Universidad del Valle de Guatemala
Redes de Computadoras
MSc. Jorge Yass
Sección 20

Laboratorio 2.2

Esquemas de detección y corrección de errores

Diego Lemus, 21469
Fabián Juárez, 21440

Guatemala, 01 de agosto 2024

DESCRIPCIÓN DE LA PRÁCTICA

La actividad consiste en abordar el diseño e implementación de esquemas de detección y corrección de errores en la transmisión de datos a través de una red no confiable. El laboratorio inicia con una revisión de los antecedentes, destacando la importancia de manejar adecuadamente el ruido y los errores de transmisión en cualquier sistema de comunicación. Se enfatiza en la evolución de Internet y en cómo se han desarrollado diversos mecanismos para la detección y corrección de errores para mejorar la integridad de la información transmitida.

El objetivo principal del laboratorio es comprender y aplicar un modelo de capas para la transmisión de datos, implementando sockets para enviar y recibir información. Los estudiantes trabajarán en grupos para desarrollar una aplicación que simule la transmisión de mensajes a través de un canal no confiable, analizando cómo los algoritmos de detección y corrección implementados en la primera parte del laboratorio se comportan bajo estas condiciones. La arquitectura de la aplicación incluye varias capas, cada una con funciones específicas como solicitar y mostrar mensajes, codificar y decodificar datos en ASCII binario, calcular y verificar la integridad de los mensajes, aplicar ruido y enviar la información mediante sockets.

El desarrollo práctico implica la interacción de las capas de aplicación, presentación, enlace, ruido y transmisión. Los estudiantes experimentarán cómo los errores inducidos por el ruido afectan la integridad de los mensajes y cómo los algoritmos de detección y corrección pueden mitigar estos problemas. Esta experiencia práctica no solo refuerza la comprensión teórica de los esquemas de detección y corrección de errores, sino que también proporciona habilidades técnicas esenciales para implementar y evaluar soluciones efectivas en sistemas de comunicación real.

RESULTADOS

Repositorio: https://github.com/TheDeloz-v2/RC-LAB2-2

Se definió que el emisor utilizase el lenguaje de programación Javascript y que el receptor utilizase Python. Para la simulación de capas, tanto el emisor como el receptor, tienen una serie de carpetas, cada una con un archivo que realiza la función de la capa. Se utilizó el socket 127.0.0.1:3000, el emisor espera que el usuario ingrese un mensaje, por ejemplo “Hello world”, realizaba el proceso de conversión, codificación y ruido, luego lo enviaba por el socket donde el receptor espera la información y realiza el proceso inverso para comprobar la integridad del mensaje.

Se realizaron un total de 60 pruebas, 30 para cada algoritmo. Cada prueba consistió en ingresar 5 veces seguidas un mensaje con una determinada cantidad de bits, en este caso para: 8, 16, 24, 32, 40 y 48. Dependiendo del algoritmo se evaluó la cantidad de éxitos que tuvo, ya sea para detectar, o detectar y corregir el mensaje. Este conteo de éxitos y fracasos se trasladó a porcentajes y se compararon con la cantidad de bits.

En el caso de Hamming la evaluación de éxito se llevó a cabo al esperar que el receptor indicará si se encontró ruido en el mensaje recibido, y que de hacerlo, corrigiera correctamente el mensaje binario de forma que al transformarse mostrará exactamente el mensaje ingresado por el emisor.

