Universidad del Valle de Guatemala
Departamento de Ciencias de la Computación
CC3067 Redes
Jorge Andrés Yass Coy

# Laboratorio No. 2 Parte 2

*Esquemas de detección y corrección de errores*

Ana Paola de León 203061
Gabriela Paola Contreras 20213

Guatemala, 10 de agosto de 2023

## Introducción

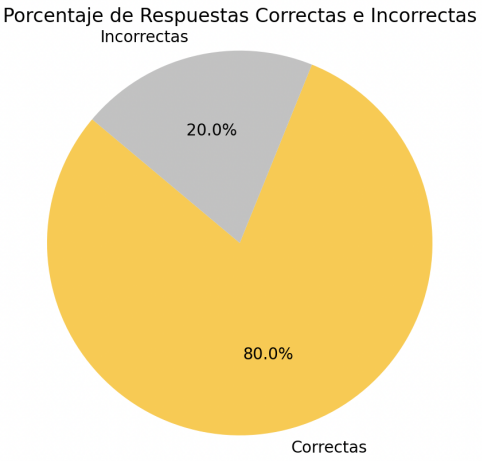
El objetivo de esta práctica fue comprender y poner en práctica el funcionamiento de un modelo de capas y sus servicios mediante la implementación de sockets para lograr la transmisión de datos para así poder analizar el funcionamiento de los algoritmos para la corrección y dirección de errores.

Esta actividad fue realizada con las mismas parejas de la parte uno, para ello se utilizaron los algoritmos realizados previamente sólo que a estos se les añadieron sockets para así poder enviar los datos en una sola ejecución y no depender de archivos. Así mismo, los programas fueron mejorados con la finalidad de poder aceptar palabras y no un conjunto de bits. Es importante destacar que antes de enviar al receptor el mensaje se aplicó una capa de ruido que incorporó probabilidades con el propósito de simular que el mensaje había sido expuesto a interferencias. Por otra parte, este también consistió en simular una serie de mensajes para así poder analizar y comparar el funcionamiento de los algoritmos. Finalmente, se puede concluir que usar modelos de capas en la transmisión de datos, son esenciales para una comunicación exitosa, no obstante cuando estos se exponen a canales inseguros, se resalta la importancia de tener métodos de detección y corrección de errores como defensa.

