Manuel Archila 161250
Juan Diego Avila 20090

Esquema de Detección y Corrección de Errores

Descripción de la Práctica

Para la elaboración de esta práctica se realizaron implementaciones de dos algoritmos, uno de detección de errores y otro de corrección de errores. Para el algoritmo de detección de errores se utilizó el algoritmo CRC-32. Este algoritmo en el lado del emisor, se solicita una cadena entrante en binario a la cual se le suman a la derecha 32 bits seteados en cero. A esta nueva cadena se le aplica una serie de operaciones XOR hasta lograr codificar todo el mensaje. Luego por parte del receptor, es necesario aplicar el proceso inverso. Entonces al recibir la cadena codificada hay que hacer nuevamente la operación XOR en todos los bits. Si al final de la operación todos los bits resultantes tienen el valor de 0 no hay errores en el mensaje, de haber un bit que tiene el valor de 1, el mensaje contiene un error.

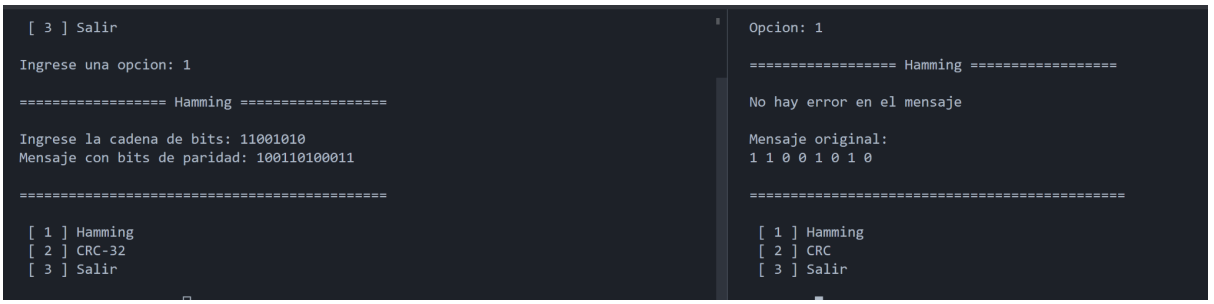
Por otro lado, para el algoritmo de detección de errores se usó el algoritmo de Hamming. Este algoritmo consiste en recibir una cadena de bits, para luego evaluar la desigualdad $2^p >= p + i + 1$, donde p son los bits de paridad e i es la longitud de bits de la trama original. Para el valor de p se utiliza el valor más pequeño que cumpla con la desigualdad. Como siguiente paso es necesario reacomodar el mensaje. Esto se hace obteniendo el tamaño de la nueva cadena, el cual tiene tamaño data+p y luego colocar los bits de izquierda a derecha dejando vacías los bits en posiciones de potencias de dos. Conociendo cuántos bits de paridad se tienen, es posible realizar la tabla de paridad. Esta tabla tiene la función de determinar en base a cuáles bits se realizará el cálculo del valor respectivo de cada bit de paridad. Al tener estos valores, el mensaje podrá ser codificado para que el emisor pueda enviarlo.

En este caso el receptor necesita hacer el proceso inverso. Para esto se comenzará haciendo la tabla de paridad para encontrar en base a qué bits están calculadas las paridades. Después, de tener la tabla y calcular la paridad de los bits, si se tienen que todos los bits son 0 indica que no hay error. En caso de que los bits de paridad no sean todos 0 es necesario convertir los bits de paridad en un número binario y el número resultante convertido a decimal es el bit que hay que cambiar para corregir el mensaje.

