Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 - Redes

Laboratorio 2 - Segunda parte

Esquemas de Detección y Corrección de Errores

Stefano Alberto Aragoni Maldonado (20261)
Carol Andreé Arévalo Estrada (20461)

1. Introducción

En la primera etapa del laboratorio, se llevó a cabo la implementación de dos algoritmos: el CRC32, destinado a la detección de errores, y el algoritmo de Hamming, utilizado para la corrección de errores en datos. Basándose en estos avances, en la presente práctica se procedió a desarrollar una arquitectura de capas, que consta de las capas de aplicación, presentación, enlace, ruido y transmisión. El objetivo principal fue lograr la codificación y decodificación efectiva de mensajes –entre un emisor y receptor– empleando sockets como medio de comunicación.

Para finalizar, se llevaron a cabo exhaustivas pruebas automatizadas que tuvieron como propósito evaluar la solidez y confiabilidad de los algoritmos implementados y la arquitectura de capas diseñada. Es importante destacar que este proceso se realizó con el propósito de adquirir una comprensión profunda sobre el funcionamiento de un modelo de capas y para enfrentar los desafíos inherentes de este enfoque.

2. Objetivos

a. Comprender el funcionamiento de un modelo de capas y sus servicios
b. Implementar sockets para la transmisión de información
c. Experimentar la transmisión de información expuesta a un canal no confiable
d. Analizar el funcionamiento de los esquemas de detección y corrección

3. Descripción

En esta práctica se simuló un proceso de comunicación mediante una arquitectura de capas. Se aprovecharon los algoritmos previamente desarrollados en el laboratorio anterior, específicamente, Código Hamming y CRC-32. La metodología empleada se basa en la colaboración de distintas clases, cada una desempeñando un papel crucial en las diversas etapas de la comunicación. Se emplea una estructura jerárquica de capas, cada una con funciones específicas que contribuyen al proceso integral de transmisión, presentación, detección y corrección de errores.

Es importante resaltar que para esta implementación se optó por programar las capas del emisor en Python, mientras que las del receptor se desarrollaron en Java. Esta logró demostrar que, independientemente del lenguaje de programación utilizado, una comunicación efectiva puede lograrse mediante un protocolo establecido y una arquitectura bien implementada.

A continuación se describe el funcionamiento de los servicios de las capas utilizados por el emisor.

- *Capa de Aplicación:*

  En primer lugar, la capa de aplicación interactúa con el usuario, permitiéndole ingresar un mensaje, elegir un método de codificación (Hamming o CRC-32) y definir la probabilidad de ruido.

- *Capa de Presentación:*

  La capa de presentación se encarga de convertir el mensaje de texto ingresado en su representación binaria utilizando el código ASCII. Esta etapa es crucial para la posterior transmisión y codificación de datos.

- *Capa de Enlace:*

  La capa de enlace implementa el método de detección o corrección de errores seleccionado por el usuario en la capa de aplicación. Para el método Hamming, se calculan los bits redundantes necesarios para detectar y corregir errores, mientras que para el método CRC-32 se genera un código de detección de errores. Esta capa permite fortalecer la integridad de los datos antes de la transmisión.

- *Capa de Ruido:*

  La capa de ruido introduce aleatoriamente ruido en la trama binaria para simular posibles interferencias durante la transmisión del lado del receptor. La probabilidad de ruido se ajusta según la especificación del usuario, contribuyendo a evaluar la resistencia de los algoritmos de detección y corrección de errores.

- *Capa de Transmisión:*

  Finalmente, la capa de transmisión establece una conexión a través de sockets con el receptor. Envía la trama binaria con ruido, reflejando las condiciones reales de una comunicación. Este proceso demuestra cómo los datos se transmiten a través de la red y llegan al receptor.

En el caso del lado del receptor, se utilizaron las mismas capas (exceptuando la capa de ruido). Sin embargo, estas presentaron diferentes servicios para el receptor. A continuación se describe el funcionamiento de estos:

- *Capa de Transmisión:*

  Del lado del receptor, la capa de transmisión se encarga de recibir el mensaje enviado por el emisor. Esto a través de primero establecer una conexión con el emisor a través de un socket definido. Cabe destacar que el receptor siempre debe de estar “escuchando”, a la espera de un mensaje.

