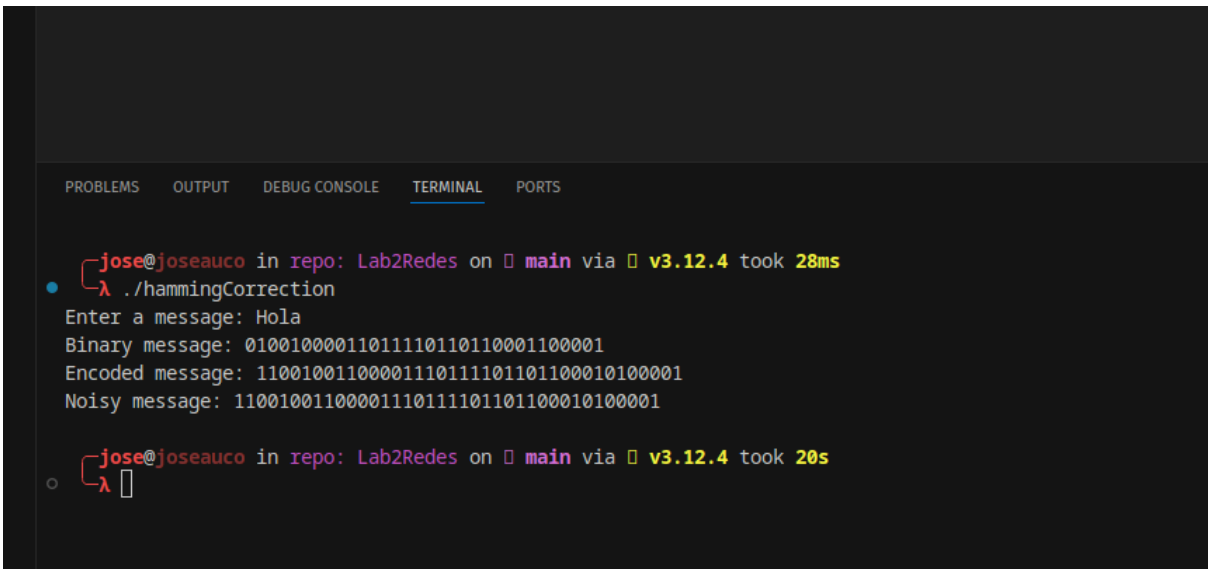
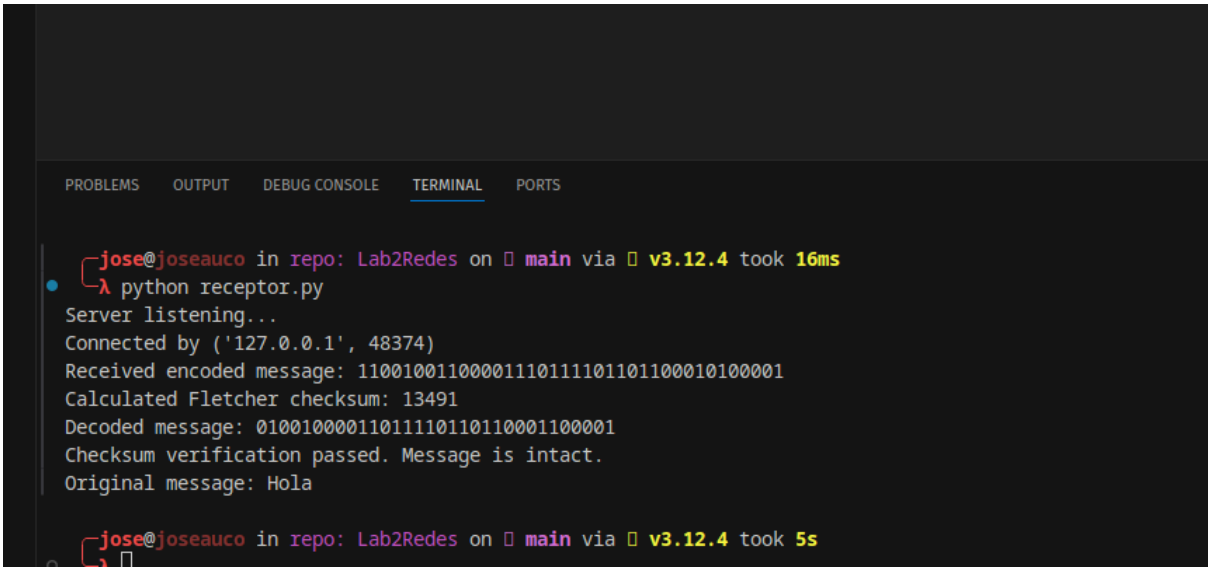
Facultad de Ingeniería
Departamento de Ciencias de la Computación
Universidad del Valle de Guatemala
CC3067 - Redes
Catedrático: Miguel Novella
Julio 25, 2024
José Andrés Auyón - 201579

