Universidad del Valle de Guatemala
Facultad de Ingeniería

Laboratorio #2
Esquemas de detección y corrección de errores

Marco Antonio Jurado Velasquez 20308
Cristian Eduardo Aguirre Duarte 20231

Introducción

El ruido y los errores de transmisión no son sólo inevitables sino que también representan un desafío continuo para las aplicaciones de la comunicación. A medida que la digitalización del mundo avanza, la necesidad de una transmisión de datos confiable y eficiente se vuelve cada vez más crítica, pero a la vez el problema del ruido y los errores en la transmisión sigue siendo relevante. Para estas situaciones se han desarrollado procesos diseñados específicamente para detectar y corregir estos errores, garantizando así la integridad y la confiabilidad de la información transmitida.

En este laboratorio, se detalla y trabaja en torno a estos mecanismos, desde su conceptualización teórica hasta su aplicación práctica. La primera parte del laboratorio se centró en implementar algoritmos para la detección y corrección de errores, con la cual pudimos comprender de mejor manera cómo funcionan estos algoritmos y en qué casos son más efectivos. Ahora, en la segunda parte, nos enfocamos en conocer el proceso de comunicación por capas en un contexto más real y cercano a la industria, en donde se hacen uso de sockets que siguen una arquitectura de capas definida y que permiten la comunicación directa a través de la red entre programas.

Finalmente, a través de este informe, buscamos conocer más a detalle los procesos que garantizan comunicaciones seguras y confiables entre dos programas completamente distintos y que tienen como única característica en común, estar conectados a una misma conexión de red.

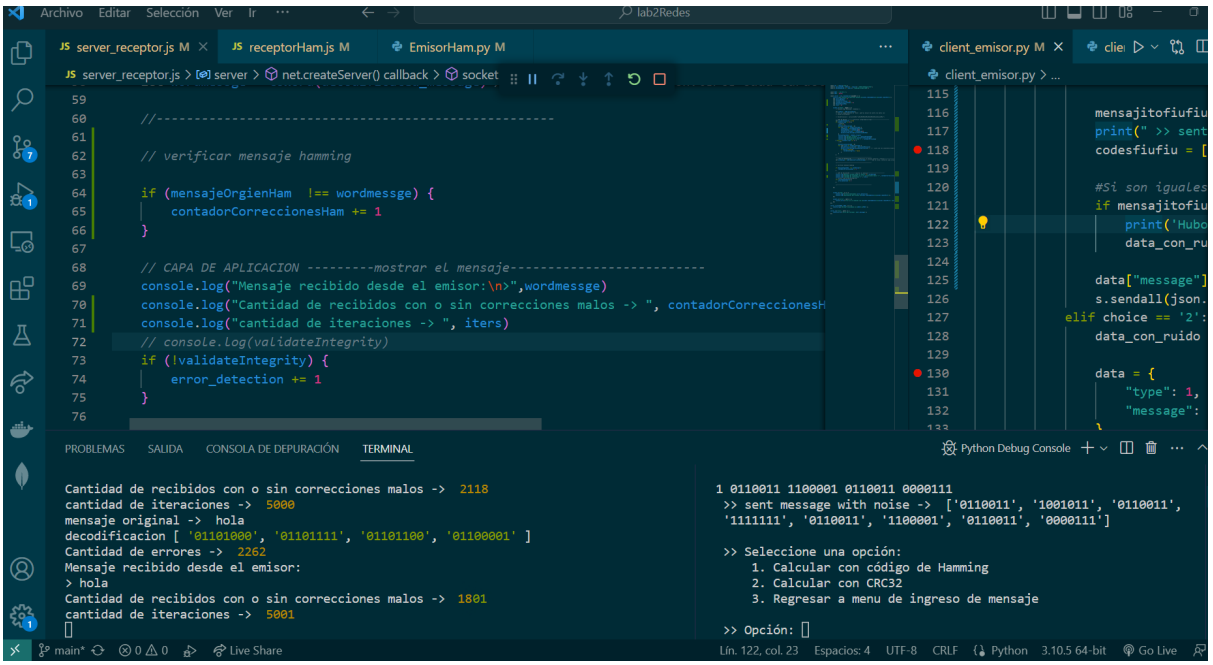
Resultados

Algoritmo Hamming

- **No. Mensajes: 5000**
- **Longitud de cadena por mensaje codificado: 4 bits**
- **Longitud de cadena por carácter generador de palabra: 8 bits (dos grupos de 4 bits en hamming para conformar una letra)**
- **Prob. de error: 0.01**
- **No. Mensajes con ruido: 2262**
- **No. Mensajes con ruido detectados: 2262**

