Universidad del Valle de Guatemala
18 Avenida 11-95 Guatemala
Facultad de Ingeniería
Departamento de Computación

Laboratorio 2-parte 2

Integrantes:
Astrid Marié Glauser Oliva,21299
Osmin Josue Sagastume Orellana, 18173

Curso:
Redes

Sección:
Sección 10

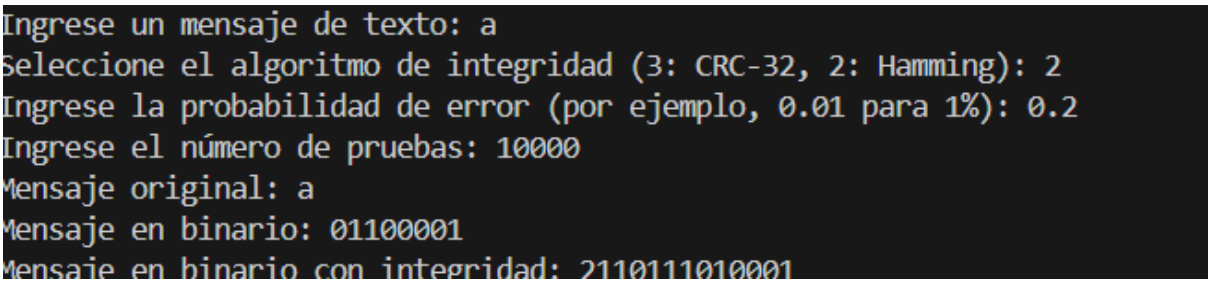
Guatemala, 1 de agosto de 2024

Descripción de la práctica y metodología utilizada

En esta práctica, se ha implementado un análisis exhaustivo de algoritmos de detección y corrección de errores en la transmisión de datos, utilizando un emisor desarrollado en Python y un receptor en Java. El emisor cuenta con la capacidad de utilizar algoritmos como CRC32 y Hamming para procesar los datos antes de su transmisión. Este emisor genera, codifica y envía los mensajes, además de realizar pruebas automatizadas a gran escala (10k, 100k+), asegurando la robustez de los resultados. Durante estas pruebas, se varían el tamaño de las cadenas enviadas, la probabilidad de error y la redundancia/tasa de código para evaluar la efectividad de cada algoritmo. El receptor en Java, por su parte, se encarga de recibir, decodificar y verificar la integridad de los mensajes, detectando y corrigiendo errores cuando es posible. Los datos generados se someten a un análisis estadístico y se visualizan mediante gráficos de pastel, barras y histogramas, permitiendo evaluar la eficacia de cada algoritmo en función de la tasa de errores detectados y corregidos, considerando la interoperabilidad entre el emisor en Python y el receptor en Java.

