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

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

Lab2.2 - Esquemas de detección y corrección de errores

(CC 3067) Redes
Catedrático: Jorge Andrés Yass Coy

Objetivos

- Comprender el funcionamiento de un modelo de capas y sus servicios
- Implementar sockets para la transmisión de información
- Experimentar la transmisión de información expuesta a un canal no confiable
- Analizar el funcionamiento de los esquemas de detección y corrección

Descripción

En la primera parte del laboratorio se implementaron por lo menos dos algoritmos, uno de detección y otro de corrección. En esta segunda parte se desarrollará una aplicación para la transmisión y recepción de mensajes, en base a una arquitectura de capas con distintos servicios. Para ello se consideró la realización de un socket, el cuál, a través de un emisor recibiría la data o información que el usuario pudiera ingresar, y con ello, utilizar el receptor ya sea del algoritmo de hamming o crc para poder decodificar y mostrar los mensajes que fueron transmitidos por el usuario.

Respuestas de las preguntas dadas:

- ¿Qué algoritmo tuvo un mejor funcionamiento y por qué?

  El algoritmo que tuvo un mejor funcionamiento al momento de realizar tanto la simulación, como también, el enviar un solo mensaje fue le de Hamming, esto se debe a la capacidad que tiene Hamming de poder resolver tramas o errores que esta presenta al momento de ser pasada al receptor, por lo que, a comparación del algoritmo de CRC que solamente es para una detección de errores, se podría decir que por ese mismo factor el algoritmo de Hamming tuvo un mejor funcionamiento.
- ¿Qué algoritmo es más flexible para aceptar mayores tasas de errores y por qué?

  El algoritmo de CRC , y esto se debe a que dicho algoritmo no posee la capacidad de corregirlos, solamente de identificarlos y decirle al usuario en donde se encuentra dicho error.
- ¿Cuándo es mejor utilizar un algoritmo de detección de errores en lugar de uno de corrección de errores y por qué?

  Utilizó un algoritmo de detección de errores cuando quiero detectar posibles problemas de corrupción de datos de manera eficiente, pero prefiero un algoritmo de corrección de errores cuando la integridad de los datos es fundamental y estoy dispuesto a gastar más recursos para lograrlo.

Resultados

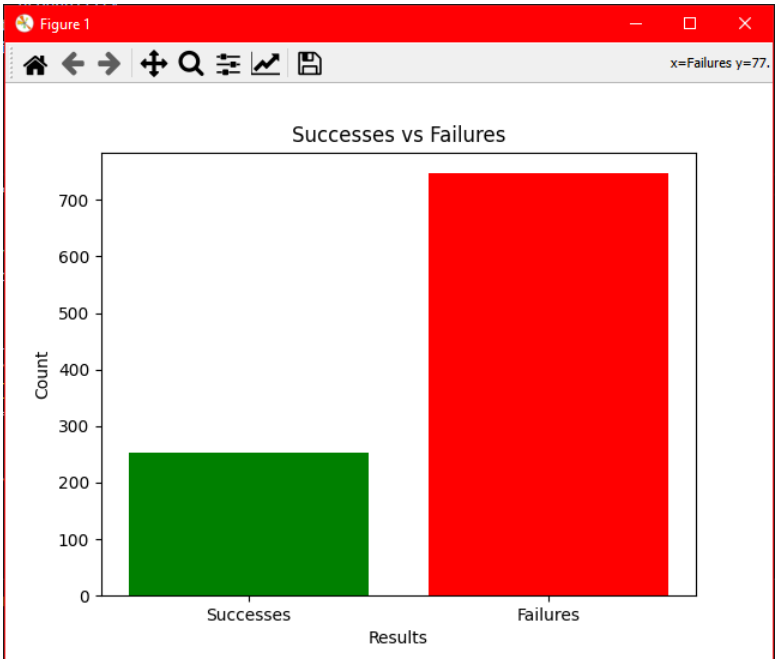

Imagen No.1: Simulación con Algoritmo CRC32

Para la primera prueba se obtuvo que el algoritmo de CRC32 obtuvo un total de 747 errores con respecto a 253 aciertos

