Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación

Autores:
Oscar Oswaldo Estrada Morales (20565)
Gabriel Alejandro Vicente Lorenzo (20498)

Laboratorio 2.2 - Esquemas de detección y corrección de errores

Guatemala 11 de agosto de 2023
CC 3067 Redes
Sección 20

Laboratorio 2 Parte 2
Esquema de detección y corrección de errores

Descripción

En esta fase de continuación del proyecto, se ha ampliado y mejorado significativamente el sistema previamente desarrollado, que consistía en los archivos Sender.java y receiver.py para la codificación y decodificación de mensajes binarios mediante los algoritmos Hamming 7-4 y CRC-32. El enfoque principal se ha desplazado hacia la comprensión profunda del modelo de capas y sus servicios, con la implementación clave de sockets para la transmisión de información entre el emisor y el receptor.

El proyecto se ha enriquecido mediante la adición de nuevos servicios en cada capa del modelo OSI. En la capa de Aplicación, se ha habilitado la interacción del usuario con el sistema, permitiendo la solicitud y visualización de mensajes a través de la selección de algoritmos de comprobación de integridad. La capa de Presentación se encarga ahora de codificar y decodificar mensajes en ASCII binario, garantizando que los caracteres sean transmitidos y recibidos de manera precisa. Por su parte, la capa de Enlace se ha fortalecido con la capacidad de calcular, verificar y, en algunos casos, corregir la integridad de los mensajes transmitidos, haciendo uso de los algoritmos Hamming 7-4 y CRC-32 previamente implementados. La capa de Ruido simula posibles interferencias y ruido en la transmisión (de forma modificable), mientras que la capa de Transmisión gestiona la transferencia de tramas de información a través de sockets.

En consonancia con estos avances, se han modificado los archivos originales Sender.java y receiver.py para incorporar los nuevos servicios y aprovechar los algoritmos de detección y corrección existentes. Además, se ha agregado un apartado de pruebas exhaustivas para verificar el funcionamiento de los algoritmos implementados y del sistema en su conjunto. Se han ejecutado múltiples pruebas de envío y recepción, automatizando el proceso y aplicando análisis estadísticos para evaluar el rendimiento y la eficacia de los algoritmos bajo diversas condiciones. Como resultado, se espera obtener una comprensión más profunda de los conceptos de comunicación en capas, la transmisión de datos mediante sockets y la robustez de los esquemas de detección y corrección en un entorno simulado.

Metodología

Diseño y Modificación del Código: Se tomaron los códigos previamente desarrollados (Sender.java y Receiver.py) como punto de partida y se realizaron modificaciones significativas para integrar los nuevos servicios y la comunicación a través de sockets. Se implementaron funciones para codificar y decodificar mensajes en ASCII binario, así como para aplicar algoritmos de detección y corrección, como el CRC-32 y el Hamming 7-4.

Generación de Datos de Prueba y Simulación de Ruido: Se incorporó un generador de palabras aleatorias en Java para simular mensajes de entrada. Además, se introdujo un elemento de realismo en las pruebas al simular ruido en la transmisión. Se estableció que con cada bit enviado, había una probabilidad de 1/100 de que el bit cambiará debido al ruido.