- *Capa de Enlace:*

  La capa de enlace hace uso de los métodos de detección o corrección de errores. Aquí, se analiza la integridad de los mensajes recibidos y se detectan posibles errores. En caso de haber utilizado el método de Código de Hamming, se hacen las respectivas correcciones para eliminar los errores detectados.

- *Capa de Presentación:*

  La capa de presentación, en caso de no haber errores, se encarga de convertir el mensaje binario en caracteres. En caso que se hayan detectado errores, esto se debe indicar en la capa de aplicación.

- *Capa de Aplicación:*

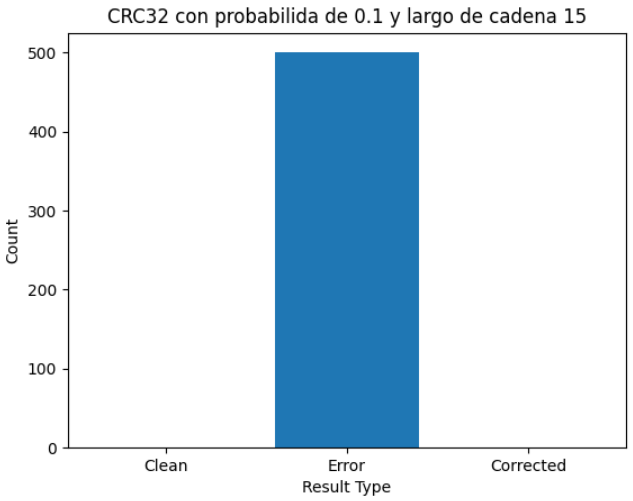
  Finalmente, la capa de aplicación se encarga de interactuar con el receptor. En caso el mensaje no haya tenido errores (o se hayan corregido), se muestra el mismo al receptor. Si se detectaron errores y no fue posible corregirlos, se indica un mensaje de error.

A partir de esto, con el objetivo de evaluar el desempeño de los algoritmos y el sistema de capas desarrollado, se desarrolló un programa para llevar a cabo pruebas automatizadas. Estas pruebas consistieron en la transmisión de mensajes desde un emisor hasta un receptor, empleando los métodos de CRC32 y Código Hamming. Además, se exploraron distintas tasas de probabilidad de ruido, variados tamaños de cadenas de datos, así como niveles diversos de redundancia de código. A continuación se presentan los resultados de las pruebas realizadas.

4. Resultados

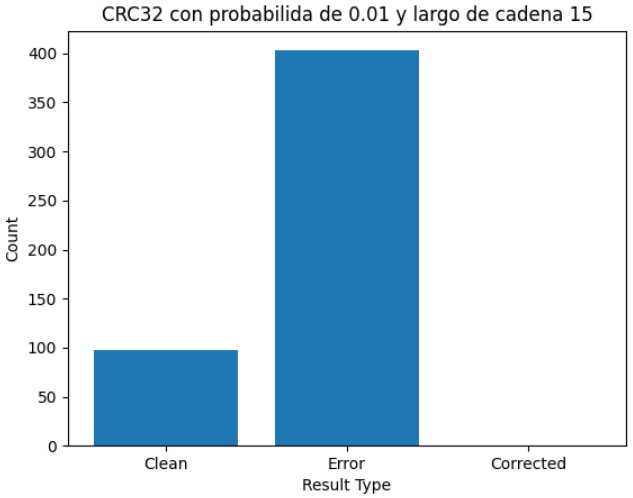
La práctica involucró la simulación de la comunicación mediante una arquitectura de capas, utilizando los algoritmos de Hamming y CRC32 previamente implementados. Se realizaron pruebas automatizadas con variaciones en la probabilidad de ruido, el algoritmo utilizado y la longitud de la cadena. Cada tipo de prueba se repitió 500 veces para obtener resultados consistentes y significativos. El análisis de los resultados se presenta a continuación, destacando las tendencias observadas y las conclusiones obtenidas.