Imagen No.2: Simulación con Algoritmo Hamming

Para la primera prueba se obtuvo que el algoritmo de Hamming obtuvo un total de 13 errores con respecto a 987 aciertos

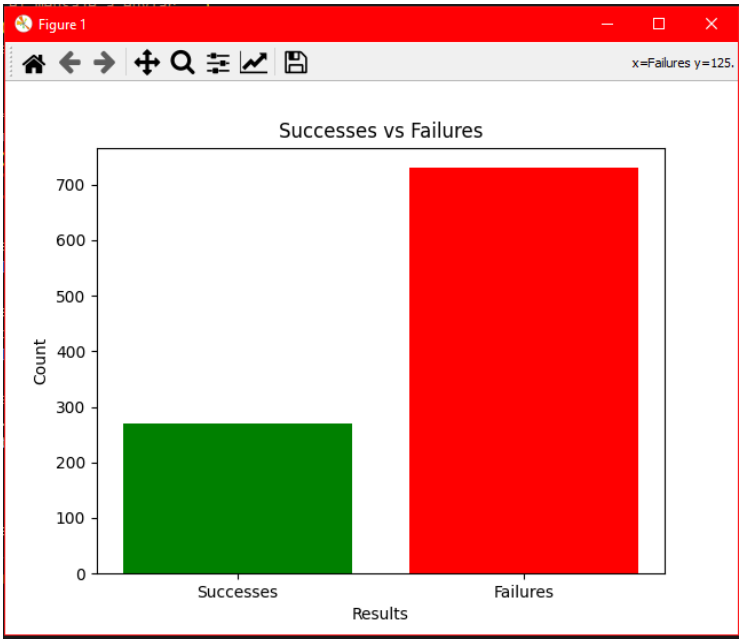

Imagen No.3: Simulación con Algoritmo CRC32

Para la segunda prueba se obtuvo que el algoritmo de CRC32 obtuvo un total de 730 errores con respecto a 270 aciertos

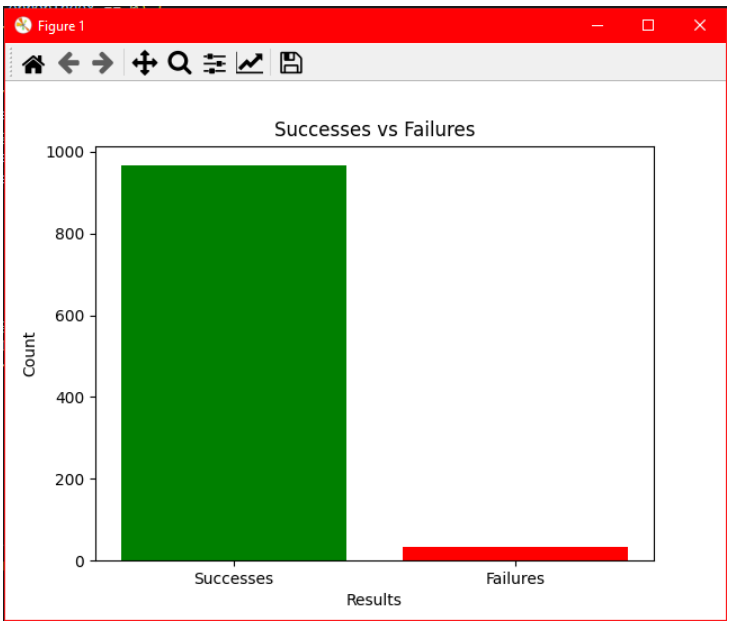

Imagen No.4: Simulación con Algoritmo Hamming

Para la segunda prueba se obtuvo que el algoritmo de Hamming obtuvo un total de 34 errores con respecto a 966 aciertos