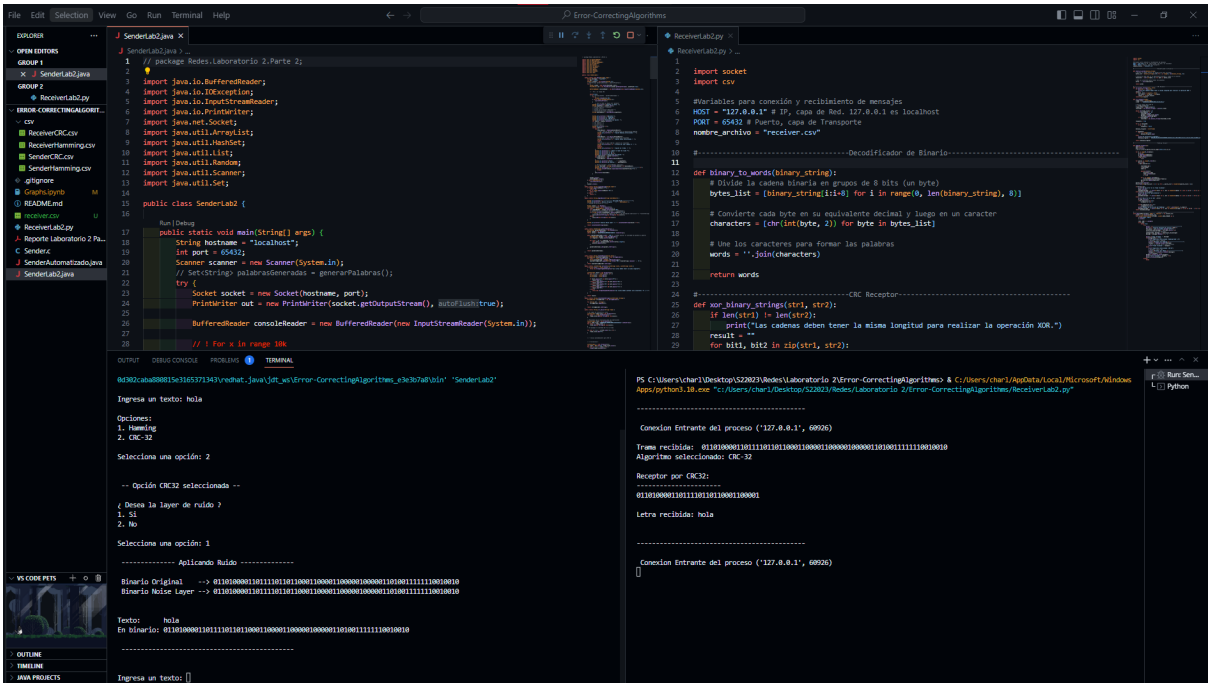
Integración de Sockets: Se implementaron conexiones de sockets tanto en el emisor como en el receptor. Esto permitió la transferencia de tramas binarias entre las dos entidades a través de una red simulada, lo que constituye una representación de la capa de transmisión.

Pruebas y Análisis: Se realizaron pruebas masivas automatizadas, utilizando miles de mensajes generados aleatoriamente, para evaluar la efectividad y confiabilidad de los algoritmos de detección y corrección. Además, se consideró el impacto del ruido simulado en la integridad de los datos. Se recopilaron datos estadísticos para analizar el rendimiento y la eficiencia del sistema en términos de integridad y transmisión exitosa.

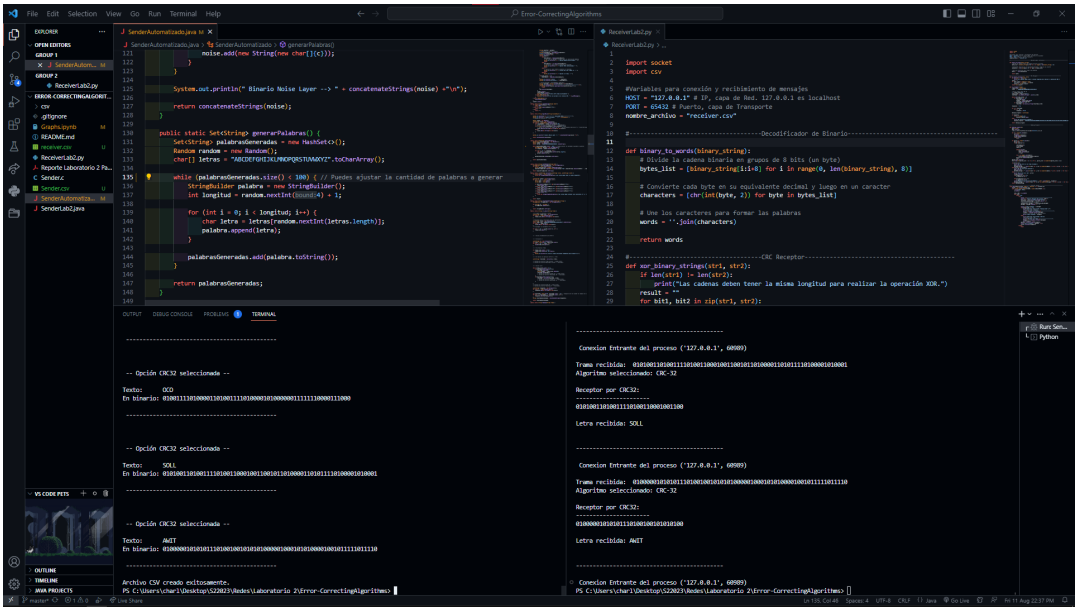
Verificación de Resultados: Los resultados de las pruebas se compararon con las expectativas teóricas, teniendo en cuenta la posibilidad de cambio de bits debido al ruido simulado. Se verificó que los algoritmos de detección y corrección funcionaran correctamente, incluso en presencia de ruido, y su habilidad para corregir los errores detectados.