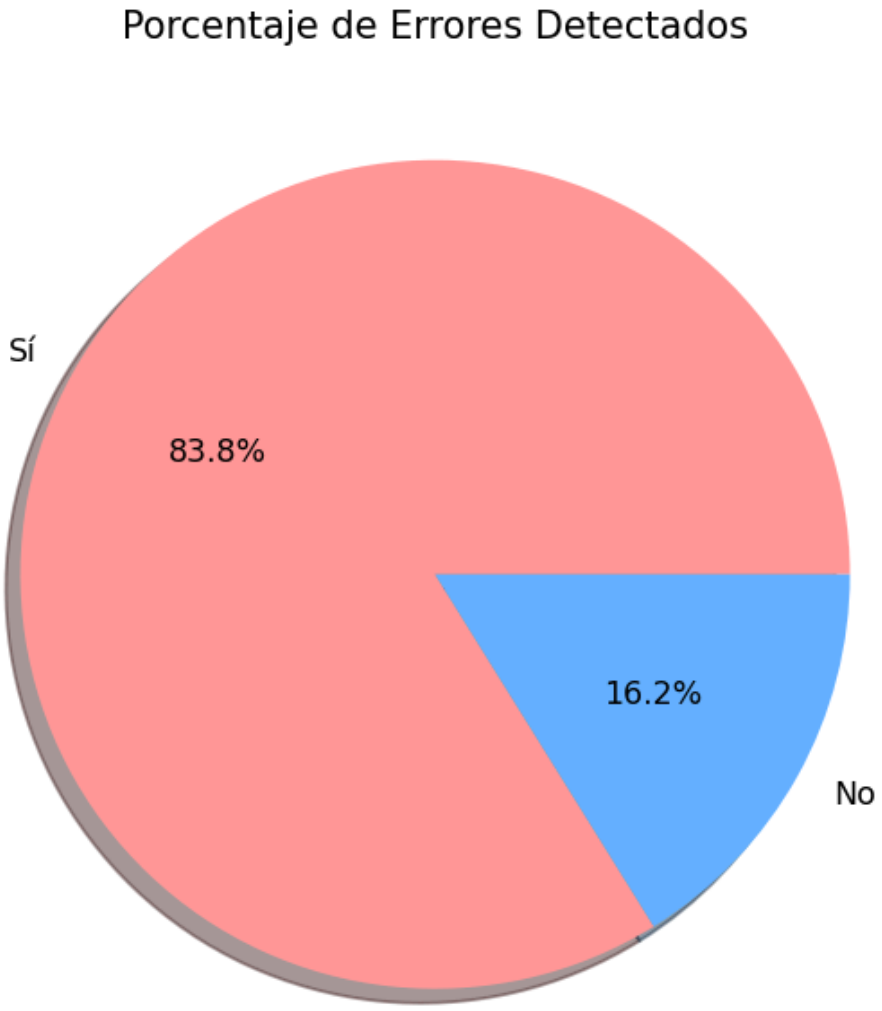
Resultados

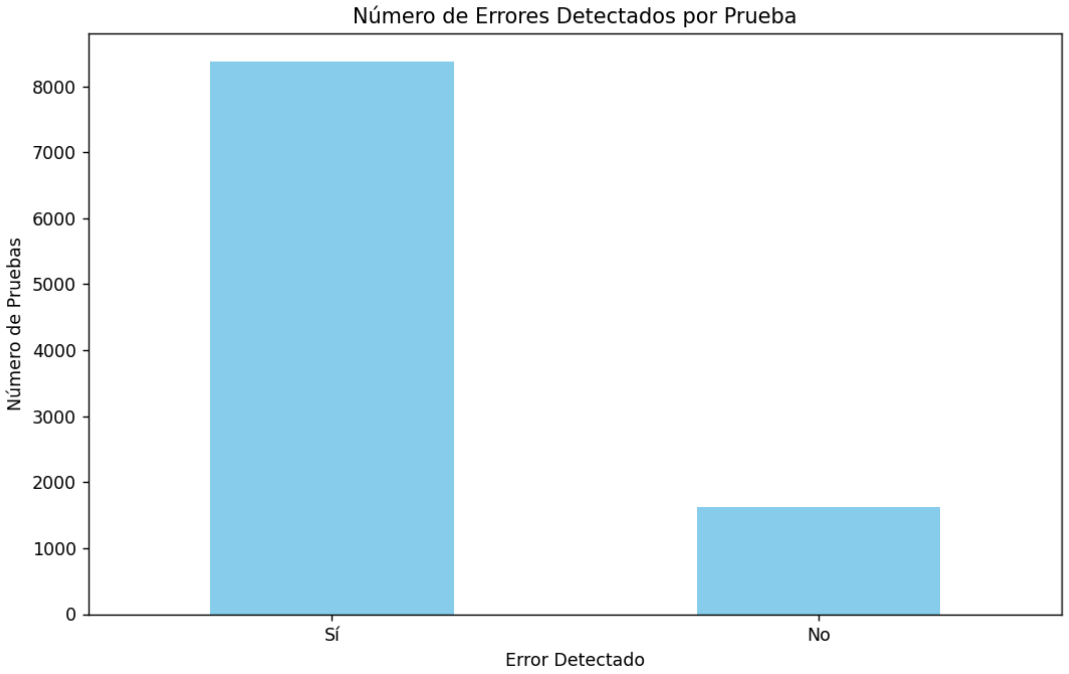
Archivo:Prueba1.csv



Mensaje : a
algoritmo:Hamming
probabilidad: 0.2
número de pruebas 10000
Mensaje binario: 01100001

