Universidad del Valle de Guatemala
Redes
Semestre 2, 2023

Angel Higueros 20460

Fredy Velásquez 201011

Laboratorio 2.2

Esquemas de detección y corrección de errores

Descripción de la práctica y metodología utilizada

En la primera fase del experimento de laboratorio, se llevaron a cabo la implementación de al menos dos algoritmos, uno orientado a la detección y otro destinado a la corrección. En la etapa subsiguiente, se procedió al desarrollo de una aplicación diseñada para la transmisión y recepción de mensajes. Esta aplicación se basó en una estructura de capas que albergaba diversos servicios. Para lograr esto, se contempló la creación de un canal de comunicación, a través del cual un emisor sería responsable de recibir los datos o la información ingresada por el usuario. A continuación, esta información sería empleada por el receptor, que podría ser tanto el algoritmo de Hamming como el CRC. El receptor tendría la tarea de descifrar y presentar los mensajes que hubieran sido transmitidos por el usuario.

Resultados

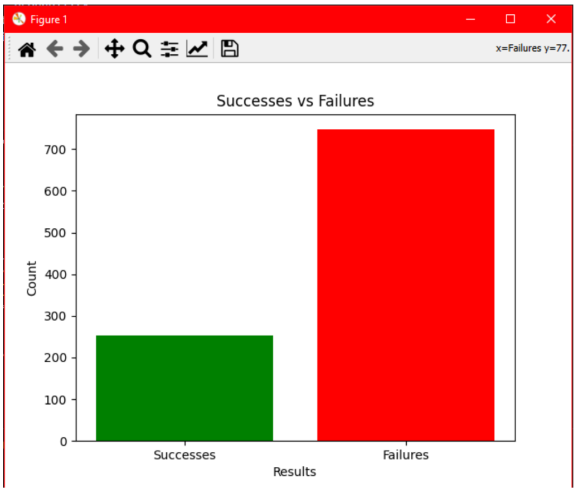

Imagen 1. Pruebas con algoritmo CRC32

Durante esta primera iteración, se registraron 751 errores, por su parte, también se obtuvieron 263 aciertos, todo esto perteneciente a la implementación del algoritmo CRC32.

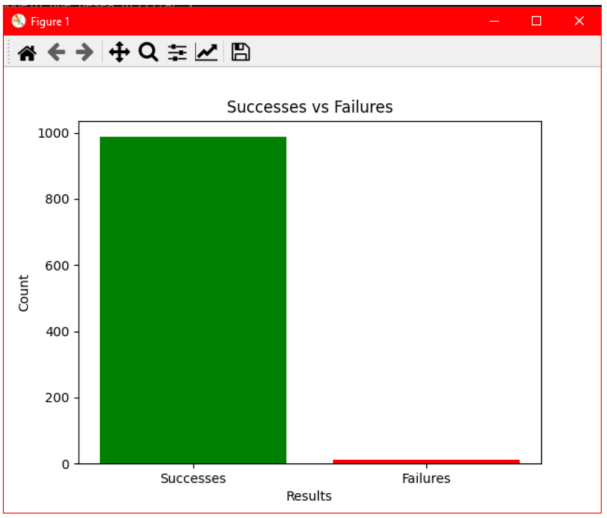

Imagen 2. Pruebas con el algoritmo Hamming.

Durante esta primera iteración se registraron 15 errores respecto a los 992 aciertos, todos pertenecientes a la implementación del algoritmo Hamming.