Laboratorio 2 - Detección y corrección de errores - PT2

Link de github: https://github.com/auyjos/Lab2Redes.git

Funcionamiento antes de realizar pruebas:





- ¿Qué algoritmo tuvo un mejor funcionamiento y por qué?
El rendimiento de un algoritmo de detección o corrección de errores depende de varios factores, incluyendo el tipo de errores que se esperan, la tasa de error en la transmisión y la capacidad del algoritmo para corregir o detectar errores.

Ya que hamming es un algoritmo de corrección de errores. Es particularmente eficaz para corregir errores en situaciones donde la tasa de error es baja a moderada. El código de Hamming puede detectar hasta dos errores y corregir un solo error en un mensaje, lo cual lo hace adecuado para escenarios donde se espera un número relativamente bajo de errores. Su principal ventaja es su capacidad para corregir errores, pero tiene una limitación en términos de flexibilidad y sobrecarga (bits de paridad necesarios) cuando la tasa de errores aumenta.

Checksum, al ser un un algoritmo de detección de errores, no está diseñado para corregir errores, solo para detectarlos. Fletcher es más simple y requiere menos sobrecarga en términos de bits adicionales que el código de Hamming.

En el caso de nuestra aplicación, Checksum de Fletcher resultó siendo un algoritmo más eficiente a medida que la probabilidad de error incrementó al igual que los mensajes, al introducir más probabilidad de error para el mensaje resultaba más complejo de corregir, especialmente, mensajes con mayor longitud ya que había una probabilidad más alta que se cambiaran dos bits del mensaje y Hamming solo puede corregir uno. Checksum, aunque no puede corregir, logró detectar los errores con mayor precisión, hasta el punto de poder determinar en que bits estaban ocurriendo.

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

Checksum de Fletcher es generalmente más flexible para aceptar mayores tasas de errores en comparación con el código de Hamming.

A medida que la tasa de errores aumenta, la probabilidad de que los errores no sean corregidos también aumenta. El código de Hamming tiene limitaciones en términos de la cantidad de errores que puede corregir y detectar. Para tasas de error más altas, puede requerir una cantidad significativamente mayor de bits de paridad para mantener su capacidad de corrección, lo cual puede no ser práctico.

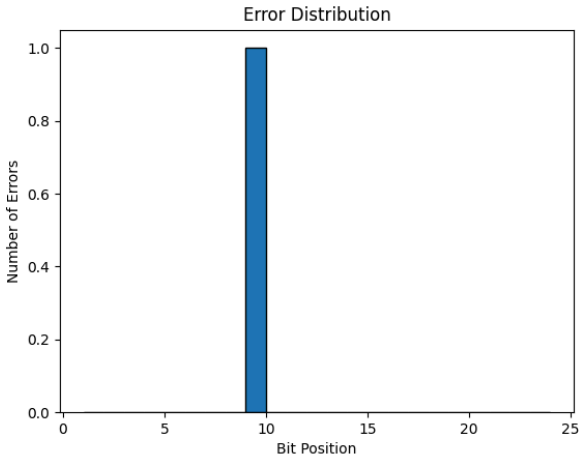
Aunque no corrige errores, Checksum es más flexible en términos de manejar mayores tasas de errores debido a su menor sobrecarga en términos de bits adicionales. Además, su diseño permite una evaluación relativamente rápida del mensaje y puede detectar errores en la mayoría de los casos, aunque no puede corregirlos.