Figura 1. *Algoritmo de CRC32 con probabilidad de ruido de 0.1 y un largo de cadena de 15 caracteres.*



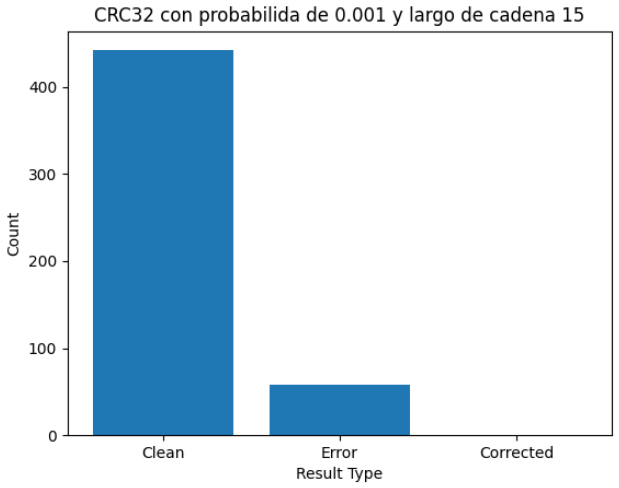
Como se puede observar en la *Figura 1*, el 100% de las tramas procesadas con CRC32 con una probabilidad de ruido relativamente alta y un largo de cadena medio fueron descartadas por errores. Esto debido a que este algoritmo tiene la limitante de no poder corregir errores en las tramas recibidas.

Figura 2. *Algoritmo de CRC-32 con probabilidad de ruido de 0.01 y un largo de cadena de 15 caracteres.*



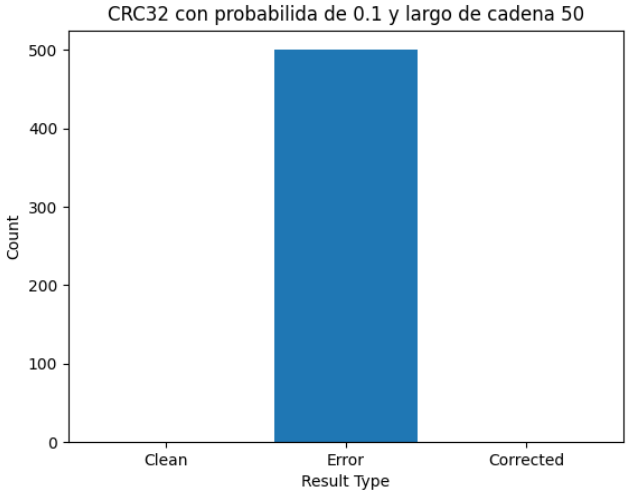
Como se puede observar en la *Figura 2*, la mayoría de las tramas procesadas con CRC32 con una probabilidad de ruido media y un largo de cadena medio fueron descartadas por errores. Sin embargo, un 20% de las tramas resultaron correctas. Esto indica que mientras más baja es la probabilidad de ruido, menor es la posibilidad de que un mensaje sea posteriormente descartado.

Figura 3. *Algoritmo de CRC-32 con probabilidad de ruido de 0.001 y un largo de cadena de 15 caracteres.*



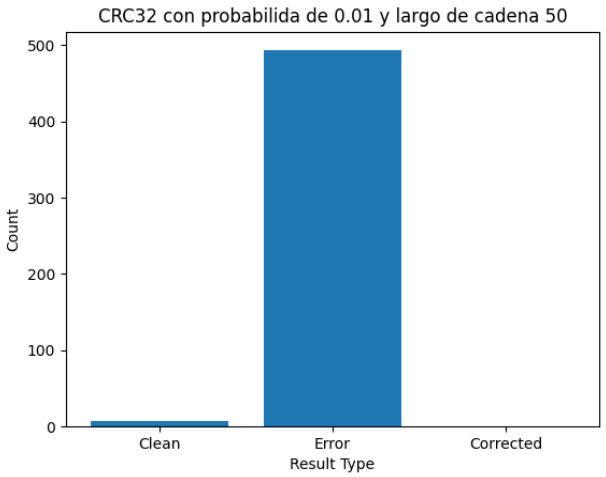
Como se puede observar en la *Figura 3*, la mayoría de las tramas procesadas con CRC32 con una probabilidad de ruido baja y un largo de cadena medio fueron decodificadas exitosamente y solo un 10% resultaron con errores. Nuevamente confirmando lo anteriormente mencionado, que una probabilidad de ruido baja garantiza menos errores.