Discusión

Para la realización de este laboratorio, se consideró realizar un socket, el cual se iba a encargar de recibir un mensaje por medio del emisor, y este, a través del receptor poder calcular si la trama recibida del emisor es correcta, y de esa forma, poder decodificarla y retornar el mensaje original que el usuario había enviado. En donde, para poder llevar la validación de su funcionamiento, se consideró realizar una simulación, en donde, se obtuvo que el algoritmo de Hamming tuvo un mejor desempeño al momento de realizar la simulación, ya que, tal y como se puede apreciar en los resultados, por ejemplo, en las imagenes “Imagen No.1” e “Imagen No.2” se tiene que Hamming posee un mejor desempeño comparado con el algoritmo CRC.

Esto se debe a que el algoritmo CRC se utiliza principalmente para la detección de errores. Su objetivo es detectar cambios accidentales en los datos debido a ruido o interferencia durante la transmisión. Puede identificar errores, pero no puede corregirlos directamente. A comparación con el algoritmo de Hamming, el cual se centra en la corrección de errores en los datos transmitidos. Está diseñado para detectar y corregir un número limitado de errores en una secuencia de bits. Esto significa que puede recuperar los bits originales incluso si algunos de ellos se corrompen durante la transmisión.

Ahora bien, dependiendo de los objetivos que se quieran para la detección de errores, se tiene que CRC es mucho más eficiente que Hamming y esto se debe a que Hamming debido a la necesidad de añadir bits redundantes para la corrección, dicho algoritmo introduce un overhead adicional en la transmisión de datos. Resultando en un mayor uso de ancho de banda y recursos. A diferencia con el algoritmo CRC, el que a pesar que introduce cierto overhead debido al cálculo del valor CRC, este es generalmente menor en comparación con los algoritmos de corrección. CRC32 se utiliza comúnmente en protocolos de red y sistemas de almacenamiento donde la detección rápida y eficiente de errores es esencial.

Por lo que se concluye que la elección entre el algoritmo de Hamming y CRC32 depende de las prioridades específicas de la transmisión de datos. Si mi objetivo principal es garantizar la integridad de los datos de manera eficiente, optaría por el algoritmo de detección CRC32. Su enfoque rápido y confiable en la detección de errores es valioso en situaciones donde la corrección no es esencial, pero la identificación temprana de posibles corrupciones es crucial.

Comentario grupal

Después de analizar detenidamente los algoritmos de detección y corrección de errores, llegamos a algunas conclusiones clave como grupo. En primer lugar, entendimos que la elección entre el algoritmo de Hamming y CRC32 depende de nuestras prioridades específicas. Si nuestra principal preocupación es detectar cambios no deseados en los datos de manera rápida y eficiente, optaríamos por CRC32. Este enfoque nos permite identificar errores y tomar medidas adecuadas, aunque no pueda corregirlos directamente.

Por otro lado, si la precisión y la corrección de los datos son esenciales, el algoritmo de Hamming se convierte en una opción valiosa. Aunque introduce cierto overhead debido a los bits redundantes, su capacidad para corregir un número limitado de errores puede ser vital en situaciones donde los datos incorrectos pueden tener consecuencias significativas.

Conclusiones

1. Al comparar el algoritmo de detección de errores CRC32 con el de corrección de errores Hamming, comprendo que la elección depende de la importancia de la precisión y la eficiencia en la transmisión de datos.

2. Si mi objetivo es identificar rápidamente cambios no deseados en los datos, opto por CRC32; mientras que si necesito corregir errores en datos críticos, elijo Hamming, aunque esto implique cierto sobrecargo.

3. Al considerar la rapidez de detección frente a la capacidad de corrección, estoy mejor preparado para tomar decisiones informadas según el contexto y los recursos disponibles.

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.