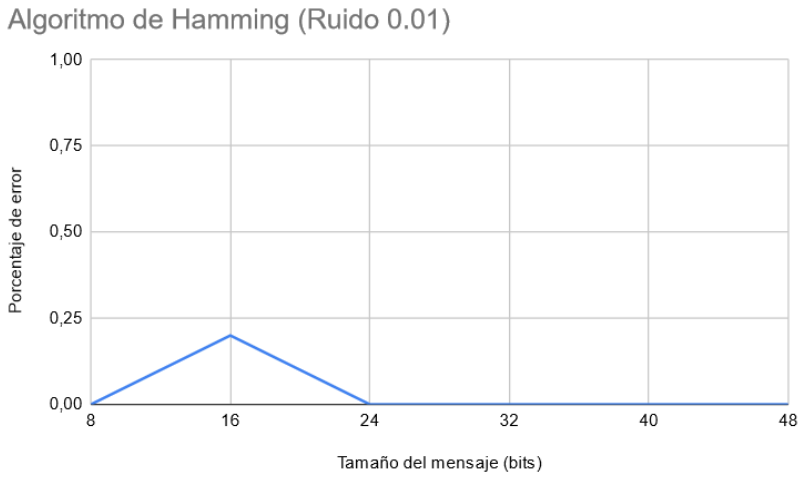

Figura 1: Porcentaje de error con Hamming para una probabilidad de ruido de 0.01

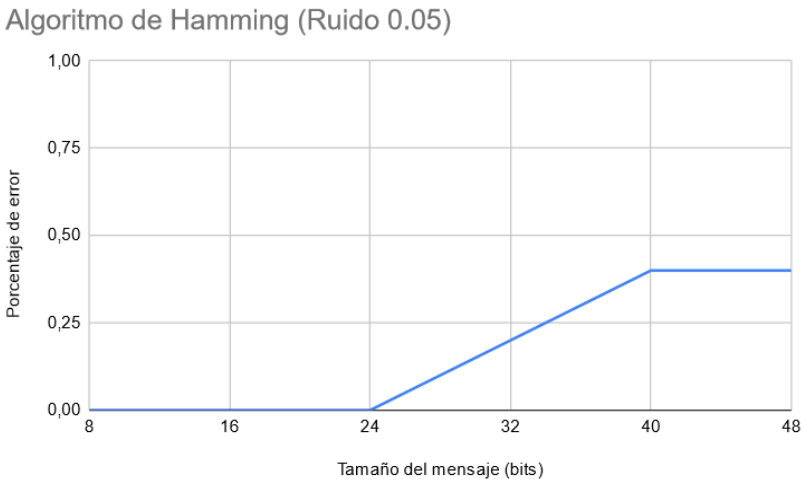

Figura 2: Porcentaje de error con Hamming para una probabilidad de ruido de 0.05

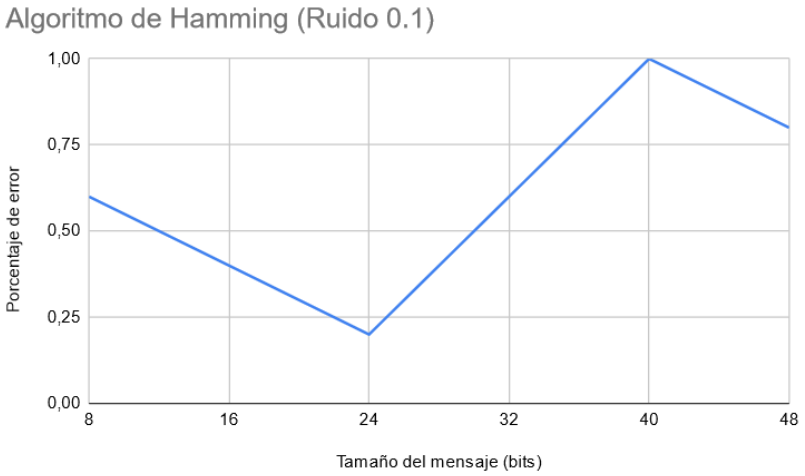

Figura 3: Porcentaje de error con Hamming para una probabilidad de ruido de 0.1

En el caso del checksum Fletcher, dado que solo es un algoritmo de detección, la evaluación de éxito se llevó a cabo al esperar que el receptor indicara que se detectaron errores en base a la comparación del checksum generado.

