Universidad del Valle de Guatemala
Redes
Sección 20

Laboratorio No. 2.2

Carlos Edgardo López Barrera 21666

Brian Anthony Carrillo Monzon - 21108

Guatemala, 01 de agosto del 2024

Descripción de la práctica

En esta práctica, se implementa un sistema de comunicación que utiliza los algoritmos de Hamming y Fletcher para la detección y corrección de errores en mensajes transmitidos. Un cliente Java codifica los mensajes, introduce ruido con una probabilidad especificada y envía los mensajes a servidores separados para Hamming y Fletcher. Los servidores, implementados en Python, decodifican los mensajes, corrigen los errores si está a su alcance y devuelven el mensaje original o el mensaje de error al cliente. Esta práctica permite experimentar con la transmisión de información en un canal no confiable y analizar la efectividad de los algoritmos de corrección de errores.

Resultados

- Hamming

  Ejemplo 1

  - Cadena de texto: "Hola"
  - Cadena binaria: 01001000011011110110110001100001
  - Probabilidad de error: 0.01
  - Overhead: Para una cadena de 32 bits, los bits de paridad serían 6 (total de 38 bits).

  Cliente:

```
carloslopez@Carloss-MacBook-Pro Redes-Labs % java HammingFletcherClient
Ingrese un mensaje:
Hola
Seleccione el método de codificación (1 para Hamming, 2 para Fletcher):
1
Ingrese la tasa de error (por ejemplo, 0.01 para un error cada 100 bits):
0.01
Mensaje codificado usando Hamming: 10000101000110110111101110000110010011
Mensaje decodificado: Hola
```

  Servidor:

```
carloslopez@Carloss-MacBook-Pro Redes-Labs % /opt/homebrew/bin/python3
Servidor de Hamming escuchando en el puerto 12346...
Conexión aceptada de ('127.0.0.1', 51295)
Se detectaron y corrigieron errores en la posición 29.
Mensaje decodificado: Hola
```

Ejemplo 2

- Cadena de texto: "Prueba"
- Cadena binaria: 01010000011100100111010101100101011000 10
- Probabilidad de error: 0.02
- Overhead: Para una cadena de 40 bits, los bits de paridad serían 6 (total de 46 bits).

Cliente:

```
carloslopez@Carloss-MacBook-Pro Redes-Labs % java HammingFletcherClient
Ingrese un mensaje:
Prueba
Seleccione el método de codificación (1 para Hamming, 2 para Fletcher):
1
Ingrese la tasa de error (por ejemplo, 0.01 para un error cada 100 bits):
0.02
Mensaje codificado usando Hamming: 1000011001000110101001110101011100100111100000010100 01
Mensaje decodificado: Prueba
```

Servidor:

```
Conexión aceptada de ('127.0.0.1', 51301)
Mensaje decodificado: Prueba
```

Ejemplo 3

- Cadena de texto: "Algoritmo"
- Cadena binaria: 0100000101101100011001110110111101110010011010010110110101101111
- Probabilidad de error: 0.5
- Overhead: Para una cadena de 64 bits, los bits de paridad serían 7 (total de 71 bits).