Figura 4. *Algoritmo de CRC-32 con probabilidad de ruido de 0.1 y un largo de cadena de 50 caracteres.*



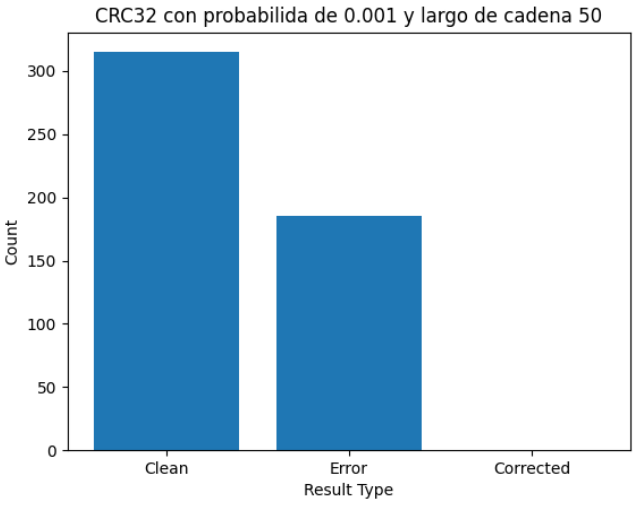
Al igual que la Figura 1, la *Figura 4* tiene una probabilidad de ruido relativamente alta, lo que resultó en un 100% de las tramas con errores.

Figura 5. *Algoritmo de CRC-32 con probabilidad de ruido de 0.01 y un largo de cadena de 50 caracteres.*



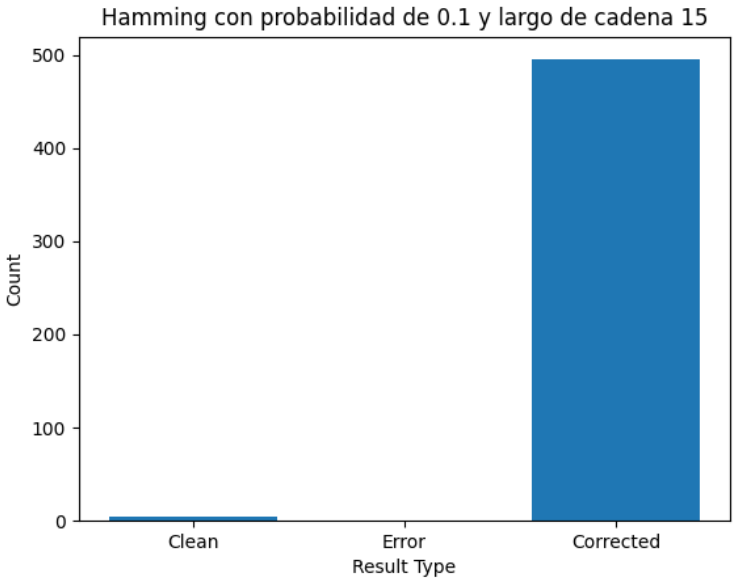
Como se puede observar en la *Figura 5*, la mayoría de tramas procesadas con CRC32 con una probabilidad media y una cadena larga resultaron con errores. Sin embargo, hubo un único mensaje atípico que no tuvo ningún error. Esto a comparación de la *Figura 2*, donde una mayor cantidad de mensajes no tenían errores –dada la misma probabilidad de ruido–.

Figura 6. Algoritmo de CRC-32 con probabilidad de ruido de 0.001 y un largo de cadena de 50 caracteres.



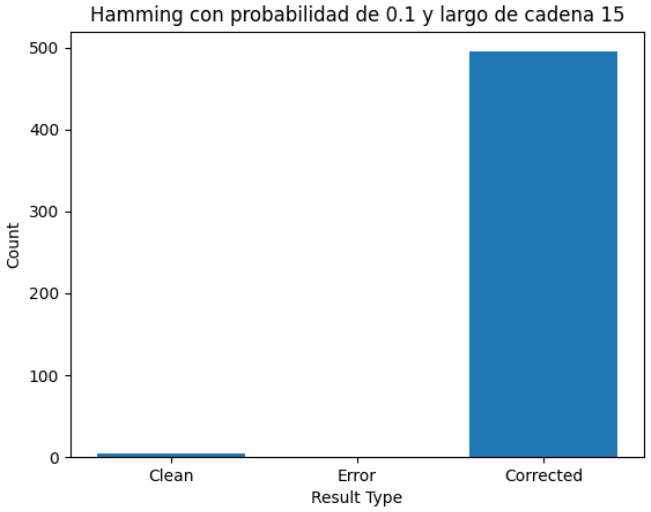
Como se puede observar en la *Figura 6*, la mayoría de tramas procesadas resultaron sin errores. Sin embargo, al compararlo con la *Figura 3*, a pesar de tener la misma probabilidad de ruido, al tener una cadena más larga se obtuvo una mayor cantidad de tramas con errores.