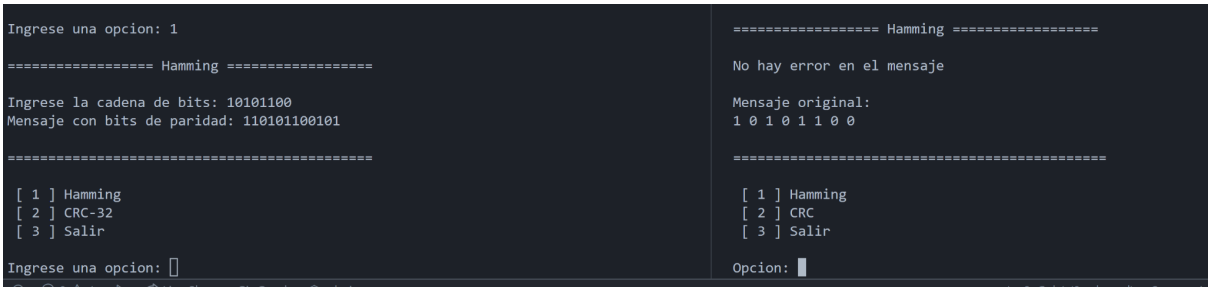
Resultados

Hamming:

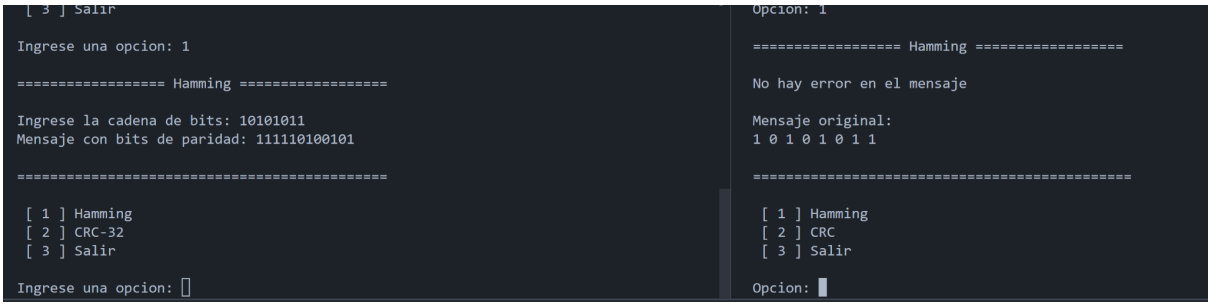
- Tramas sin manipular:
    - 11001010



    - 10101100

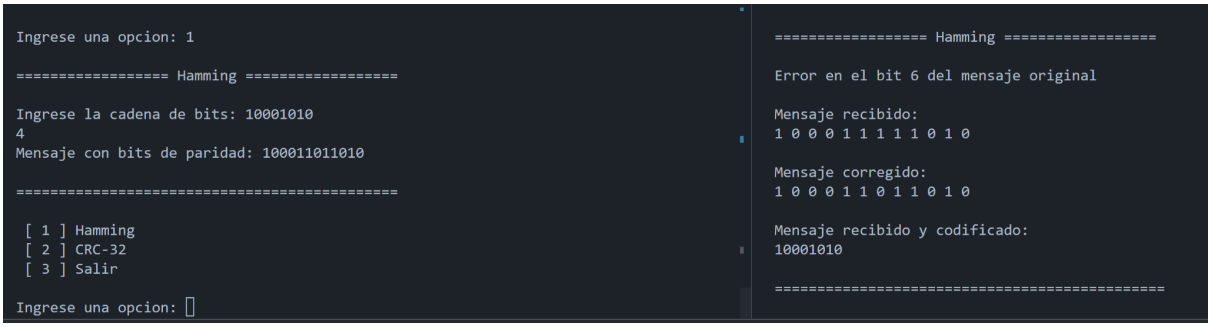


    - 10101011

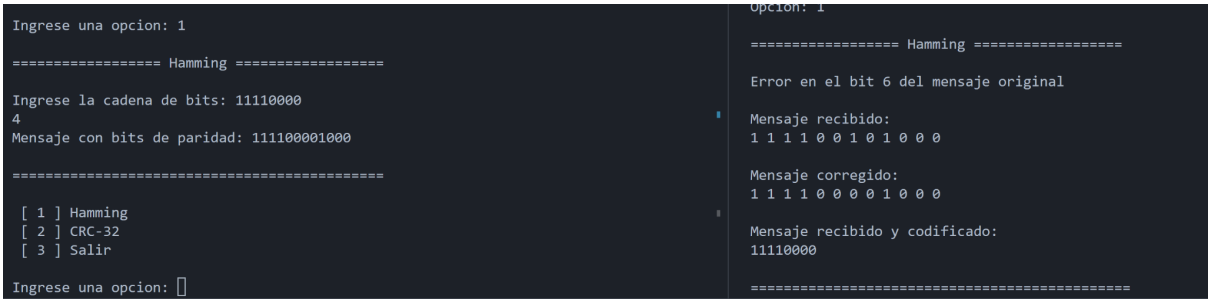


- Tramas con 1 bit manipulado:
    - Original: 11100010
    - Bit modificado en mensaje con bits de paridad: 111010010001