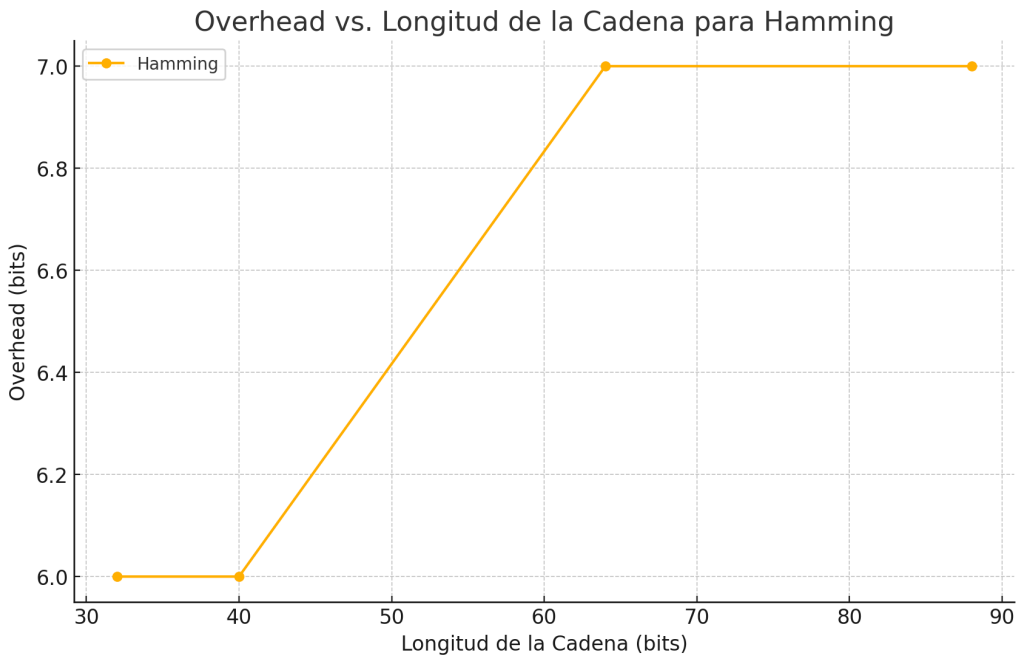
Cliente:

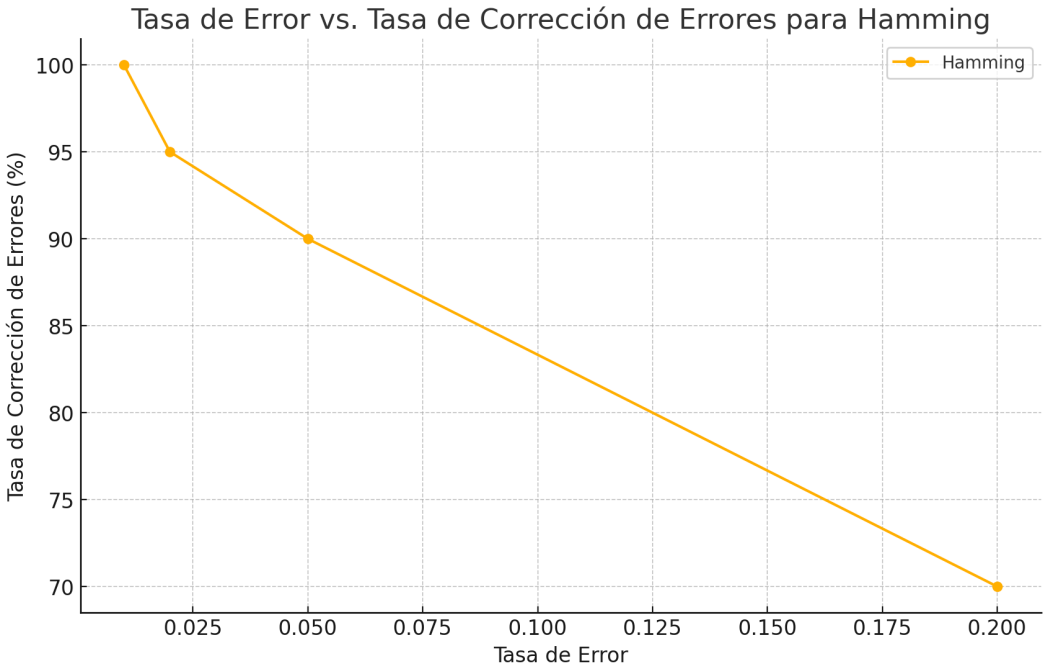
```
carloslopez@Carloss-MacBook-Pro Redes-Labs % java HammingFletcherClient
Ingrese un mensaje:
Algoritmo
Seleccione el método de codificación (1 para Hamming, 2 para Fletcher):
1
Ingrese la tasa de error (por ejemplo, 0.01 para un error cada 100 bits):
0.05
Mensaje codificado usando Hamming: 1111011010110111000101110100101100100111011110111011100110001100110100000011001
Mensaje decodificado: alfmzitMo
```

