Universidad del valle de Guatemala
Redes

Diego Córdova: 20212
Andrea Lam: 20102

Laboratorio #2
Algoritmos de detección y corrección de errores

Descripción de la práctica

Esta práctica consistió en la implementación de 2 algoritmos, uno de corrección de errores y otro de detección de errores. Cabe destacar que se implementó un emisor y un receptor para cada algoritmo. En este caso los emisores fueron implementados en el lenguaje de programación *python* y los receptores en el lenguaje de programación *javascript*. A continuación se describen los algoritmos en cuestión:

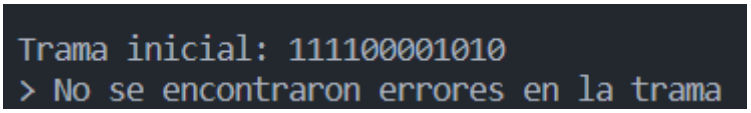
- CRC-32
  Este algoritmo de detección de errores se basa en la división polinomial de una secuencia de bits. El largo de este polinomio depende de la implementación a realizar, en este caso se utilizó un algoritmo de grado 32 dado que se implementó el algoritmo *CRC-32*. En términos prácticos, el emisor divide el mensaje original por el polinomio de grado 32 y lo manda al receptor, este realiza el mismo procedimiento y dependiendo del residuo resultante realiza la detección de errores en el envío del mensaje.

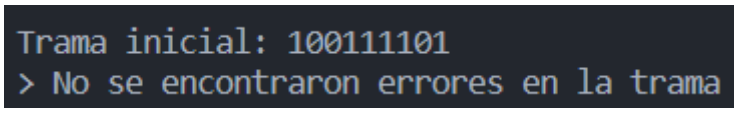
- Hamming
  Este algoritmo de corrección de errores se basa en el concepto de los bits de paridad combinado con potencias de 2 para detectar y corregir errores en mensajes enviados. En este caso específico se implementó la variante 7-4, en donde el emisor empaqueta los mensajes en grupos de 4 para luego aplicar el algoritmo de codificación a las sub-tramas individuales y luego juntarlas para mandar el mensaje. Por la naturaleza del algoritmo las tramas resultantes de la codificación tienen un largo de 7 bits. Por tanto el emisor recibe la trama total y la procesa en sub-tramas de 7 bits para lograr la detección y corrección de errores.

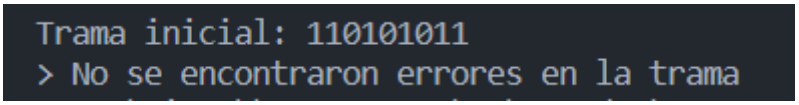
Resultados:

CRC-32

- Pruebas con trama sin errores:



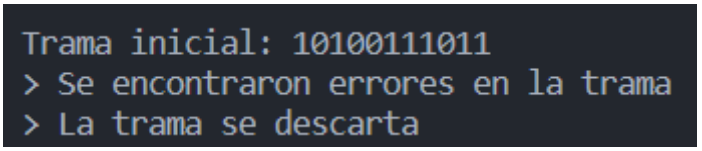


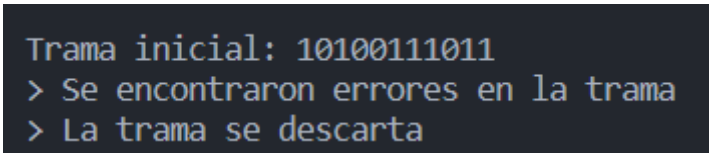


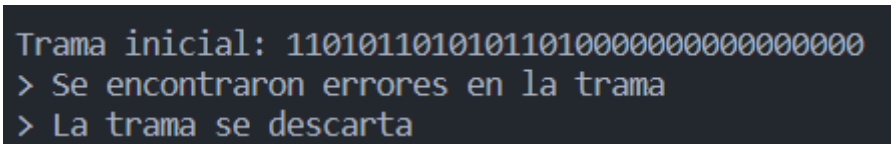
Las tramas y polinomios utilizados son los siguientes:

