Universidad del Valle de Guatemala
Facultad de ingeniería

Laboratorio 2 Primera Parte: Esquemas de detección y corrección de errores

Javier Ramírez Cospin 18099
Priscilla Gonzalez 20689

Guatemala 31 de julio del 2023

Descripción de la práctica

En esta práctica, se tendrá como base implementar algoritmos de corrección y detección de errores. Estos ayudarán a la detección de errores que pueden ser ocasionados por ruido, interferencia o hasta fallos en los canales de comunicación, así como corregir dichos errores para poder garantizar que los datos recibidos o almacenados sean los mismos que los datos originales. Dicho esto, se utilizó el algoritmo de corrección de errores Código Hamming y el de detección de errores CRC-4 tanto del lado del emisor como del receptor. Los emisores para dichos algoritmos se realizaron en Python y los receptores en Java. Para el código Hamming del lado del emisor, lo que se buscó fueron los bits de paridad al ingresar una cadena de binarios y las posiciones que estos ocupaban -siendo estas p0, p1, p2…- utilizando esta ecuación *(m + p + 1) <= 2^p*. Una vez teniendo estos bits de paridad por medio de consola, se procedió a pasarlos al receptor.

Ahora bien, para la detección de errores CRC nuevamente del lado del emisor, se ingresa una cadena binario cualquiera y este lo que hacía es retornar la trama. Para la realización de este algoritmo es necesaria la operación XOR bit a bit entre dos cadenas binaria -en este caso, la cadena que se ingresa y la cadena que surge por medio del polinomio que se estará usando 1001-. El resultado de esta operación se almacena en una nueva cadena y se devuelve.

Para el receptor, se aplica el mismo procedimiento que el emisor para identificar errores en una trama provista como primer argumento, es decir, se aplica la operación XOR entre la trama y el polinomio establecido. El receptor identifica si hay error en la trama si existe al menos 1 bit con valor de 1 en las últimas *n - 1* posiciones del resultado final, en donde *n* es la longitud del polinomio establecido. En caso de no haber error, todas las *n - 1* posiciones del resultado tienen valor de 0.

En caso de que el receptor detecte que hay error en la trama, este intenta corregir dicho error. Esto lo hace aplicando el algoritmo de Hamming sobre los datos de la trama y los datos de paridad provistos como segundo argumento. Con una tabla de verdad, se verifican los *p* bits de paridad, en donde *p* es el resultado de calcular *(m + p + 1) <= 2^p* con *m* siendo la cantidad de datos provista en la trama. Estos *p* valores se convierten a formato decimal y el resultado de esa operación es el índice del bit en la trama que se indica como erróneo y el receptor procede a cambiar dicho bit y a aplicar el proceso nuevamente hasta encontrar la trama correcta o alcanzar un límite de ciclos establecido.

A cada uno de los algoritmos se le indicaba qué trama se usaría y el polinomio correspondiente. La trama básicamente fue el resultado del emisor al aplicar el CRC sobre el input. Por último, se verán las ventajas y desventajas que dichos algoritmos pueden llegar a tener y cómo de alguna manera se podrían mejorar.

Resultados

Con la intención de comprobar la eficiencia de los algoritmos de deteccion y correccion de errores implementados, se probaron los siguientes escenarios:

1. Cadena “1101” y polinomio “11001”
    - Emisor

```
priscillagonzalez@MacBook-Pro-de-Priscilla Emisor-Receptor-Redes %
Ingrese la data que desea enviar->1101
Encoded data: 11010001
```

    - Receptor:
      Trama: 11010001 (Sin manipulación)
      Bits de paridad: 010

```
PS D:\Documents\UVG\12vo Semestre\REDES\Emisor-Receptor-Redes\Emisor-Receptor-Redes\src\main\java> java Main 11010001 010
> Trama converted succesfully

Received trama: [1, 1, 0, 1, 0, 0, 0, 1]
Converted trama: [1, 1, 0, 1, 0, 0, 0, 0]
```