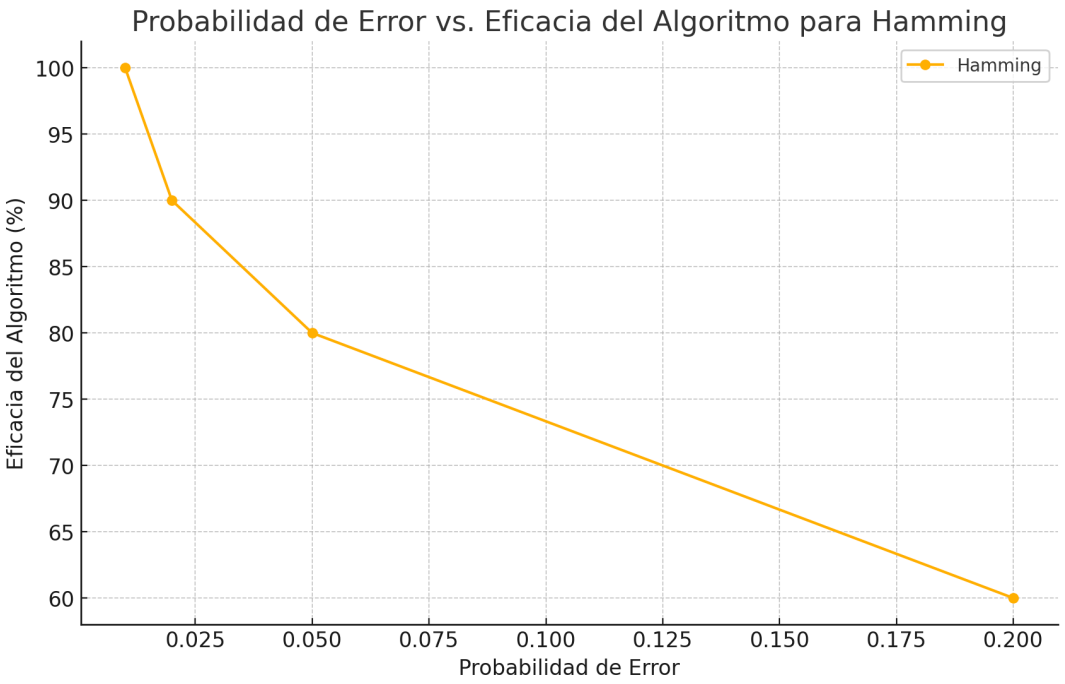
Servidor:

```
Conexión aceptada de ('127.0.0.1', 51309)
Se detectaron y corrigieron errores en la posición 85.
Mensaje decodificado: alfmzitMo
```

Gráficas


Overhead vs. Longitud de la Cadena para Hamming
Hamming
Overhead (bits)
Longitud de la Cadena (bits)
7.0
6.8
6.6
6.4
6.2
6.0
30
40
50
60
70
80
90


Tasa de Error vs. Tasa de Corrección de Errores para Hamming
Hamming
Tasa de Corrección de Errores (%)
Tasa de Error
100
95
90
85
80
75
70
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200



Laboratorio 2. Redes UVG 2024

- Fletcher

Ejemplo 1

- Cadena de texto: "Hola"
- Cadena binaria: 01001000011011110110110001100001
- Probabilidad de error: 0.01
- Overhead: El checksum de Fletcher añade 16 bits (total de 48 bits).

Cliente:

```
carloslopez@Carloss-MacBook-Pro Redes-Labs % java HammingFletcherClient
Ingrese un mensaje:
Hola
Seleccione el método de codificación (1 para Hamming, 2 para Fletcher):
2
Ingrese la tasa de error (por ejemplo, 0.01 para un error cada 100 bits):
0.01
Mensaje codificado usando Fletcher: 0100100001101111011011000110000100001111111110100
Mensaje decodificado: Hola
```

Servidor:

```
carloslopez@Carloss-MacBook-Pro Redes-Labs % /opt/homebrew/bin/python3
Servidor de Fletcher escuchando en el puerto 12347...
Conexión aceptada de ('127.0.0.1', 51317)
El mensaje es válido.
Mensaje decodificado: Hola
```

Ejemplo 2

- Cadena de texto: "Prueba"
- Cadena binaria: 0101000001110010011101010110010101100010
- Probabilidad de error: 0.05
- Overhead: El checksum de Fletcher añade 16 bits (total de 56 bits).