- Prueba 1:
  - Trama = '111100001010'
  - Polinomio = '10011'
- Prueba 2:
  - Trama = '100111101'
  - Polinomio = '10101'
- Prueba 3:
  - Trama = '110101011'
  - Polinomio = '1001'

- Pruebas con trama con errores:







Las tramas y polinomios utilizados son los siguientes:

- Prueba 1:
    - Trama = '10100111011'
    - Polinomio = '1101'
- Prueba 2:
    - Trama = '10100111011'
    - Polinomio = '1101'
- Prueba 3:
    - Trama = '11010110101011010000000000000000'
    - Polinomio = '100000100110000010001110110110111'

Hamming

- Pruebas con trama sin errores:

```
---- Pruebas con tramas correctas ----

Trama inicial: 0110011
> No se encontraron errores en la trama

Trama inicial: 1001100
> No se encontraron errores en la trama

Trama inicial: 1010010
> No se encontraron errores en la trama
```

Para estas pruebas se utilizaron las siguientes tramas: 0110011, 1001100 y 1010010

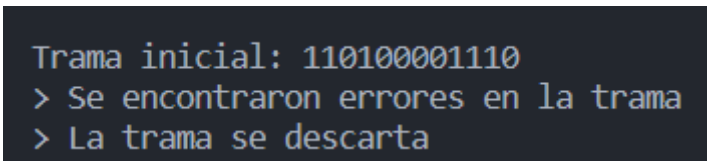
- Pruebas con trama con errores:

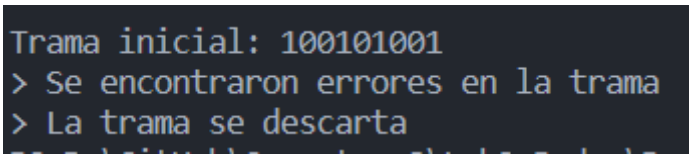
```
---- Pruebas con tramas con error ----
Trama inicial: 0100011
> Se encontraron errores en el bit: 5
Trama correcta: 0110011

Trama inicial: 1001110
> Se encontraron errores en el bit: 2
Trama correcta: 1001100

Trama inicial: 0010010
> Se encontraron errores en el bit: 7
Trama correcta: 1010010
```

Para estas pruebas se utilizaron las siguientes tramas: 0100011, 1001110 y 0010010

Pruebas Cambiando 2 bits

- CRC-32



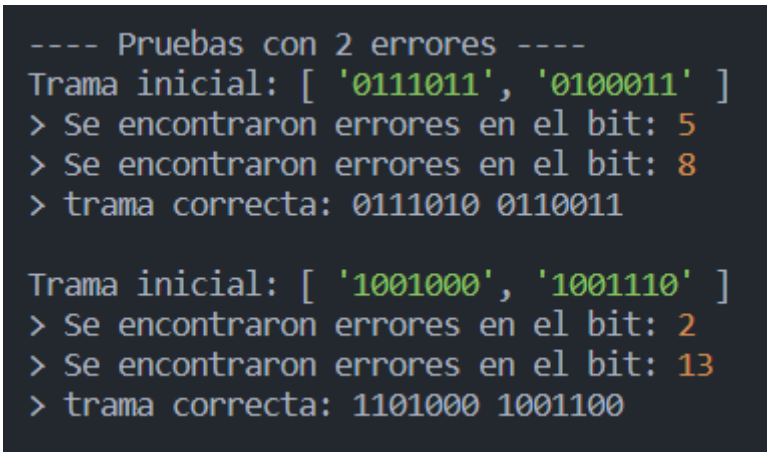


Las tramas y polinomios utilizados son los siguientes:

- Prueba 1:
  - Trama = '11010000111 0'
  - Polinomio = '10011'
- Prueba 2:
  - Trama = '100101001'
  - Polinomio = '10101'

Los bits cambiados se marcan en rojo.