2. Cadena “10011” y polinomio “10101”
    - Emisor

```
priscillagonzalez@MacBook-Pro-de-Priscilla Emisor-Receptor-Redes %
Ingrese la data que desea enviar->10011
Encoded data: 100111011
```

    - Receptor
      Trama: 100111011 (Sin manipulación)
      Bits de paridad: 1111

```
PS D:\Documents\UVG\12vo Semestre\REDES\Emisor-Receptor-Redes\Emisor-Receptor-Redes\src\main\java> java Main 100111011 1111
> Trama converted succesfully

Received trama: [1, 0, 0, 1, 1, 1, 0, 1, 1]
Converted trama: [1, 0, 0, 1, 1, 0, 0, 0, 0]
```

3. Cadena “11100” y polinomio “11111”
    - Emisor

```
priscillagonzalez@MacBook-Pro-de-Priscilla Emisor-Receptor-Redes %
Ingrese la data que desea enviar->11100
Encoded data: 111001110
```

    - Receptor
      Trama: 111000001 (Sin manipulación)
      Bits de paridad: 1000

```
PS D:\Documents\UVG\12vo Semestre\REDES\Emisor-Receptor-Redes\Emisor-Receptor-Redes\src\main\java> java Main 111000001 1000
> Trama converted succesfully

Received trama: [1, 1, 1, 0, 0, 0, 0, 0, 1]
Converted trama: [1, 1, 1, 0, 0, 0, 0, 0, 0]
```

4. Cadena “1101” y polinomio “10111”
    - Emisor

```
priscillagonzalez@MacBook-Pro-de-Priscilla Emisor-Receptor-Redes %
Ingrese la data que desea enviar->1101
Encoded data: 11011101
```

    - Receptor:
      Trama: 10011101 (Segundo bit de izquierda a derecha fue manipulado)
      Bits de paridad: 010

```
PS D:\Documents\UVG\12vo Semestre\REDES\Emisor-Receptor-Redes\Emisor-Receptor-Redes\src\main\java> java Main 10011101 010
> Error: Trama has a mismatch ...
> Applying correction ...
> Error detected in position (6)
> Converting trama again ...
> Trama converted succesfully

Received trama: [1, 0, 0, 1, 1, 1, 0, 1]
Converted trama: [1, 1, 0, 1, 0, 0, 0, 0]
```

5. Cadena “1111” y polinomio “11011”
    - Emisor

```
priscillagonzalez@MacBook-Pro-de-Priscilla Emisor-Receptor-Redes %
Ingrese la data que desea enviar->1111
Encoded data: 11110101
```

    - Receptor
      Trama: 01110101 (Primer bit de izquierda a derecha fue manipulado)
      Bits de paridad: 111

```
PS D:\Documents\UVG\12vo Semestre\REDES\Emisor-Receptor-Redes\Emisor-Receptor-Redes\src\main\java> java Main 01110101 111
> Error: Trama has a mismatch ...
> Applying correction ...
> Error detected in position (7)
> Converting trama again ...
> Trama converted succesfully

Received trama: [0, 1, 1, 1, 0, 1, 0, 1]
Converted trama: [1, 1, 1, 1, 0, 0, 0, 0]
```

6. Cadena “10101” y polinomio “11001”
    - Emisor

```
priscillagonzalez@MacBook-Pro-de-Priscilla Emisor-Receptor-Redes %
Ingrese la data que desea enviar->10101
Encoded data: 101011000
```

    - Receptor
      Trama: 101110001
      Bits de paridad: 1100

```
PS D:\Documents\UVG\12vo Semestre\REDES\Emisor-Receptor-Redes\Emisor-Receptor-Redes\src\main\java> java Main 101110001 1100
> Error: Trama has a mismatch ...
> Applying correction ...
> Error detected in position (5)
> Converting trama again ...
> Trama converted succesfully

Received trama: [1, 0, 1, 1, 1, 0, 0, 0, 1]
Converted trama: [1, 0, 1, 0, 1, 0, 0, 0, 0]
```