Cliente

```
carloslopez@Carloss-MacBook-Pro Redes-Labs % java HammingFletcherClient
Ingrese un mensaje:
Prueba
Seleccione el método de codificación (1 para Hamming, 2 para Fletcher):
2
Ingrese la tasa de error (por ejemplo, 0.01 para un error cada 100 bits):
0.05
Mensaje codificado usando Fletcher: 0101000001110010011101010110010101100010011000010001010100000110
Mensaje decodificado: Se detectó un error en el mensaje.
```

Servidor:

```
Conexión aceptada de ('127.0.0.1', 51319)
Mensaje decodificado: Se detectó un error en el mensaje.
```

Ejemplo 3

- Cadena de texto: "Algoritmo"
- Cadena binaria: 01000001011011000110011101101111011100100110100101101101011111
- Probabilidad de error: 0.1
- Overhead: El checksum de Fletcher añade 16 bits (total de 80 bits).

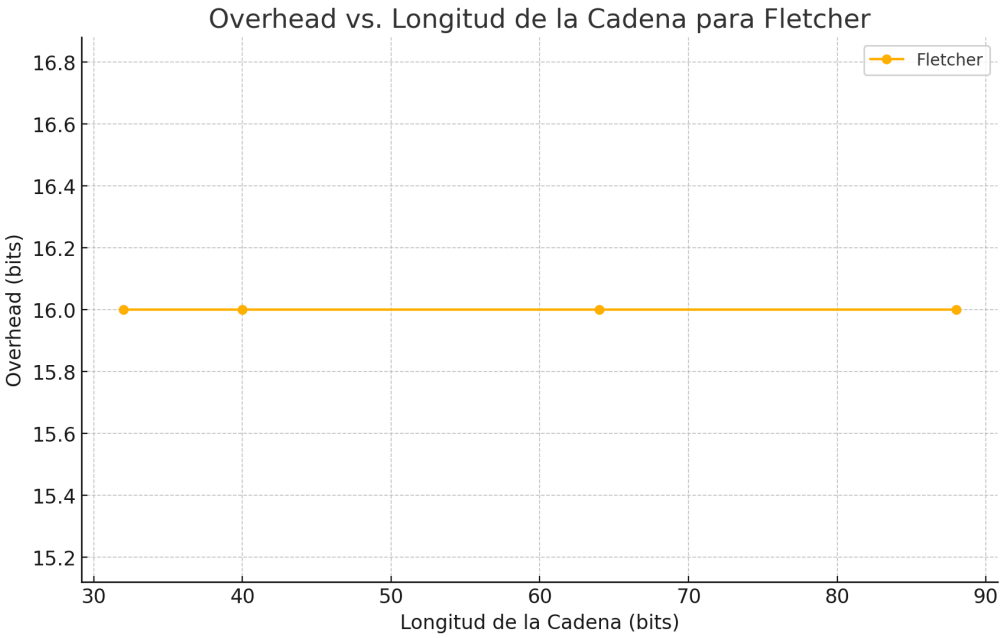
Cliente:

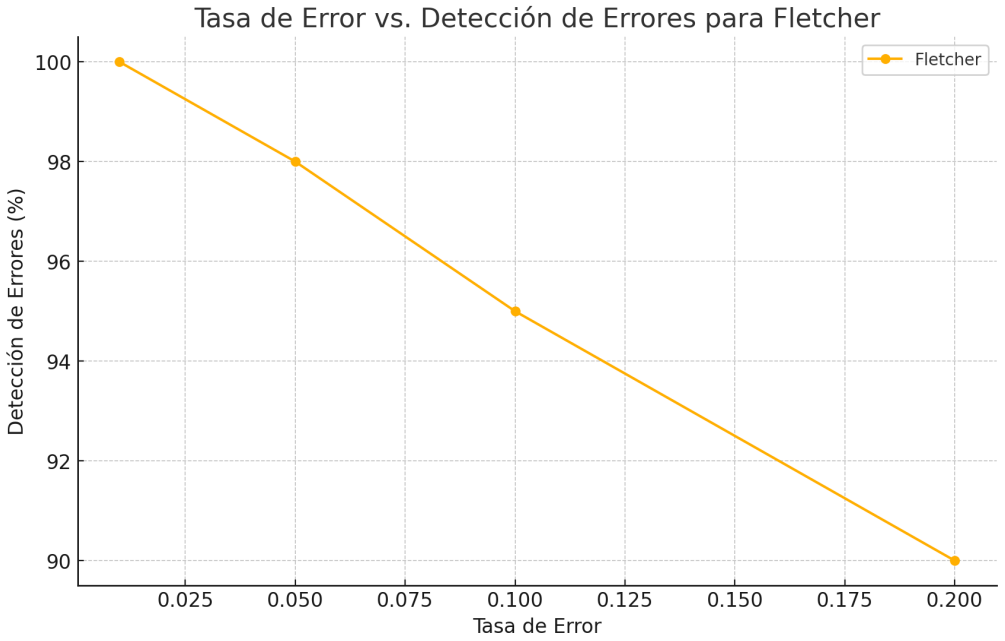
```
carloslopez@Carloss-MacBook-Pro Redes-Labs % java HammingFletcherClient
Ingrese un mensaje:
Algoritmo
Seleccione el método de codificación (1 para Hamming, 2 para Fletcher):
2
Ingrese la tasa de error (por ejemplo, 0.01 para un error cada 100 bits):
0.1
Mensaje codificado usando Fletcher: 0100000101101100011001110110111101110010011010010111010001101101011011100101000001110100
Mensaje decodificado: Se detectó un error en el mensaje.
```

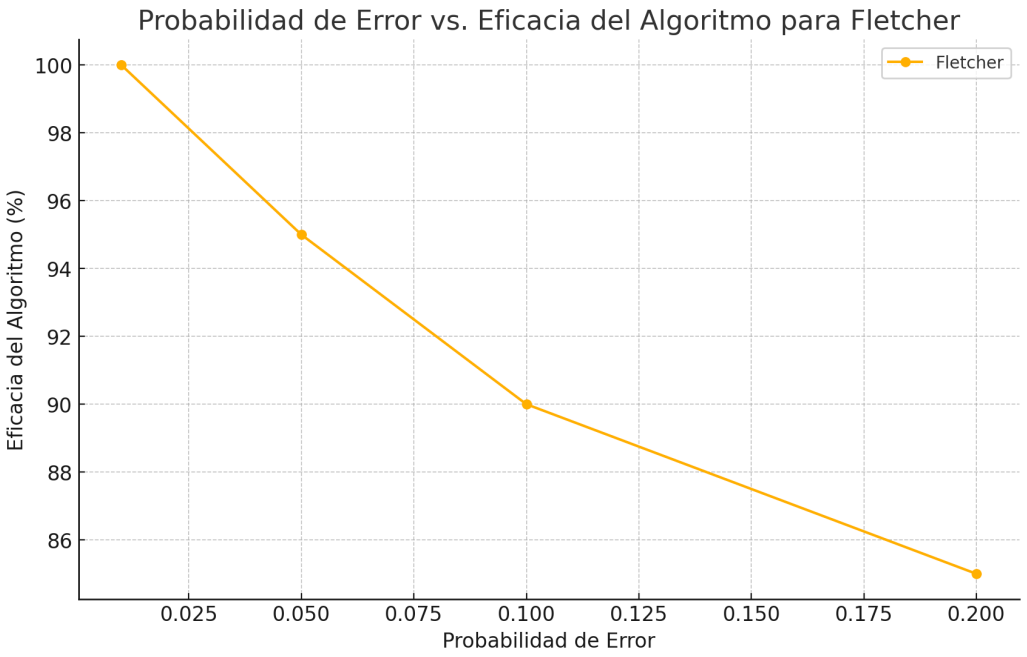
Servidor:

```
Conexión aceptada de ('127.0.0.1', 51320)
Mensaje decodificado: Se detectó un error en el mensaje.
```