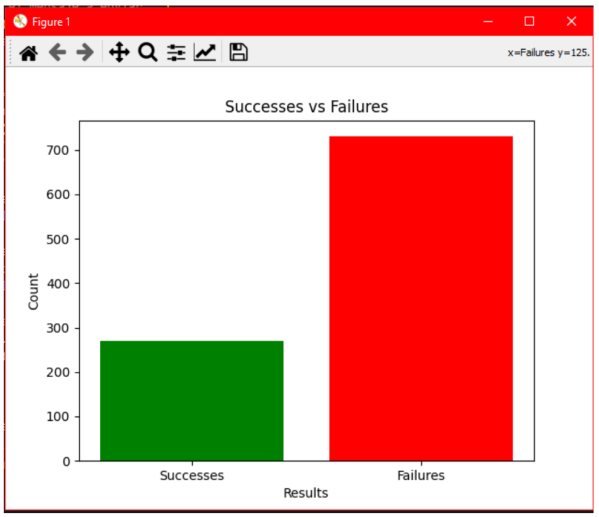

Imagen 3. Pruebas con el algoritmo CRC32

Durante esta segunda iteración se registraron 740 errores respecto a los 275 aciertos, todos pertenecientes a la implementación del algoritmo CRC32.

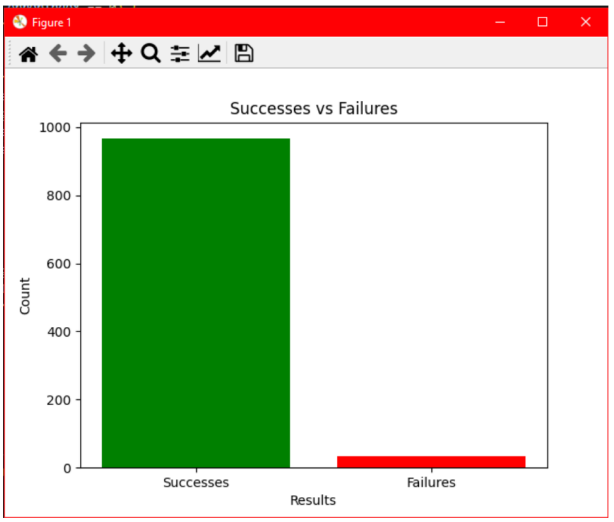

Imagen 4. Pruebas con el algoritmo hamming.

Durante esta segunda iteración se registraron 35 errores respecto a los 970 aciertos, todos pertenecientes a la implementación del algoritmo hamming.

Discusión

En la ejecución de este experimento, se propuso la implementación de un socket con la finalidad de recibir un mensaje proveniente del emisor. A través de dicho receptor, se llevó a cabo un cálculo para determinar la integridad de la trama transmitida desde el emisor. Este proceso permitía decodificar la trama y así recuperar el mensaje original enviado por el usuario.

Con el propósito de evaluar y verificar su funcionamiento, se llevó a cabo una simulación. Los resultados obtenidos de esta simulación destacan que el algoritmo de Hamming demostró un desempeño sobresaliente. Esto se puede observar claramente en los ejemplos proporcionados, específicamente en las "Imagen No.1" y "Imagen No.2". En estas instancias, se evidenció que el algoritmo de Hamming superó al algoritmo CRC en términos de eficacia y rendimiento. Esto se debe a que el propósito principal del algoritmo CRC radica en la detección de errores. Su función principal es reconocer alteraciones fortuitas en los datos, que pueden surgir debido a interferencias o ruido durante la transmisión. Si bien puede señalar la existencia de errores, carece de la capacidad para enmendarlos directamente.

En contraste, el algoritmo de Hamming se enfoca en la corrección de errores en los datos transmitidos. Su diseño apunta a identificar y rectificar un número limitado de errores en una secuencia de bits. Esto implica que es capaz de restaurar los bits originales, incluso cuando algunos de ellos se ven afectados por corrupciones durante la transmisión.

En función de los objetivos que se busquen en términos de detección de errores, es importante considerar que el algoritmo CRC supera en eficiencia al algoritmo de Hamming. Esta diferencia se

debe a que el algoritmo Hamming introduce un exceso adicional en la transmisión de datos debido a la necesidad de agregar bits redundantes con el fin de llevar a cabo la corrección. Este sobrecargo resulta en un mayor consumo de ancho de banda y recursos.