Figura 7. *Algoritmo de Hamming con probabilidad de ruido de 0.1 y un largo de cadena de 15 caracteres.*



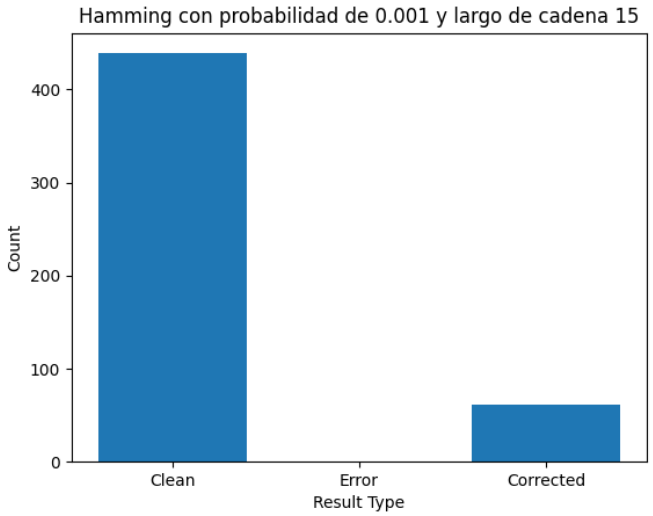
Como se puede observar en la *Figura 7*, casi el 100% de las tramas procesadas con Hamming con una probabilidad de ruido alta y un largo de cadena medio tuvieron errores. Como resultado, el algoritmo corrigió dichos errores detectados.

Figura 8. *Algoritmo de Hamming con probabilidad de ruido de 0.01 y un largo de cadena de 15 caracteres.*



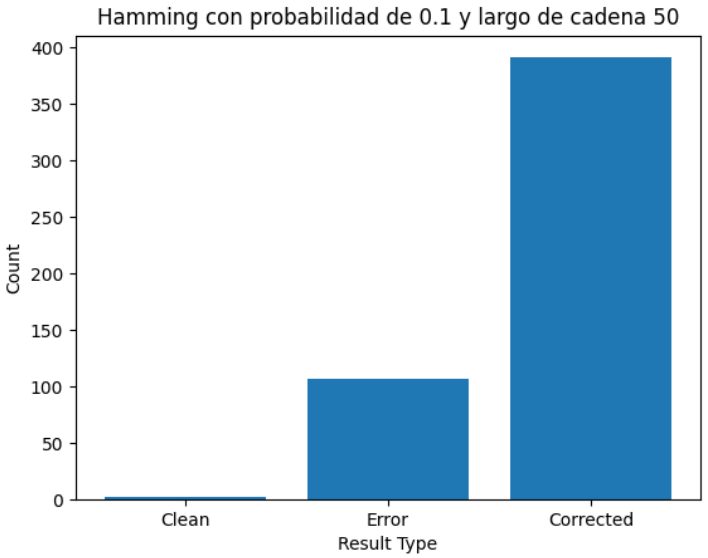
Como se puede observar en la *Figura 8*, casi el 100% de las tramas procesadas con Hamming con una probabilidad de ruido media y un largo de cadena medio fueron corregidas. A diferencia del CRC32, el algoritmo de Hamming puede corregir errores por lo que a pesar de que la probabilidad de ruido es media, las tramas pudieron ser corregidas exitosamente.

Figura 9. *Algoritmo de Hamming con probabilidad de ruido de 0.001 y un largo de cadena de 15 caracteres.*



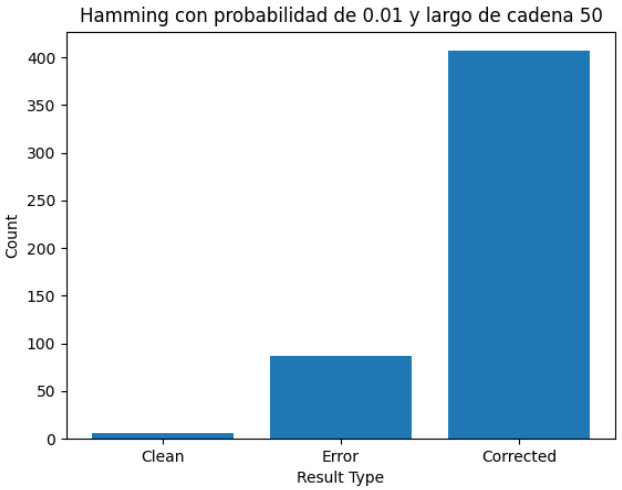
Como se puede observar en la *Figura 9*, la mayoría de las tramas procesadas con Hamming con una probabilidad de ruido baja y un largo de cadena medio estaban correctas. Las cadenas que presentaron errores fueron corregidas exitosamente.