- Hamming



Las tramas y polinomios utilizados son los siguientes:

- Prueba 1: 0111011 0100011
- Prueba 2: 1001000 1001110

Los bits cambiados se marcan en rojo.

Pruebas donde no se detectan errores

- CRC-32

```
Trama inicial: 00100111011
> No se encontraron errores en la trama
PS D:\GitHub\Semestre-8\Lab2-Redes\Receptor>
```

```
Trama inicial: 101100001000
> No se encontraron errores en la trama
```

Las tramas y polinomios utilizados son los siguientes:

- Prueba 1:
  - Trama = 00100111000
  - Polinomio = '1101'
- Prueba 2:
  - Trama = 101100001000
  - Polinomio = '10101'

Los bits cambiados se marcan en rojo.

- Hamming

```
---- Pruebas con cambios sin errores ----
Trama inicial: [ '1010010' ]
> No se detectaron errores en la trama

Trama inicial: [ '1100110' ]
> No se detectaron errores en la trama
```

Las tramas y polinomios utilizados son los siguientes:

- Prueba 1: 1010010
- Prueba 2: 1100110

Los bits cambiados se marcan en rojo.

Discusión

- Análisis exhaustivo de los resultados, errores, algoritmos etc..
    - Hamming
        - Para el programa de Hamming se utilizaron como tramas correctas las siguientes: '0110011', '1001100', '1010010'. Las cuales como se esperaba dieron que era una trama correcta ya que en el código del receptor del hamming está la función “process_hamming” la cual se encarga de verificar si hay errores en una subtrama, para eso se calcula el bit de la paridad para las posiciones 0,1,2 y se comparan con los bits de paridad que están almacenados en la trama recibida.
        - Seguidamente para las tramas que dan error se utilizaron las siguientes: '0100011', '1001110', '0010010'. Las cuales como se mencionó anteriormente dan error. Esto pasa ya que el algoritmo de Hamming usa los bit de paridad para detectar los errores, para este codigo se utilizo el Hamming (7,4), donde se trabaja con tramas de 7 bits y 4 bits de datos.
            - La trama 0100011, está mal porque el bit de la paridad en la posición 0 debería ser 1 para que el número total de bits 1 sea par. Pero en esta trama hay 3 bits 1 en las posiciones 2,3 y 7, lo que da como resultado un número impar de bits 1.
            - La trama 1001110, está mal ya que el bit de paridad en la posición 1 debería ser 1 para que el número total de bits de 1 sea par.
            - La trama 0010010, está mal porque el bit de paridad en la posición 2 debería ser 1, para que el número de bits totales sea par.
    - CRC-32
        - Para el programa de CRC-32 se utilizaron las siguientes tramas correctas : Trama = '111100001010' Polinomio = '10011' , Trama = '100111101' Polinomio = '10101' , Trama = '110101011' Polinomio = '1001'. En las cuales como se mencionó anteriormente dieron que fueron tramas correctas.Esto se debe a que en la función “process_trama” convierte la trama y el polinomio en una representacion de bits y luego realiza el cálculo del CRC. Seguidamente en la función trama_xor se realizan las operaciones XOR utilizando el polinomio dado. Después en la función “get_usable_trama” se separa la trama original en dos partes, la que se puede utilizar la parte “útil” y la otra parte que es para operar. Por último en la función “crc” se imprimen la trama original y el resultado de estas que es el correcto.