Nota: Se le agrega 2 o 3 para que el receptor sepa de qué algoritmo se trata


Porcentaje de Errores Detectados
Sí
83.8%
16.2%
No

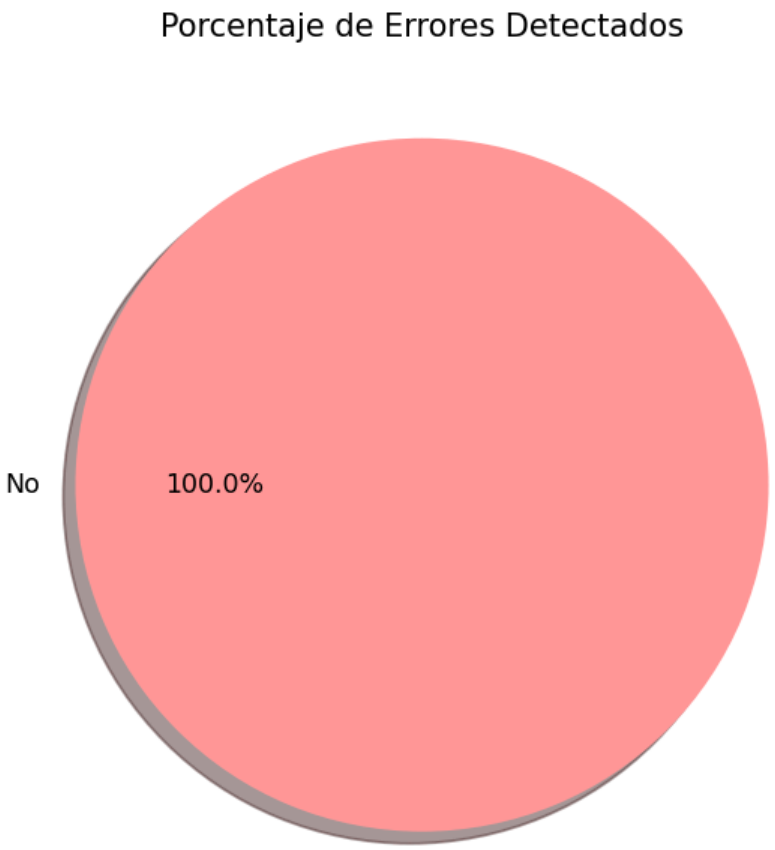


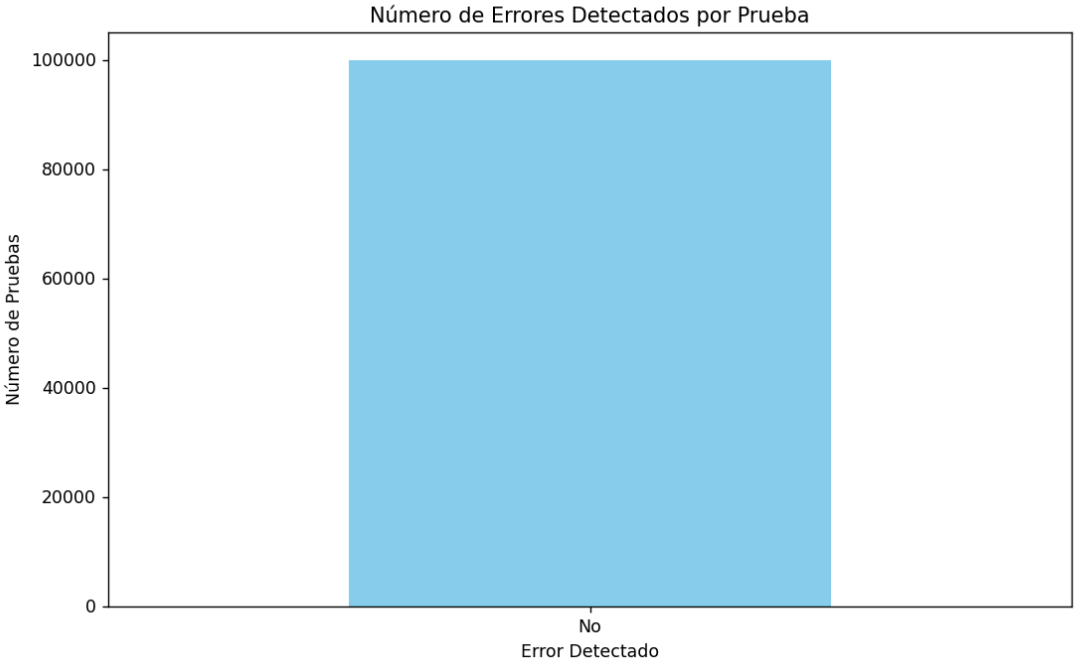
Sí/No, con 8,384 casos de "Sí".

Tiempo total:19.3 segundos.