Figura 10. *Algoritmo de Hamming con probabilidad de ruido de 0.1 y un largo de cadena de 50 caracteres.*



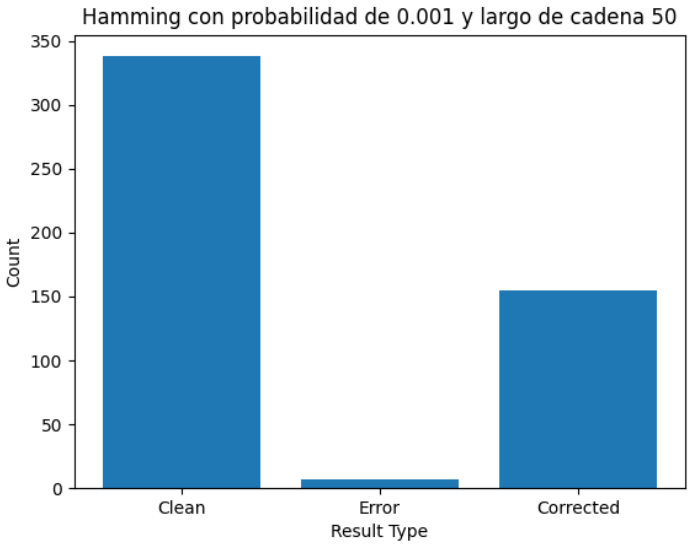
Como se puede observar en la *Figura 10*, a pesar de tener una probabilidad realmente alta de ruido y una cadena larga, el algoritmo pudo corregir la mayoría de las tramas con errores. Sin embargo, hubo otras tramas donde detectó errores pero no pudo corregirlos debido a la cantidad de los mismos.

Figura 11. *Algoritmo de Hamming con probabilidad de ruido de 0.01 y un largo de cadena de 50 caracteres.*



Como se puede observar en la *Figura 11*, esta se parece mucho a la *Figura 10*. A pesar del ruido y el largo de la cadena, la mayoría de tramas fueron corregidas exitosamente.

Figura 12. *Algoritmo de Hamming con probabilidad de ruido de 0.001 y un largo de cadena de 50 caracteres.*



Como se puede observar en la *Figura 12*, con una probabilidad de ruido baja la mayoría de tramas estaban correctas. Sin embargo, de las que no estaban correctas, una gran parte fueron corregidas exitosamente.

En resumen, se pueden destacar los siguientes puntos:

1. Algoritmo de CRC32:
    a. A medida que la probabilidad de ruido aumentó, el porcentaje de tramas con errores también aumentó considerablemente
    b. Con una probabilidad de ruido alta (0.1), todas las tramas generadas fueron descartadas debido a errores.
    c. A medida que la longitud de la cadena aumentó, se observó una un aumento en la cantidad de tramas con errores; incluso con una probabilidad de ruido baja.

2. Algoritmo de Hamming
    a. El algoritmo de Hamming demostró una mayor robustez en comparación con CRC32 por su capacidad de corrección de errores.
    b. A pesar de la probabilidad de ruido, el algoritmo de Hamming tuvo un buen desempeño en la corrección de errores.
    c. A medida que la longitud de la cadena aumentó, el algoritmo de Hamming mantuvo una notable capacidad para corregir errores, incluso en condiciones de alta probabilidad de ruido.
        i. También se observó cómo al aumentar la longitud de la cadena, aumentó la cantidad de tramas con errores. Esto incluso con una probabilidad baja de ruido.

5. Discusión

En la primera parte de esta práctica, se desarrolló una aplicación para la transmisión y recepción de mensajes, en base a una arquitectura de capas con distintos servicios. A través de esta experiencia, se logró comprender profundamente la importancia de un modelo de capas y sus servicios. Asimismo, se logró correctamente implementar sockets para experimentar con la transmisión de información. Finalmente, también se logró utilizar nuevamente los métodos de detección y corrección de errores (CRC-32 y Código Hamming) anteriormente desarrollados.