## Resultados

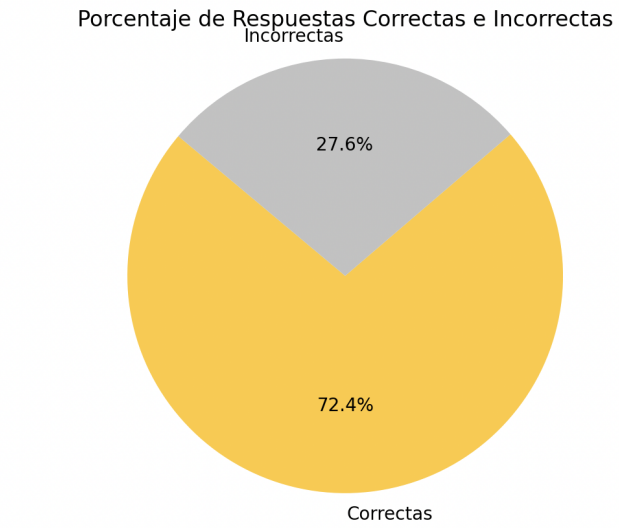
- **Algoritmo CRC**

*Figura No.1 Resultados simulación 50 palabras*



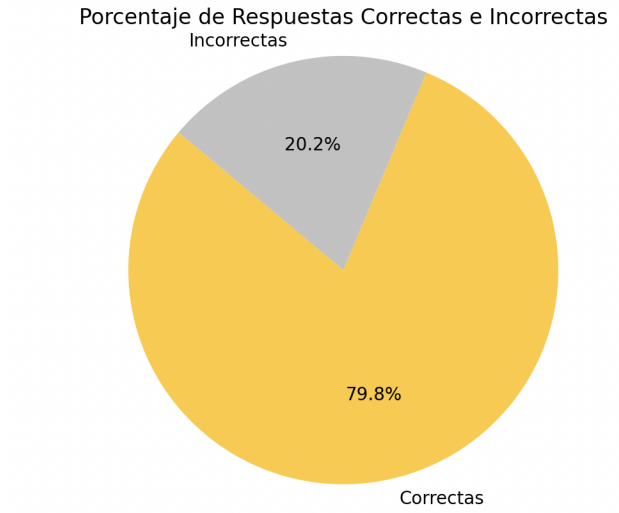

Para esta se simularon 50 palabras de diferentes tamaños obteniendo así 80% palabras decodificadas correctamente y 20% de manera incorrecta

*Figura No.2 Resultados simulación 250 palabras*



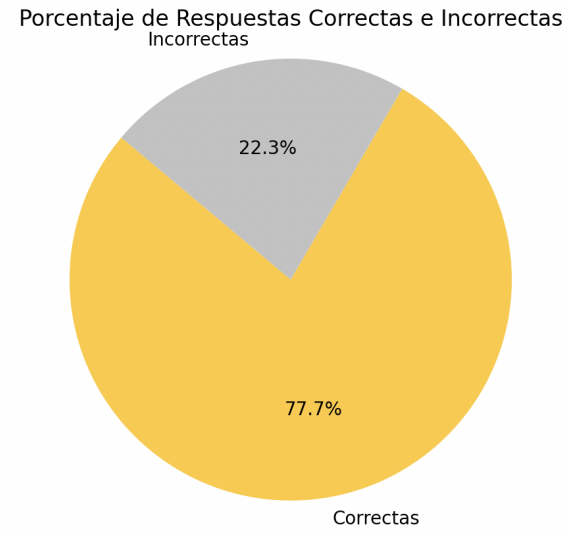

Para esta se simularon 250 palabras de diferentes tamaños obteniendo así 72.4% palabras decodificadas correctamente y 27.4% de manera incorrecta

*Figura No.3 Resultados simulación 500 palabras*




Para esta se simularon 500 palabras de diferentes tamaños obteniendo así 79.8% palabras decodificadas correctamente y 20.2% de manera incorrecta

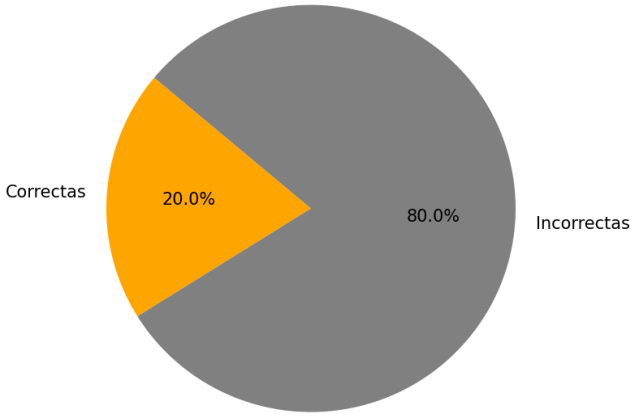
*Figura No.4 Resultados simulación 1000 palabras*




Para esta se simularon 1000 palabras de diferentes tamaños obteniendo así 77.7% palabras decodificadas correctamente y 22.3% de manera incorrecta