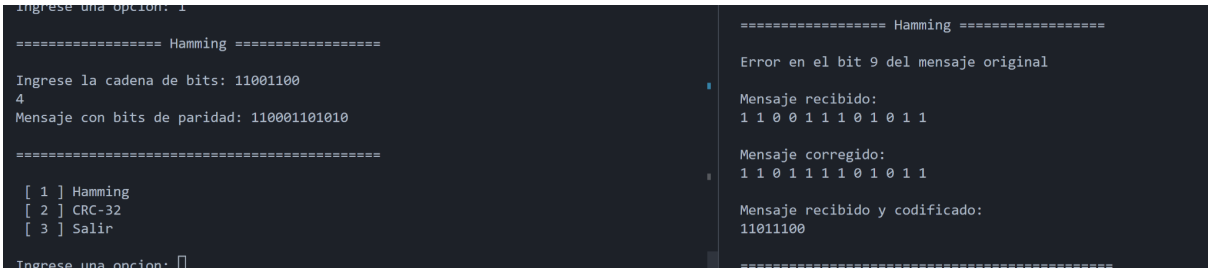
    - Original: 10001010
    - Bit modificado en mensaje con bits de paridad: 100011111010



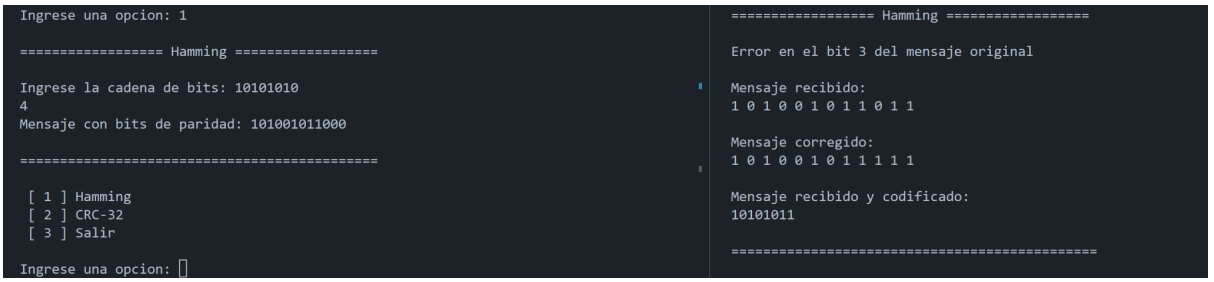
    - Original: 11100010
    - Bit modificado en mensaje con bits de paridad: 111010010001



- Tramas con 2 bits manipulado:
    - Original: 11001100
    - Bits modificados en el mensaje con bits de paridad: 110011101011



- Original: 10101010
- Bits modificados en el mensaje con bits de paridad: 101001011011



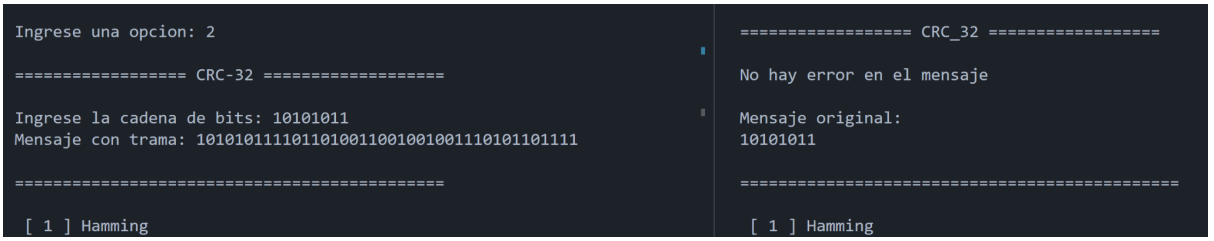
- Original: 10010010
- Bits modificados en el mensaje con bits de paridad: 100100010101