A lo largo del proceso de desarrollo, se pudo apreciar cómo la arquitectura de capas es un enfoque efectivo para estructurar y modularizar las diferentes etapas de la comunicación. Esta organización nos facilitó la comprensión de las responsabilidades de cada capa y permitió la integración correcta de los algoritmos de detección y corrección de errores.

Por otro lado, con el propósito de evaluar la eficacia de la arquitectura y los algoritmos implementados, se realizaron una serie de pruebas automatizadas. Para esto, se enviaron diferentes mensajes del emisor al receptor, variando la longitud del mensaje, la probabilidad de ruido, y el método de detección/corrección utilizado. Esto con el propósito de determinar cómo reaccionaba cada algoritmo bajo diferentes circunstancias. Cabe destacar que cada combinación de variables fue

sometida a 500 iteraciones, cuyos resultados fueron graficados para un análisis posterior.

La automatización de las pruebas, aunque fue desafiante, resultó esencial para obtener resultados significativos y consistentes. La implementación de un script en Python y la aplicación múltiples hilos nos permitió ejecutar un gran número de pruebas de manera eficiente. Sin embargo, se evidenció la necesidad de mantener un equilibrio entre la cantidad de pruebas y el tiempo disponible, ya que la ejecución de un gran volumen de pruebas aumentó significativamente el tiempo de procesamiento.

En base a los resultados obtenidos de las pruebas, se puede concluir que el algoritmo de Código Hamming tuvo un mejor funcionamiento. Esto debido a que dicho algoritmo es capaz de no solo detectar errores, sino también corregirlos. Asimismo, es importante mencionar que el método Código de Hamming ofrece una ventaja en términos de velocidad y uso eficiente de recursos en comparación con CRC32. A diferencia de CRC32, este no agrega una gran cantidad de bits adicionales a la trama original. Esto resulta en que dicho algoritmo sea igual de robusto pero muchísimo más eficiente.

Cabe destacar que el método de Código de Hamming tuvo un mejor funcionamiento debido a que este algoritmo era menos propenso a ser afectado por ruido (sin importar la probabilidad de ruido) a comparación de CRC32. Como se puede observar en la *Figura 12*, con largos de cadena de 50 caracteres y probabilidad de ruido de 0.001, el Código de Hamming tuvo aproximadamente 350 mensajes que correctamente identificó como libres de error. Esto a diferencia del método CRC32, que con los mismos parámetros únicamente identificó a 300 mensajes –aproximadamente– como libres de error.

Se puede concluir que esto es el resultado del *overhead* introducido por cada método. Por ejemplo, CRC32 agrega 32 bits adicionales a la trama binaria original. Sin embargo, Código de Hamming únicamente agrega una pequeña cantidad de bits en posiciones específicas. Como resultado, incluso con probabilidades de ruido bajas, las tramas procesadas por CRC32 son más propensas a ser alteradas debido a que hay una mayor cantidad de bits que pueden ser modificados. Por tal razón, aunque la probabilidad de ruido sea extremadamente baja, un canal no confiable nunca podrá garantizar la integridad de un mensaje.

Por otro lado, en el caso de mayores tasas de error, se pudo observar nuevamente que el Código de Hamming tenía más flexibilidad. Esto debido a que dicho algoritmo –nuevamente– introducía menos bits adicionales que podían ser afectados por ruido. Como resultado, este algoritmo era capaz de detectar dichos errores y corregirlos fácilmente. Incluso, debido a su reducido *overhead*, en varias ocasiones la tasa alta de ruido no afectó la trama binaria procesada. Como se puede observar en la *Figura 7* (cadenas de longitud 15, probabilidad de ruido de 0.1), el Código Hamming logró corregir las cadenas con error y correctamente identificó varias cadenas que no presentaban errores. Esto a diferencia de la *Figura 1*, que

con los mismos parámetros el método CRC32 no obtuvo ninguna cadena libre de errores.

Finalmente, es importante mencionar que estos algoritmos tienen objetivos diferentes; uno de ellos busca únicamente detectar errores, mientras que el otro busca poder corregir errores detectados. En este caso, es mejor utilizar el algoritmo CRC32 (detección de errores) cuando la prioridad es identificar si existe algún error en la trama transmitida, pero no es importante la corrección. Por ejemplo, este podría ser útil cuando no se necesita garantizar la integridad de los datos con absoluta certeza, como en casos donde la trama se puede retransmitir fácilmente si se detecta un error. Por otro lado, el algoritmo Código de Hamming (detección y corrección de errores) es mejor cuando se requiere garantizar la integridad de los datos y no necesariamente es factible retransmitir el mensaje. Asimismo, este sería de mayor utilidad cuando los datos transmitidos son muy críticos y se necesita asegurar que cualquier error sea corregido.