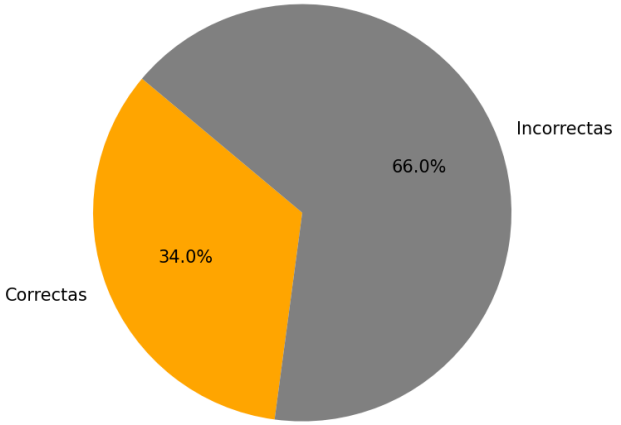
- **Algoritmo Hamming**

*Figura No.5 Resultados simulación 50 palabras*



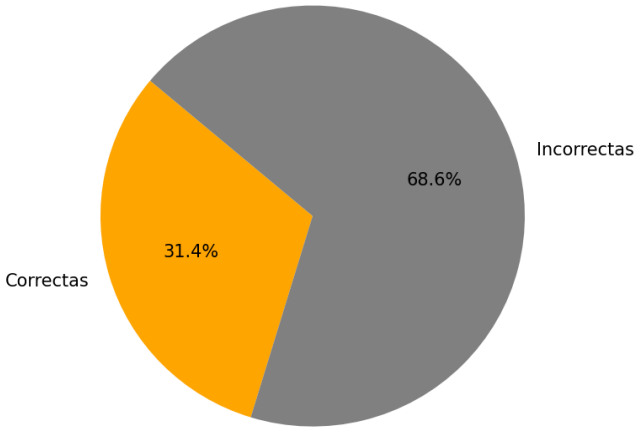

Para esta se simularon 50 palabras de diferentes tamaños obteniendo así 12 palabras decodificadas correctamente y 38 de manera incorrecta

*Figura No.6 Resultados simulación 250 palabras*



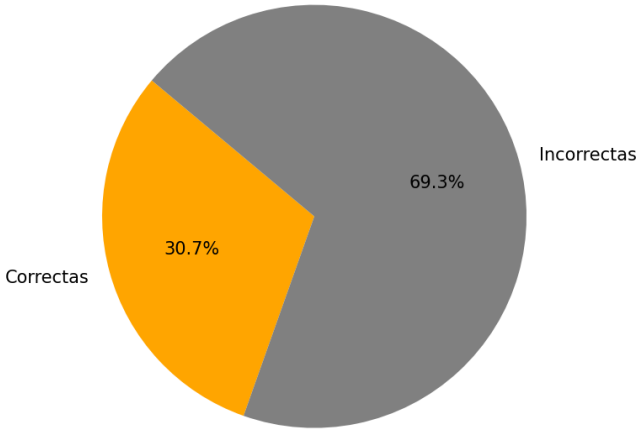

Para esta se simularon 100 palabras de diferentes tamaños obteniendo así 85 palabras decodificadas correctamente y 165 de manera incorrecta

*Figura No.7 Resultados simulación 500 palabras*




Para esta se simularon 500  palabras de diferentes tamaños obteniendo así 157 palabras decodificadas correctamente y 343 de manera incorrecta

*Figura No.8 Resultados simulación 1000 palabras*




Para esta se simularon 1000 palabras de diferentes tamaños obteniendo así 307 palabras decodificadas correctamente y 693 de manera incorrecta

**Discusión**

El objetivo de este laboratorio fue conocer el funcionamiento de un modelo de capas y sus servicios mediante la implementación de sockets para la transmisión de mensajes, para cumplir dicho objetivo se modificaron los algoritmos de Hamming y CRC realizados previamente para poder utilizar sockets, además se automatizo el ruido y por último la codificación y decodificación de palabras.