Todos los mensajes con error fueron detectados

- **No. mensajes corregidos exitosamente 1801**



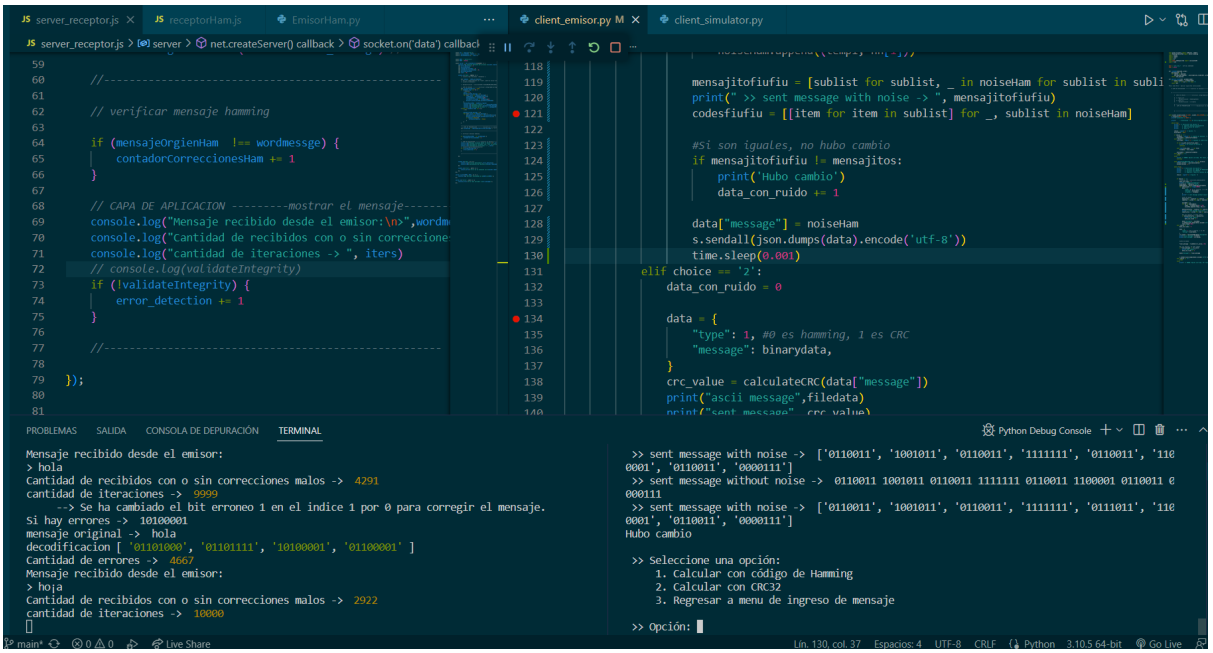
Con los datos mostrados en las 5000 pruebas podemos decir que de 2262 mensajes con error generado se logró corregir correctamente 1801 de ellos dando así:

- **Porcentaje de efectividad: 20.4%**

- **No. Mensajes: 10000**
- **Longitud de cadena por mensaje codificado: 4 bits**
- **Longitud de cadena por carácter generador de palabra: 8 bits (dos grupos de 4 bits en hamming para conformar una letra)**
- **Prob. de error: 0.01**
- **No. Mensajes con ruido: 4667**
- **No. Mensajes con ruido detectados: 4667**