Crc_32:

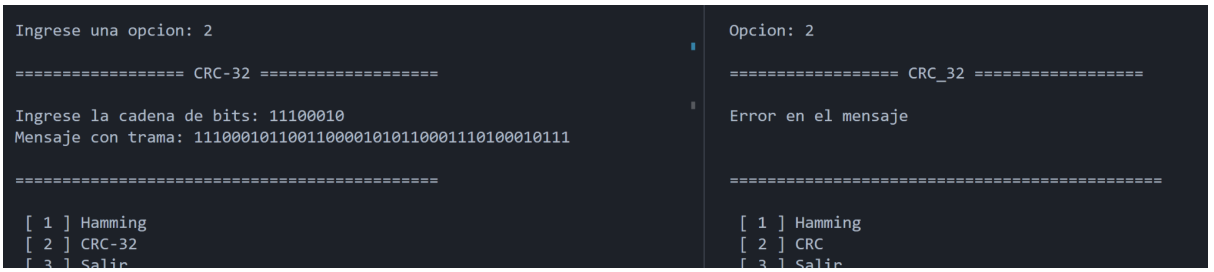
- Tramas sin manipular:
  - 11001010

  - 10101100

  - 10101011

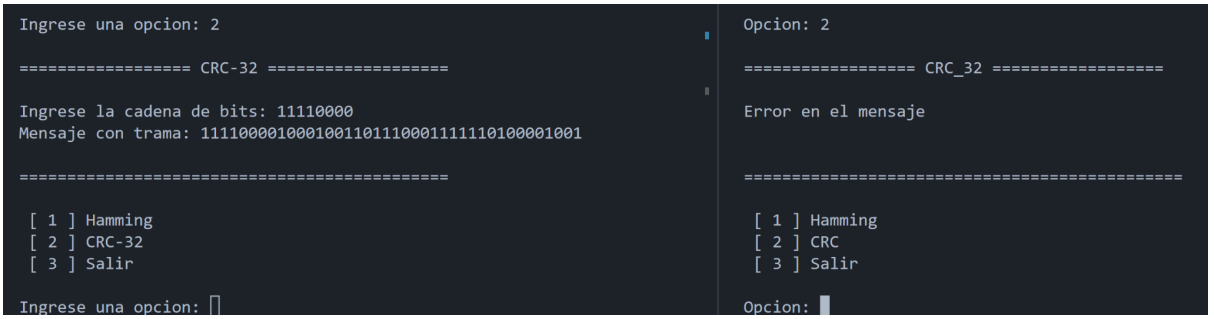


- Tramas con 1 bit manipulado:
  - Original: 11100010
  - Bit modificado en mensaje con trama: 11101010110011000010101100011101000101111

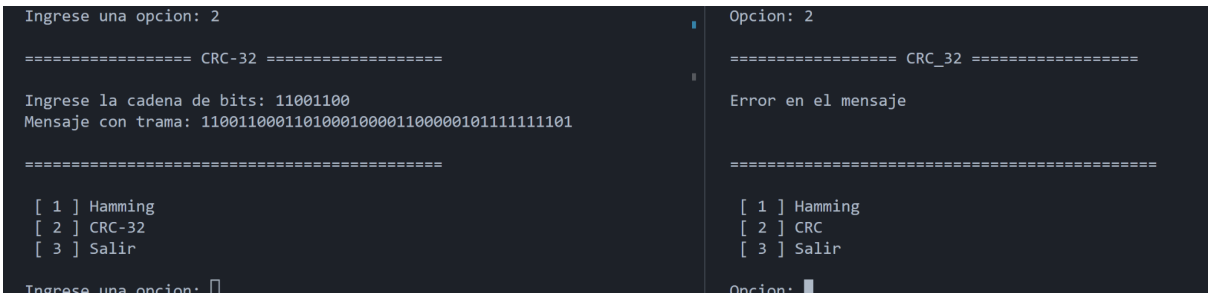


  - Original: 10001010
  - Bit modificado en mensaje con trama: 11001010010001101000011000110110001110000