Al desarrollar el algoritmo CRC-32 se optó por realizar una codificación en donde primero se solicitaba el mensaje al usuario, luego cada carácter se convertía a una representación binaria de 8 bits (utilizando ASCII) para luego concatenar todas las representaciones binarias en un solo string y así enviarlo por medio de un socket al receptor. Sin embargo, al realizar esta implementación e implementar el algoritmo de ruido pudimos observar que la decodificación de los mensajes no siempre era la correcta y que el ruido si tenía un impacto en el resultado final a diferencia del laboratorio pasado donde se tuvo una precisión del 100%. Pues al realizar las pruebas, se pudo observar que el algoritmo tuvo una precisión de cerca del 70%.

El algoritmo de hamming con el cual se trabajó fue (7:4) en donde 4 bits son de data y 3 bits son de paridad. Debido a que la cantidad de bits que cada letra generaba se tuvo que dividir la cantidad la trama cada 4 caracteres para poder ser procesada por el emisor, seguidamente con la ayuda de sockets se envió un string, el cual contenía todos los mensajes codificados. Es importante destacar que los mensajes codificados fueron procesados por una capa de ruido, en la cual se usó la probabilidad del 0.3% de que un bit fuera cambiado, esto con la finalidad de evitar realizar cambios en más de un bit y de esta manera no perjudicar la eficiencia del algoritmo. Luego de todo esto ya el receptor realizaba su parte y decodificaba los mensajes para poder ser desplegados en la terminal.

Ahora bien, en cuanto a las simulaciones se puede mencionar que se realizaron cuatro con diferentes cantidades de palabras, en total se simularon 1800 palabras de las cuales la cantidad de incorrectas predominaron como se puede observar en los gráficos. La razón por la cual la mayoría de resultados fueron incorrectos puede deberse al porcentaje de ruido que se le aplicó a cada bit, puesto a que esta era pequeña no se nos garantizó que en una secuencia de 7 bits únicamente iba a encontrarse un error. Lo cual hace que este algoritmo sea un poco ineficiente al momento de enviar mensajes con transferencia de por medio.

Al comparar ambos resultados de las pruebas del algoritmo Hamming con el algoritmo CRC-32 se puede decir que existe una diferencia bastante grande entre ambos algoritmos pues, en general, al utilizar CRC32 hubo una menor cantidad de mensajes incorrectos. Sin embargo, para futuras implementaciones, recomendamos que cada uno de los caracteres de los mensajes sean analizados individualmente para así disminuir la tasa de error y mejorar la precisión del algoritmo.

**Conclusiones**

- Para futuras prácticas se recomienda considerar la forma y/o tasa de error pues a pesar de que el algoritmo de hamming demostró ser efectivo en algunos contextos es importante considerar alternativas más efectivas para entornos con niveles más altos de ruido.

- El algoritmo CRC32 demostró ser un algoritmo con una precisión de alrededor del 70%, lo cual indica que los mensajes en su mayoría son decodificados de manera correcta. Sin embargo, para futuras implementaciones, recomendamos que cada uno de los caracteres de los mensajes sean analizados individualmente para así disminuir la tasa de error y mejorar la precisión del algoritmo.

- Se puede decir que existe una diferencia bastante grande entre ambos algoritmos pues, en general, al utilizar CRC32 hubo una menor cantidad de mensajes incorrectos, mientras que con el algoritmo de Hamming la tasa de error fue alta.

**Cita y referencias**


- Wright, G. (2022). Hamming code. *WhatIs.com*. https://www.techtarget.com/whatis/definition/Hamming-code#:~:text=Hamming%20code%20is%20an%20error,data%20is%20stored%20or%20transmitted.

- *Comprender los errores de verificación de redundancia cíclica en los switches Nexus*. (2022, 9 marzo). Cisco. https://www.cisco.com/c/es_mx/support/docs/ios-nx-os-software/nx-os-software/217554-understand-cyclic-redundancy-check-crc.html

- *Sockets con Java (servidor) y Python (Cliente) sin flush en Python*. (s. f.). Stack Overflow en español. https://es.stackoverflow.com/questions/84552/sockets-con-java-servidor-y-python-cliente-sin-flush-en-python