Todos los mensajes con error fueron detectados

- **No. mensajes corregidos exitosamente 2922**

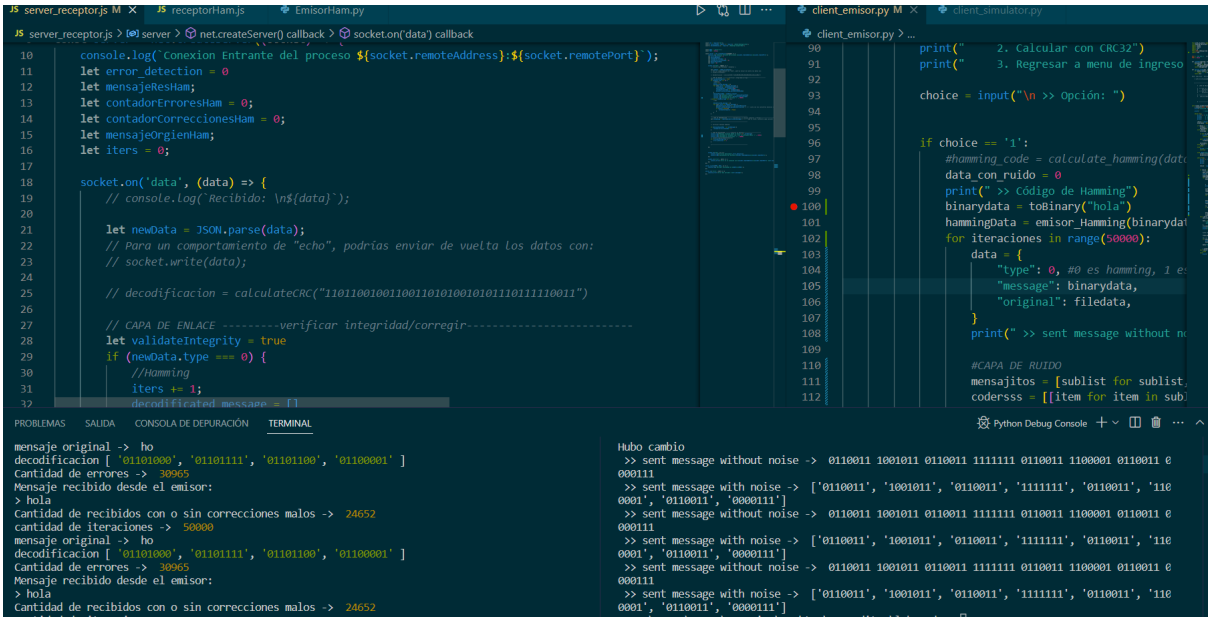


- **porcentaje de error: 37.4%**

- **No. Mensajes: 50000**
- **Longitud de cadena por mensaje codificado: 4 bits**
- **Longitud de cadena por carácter generador de palabra: 8 bits (dos grupos de 4 bits en hamming para conformar una letra)**
- **Prob. de error: 0.01**
- **No. Mensajes con ruido: 30965**
- **No. Mensajes con ruido detectados: 30965**

Todos los mensajes con error fueron detectados

- **No. mensajes corregidos exitosamente 24652**



- **porcentaje de error: 20.38%**

Algoritmo CRC32

- **No. Mensajes: 10000**
- **Longitud de cadena: 12bits**
- **Prob. de error: 0.01**
- **No. Mensajes con ruido: 3531**
- **No. Mensajes con ruido detectados: 3531**

Porcentaje de error: 0%
Porcentaje de efectividad: 100%

client_simulator.py > ...

```python
from word_to_binary import toBinary, toWord
from Hamming.EmisorHam import emisor_Hamming
import socket
import random
import json
import os
from CRC32.CRCAlgorithm import calculateCRC
import string

HOST = "127.0.0.1"   #IP DEL SERVIDOR
PORT = 65123
n_loops = 10000 #Cantidad de iteraciones
n_length = 12 #Longitud de las cadenas
n_prob = 0.01

def generar_palabra(longitud):
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

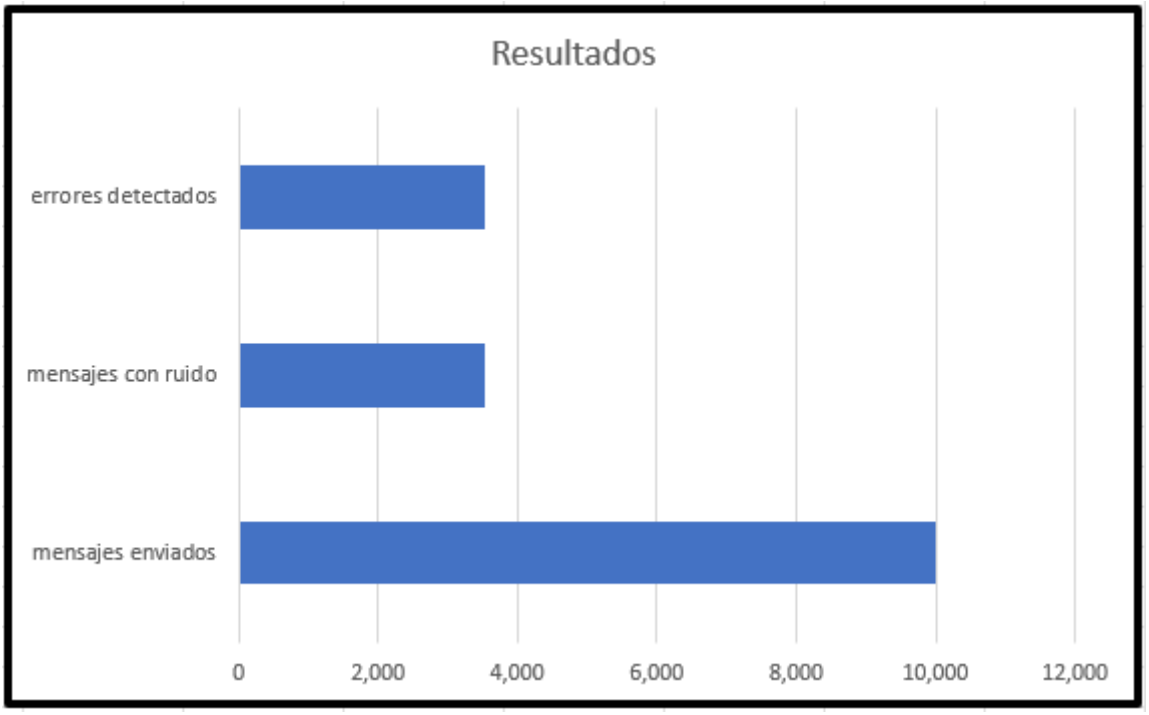
```
    1. Calcular con código de Hamming
    2. Calcular con CRC32
    3. Salir