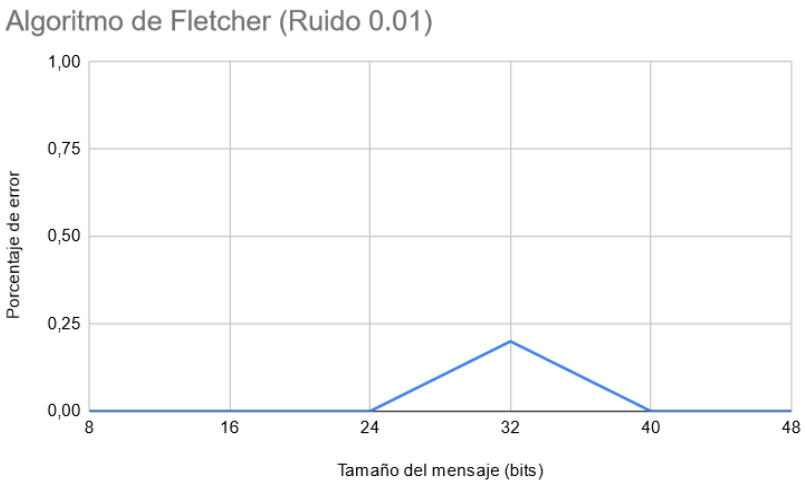

Figura 4: Porcentaje de error con Fletcher para una probabilidad de ruido de 0.1

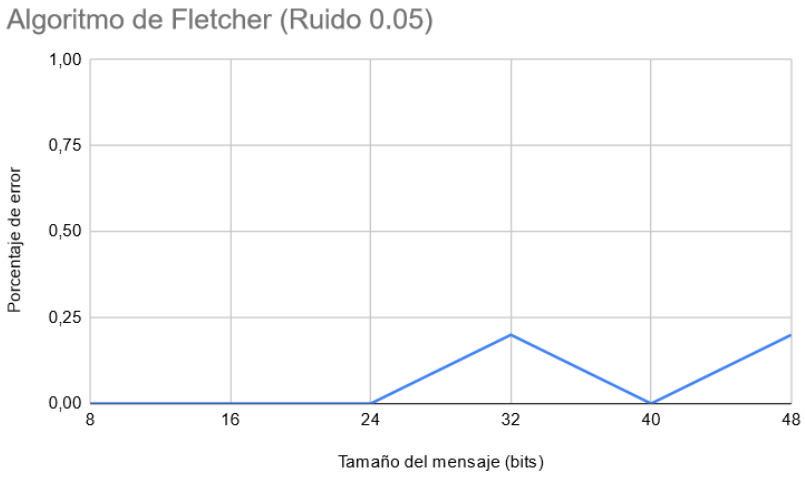

Figura 5: Porcentaje de error con Fletcher para una probabilidad de ruido de 0.05

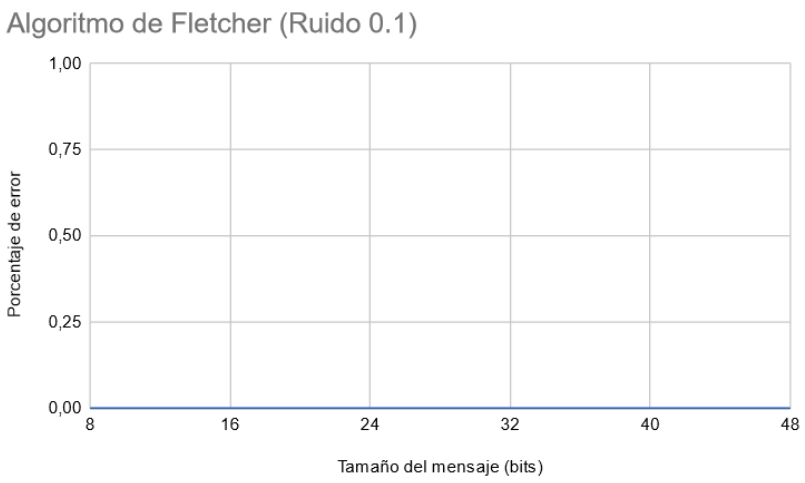

Figura 6: Porcentaje de error con Fletcher para una probabilidad de ruido de 0.1

Por último, se generó un gráfico que recopiló los resultados de los anteriores experimentos, en este caso, este detalla el promedio de error durante cada prueba para cada algoritmo, es decir, compara el error medio dependiendo de la probabilidad de ruido a la que se sometieron los mensajes.

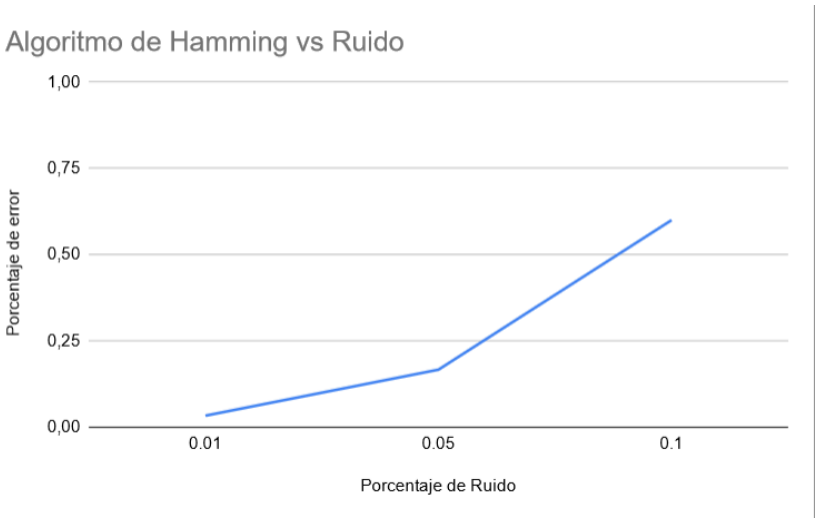

Figura 7: Porcentaje de error con Hamming vs Probabilidad de ruido

.