7. Cadena “1101” y polinomio “11001”
    - Emisor

```
priscillagonzalez@MacBook-Pro-de-Priscilla Emisor-Receptor-Redes %
Ingrese la data que desea enviar->1101
Encoded data: 11010001
```

    - Receptor
      Trama: 10110001 (Segundo y tercer bits fueron manipulados)
      Bits de paridad: 010

```
PS D:\Documents\UVG\12vo Semestre\REDES\Emisor-Receptor-Redes\Emisor-Receptor-Redes\src\main\java> java Main 10110001 010
> Applying correction ...
> Error detected in position (6)
> Applying correction ...
> Error detected in position (5)
> Trama converted succesfully

Received trama: [1, 0, 1, 1, 0, 0, 0, 1]
Converted trama: [1, 1, 0, 1, 0, 0, 0, 0]
```

8. Cadena “10011” y polinomio “10101”
    - Emisor

```
priscillagonzalez@MacBook-Pro-de-Priscilla Emisor-Receptor-Redes %
Ingrese la data que desea enviar->10011
Encoded data: 100111011
```

    - Receptor
      Trama: 010111011 (Primer y segundo bit fueron manipulados)
      Bits de paridad: 1111

```
PS D:\Documents\UVG\12vo Semestre\REDES\Emisor-Receptor-Redes\Emisor-Receptor-Redes\src\main\java> java Main 010111011 1111
> Applying correction ...
> Error detected in position (7)
> Applying correction ...
> Error detected in position (9)
> Trama converted succesfully

Received trama: [0, 1, 0, 1, 1, 1, 0, 1, 1]
Converted trama: [1, 0, 0, 1, 1, 0, 0, 0, 0]
```

9. Cadena “1111” y polinomio “11011”
    - Emisor

```
priscillagonzalez@MacBook-Pro-de-Priscilla Emisor-Receptor-Redes %
Ingrese la data que desea enviar->1111
Encoded data: 11110101
```

    - Receptor
      Trama: 11000101 (Tercer y cuarto bit fueron manipulados)
      Bits de paridad: 111

```
PS D:\Documents\UVG\12vo Semestre\REDES\Emisor-Receptor-Redes\Emisor-Receptor-Redes\src\main\java> java Main 11000101 111
> Applying correction ...
> Error detected in position (3)
> Applying correction ...
> Error detected in position (5)
> Trama converted succesfully

Received trama: [1, 1, 0, 0, 0, 1, 0, 1]
Converted trama: [1, 1, 1, 1, 0, 0, 0, 0]
```

10. Cadena “0101” y polinomio “10000” (NO ENCUENTRA ERROR)
    - Emisor

```
priscillagonzalez@MacBook-Pro-de-Priscilla Emisor-Receptor-Redes %
Ingrese la data que desea enviar->0101
Encoded data: 01010000
```

    - Receptor
      Trama: 10100000 (Los primeros 4 bits fueron manipulados)
      Bits de paridad: 101

```
PS D:\Documents\UVG\12vo Semestre\REDES\Emisor-Receptor-Redes\Emisor-Receptor-Redes\src\main\java> java Main 10100000 101
> Trama converted succesfully

Received trama: [1, 0, 1, 0, 0, 0, 0, 0]
Converted trama: [1, 0, 1, 0, 0, 0, 0, 0]
```

Discusión

El presente laboratorio se realizó con los algoritmos de corrección y detección de errores, Código Hamming y CRC respectivamente, los cuales fueron brindados en el documento de instrucciones del laboratorio. Se implementaron ambos algoritmos tanto para el receptor, como para el emisor.

Para los casos de prueba 1 - 3, los resultados obtenidos del emisor fueron proporcionados directamente al receptor sin ningún tipo de manipulación. En todos los casos, el receptor fue capaz de detectar que no hubo errores al momento de revisar errores en la trama provista.