 >> Opción: 2
calculando...
Se envio 3531 datos con ruido de 10000

 >> Seleccione una opción:
    1. Calcular con código de Hamming
    2. Calcular con CRC32
    3. Salir

 >> Opción: 3
```

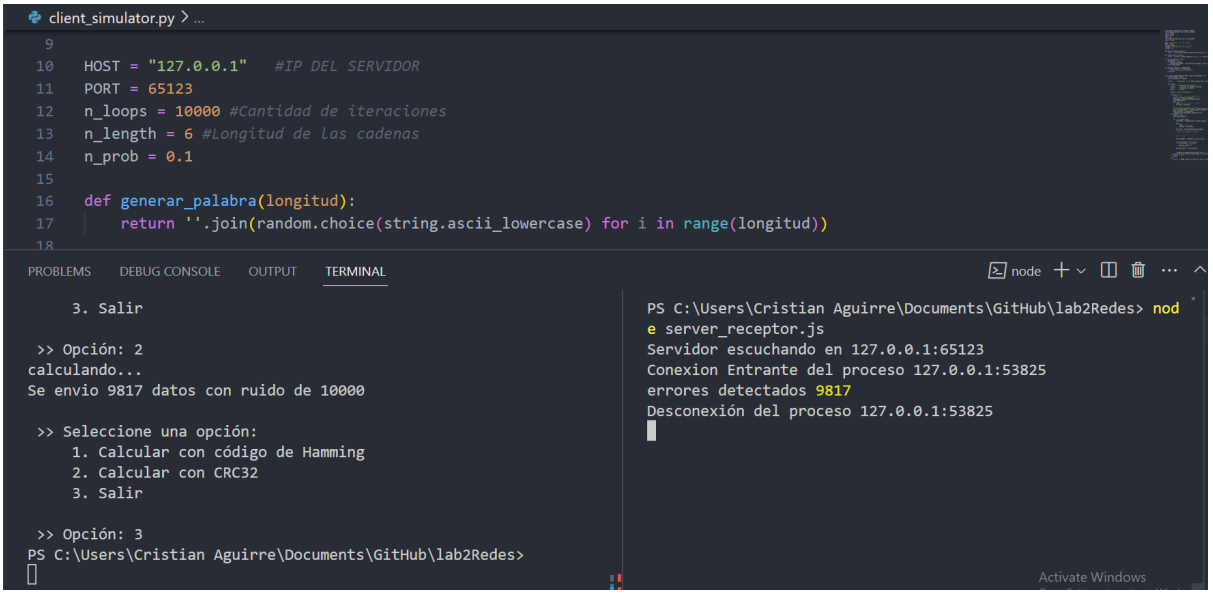
```
PS C:\Users\Cristian Aguirre\Documents\GitHub\lab2Redes> node server_
receptor.js
Servidor escuchando en 127.0.0.1:65123
Conexion Entrante del proceso 127.0.0.1:52639
errores detectados 3531
Desconexión del proceso 127.0.0.1:52639
```

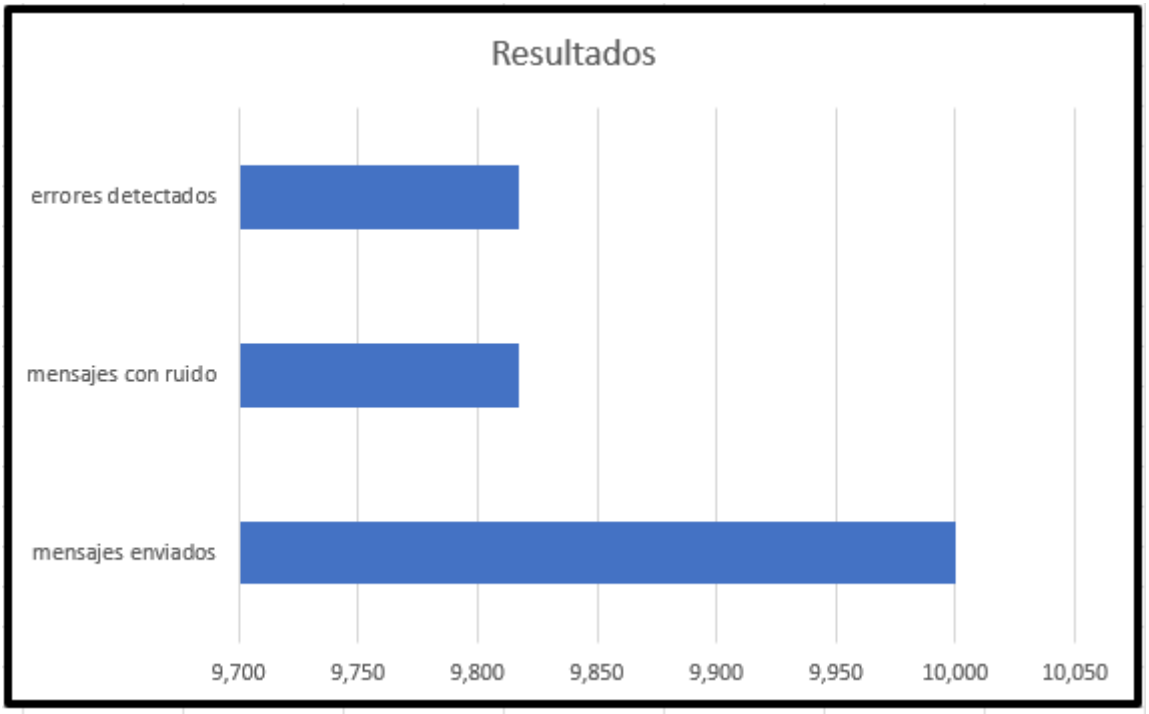


- **No. Mensajes: 10000**
- **Longitud de cadena: 6 bits**
- **Prob. de error: 0.1**
- **No. Mensajes con ruido: 9817**
- **No. Mensajes con ruido detectados: 9817**

Porcentaje de error: 0%

Porcentaje de efectividad: 100%

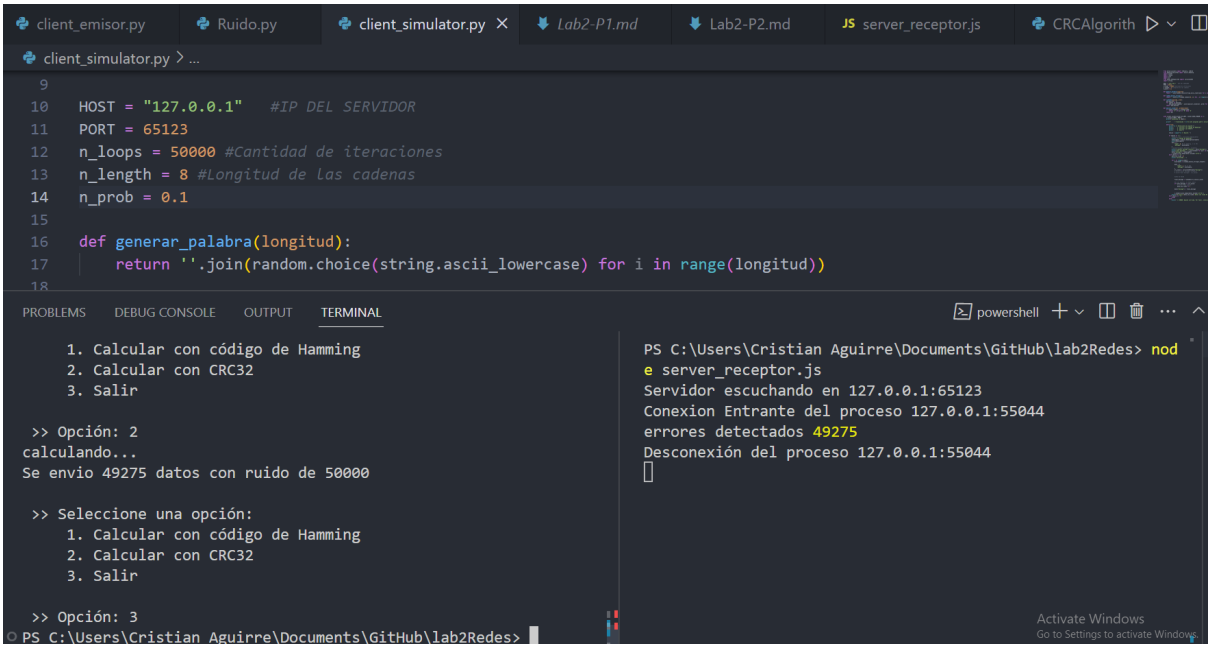


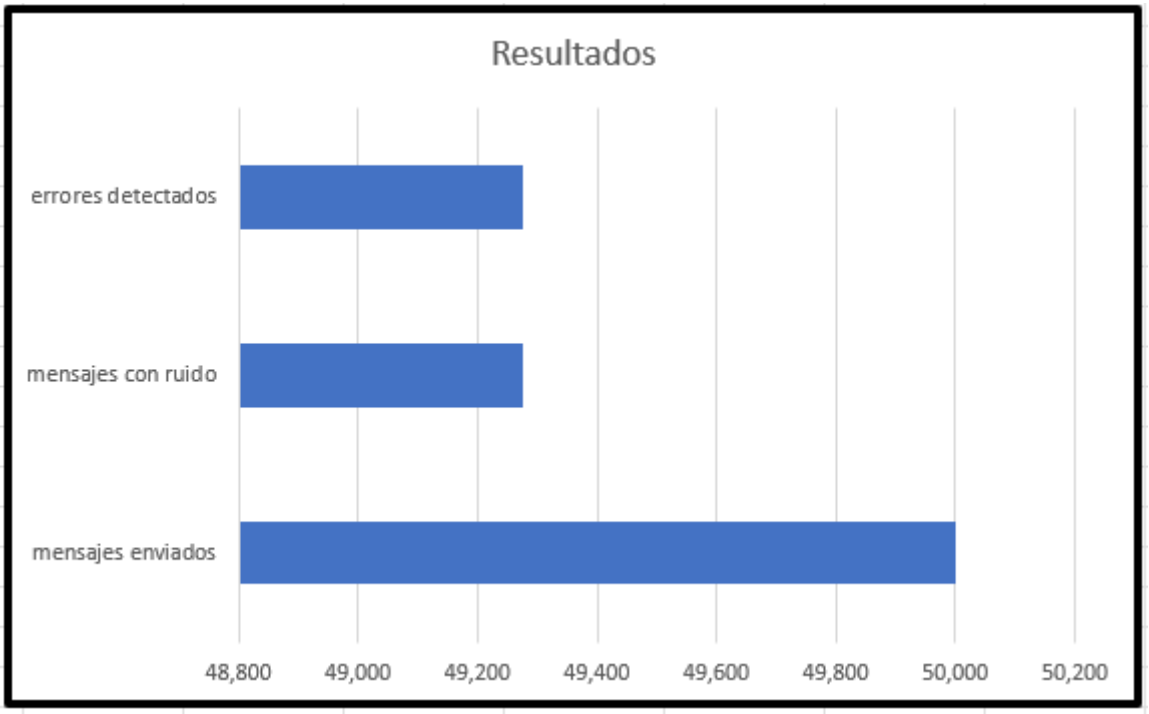


- **No. Mensajes: 50,000**
- **Longitud de cadena: 8 bits**
- **Prob. de error: 0.1**
- **No. Mensajes con ruido: 49,275**
- **No. Mensajes con ruido detectados: 49,275**

Porcentaje de error: 0%

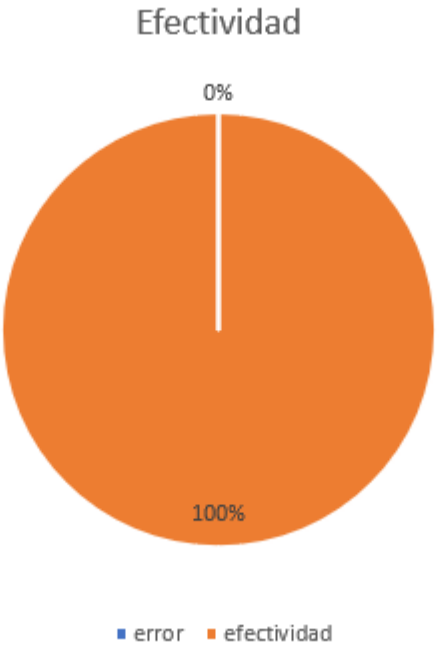
Porcentaje de efectividad: 100%



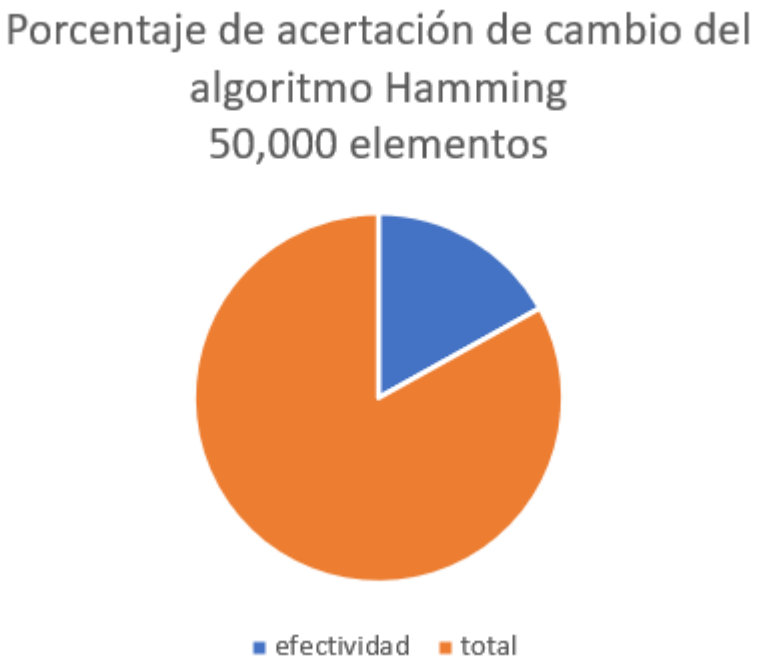


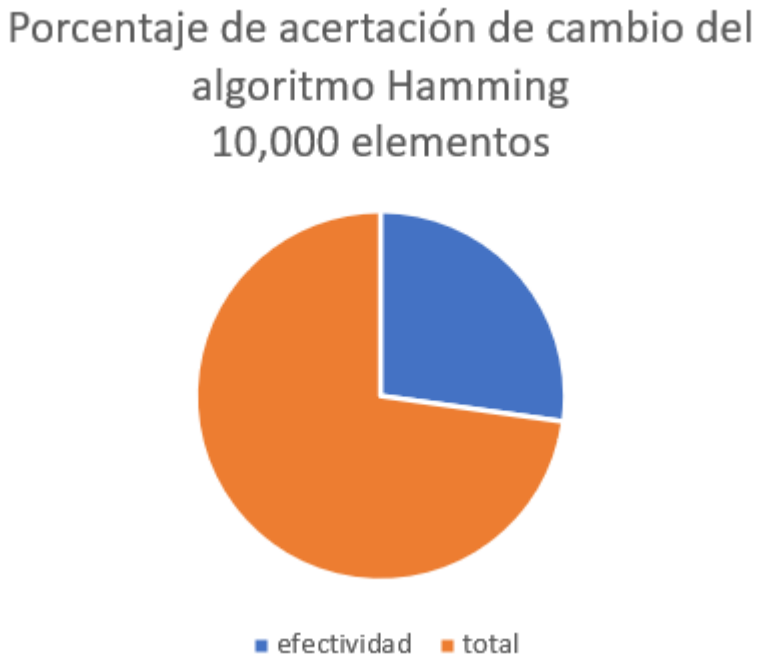
Estadísticas

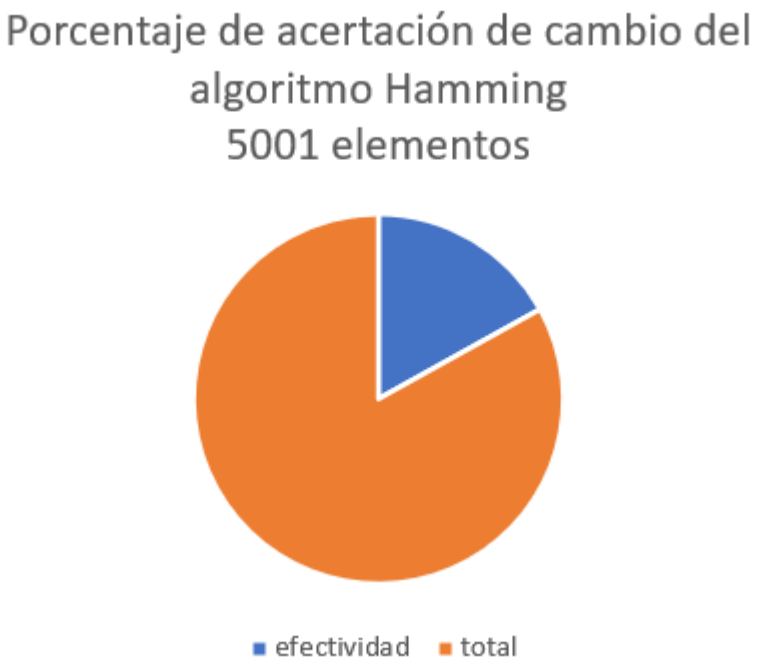
CRC32



Hamming


Porcentaje de acertación de cambio del algoritmo Hamming
50,000 elementos
efectividad
total


Porcentaje de acertación de cambio del algoritmo Hamming