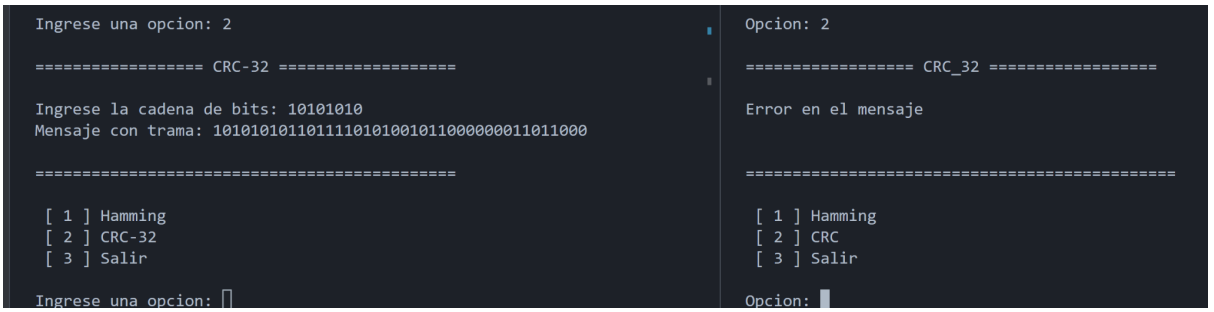
  - Original: 11100010
  - Bit modificado en mensaje con trama: 11110000100010011011100011111101000010 11



- Tramas con 2 bits manipulado:
  - Original: 11001100
  - Bits modificados en el mensaje con bits de paridad: 11111100011010001000011000001011111111101

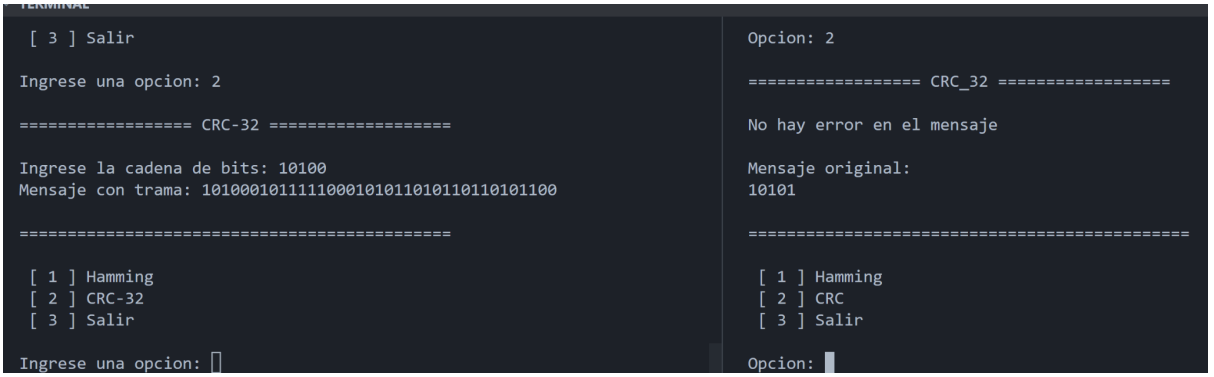


- Original: 10101010
- Bits modificados en el mensaje con bits de paridad: 1111101011011110101001011000000011011000



- Original: 10010010
- Bits modificados en el mensaje con bits de paridad: 1001001000101100100111111001000 1 011110000

- Trama con error indetectable:
  - Original: 1000
  - Trama original: 10100010111110001010110101101101011 00
  - Trama modificada: 10101010110111101010011000000011011



Discusión

Para comenzar ambos algoritmos tanto el de Hamming como el CRC_32 descifran correctamente los mensajes siguiendo el algoritmo descrito anteriormente para cada uno. Ya que el algoritmo de Hamming es de detección y corrección de errores, se puede observar en las capturas que para 1 bit cambiado muestra el mensaje original, el bit en el que hubo error y lo corrige dando el mensaje original de forma certera, pero con 2 bits no marca error y descifra un mensaje totalmente diferente. Esto sucede ya que al alterar dos bits ya no es posible obtener el mensaje original por la forma en la que funciona el algoritmo, por lo que cambia totalmente el contexto del mensaje y ya no lo toma como error sino descifra otro mensaje totalmente diferente. Para poder corregir más de un error es necesario implementar una versión extendida o modificada de dicho algoritmo. Este podría ser el algoritmo de Hamming extendido el cual puede detectar y corregir desde un error único hasta un error doble. Esto se logra calculando un bit extra de paridad en base a todos los bits de data y bits de paridad, el cual permite saber si hay más de un error.