Resultados

Teniendo en cuenta que la transmisión de mensajes (de manera manual) tiene la siguiente salida en terminal:

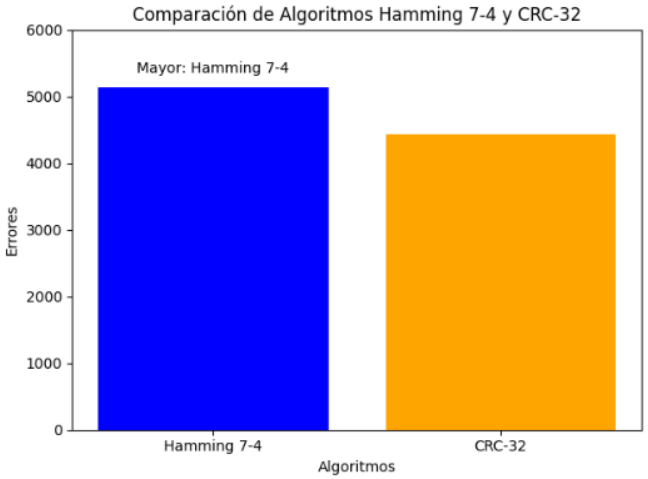


Las pruebas automatizadas constan de las mismas partes pero simplemente con la selección previa antes de mandar mensajes aleatorios repetidas veces (para este caso 10k). Terminando con la transmisión de palabras y las terminales de la siguiente forma:



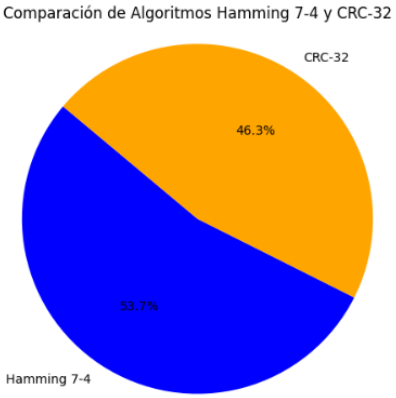
Se puede apreciar que independientemente de la manera de transmisión de mensajes que se ejecute, se deja un registro de las palabras enviadas y recibidas, esto nos permite hacer un análisis más profundo del funcionamiento del laboratorio.

Uno de los resultados más puntuales que se pudo obtener es la cantidad de errores que marcaron los algoritmos cuando en la prueba automatizada se seleccionaba que si tuviera ruido (teniendo en cuenta que el ruido aplica una probabilidad de 0.01% de cambiar el bit). En esta gráfica de barras se puede apreciar mejor los resultados.



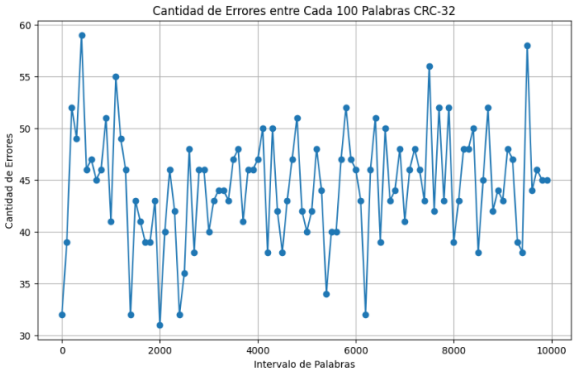
Gráfica #1 **Cantidad de errores por algoritmo**

Si comparamos porcentajes respecto a cada prueba, directamente podemos observar que el algoritmo de CRC-32 es el que detecta “menos errores” respecto a las 10k transmisión de mensajes que realizó, obteniendo un 7.4% menos a comparación de Hamming.