En conclusión, en este laboratorio se pusieron en práctica los conocimientos teóricos adquiridos sobre la arquitectura de capas y los algoritmos de detección y corrección de errores. Asimismo se experimentó con los desafíos que puede llegar a conllevar la comunicación con esta arquitectura y sus posibles efectos.

6. Comentario

Esta práctica ha demostrado ser un valioso refuerzo para nuestros conocimientos teóricos al permitirnos aplicar conceptos y métodos aprendidos en clase en un entorno práctico y tangible. A medida que trabajamos en la implementación de la arquitectura de capas y la ejecución de las pruebas, profundizamos nuestra comprensión de cómo estos conceptos se traducen en acciones concretas en el mundo real de las comunicaciones.

Uno de los aspectos más reveladores de esta experiencia fue la comprensión más profunda de las dificultades y complicaciones que los factores externos, como el ruido en la transmisión, pueden introducir en el proceso de comunicación. Observamos cómo incluso pequeños cambios en la probabilidad de ruido pueden tener un impacto significativo en la integridad de los datos transmitidos, lo que destaca la importancia de los mecanismos de detección y corrección de errores.

La automatización de las pruebas también resultó ser un desafío.. La generación de un script en Python para la ejecución de las pruebas y la implementación de enfoques asincrónicos y de múltiples hilos fue una tarea que demandó un esfuerzo considerable. A pesar de esta complejidad, logramos agilizar el proceso de pruebas y obtuvimos resultados valiosos para nuestro análisis.

Es importante destacar que, a pesar de nuestros esfuerzos por optimizar el proceso de pruebas, la ejecución de seis mil pruebas resultó en un tiempo de ejecución bastante prolongado. Este desafío nos recordó la importancia de considerar tanto la eficiencia en la implementación de las pruebas como el equilibrio entre la cantidad de pruebas y el tiempo disponible.

7. Conclusiones

1. En esta práctica, hemos logrado alcanzar de manera exitosa los objetivos establecidos. Hemos profundizado nuestra comprensión del funcionamiento de un modelo de capas y sus servicios en el contexto de las comunicaciones. Asimismo, hemos abordado y enfrentado de manera directa las dificultades inherentes a la transmisión de información en un entorno real.

2. Se puede concluir que una arquitectura de capas es un enfoque efectivo para estructurar y modularizar las diferentes etapas de la comunicación. Esta organización facilita la comprensión de las responsabilidades de cada capa, así como la integración de servicios y algoritmos en cada etapa del proceso de comunicación.

3. A través de las pruebas realizadas, se pudo determinar que aunque la probabilidad de ruido sea extremadamente baja, un canal no confiable nunca podrá garantizar la integridad de un mensaje. Esto nuevamente resalta la importancia de implementar mecanismos de detección y corrección de errores.

4. Un hallazgo importante que surgió de las pruebas realizadas es la relación entre la longitud de la cadena y la tasa de errores en las tramas. Observamos que a medida que aumenta la longitud de la cadena transmitida, aumenta la probabilidad de que se introduzcan errores en la comunicación (aunque la probabilidad de ruido sea baja). Esto debido a que hay más bits que pueden ser alterados.

5. Si bien ambos algoritmos, CRC-32 y Hamming, se vieron afectados por la variación en la probabilidad de ruido, se puede observar un rendimiento más sólido y adaptable por parte del algoritmo de Hamming. Esto debido a que dicho algoritmo –nuevamente– introducía menos bits adicionales. Como resultado, había menos probabilidad de que alguno de ellos fuera afectado por ruido. Asimismo, este algoritmo era capaz de detectar dichos errores y corregirlos fácilmente.

6. Se puede concluir que es mejor utilizar el algoritmo CRC32 (detección de errores) cuando la prioridad es identificar si existe algún error en la trama transmitida, pero no es importante la corrección. Por otro lado, es mejor utilizar el algoritmo Código de Hamming (detección y corrección de errores) cuando se requiere garantizar la integridad de los datos y se necesita asegurar que cualquier error sea corregido.