10,000 elementos
efectividad
total



Por lo tanto la efectividad promedio para hamming es de 26.06666667%

Discusión

-¿Qué algoritmo tuvo un mejor funcionamiento y porqué?

De acuerdo con los resultados obtenidos, podemos ver que el algoritmo CRC32 presentó un mejor rendimiento. Pues a lo largo de todas las pruebas realizadas, a pesar de variar la longitud del mensaje y la cantidad de errores, el CRC32 mantuvo una efectividad del 100% en la identificación y detección de errores, lo cual demuestra su confiabilidad y solidez como herramienta de control de integridad en sistemas de comunicación.

Por otro lado, el algoritmo de Hamming demostró una eficacia del 100% en la detección de errores, aunque su capacidad para corregirlos fue limitada, logrando apenas un 26.06% de efectividad en este aspecto. Esto implica que, si bien es capaz de identificar errores de manera muy precisa, su capacidad para corregirlos aún presenta un porcentaje relativamente bajo de éxito.

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

Con fundamento en la teoría vista en clase sabemos que el algoritmo CRC32, está basado en una base matemática que se basa en la división polinómica. Al representar la data como un polinomio, se la divide por el polinomio generador con lo cual cualquier error, incluso uno minúsculo, es equivalente a un cambio en el polinomio de la data. Gracias a la naturaleza