En el algoritmo de Hamming no se pueden introducir errores indetectables por el algoritmo. Esto se debe al cálculo que realiza con la tabla y los bits de paridad. Este cálculo involucra revisar cada una de las posiciones necesarias en el mensaje para verificar que los bits de paridad sean los correctos. Si no se cambia totalmente de trama y de mensaje, está hecho para detectar cualquier tipo de error.

Una dificultad presentada en este algoritmo fue en la parte de descifrado. Esto sucedió debido a la creación incorrecta del número binario resultante de los bits de paridad. Cuando se crea este número es necesario ordenar los bits paridad en orden de manera que los bit de mayor paridad va hasta el lado izquierdo y continuamente descendiendo, de lo contrario el bit que se cambiaría sería uno completamente distinto.

Por otro lado, el algoritmo de CRC32 funciona en los casos de 1 o 2 bits ya que se encarga solamente de la detección de errores. Al hacer XOR con el código del polinomio a lo largo de todo el mensaje, es más fácil la detección de errores ya que depende totalmente del cálculo de esta operación y que el resultado sea 0, de lo contrario es error. Este algoritmo no muestra bit de error ni corrección ya que, aunque es fácil detectar el error, el algoritmo no está hecho para poder detectarlo por lo que no logra descifrar el mensaje original con trama corregida.

Con el algoritmo de CRC-32 existe la posibilidad de ingresar cadenas que contengan errores y no ser detectados. Por ejemplo, en la última prueba de CRC-32 la cadena contenía errores, sin embargo, no se detectaron errores y se devolvió un mensaje erróneo. Esto ocurre

debido a cómo las operaciones XOR se ejecutan en la decodificación del mensaje. Con esta cadena las operaciones XOR resultan en residuo 0, indicando que no hay error, pero como al final del algoritmo devuelve desde el primer bit de la izquierda hasta el bit que cubra el tamaño de la cadena original el mensaje devuelto es uno diferente.

Un problema que se tuvo en este algoritmo fue en la parte del cifrado del mensaje. Esto sucedía cuando el mensaje inicial se quedaba sin elementos para realizar el XOR. Esto provocaba que el receptor no funcionará correctamente al descifrar el mensaje. Cuando realizamos varias pruebas se encontró que una solución para este problema era insertar 0 hacia la izquierda del mensaje para que igualara la longitud del código dado por el polinomio. Así se pudo obtener la cadena completamente cifrada y que la detección de errores funcionara correctamente.

Comentario Grupal:

Esta actividad fue bastante interesante porque se logró concretar una idea de los distintos tipos de algoritmos existentes para el manejo de errores en los sistemas de redes. Estos algoritmos al principio parecían un reto pero en el momento de discutir el procedimiento de cada uno de los algoritmos fue fácil realizar las implementaciones. Sin embargo, este ejercicio logró dejar en claro las ventajas que tiene un algoritmo sobre otro, pero lo más importante son las limitaciones que tiene cada uno. Por ejemplo con Hamming está la capacidad de corregir errores, siendo en parte mejor que CRC-32, pero este se limita únicamente a un error. Sería interesante conocer más algoritmos que logren arreglar más de un error y ver cómo se aplican estos en la vida real para poder tener una mejor imagen de este campo de las redes.

Conclusiones

- Después de realizar las diferentes implementaciones fue posible comprender el funcionamiento de dichos algoritmos.
- Al finalizar las pruebas con los algoritmos se logró entender que ventajas, desventajas y limitaciones tenía cada uno de los algoritmos
- A pesar de que estos algoritmos son bastante conocidos no son infalibles ya que hay casos donde no es posible detectar o corregir los errores presentados.