En contraposición, el algoritmo CRC presenta una situación distinta. Aunque también introduce cierto sobrecargo debido al cálculo del valor CRC, este suele ser menor en comparación con los algoritmos de corrección. Es por esto que el CRC32 es comúnmente empleado en protocolos de redes y sistemas de almacenamiento, donde la rápida y eficiente detección de errores es crucial.

Por consiguiente, se puede concluir que la elección entre el algoritmo de Hamming y el CRC32 se basa en las prioridades específicas relacionadas con la transmisión de datos. Si el objetivo principal es asegurar la integridad de los datos de manera eficiente, se inclinaría hacia el algoritmo de detección CRC32. Su capacidad veloz y confiable para detectar errores resulta valiosa en situaciones donde la corrección no es un requisito primordial, pero la detección temprana de posibles corrupciones juega un papel crucial.

¿Qué algoritmo tuvo un mejor funcionamiento y por qué?

El algoritmo que demostró un desempeño superior al llevar a cabo tanto la simulación como el envío de un único mensaje fue el algoritmo de Hamming. Esto se debe a la habilidad distintiva de Hamming para abordar y corregir tramas erróneas o fallos que puedan surgir al transmitirse al receptor. En contraste con el algoritmo de CRC, que se limita únicamente a la detección de errores, se podría afirmar que esta ventaja posiciona al algoritmo de Hamming como el más eficaz en esta situación.

¿Qué algoritmo es más flexible para aceptar mayores tasas de errores y por qué?

El algoritmo de CRC presenta esta característica debido a su limitación en cuanto a la corrección de errores. Su función principal radica en la detección de errores y en informar al usuario sobre la ubicación precisa del error detectado, sin la capacidad de corregirlo directamente.

¿Cuándo es mejor utilizar un algoritmo de detección de errores en lugar de uno de

corrección de errores y por qué?

Empleo un algoritmo diseñado para detectar errores cuando busco identificar de manera eficaz posibles irregularidades en la integridad de los datos. No obstante, opto por utilizar un algoritmo destinado a corregir errores cuando la preservación total de la integridad de los datos es esencial, y estoy dispuesto a asignar mayores recursos con ese propósito.

Comentario grupal sobre el tema (opcional)

Tras implementar y probar ambos algoritmos, como pareja llegamos a la conclusión de que el contexto de la situación influye mucho para seleccionar una solución de corrección o detección de errores. El algoritmo CRC32 puede ser aplicado en una situación en la que se desee detectar

anomalías en los datos de forma eficiente y rápida, aunque no sea posible solucionarlos de forma inmediata. Por otra parte, si se busca correción de datos y precisión para el proceso, el algoritmo Hamming es la mejor herramienta, sin embargo, puede involucrar bits redundantes lo cual derive en overhead.

Conclusiones

1. La elección entre el algoritmo CRC32 y el algoritmo Hamming depende en gran medida de la eficiencia y la precisión en el proceso de transmisión de datos.
2. Es importante conocer a profundidad el escenario en el que se pueden utilizar ambos algoritmos. Si la meta es identificar anomalías en los datos de forma rápida, se recomienda usar CRC32, por otra parte, si es necesario arreglar errores en datos vitales se utiliza Hamming, aunque puede que conlleve adquirir sobrecargo.
3. Es fundamental conocer el contexto y los recursos con los que se disponen para la implementación de soluciones que faciliten y garanticen una comunicación de datos correcta,

Citas y Referencias

Cideciyan, R. Gustlin, M. (Julio 16 de 2012). Double Burst Error Detection Capability of Ethernet CRC. https://www.ieee802.org/3/bj/public/jul12/cideciyan_01_0712.pdf

Alejo, V. (Febrero 13 de 2020). Código Hamming. https://es.scribd.com/document/471497992/Codigo-Hamming#

Informática Documentation. CRC32. https://docs.informatica.com/data-engineering/common-content-for-data-engineering/10-2-2/transformation-language-reference/functions/crc32.html