de la división polinómica, este cambio, por más pequeño que sea, produce un residuo notablemente distinto. Esta alta sensibilidad a las variaciones garantiza que incluso alteraciones accidentales en un solo bit de la data resulten en un resultado completamente diferente, proporcionando así una herramienta de detección de errores sumamente confiable y eficaz.

Respecto al algoritmo de Hamming, la posibilidad de que no pueda corregir errores de manera precisa cuando se altera más de un bit en la secuencia contribuye significativamente a una tasa de error elevada. Este fenómeno se deriva de la realidad de que la versión 4,7 del algoritmo de Hamming carece de la infalibilidad presente en su contraparte más robusta. No obstante, esta iteración se destaca por su velocidad de ejecución superior. Es relevante señalar que durante las pruebas de ejecución, fue imperativo asignar un período de espera al socket para asegurar la finalización adecuada de la decodificación. Esto permitió obtener resultados exentos de condiciones de carrera en los datos transmitidos.

-¿Cuándo es mejor utilizar un algoritmo de detección de errores en lugar de uno de corrección de errores y por qué?

A pesar de que ambos algoritmos comparten un objetivo similar en el proceso de comunicación, sabemos que el algoritmo de Hamming desarrolla cierta cantidad de procesos adicionales que además de detectar errores, permite la corrección de los