Archivo:Prueba2.csv

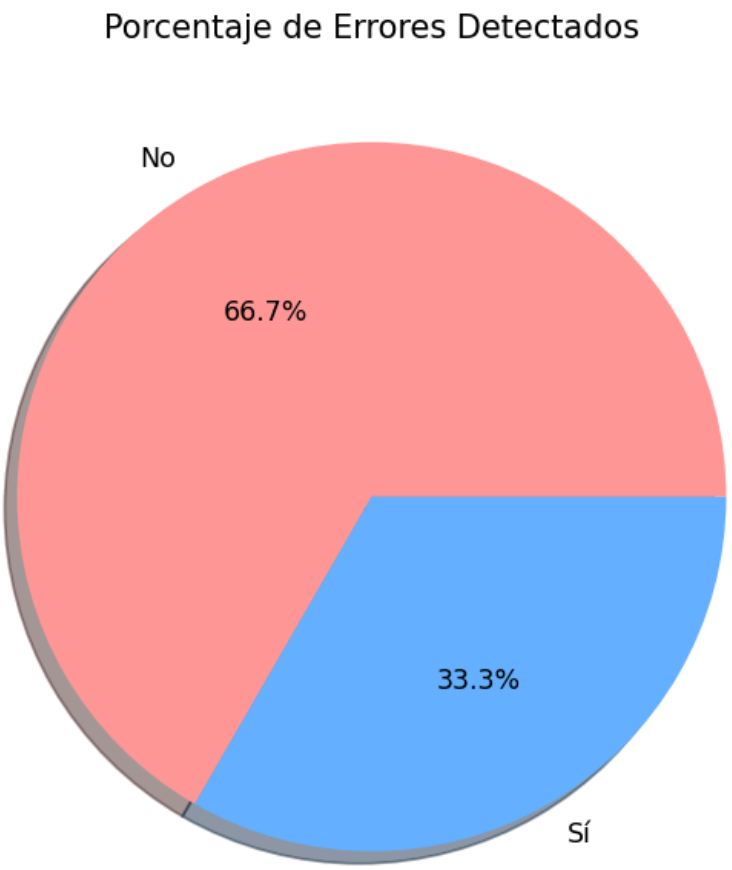
```
Ingrese un mensaje de texto: hola
Seleccione el algoritmo de integridad (3: CRC-32, 2: Hamming): 3
Ingrese la probabilidad de error (por ejemplo, 0.01 para 1%): 0.01
Ingrese el número de pruebas: 100000
```

