar al receptor:  
11010111001110101101000101011010010001
```

Receptor:

original:11010111001110101101000101011010010001

modificado: 11010111001110101101000101011010010101

```
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
```

Hamming

Emisor:

```
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'  
Data transferred is 1100101110
```

Receptor:

Original: 1100101110

Modificado: 0100101110

```
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
```

Tramas distintas donde se modifique manualmente por lo menos dos bits

111010

CRC-32

Emisor:

Receptor:

original:11101011110101001110110001101100110010

modificado: 11101011110101001110110001101100111110

```
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'  
Ingrese la trama (solo 0s y 1s):  
11101011110101001110110001101100111110  
problema! trama con problema
```

Hamming

Emisor:

```
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab_2_d4d415c1\bin' 'Emisor'  
Data transferred is 1101010001
```

Receptor:

Original: 1101010001

Modificado: 1101010111

```
Ingrese la trama recibida (trama + bit de paridad modificado manualmente): 1101010111  
Se detectó un error en la trama recibida en la posición: 10 desde la izquierda.  
Trama corregida sin el error: 110101010  
PS D:\Universidad\Redes Labs\Lab 2> █
```

1010101011

CRC-32

Emisor:

Receptor:

original:101010101111010010101110000000011000100000
modificado:10101010111101001010111000000011000111000

```
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'  
Ingrese la trama (solo 0s y 1s):  
10101010111101001010111000000011000111000  
problema! trama con problema
```

Hamming

Emisor:

```
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'  
Enter data:  
1010101011  
Data transferred is 10101011010101
```

Receptor:

Original: 10101011010101

Modificado: 10101011011111

```
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:

```
goritmos)/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
```

Receptor:

original:11111101000001011011101000011100111111001
modificado:11111101000001011011101000011100111111111

```
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
```

Hamming

Emisor:

```
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab_2_d4d415c1\bin' 'Emisor'  
Enter data:  
111111010  
Data transferred is 1111111010011
```

Receptor:

Original: 1111111010011

Modificado: 1101101010011

```
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 errores

1010101011

CRC-32

Emisor:

```
gabinetos)\crc-emisor.py  
Ingrese la trama (solo 0s y 1s):  
101010101111010010101111000000011000100000  
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:  
101010101111010010101111000000011000100000
```

Receptor:

original:1010101011101001010111000000011000100000
modificado:101010101101010000111110010000000111110101

```
5542d51be212d565af1c058466bd\redhat.java\jdt_ws\Laboratorio-2-Redes-UVG_545bf8\bin' 'crc_receptor'  
Ingrese la trama (solo 0s y 1s):  
10101010101010000111110010000000111110101  
Todo correcto  
[1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0]
```

Hamming

Emisor:

```
7edd281b67bf6349a76e987c8\redhat.java\jdt_ws\Lab 2_d4d415c1\bin' 'Emisor'  
Enter data:  
1010101011  
Data transferred is 10101011010101
```

Receptor:

Original: 10101011010101

Modificado: 0010101101010100

```
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:

Para las pruebas de esta fase se utilizaron tres tramas sin manipulación en el lado del receptor. Con las pruebas se apreció que el receptor no detectó errores e imprimió correctamente las tramas que recibió. Lo anterior refleja la efectividad del código Hamming y CRC-32 en el contexto de transmisión de datos sin errores, a la vez que resalta la capacidad de ambos algoritmos para preservar la integridad de los datos manejados.

Resultados de las pruebas con un bit modificado:

Para esta fase se utilizaron tres tramas distintas respecto al punto anterior, en las mismas se modificó manualmente un bit al momento de enviarlas al receptor. En este caso, el algoritmo CRC-32 detectó los

errores en todas las tramas que estaban modificadas. Lo anterior refleja la efectividad y precisión del receptor al momento de encontrarse con errores y de poder manejar las tramas que estén corruptas. A su vez, el algoritmo Hamming fue exitoso dado que también tuvo una capacidad alta de detección y corrección de errores.

Resultados de las pruebas con dos bits modificados:

Para esta fase se utilizaron tres tramas distintas en donde se cambiaron manualmente al menos dos bits al momento de enviarlas al receptor. En este caso, el algoritmo de Hamming no tuvo la capacidad de detectar los errores, esto se debe a que el mismo está diseñado para corregir y detectar un solo error en la trama de datos, dicho funcionamiento se vio afectado cuando detectó múltiples errores en la trama. El problema con Hamming también radica en el método utilizado para calcular los bits de paridad ya que se colocan de modo redundante en la trama.

Justificación de la capacidad de detección y corrección:

1. Hamming: Las causas por las que este algoritmo no funcione correctamente están ligadas con la mala configuración de la trama, dado que si los bits de paridad se colocan en las posiciones incorrectas el algoritmo no será capaz de funcionar adecuadamente.
2. CRC-32: Este algoritmo puede experimentar errores si el polinomio generador elegido es el incorrecto, a su vez, si el mensaje es demasiado corto no funcionará correctamente ya que no se utilizan los bits suficientes.

Trama no detectada por el algoritmo del receptor:

Para esta parte se diseñó una trama especial para que los errores no pudieran ser detectados por parte del receptor. Las razones por las cuales el algoritmo no pudo detectar las anomalías radican en la vulnerabilidad del mismo en su esquema.

Comentario grupal sobre el tema (errores)

Como grupo nos impresiona bastante que existan algoritmos que sean tan eficientes al momento del manejo de errores a un nivel tan bajo como una trama de bits. Consideramos que es fundamental conocer este tipo de algoritmos ya que proporciona una perspectiva más amplia del funcionamiento de las redes y la comunicación entre las mismas.

Respecto a los algoritmos Hamming y CRC-32 nos impresionó la efectividad de los mismos para cumplir la tarea respectiva a su funcionamiento así como la implementación en código la cual no era tan complicada, nos parece extremadamente curioso como se han ido desarrollando más herramientas en el contexto estudiado y esperamos poder aprender más con la práctica.

Conclusiones

- Por medio de la presente práctica fue posible comprender la importancia de contar con algoritmos de detección y corrección de errores en el contexto de transmisión de información. Los algoritmos estudiados y los que no fueron utilizados son vitales para que los datos operen de forma correcta en escenarios en donde puedan existir factores como ruidos y errores los cuales dificultan la llegada correcta de la información.
- Es importante conocer las limitaciones que poseen los algoritmos de detección y corrección de errores ya que permiten optimizar el contexto en el que se los utiliza para que puedan ser más eficientes.
-

Citas y referencias

IONOS. “Error de CRC: causas y soluciones”.
<https://www.ionos.es/digitalguide/servidores/know-how/error-de-crc/>

Invarato, R. (5 de octubre de 2016). “Código de Hamming: Detección y Corrección de errores”.
<https://jarroba.com/codigo-de-hamming-detectacion-y-correccion-de-errores/>