Para los casos de prueba 4 - 6, los resultados obtenidos del emisor fueron manipulados, modificando un bit de la data original para comprobar la eficiencia del receptor verificando y corrigiendo errores. En los tres casos, el receptor fue capaz de identificar que el mensaje no estaba correcto y fue capaz de identificar las posiciones de los bits en las que pudo haber error. Al cambiar el bit en la posición indicada, el proceso se aplicó nuevamente y las tramas fueron aceptadas por el receptor, lo que indica que la trama fue corregida exitosamente.

Para los casos de prueba 7 - 9, al receptor se le proporcionó los resultados obtenidos por el emisor, con dos bits modificados, los cuales se indican en las pruebas respectivas. En los tres casos el receptor fue capaz de identificar correctamente las posiciones de los bits que tuvieran error, corrigiendo y revisando la trama de manera constante hasta encontrar la trama correcta o alcanzar un límite de ciclo.

Para la prueba 10, se hizo una prueba especial para identificar si el receptor era capaz de identificar una cadena que estuviera manipulada con un polinomio que es propenso a tener errores. En este caso, el receptor no logró identificar que hubo error. Esto se debe a que el polinomio consiste en un 1 en la primera posición seguido de n bits con valor de 0. Cuando el emisor revisa la entrada provista y la convierte, esta resulta en: la entrada provista seguida de n-1 bits con 0, es decir, de la manera inicial como se intenta aplicar el algoritmo CRC. El receptor por su parte también obtiene el mismo resultado al aplicar la comprobación de errores, ya que aplica el mismo polinomio. Esto resulta en que los bits calculados para CRC sean nuevamente n bits con valor de 0, lo que resulta en que el receptor no identifique correctamente que hubo manipulación en alguna/s posición/es en los bits de data.

Comentario grupal

A través de esta práctica se ha comprendido de una mejor manera el funcionamiento y la importancia de los algoritmos de detección y corrección de errores en la transmisión y recepción de datos en los sistemas de comunicación. Además, la implementación de estos algoritmos nos ha permitido aplicar conocimientos teóricos en un contexto práctico. Por otra parte, realizar pruebas con tramas de diferentes longitudes y diferentes niveles de errores, nos ha permitido observar cómo es que funcionan estos algoritmos en situaciones reales y a su vez hemos comprendido que la elección del algoritmo adecuado depende de las necesidades específicas de la aplicación que se tenga o del programa que se esté manejando.

Conclusiones

1. Los algoritmos de corrección y detección de errores, como el Código Hamming y CRC, son herramientas fundamentales en la transmisión de datos para garantizar la integridad y confiabilidad de la información.
2. Los resultados de las pruebas realizadas demuestran que los algoritmos son efectivos en la detección y corrección de errores. En los casos donde se manipuló la data, el receptor fue capaz de identificar las posiciones de los bits que daban error y así corregir la trama de manera exitosa.
3. Se observó que en ciertos casos, la elección de un polinomio CRC específico puede afectar la capacidad del receptor para detectar errores. Es esencial seleccionar un polinomio robusto que minimice la probabilidad de que los errores pasen por desapercibido.
4. La implementación de dichos algoritmos en diferentes lenguajes de programación y la verificación de que el receptor en un lenguaje pueda validar el mensaje codificado por el emisor en otro lenguaje, demostró la flexibilidad de estos algoritmos.

Citas y referencias

- *Comprobación de redundancia cíclica en Python – Barcelona Geeks*. (2022, 5 de julio). Barcelonageeks.com. https://barcelonageeks.com/verificacion-de-redundancia-ciclica-en-python/
- Prieto, D., Perez-Aranda, R. (2015) Study of Undetected Error Probability of IEEE 802.3 CRC-32 code for MTTFPA analysis. https://www.ieee802.org/3/bv/public/Jan_2015/perezaranda_3bv_2_0115.pdf