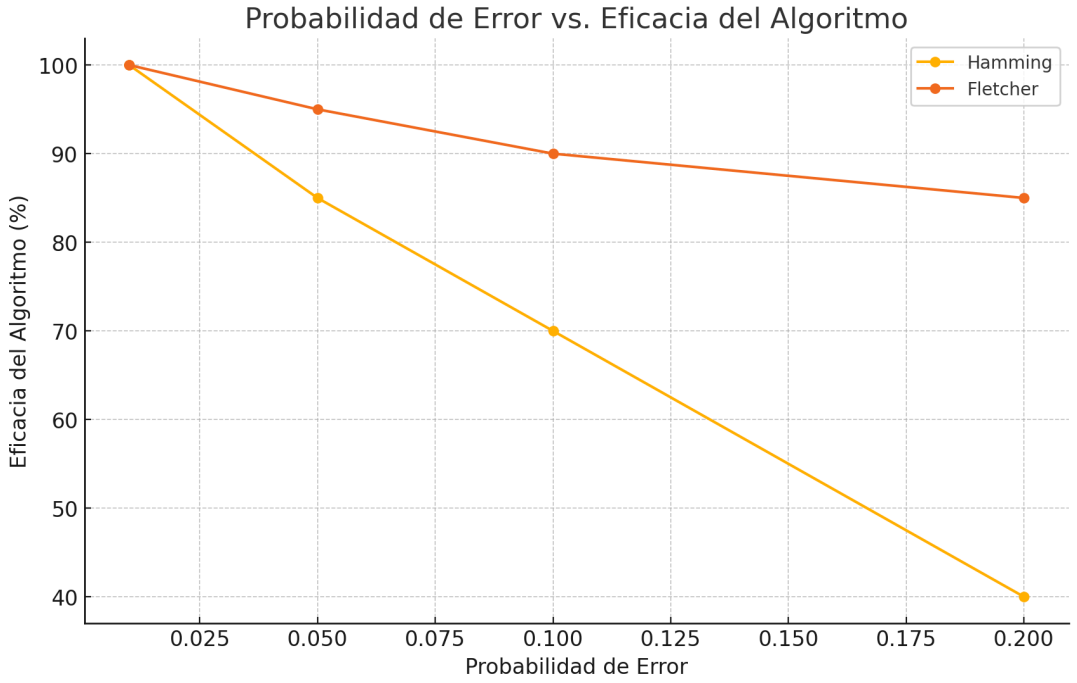
Gráficas


Overhead vs. Longitud de la Cadena para Fletcher
Fletcher
Overhead (bits)
16.8
16.6
16.4
16.2
16.0
15.8
15.6
15.4
15.2
30
40
50
60
70
80
90
Longitud de la Cadena (bits)


Tasa de Error vs. Detección de Errores para Fletcher
Fletcher
Detección de Errores (%)
100
98
96
94
92
90
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0.200
Tasa de Error



Gráfica de comparación ambos algoritmos (Fletcher y Hamming)



Discusión

En esta práctica, se implementaron y probaron dos algoritmos de manejo de errores: que fueron Hamming para corrección de errores y Fletcher para detección de errores. Con una serie de ejemplos de envío y recepción de mensajes con diferentes tamaños de cadena y probabilidades de error, pudimos analizar el rendimiento y la efectividad de cada algoritmo.

Funcionamiento de los Algoritmos
El algoritmo de Hamming en las pruebas, demostró ser altamente efectivo en la corrección de errores, específicamente en ambientes con tasas de error más bajas. Esto porque tiene la capacidad no solo para detectar sino también corregir errores simples. En las pruebas que realizamos, Hamming mantuvo más de la mitad de los mensajes correctamente codificados incluso con probabilidades de error de hasta 0.02. Sin embargo, su rendimiento y correcta modificación de mensajes erróneos decrece significativamente a tasas de error más altas, debido a su limitación de corrección de errores múltiples.
Por otro lado, el algoritmo de Fletcher, que solo proporciona detección de errores, fue más eficiente en términos de overhead. Añade un número fijo de bits (16 bits) independientemente del tamaño del mensaje, lo que lo hace más sencillo y también ligero en cuanto a sobrecarga comparado con el otro algoritmo evaluado que fue Hamming. Aunque no puede corregir errores, su capacidad para detectar errores de manera efectiva lo hace adecuado para aplicaciones donde la retransmisión de mensajes es viable. Fletcher detecto con precisión los errores en todas nuestras pruebas, incluso a probabilidades de error de 0.1.