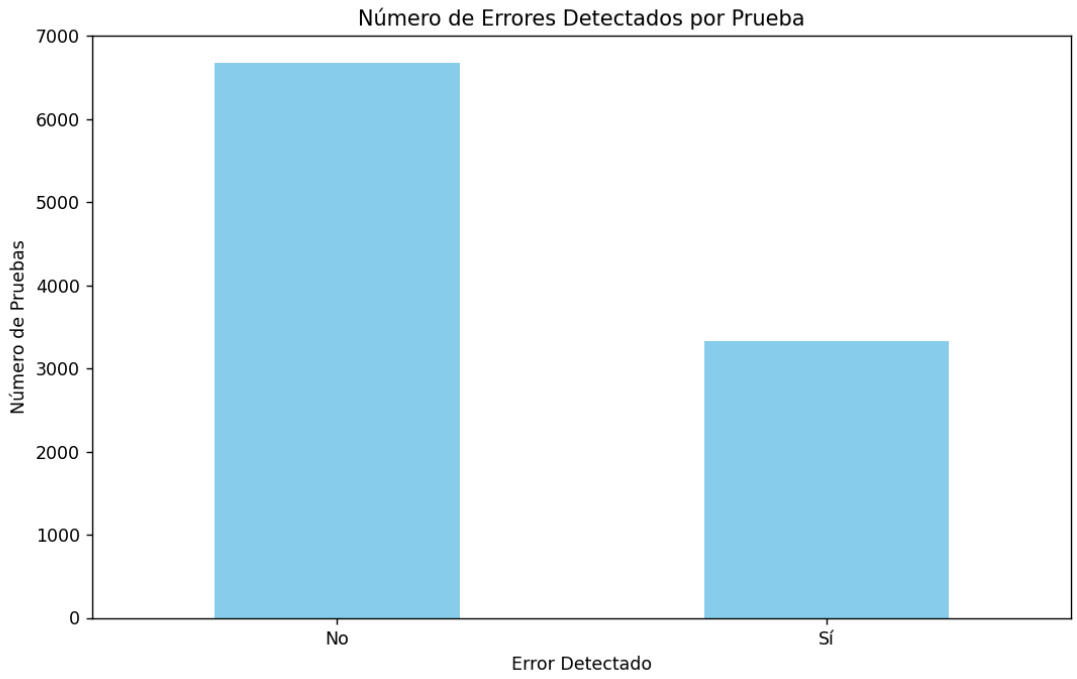

Porcentaje de Errores Detectados
No
100.0%


Número de Errores Detectados por Prueba
100000
80000
60000
40000
20000
0
Número de Pruebas
No
Error Detectado

100000 pruebas sin error
Algoritmo: Crc32
Mensaje ascii: hola
Mensaje en binario: 01101000011011110110110001100001
Duración : 2 minutos y 54.6 segundos.