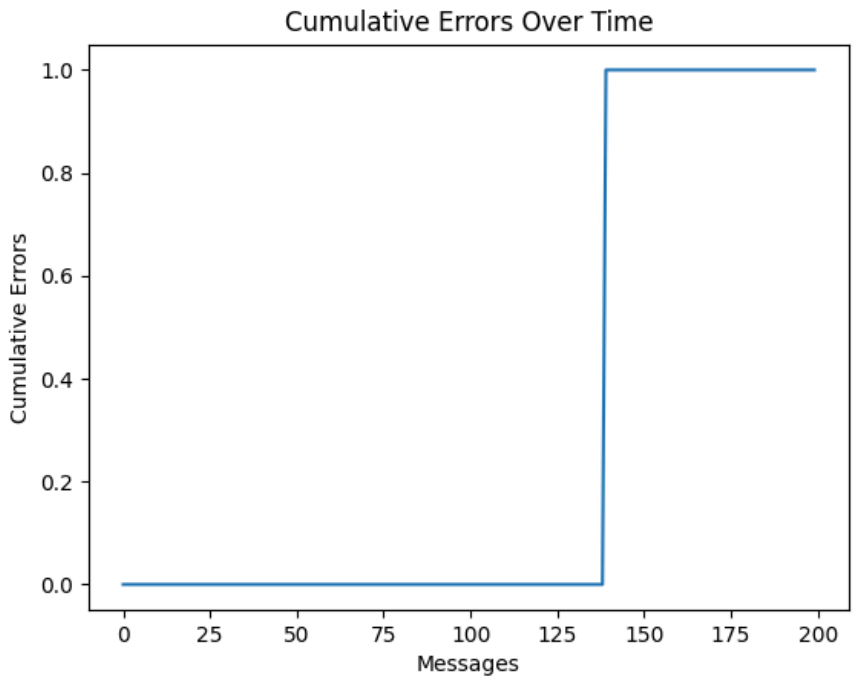
- ¿Cuándo es mejor utilizar un algoritmo de detección de errores en lugar de uno de corrección de errores y por qué?

Si el sistema puede permitir la retransmisión de datos en caso de error, un algoritmo de detección de errores como el checksum puede ser suficiente. Los protocolos de retransmisión pueden pedir nuevamente los datos cuando se detecta un error. Suelen tener una menor sobrecarga en términos de bits adicionales. Esto es útil en sistemas donde los recursos de transmisión son limitados y se necesita optimizar el ancho de banda.También pueden ser útiles en entornos donde se espera que los errores sean poco frecuentes, un algoritmo de detección de errores puede ser adecuado ya que el costo de retransmitir datos en caso de error es bajo.

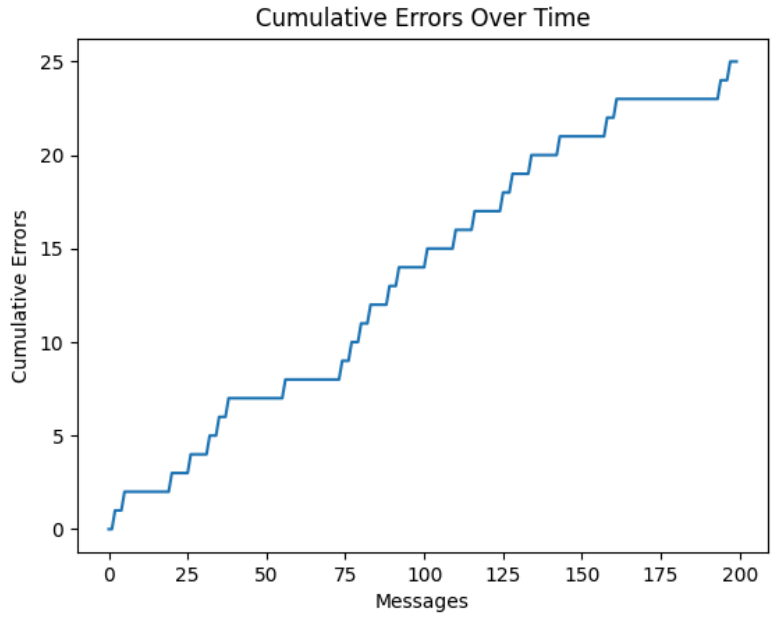
En sistemas donde retransmitir datos no es una opción viable (por ejemplo, en comunicaciones en tiempo real o en sistemas con alta latencia), los algoritmos de corrección de errores como el código de Hamming son necesarios para garantizar que los errores sean corregidos sin necesidad de retransmisión. Son especialmente importantes cuando se espera una tasa de errores alta y pueden ser necesarios para asegurar que la información se transmita correctamente sin necesidad de retransmitir los datos.