mismos. Es por ello que para determinar cual de los dos, es más conveniente utilizar, necesitamos analizar los contextos en los que nos encontramos, pues cada uno tiene características específicas que se aplican en distintas situaciones. Como sabemos el proceso de comunicación se genera a través de la transmisión de información, y en este proceso toman partido diferentes factores que aseguran la integridad y factibilidad del mismo tales como el tipo de sistema, la probabilidad estimada de errores, los recursos al alcance y las exigencias de confiabilidad y eficacia

Por ejemplo, es preferible optar por un algoritmo de detección de errores, cuando tenemos situaciones en donde no tenemos un ancho de banda limitado y la libertad de retransmitir la información a su fuente de origen, pues estos algoritmos, al introducir menos redundancia en comparación con los algoritmos de corrección de errores, ofrecen una transmisión más eficiente. Sin embargo, si nos encontramos en situaciones donde la latencia es un aspecto crucial y no podemos permitir retrasos para retransmitir datos, como en sistemas de comunicaciones en tiempo real, es más adecuado adoptar algoritmos de corrección de errores, ya que estos, a pesar de tener una mayor complejidad y coste, tienen la ventaja de corregir errores inmediatamente sin la necesidad de retransmisiones, garantizando así la continuidad y fiabilidad del flujo de datos.