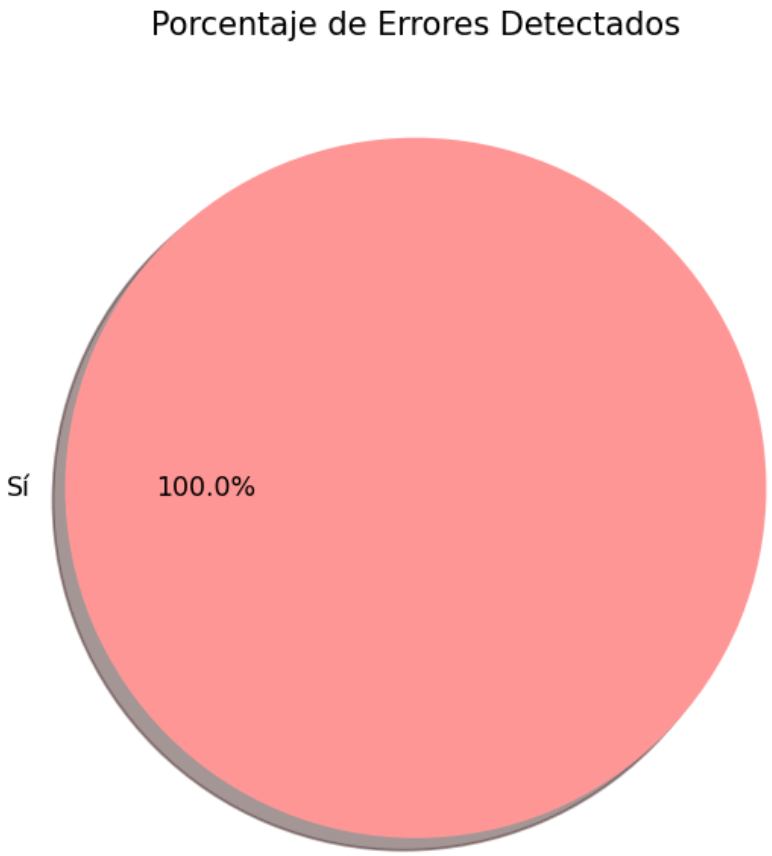
prueba3.csv
Ingrese un mensaje de texto: ea
Seleccione el algoritmo de integridad (3: CRC-32, 2: Hamming): 2
Ingrese la probabilidad de error (por ejemplo, 0.01 para 1%): 0.02
Ingrese el número de pruebas: 10000