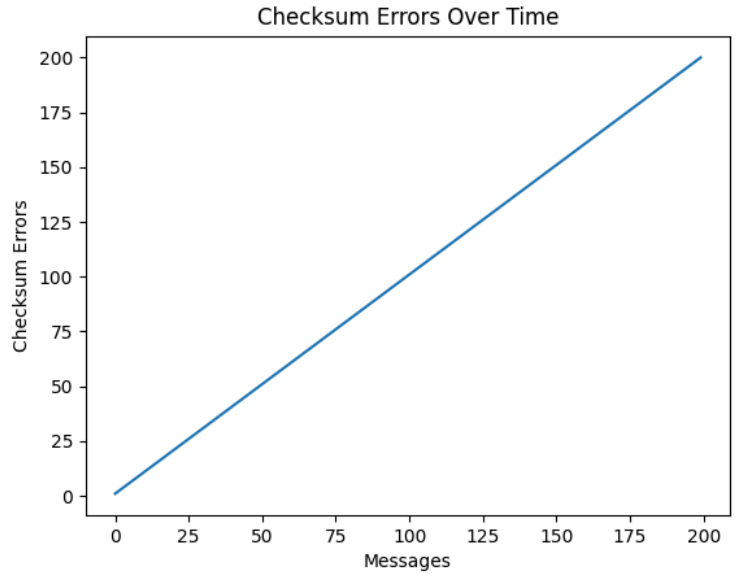
Resultados: Con probabilidad: 1/1000200 mensajes

