UNIVERSIDAD DEL VALLE DE GUATEMALA

Redes - Sección 10

Ing. Miguel Novella Linares

Laboratorio # 2 Pt.2

Maria Ramirez #21342

Gustavo González #21438

Descripción

En este laboratorio de la clase de Redes, el objetivo principal es implementar y demostrar el funcionamiento de algoritmos de corrección y detección de errores a través de un modelo cliente-servidor. Los algoritmos seleccionados para este propósito son el algoritmo de corrección de errores de Hamming y el algoritmo de detección de errores de Fletcher's checksum. Un aspecto clave de este laboratorio es la interoperabilidad entre diferentes lenguajes de programación para los emisores y receptores de cada algoritmo.

Objetivos del Laboratorio:

1. Implementar un algoritmo de corrección de errores utilizando el código de Hamming.
2. Implementar un algoritmo de detección de errores utilizando el checksum de Fletcher.
3. Establecer un modelo cliente-servidor donde los emisores y receptores se comuniquen utilizando diferentes lenguajes de programación.
4. Demostrar la capacidad de los emisores para enviar mensajes a los receptores en distintos lenguajes y realizar las verificaciones respectivas de los algoritmos de corrección y detección de errores.

Detalles de Implementación:

1. **Algoritmo de Corrección de Errores (Hamming):**

- **Emisor:** Implementado en Python.
- **Receptor:** Implementado en Java.
- **Descripción:** El emisor en Python generará un mensaje con los bits de paridad calculados usando el código de Hamming y enviará el mensaje al receptor en Java. El receptor en Java recibirá el mensaje, comprobará los bits de paridad y corregirá cualquier error detectado.

2. **Algoritmo de Detección de Errores (Fletcher's Checksum):**

- **Emisor:** Implementado en Java.
- **Receptor:** Implementado en Python.
- **Descripción:** El emisor en Java calculará el checksum de Fletcher para el mensaje y enviará tanto el mensaje como el checksum al receptor en Python. El receptor en Python recalculará el checksum del mensaje recibido y lo comparará con el checksum enviado para detectar posibles errores.

Modelo Cliente-Servidor:

- **Cliente (Emisor):** Es el responsable de enviar el mensaje junto con los datos necesarios para la corrección o detección de errores.
- **Servidor (Receptor):** Recibe el mensaje, realiza las operaciones de corrección o detección de errores y responde con el estado del mensaje (corregido, si es necesario, o indicando si se ha detectado un error).

Resultados

Pruebas:

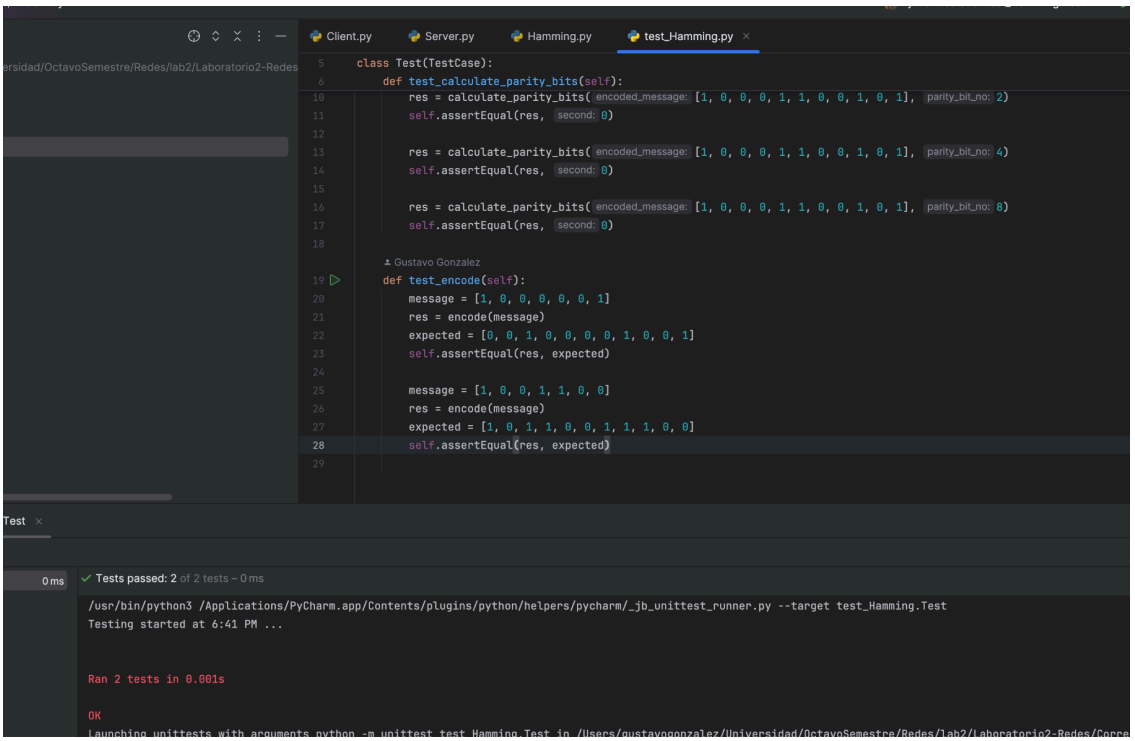

Imagen 1. Pruebas para el algoritmo de Hamming por parte del Emisor

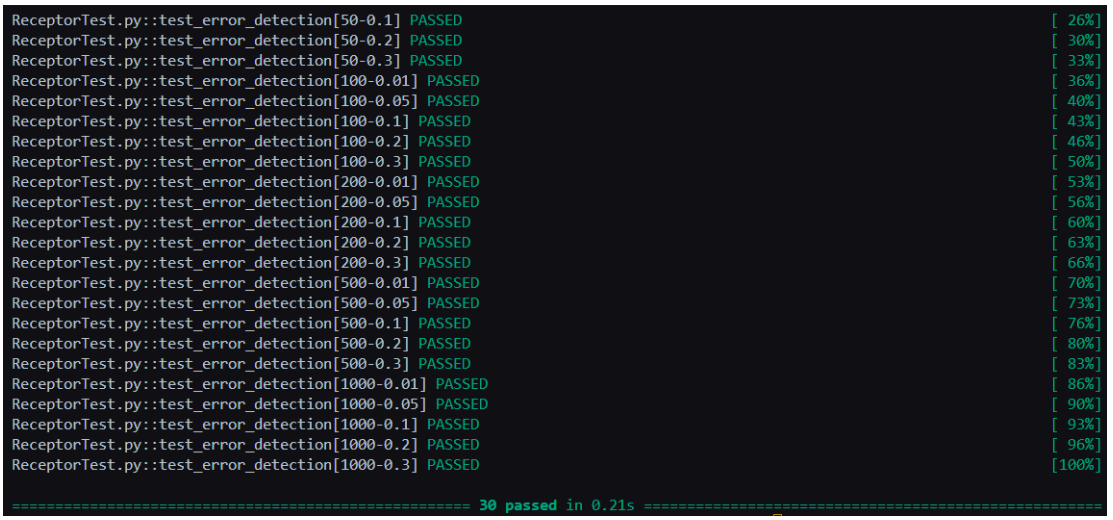

Imagen 2. Pruebas para el algoritmo de Fletcher por parte del Receptor.

Discusión

En nuestro laboratorio, hemos trabajado con dos algoritmos para el envío y recepción de mensajes: el algoritmo de Hamming y el Fletcher Checksum. Ambos tienen sus ventajas y desventajas dependiendo del tipo de errores que queremos manejar.

Por un lado Hamming fue útil para corregir errores de bits individuales. Esto significa que si hay un error en un solo bit, el algoritmo no solo detecta el error, sino que también lo corrige

automáticamente. Esto no es útil en casos donde los errores son relativamente raros y no demasiado complejos. Sin embargo, por el contrario, no es tan bueno para manejar errores en los que hay múltiples bits afectados, y el overhead para la corrección puede ser un problema si los datos son muy grandes.

Por otro lado, Fletcher Checksum se centra en detectar errores en bloques de datos. Aunque no corrige los errores, es muy efectivo para detectar errores en datos grandes y es más flexible cuando la tasa de errores es alta. Si hay muchos errores en los datos, este algoritmo puede detectarlos mejor que Hamming, pero necesitaría algún otro mecanismo para manejar los errores, como la retransmisión de datos.

Conclusiones

- Hamming es efectivo para corregir errores en bits individuales, ideal cuando los errores son simples y no demasiado frecuentes.
- La elección entre Hamming y Fletcher Checksum depende de si necesitamos corrección automática o una buena detección de errores en datos grandes y complejos.
- Fletcher Checksum es mejor para detectar errores en bloques de datos grandes y es más flexible ante una alta tasa de errores.
- Fletcher Checksum no corrige errores, pero detecta una amplia variedad de errores, siendo adecuado para manejar datos con alta tasa de errores.
- Hamming puede ser limitado en situaciones con errores complejos o cuando la tasa de errores es alta, debido a su overhead para corrección.

Link al repositorio:
https://github.com/mariaRam2003/Laboratorio2-Redes/tree/lab2

Referencias

Nakassis, A. (1988). Fletcher's error detection algorithm: how to implement it efficiently and how toavoid the most common pitfalls. *ACM SIGCOMM Computer Communication Review*, *18*(5), 63-88.

Sanjay Raghavendra (2020) *Implementation of Fletcher CheckSum* [Video]. YouTube. https://www.youtube.com/watch?v=e4nuUYiQE-A

3Blue1Brown (2020) *Pero ¿qué son los códigos Hamming? El origen de la corrección de errores* [Video]. YouTube. https://www.youtube.com/watch?v=X8jsijhllIA&t=171s