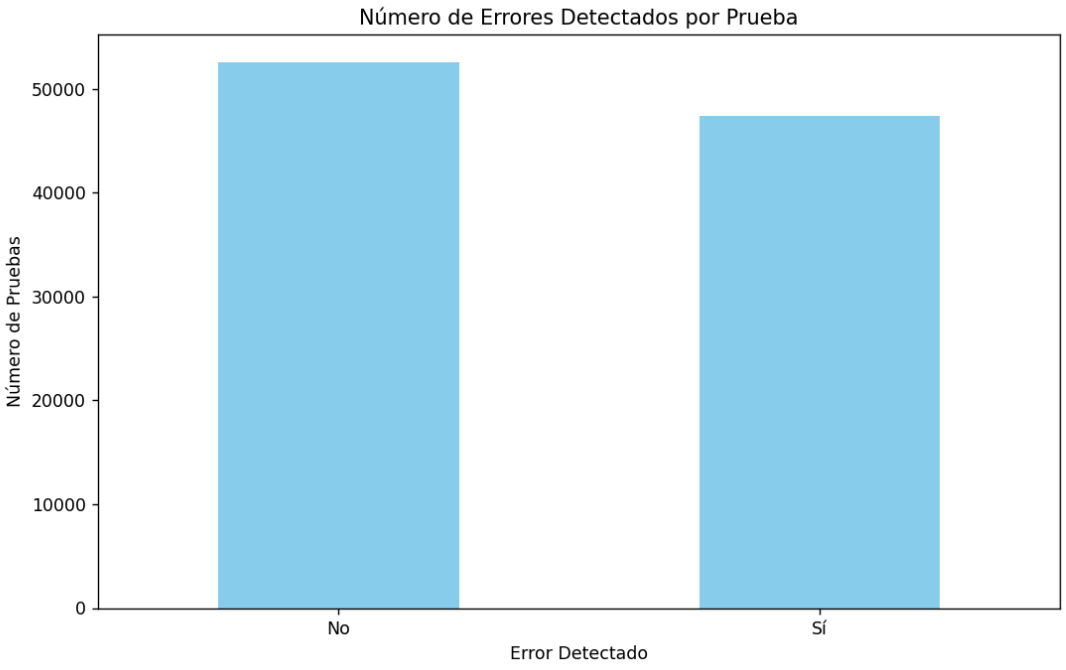




Duración: 59.2 segundos.
prueba4.csv

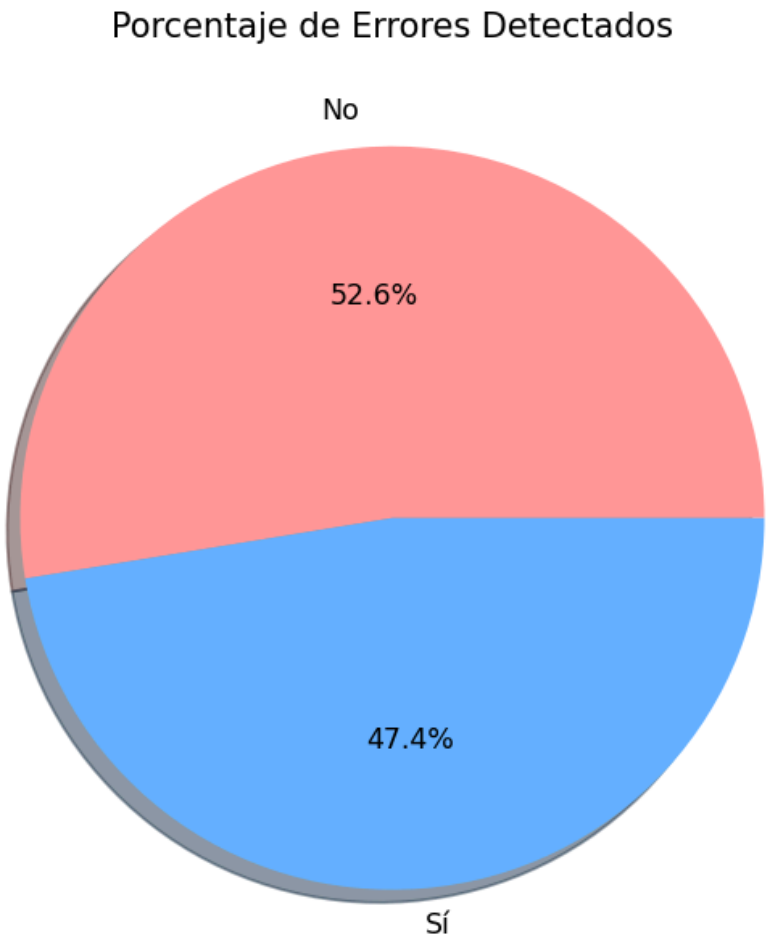
Ingrese un mensaje de texto: mundo
Seleccione el algoritmo de integridad (3: CRC-32, 2: Hamming): 3
Ingrese la probabilidad de error (por ejemplo, 0.01 para 1%): 0.2
Ingrese el número de pruebas: 100000





100,000 errores detectados de 100000

Duración: 6 minutos y 2.4 segundos
Prueba5
Ingrese un mensaje de texto: star
Seleccione el algoritmo de integridad (3: CRC-32, 2: Hamming): 3
Ingrese la probabilidad de error (por ejemplo, 0.01 para 1%): 0.01
Ingrese el número de pruebas: 100000



Duración : 7 minutos 29 segundos
Sí" un total de 47,394 veces de 100000

Discusión

En las pruebas realizadas, se analizaron los algoritmos CRC-32 y Hamming bajo diferentes probabilidades de error y se observaron los tiempos de ejecución. En la primera prueba con una probabilidad de error del 20% para el algoritmo CRC-32 y el mensaje "a", se detectaron errores en el 83.8% de las pruebas, lo cual es significativamente alto, y la duración total fue de 19.3 segundos. En otra prueba con una probabilidad de error del 2% utilizando el algoritmo Hamming y el mensaje "ea", se detectaron errores en el 33.3% de las pruebas,

con una duración total de 59.2 segundos. En una prueba adicional con el algoritmo CRC-32 y el mensaje "mundo" con una probabilidad de error del 20%, todos los errores fueron detectados (100%), con una duración total de 6 minutos y 2.4 segundos. Finalmente, en la prueba con el algoritmo CRC-32, el mensaje "star" y una probabilidad de error del 1%, se detectaron errores en el 47.394% de las pruebas, con una duración de 7 minutos y 29 segundos.