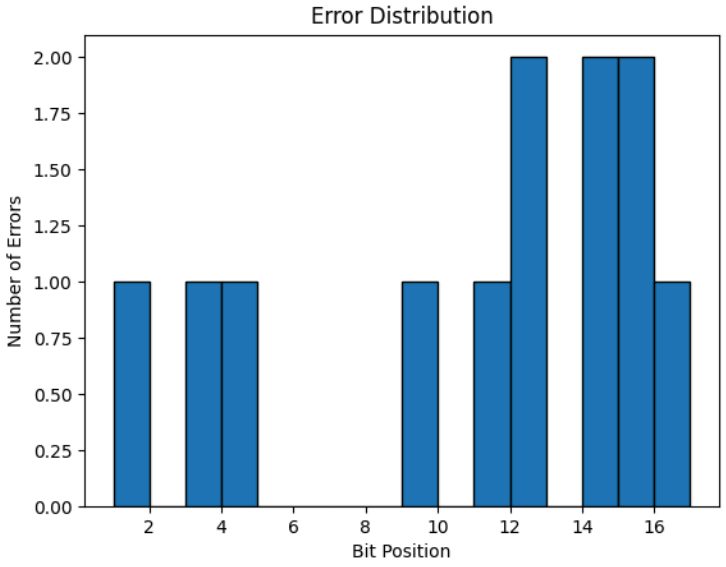


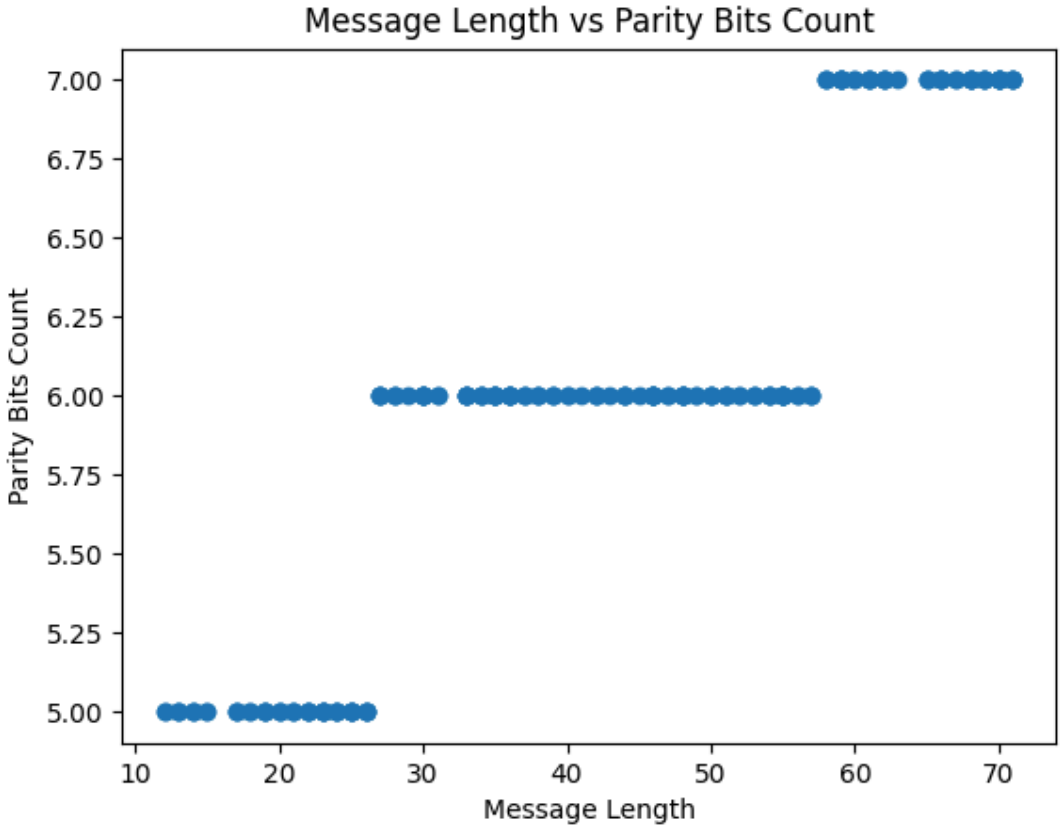


Probabilidad de : 1/100




Checksum Errors Over Time
Checksum Errors
Messages
0
25
50
75
100
125
150
175
200


Error Distribution
Number of Errors
Bit Position
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
2
4
6
8
10
12
14
16


Message Length vs Parity Bits Count
Parity Bits Count
7.00
6.75
6.50
6.25
6.00
5.75
5.50
5.25
5.00
10
20
30
40
50
60
70
Message Length

Discusión:
Luego de realizar las pruebas para ambos algoritmos, con diferencia en la cantidad de mensajes enviados, probabilidad de error y largo de la cadena de mensajes, se obtuvieron las gráficas anteriores. Al observar las gráficas y los resultados obtenidos, podemos determinar que en el caso de la aplicación que se está utilizando, el algoritmo de checksum resultó ser más eficiente ya que , aunque el algoritmo de hamming permitía corregir varios de los errores introducidos como ruido de parte del emisor, a la hora de correr múltiples pruebas con una mayor probabilidad de error, los bits de paridad cambiaban y se introducían dos errores en un solo mensaje, que por la naturaleza del algoritmo de Hamming, no se pudieron corregir, sin embargo, el checksum permitió hacer una validación mucho más rápida y con más precisión ya que éste pudo detectar los errores y en que bits estaban ocurriendo, como se puede observar en la gráfica de distribución de errores por posición de bit.

Conclusiones:

- Hamming es ideal para escenarios donde la corrección de errores es esencial y la tasa de errores es manejable. Su capacidad de corrección automática lo hace adecuado para aplicaciones críticas.
- Checksum de Fletcher es una opción eficiente y flexible para la detección de errores, especialmente en escenarios donde se permite la retransmisión de datos y donde se pueden aceptar tasas de errores más altas. Su menor sobrecarga lo hace adecuado para aplicaciones donde la simplicidad y la eficiencia son importantes.
- Hay una relación directamente proporcional entre la cantidad de errores y la cantidad de mensajes enviados

Resumen:

Los resultados indican que el algoritmo de checksum de Fletcher es más eficiente para la detección de errores en escenarios con alta probabilidad de error y mensajes largos. Aunque el código de Hamming es útil para la corrección de errores en tasas bajas a moderadas, su capacidad se ve limitada cuando la tasa de errores aumenta. El checksum, al detectar errores con mayor precisión, se ajusta mejor a entornos con alta probabilidad de errores.

Referencias bibliográficas:

Wright, G. (2022, July 11). What is hamming code and how does it work?. WhatIs. https://www.techtarget.com/whatis/definition/Hamming-code#:~:text=Hamming%20code%20is%20an%20error,after%20its%20inventor%2C%20Richard%20W.

Fletcher's checksum. Tutorialspoint. (n.d.). https://www.tutorialspoint.com/fletcher-s-checksum

PU5EPX, E. P. (n.d.). *Error detection and correction*. EPx. https://epxx.co/artigos/edc_en.html