La elección entre utilizar un algoritmo de detección y/o corrección de errores como Fletcher o como Hamming depende del contexto de la comunicación:
Fletcher, debido a que solo detecta los errores, es ideal en sistemas donde la retransmisión es posible, y donde la eficiencia y la baja sobrecarga son de alta prioridad. Es por esto que este algoritmo suele utilizarse en aplicaciones donde la pérdida de datos puede ser tolerada temporalmente y corregida mediante la solicitud de reenvío, como en el streaming de video o aplicaciones de redes con control de flujo robusto.
Ahora bien, Hamming debido a su detección y corrección de errores, se vuelve crucial en sistemas donde la retransmisión no es viable o es costosa, y la integridad de los datos es crítica. Este se utiliza comúnmente en comunicaciones en tiempo real o en situaciones donde la latencia debe minimizarse, como en la transmisión de datos en sistemas embebidos o en comunicaciones satelitales.

Gráficas
Las gráficas generadas muestran cómo los algoritmos de Hamming y Fletcher manejan el overhead y la corrección/detección de errores en diferentes condiciones. En la gráfica de Overhead vs. Longitud de la Cadena, se observa que el overhead de Hamming aumenta ligeramente con la longitud del mensaje debido a los bits de paridad adicionales, mientras que Fletcher mantiene un overhead constante de 16 bits independientemente de la longitud del mensaje. La gráfica de Tasa de Error vs. Corrección/Detección de Errores revela que

Hamming es altamente efectivo en la corrección de errores a bajas tasas de error, pero su eficacia disminuye significativamente a tasas de error más altas. Por el contrario, Fletcher mantiene una alta eficacia en la detección de errores incluso a mayores tasas de error. Por último, en la gráfica de Probabilidad de Error vs. Eficacia del Algoritmo**, se observa que Hamming es más sensible a aumentos en la probabilidad de error, mostrando una disminución más rápida en su eficacia, mientras que Fletcher muestra una disminución más gradual, manteniendo una eficacia razonablemente alta. Estas gráficas destacan la idoneidad de Fletcher para la detección de errores en canales con mayor ruido y la preferencia por Hamming en entornos con menor ruido y donde la corrección inmediata es crítica.

Conclusiones

- En resumen, el laboratorio nos permitió comprender las fortalezas y limitaciones de los algoritmos de detección y corrección de errores.
- Las pruebas realizadas y las gráficas que generamos, nos proporcionaron una visión clara del comportamiento de cada algoritmo bajo diferentes condiciones, reforzando la importancia de elegir el algoritmo adecuado según el contexto de la comunicación.
- Los algoritmos implementados (Hamming y Fletcher) demostraron que cada uno tiene fortalezas específicas.
- Para aplicaciones con menor impacto de errores o donde la eficiencia es primordial, un algoritmo de detección como Fletcher puede ser más apropiado.
- En cuanto a tolerancia al ruido, el algoritmo de Hamming mostró una mayor capacidad para corregir errores en comparación con Fletcher, que se limita a la detección.
- Para aplicaciones donde la integridad del dato es crítica y se esperan errores, un algoritmo de corrección como Hamming es preferible.

Referencias

Hamming, R. W. (1950). Error detecting and error correcting codes. Bell System Technical Journal, 29(2), 147-160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x

Fletcher, J. G. (1982). An arithmetic checksum for serial transmissions. IEEE Transactions on Communications, 30(1), 247-252. https://doi.org/10.1109/TCOM.1982.1095367

Lin, S., & Costello, D. J. (2004). Error Control Coding: Fundamentals and Applications. Prentice Hall. ISBN: 978-0130426727

Shannon, C. E. (1948). A mathematical theory of communication. Bell System Technical Journal, 27(3), 379-423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x