La decisión de optar por la detección o corrección de errores no es trivial y debe ser tomada en base con las características específicas y las demandas de la aplicación. Esto nos indica que, es crucial una evaluación cuidadosa de las necesidades y restricciones del sistema, así como los recursos que tenemos al alcance, para garantizar la integridad y eficacia de la transmisión de datos.

Conclusiones

- El algoritmo CRC es muy eficiente para la detección de errores debido a su base matemática en el uso de operaciones polinómicas que permiten detectar hasta el mínimo cambio en los bits del mensaje. .
- La versión Hamming 4,7 presenta una detección de errores con eficacia del 100%, aunque su capacidad de corrección se limita al 26.06%, lo cual muestra dificultad en la corrección de múltiples errores cuando es cambiado más de un bit. Sin embargo, esta versión sigue siendo más rápida en computar que su contraparte más robusta.
- La elección entre detección y corrección de errores depende de la naturaleza del sistema, la tasa de error anticipada, los recursos disponibles, y el grado de confiabilidad y eficiencia requeridos.

Citas y referencias bibliográficas

- Shu Lin and Daniel J. Costello, Jr. - "Error Control Coding: Fundamentals and Applications", Prentice-Hall, Inc., 1982.
- Rafael C. González y Richard E. Woods - "Tratamiento digital de imágenes", Ed. Pearson Educación, 2002.