- Seguidamente para las tramas que dan error se utilizaron las siguientes: Trama = '10100111011' Polinomio = '1101' ,
  Trama = '10100111011' Polinomio = '1101' ,
  Trama = '11010110101011010000000000000000' Polinomio = '10000010011000001000111011011 0111'.
  - Trama = '10100111011', Polinomio = '1101'. Esta trama da error debido a que al calcular el CRC la trama al convertir en bits de [1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1] y el polinomio al convertir en bits de [1, 1, 0, 1], entonces en el proceso del calculo de crc da como resultado el siguiente arreglo: [0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1]. Lo cual indica que se han encontrado errores en esas casillas de la trama, ya que para que este correcto el valor del CRC tiene que dar en todas las posiciones 0.
  - Trama = '11010110101011010000000000000000', Polinomio = '10000010011000001000111011011 0111'. Esta trama da error debido a que al calcular el CRC la trama al convertir en bits de [1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] y el polinomio al convertir en bits de [1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1], entonces en el proceso del calculo de crc da como resultado el siguiente arreglo: [1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0]. Lo cual indica que se han encontrado errores en esas casillas de la trama, ya que para que este correcto el valor del CRC tiene que dar en todas las posiciones 0.

- ¿Hubo casos donde no fue posible detectar errores? ¿Por qué?
    - CRC-32
        - Para las tramas modificadas para que dieran error y el código no las denoto se pudo dar los siguientes casos:
            - Prueba 1: Trama = '00100111000', Polinomio = '1101'
              Al calcular el CRC se obtuvieron los siguientes resultados: Trama convertida a bits: [0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0] Polinomio convertido a bits: [1, 1, 0, 1]
              Lo cual al realizar con esos datos el cálculo del CRC da como resultado [0, 0, 0, 1, 0, 1, 1, 1] . Este resultado lo que indica es que no hay errores en la trama, ya que el valor obtenido es igual a cero.
            - Prueba 2: Trama = '101100001000', Polinomio = '10101'
              Al calcular el CRC se obtuvieron los siguientes resultados: Trama convertida a bits: [1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0] Polinomio convertido a bits: [1, 0, 1, 0, 1]
              Lo cual al realizar con esos datos el cálculo del CRC da como resultado [0, 0, 0, 0, 0, 0, 0, 0]. Este resultado lo que indica es que no hay errores en la trama, ya que el valor obtenido es igual a cero.
    - Hamming
        - Para las tramas modificadas para que dieran error y el código no las denoto esto se dio ya que el codigo Hamming (7,4) es un codigo para detección y corrección de solamente 1 error. Ya que este solamente puede detectar y corregir un error en una trama de 7 bit, pero al tener más de un error o si los errores ocurren en bits diferentes a los que el algoritmo puede corregir, este no los detecta

<u>Comentario grupal sobre el tema (errores)</u>

Se tenía pensado en un inicio que sería más difícil la creación de los códigos para su verificación seguidamente, pero al momento de estar realizando la lógica de los códigos de detección y corrección, y seguidamente las pruebas, nos percatamos que es más difícil la lógica para no dejar ir tantos errores y verificar los tipos de paridad.

Conclusiones

Los polinomios generados por el cálculo del CRC son críticos, ya que su elección es fundamental para la detección de errores. Ya que un polinomio diferente puede dar lugar a un resultado diferente en el cálculo del CRC. Por lo tanto es importante verificar que el emisor y el receptor utilicen el mismo polinomio es esencial.

Cuando se implementan algoritmos como el CRC o alguna otra técnica de detección de errores, es esencial realizar una gran cantidad de pruebas y depuración para asegurarse de que funcione correctamente en diferentes escenarios.

Los bits de paridad son unos elementos importantes para la implementación de algoritmos de detección y corrección de errores, ya que se utilizan para verificar si la integridad de los datos transmitidos es correcta.

El algoritmo de Hamming demuestra la importancia de la implementación de mecanismos de detección y corrección de errores en sistemas de transmisión de datos. Logrando que este tipo de algoritmos garanticen que los datos se transmiten con alta confiabilidad.

Citas y Referencias

CRC-32. (s. f.). https://fuchsia.googlesource.com/third_party/wuffs/+/HEAD/std/crc32/README.md

Invarato, R. (2017, 12 agosto). *Código de Hamming: detección y corrección de errores - Jarroba*.

Jarroba. https://jarroba.com/codigo-de-hamming-deteccion-y-correccion-de-errores/