Las gráficas muestran claramente la efectividad de cada algoritmo en la detección de errores. El algoritmo CRC-32 tiene una alta tasa de detección de errores, especialmente con mayores probabilidades de error, pero también conlleva una mayor complejidad temporal, como se observa en las duraciones de las pruebas. Hamming, aunque menos eficaz en la detección de errores a niveles más bajos de probabilidad de error, tiene la ventaja de corregir los errores, lo cual es crítico en sistemas donde la integridad de los datos es primordial.

El algoritmo CRC-32 mostró un mejor funcionamiento en términos de detección de errores, especialmente en escenarios con altas probabilidades de error. En las pruebas, CRC-32 detectó consistentemente los errores con una alta tasa de acierto, como se observa en las gráficas donde el porcentaje de errores detectados fue muy elevado. Sin embargo, CRC-32 no corrige los errores, solo los detecta. Por otro lado, el algoritmo Hamming es más flexible para aceptar mayores tasas de errores debido a su capacidad de no solo detectar, sino también corregir errores, lo cual es crucial en sistemas donde la integridad de los datos debe ser mantenida a toda costa.

Es preferible utilizar

un algoritmo de detección de errores como CRC-32 en sistemas donde se requiere una rápida identificación de errores y la corrección puede ser manejada en una capa superior o por retransmisión de datos, como en transmisiones de datos en red. En contraste, un algoritmo de corrección de errores como Hamming es más adecuado en sistemas donde la retransmisión no es posible o deseable, y la corrección de errores debe realizarse en tiempo real, como en sistemas de almacenamiento de datos críticos.

Conclusiones

En este estudio, se realizaron diversas pruebas para evaluar el desempeño de los algoritmos CRC-32 y Hamming en la detección y corrección de errores. Se utilizaron diferentes configuraciones de mensaje, probabilidades de error y número de pruebas para generar datos que respaldaran la discusión y conclusiones. Las gráficas obtenidas mostraron claramente el comportamiento de ambos algoritmos bajo distintas condiciones.

Los resultados indican que el algoritmo CRC-32 es altamente eficiente en la detección de errores, especialmente en escenarios con altas tasas de error, donde mostró un porcentaje significativo de detección de errores. Sin embargo, CRC-32 no ofrece capacidad de corrección, lo que limita su aplicabilidad en situaciones donde es necesario recuperar datos erróneos sin retransmisión. Por otro lado, el algoritmo Hamming demostró ser más flexible

en términos de aceptar y corregir errores, aunque su complejidad y el tiempo de procesamiento fueron mayores en comparación con CRC-32.

Se observó que el tiempo de procesamiento para el algoritmo CRC-32 es considerablemente menor en comparación con el algoritmo Hamming, haciendo que CRC-32 sea más adecuado para aplicaciones en tiempo real donde la velocidad es crítica. Por ejemplo, en una prueba con 100,000 envíos de mensajes sin errores, CRC-32 tomó 2 minutos y 54.6 segundos, mientras que Hamming mostró una mayor duración en pruebas similares debido a su capacidad de corrección.

Citas y Referencias

Hamming, R. W. (1950). Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, 29(2), 147-160. doi:10.1002/j.1538-7305.1950.tb00463.x

Millman, C. (2021). *CSV File Format: Understanding the Basics*. Retrieved from https://www.csvreader.com/csv-file-format

Git:https://github.com/AGM54/redes/tree/labparte2