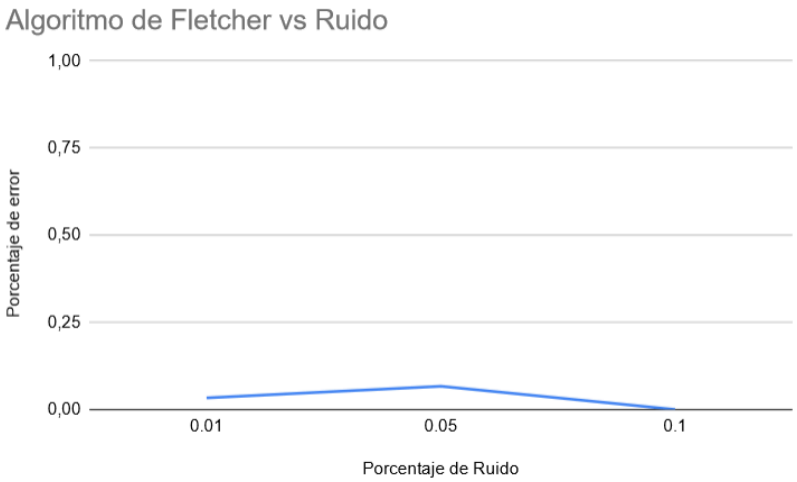

Figura 8: Porcentaje de error con Fletcher vs Probabilidad de ruido

DISCUSIÓN

El algoritmo de checksum Fletcher tuvo un mejor funcionamiento en comparación con el algoritmo de Hamming. Durante las pruebas, se observó que el porcentaje de éxito en la congruencia de la solicitud con la identificación de errores por parte del algoritmo de Fletcher fue significativo. En cambio, el algoritmo de Hamming en la corrección de errores, a pesar de tener la capacidad de corregirlos, limita su efectividad en la congruencia a pesar que esta ayuda a que se pueda mantener la integridad de los datos en un entorno ruidoso.

El algoritmo de Hamming es más flexible para aceptar mayores tasas de errores. Aunque el algoritmo de Hamming muestra un aumento en el porcentaje de errores con el incremento de la probabilidad de ruido, Hamming puede corregir los errores hasta cierto punto, lo que lo hace más robusto en condiciones de alta interferencia. Esto lo hace más adecuado para entornos donde se espera un nivel significativo de errores en la transmisión de datos.

Es mejor utilizar un algoritmo de detección de errores, como el checksum de Fletcher, en escenarios donde:

- La transmisión de datos es relativamente confiable y la probabilidad de errores es baja.
- Es crítico identificar la presencia de errores rápidamente sin necesidad de corregirlos automáticamente, permitiendo que las decisiones de corrección se manejen manualmente o a través de retransmisiones controladas.
- El costo de la corrección de errores es alto en términos de procesamiento o tiempo, y la retransmisión de datos es más eficiente.

CONCLUSIONES

Hamming es más robusto ante mayores niveles de interferencia dada su capacidad de corregir errores que le permite mantener la integridad de los datos hasta cierto punto, si la interferencia aumenta significativamente, su precisión para la congruencia de la corrección disminuye.

El algoritmo de Fletcher demostró ser más eficaz en la identificación de errores, presentó un alto porcentaje de éxito mucho mayor, lo que lo hace adecuado para entornos con baja probabilidad de errores donde es crucial detectar rápidamente la presencia de errores sin necesidad de corrección automática.

Los algoritmos de detección de errores, como el de Fletcher, son más adecuados en situaciones con alta probabilidad de errores, ya que nos indicará estrictamente si el mensaje fue interferido, pero no brinda la opción de corregirlos inmediatamente. En cambio, en entornos con una probabilidad de errores media, el algoritmo de Hamming es más adecuado debido a su capacidad de corrección de errores, proporcionando una solución más eficiente para mantener la integridad de los datos.

COMENTARIO

La implementación práctica de los algoritmos en el laboratorio nos permitió entender no solo la teoría detrás de la detección y corrección de errores, sino también de sus aplicaciones reales y sus limitaciones.

REFERENCIAS

NPM (2024). *dotenv*. https://www.npmjs.com/package/dotenv

Wikipedia (2024). *Método de Hamming.* https://es.wikipedia.org/wiki/M%C3%A9todo_de_Hamming