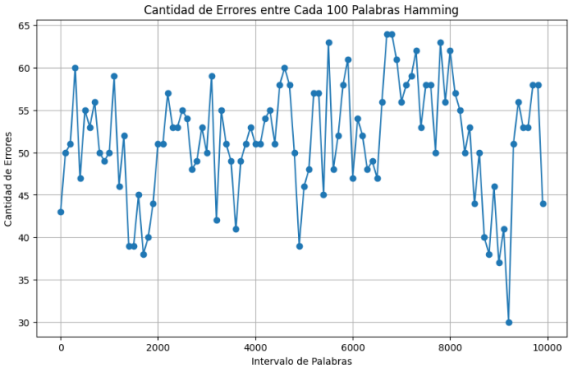


Gráfica #2 **Porcentaje de errores por algoritmo**

Sin embargo, al momento de ver la cantidad de errores en cada 100 palabras (es decir, cuantos errores aparecen en intervalos de 100 mensajes transmitidos aplicando ruido) con cada algoritmo, se puede apreciar que en realidad no existe una tendencia de la cantidad de errores que aparecen, lo cual es fiel a la capa de ruido programada ya que esta es una probabilidad de 0.01% sin tener condicionales, explicando el comportamiento errático de la aparición de errores entre los distintos algoritmos.

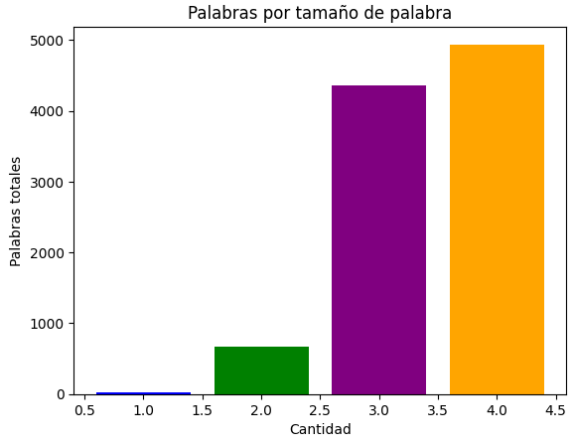


Gráfica #3 **Errores en cada 100 mensajes CRC-32**

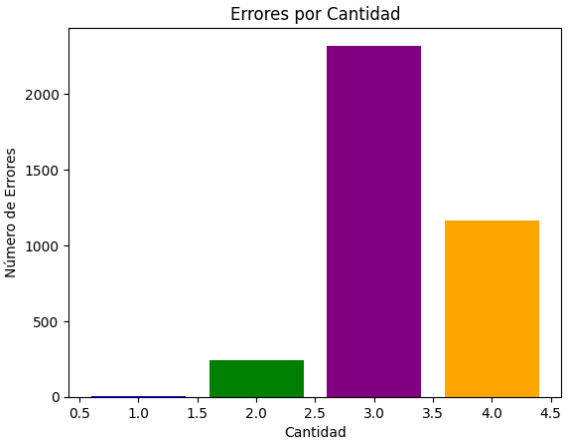


Gráfica #4 **Errores en cada 100 mensajes Hamming**

Cómo se puede apreciar, la manera en que se presentan errores no va ligada a la cantidad de mensajes que ya se han transmitido o a alguna otra dependencia del contexto en el que se ejecuta el sender y receiver, solo a la capa de ruido con probabilidad de 0.01%.



Gráfica #5 **Cantidad de palabras según longitud**



Gráfica #6 **Cantidad de errores según longitud**

De igual manera, tras analizar el algoritmo de Hamming, se pudo encontrar que el error va de la mano según el largo de la palabra ingresada. Como se menciona en la teoría de Hamming Algorithm, según más largo es el tamaño de bits estudiado es más propenso a que este falle, por lo que el comportamiento es adecuado. Si se estudiara el comportamiento de ambos algoritmos con un largo de palabra de 2, es muy probable que los papeles se intercambien y Hamming presente una mejora bastante significativa.

Discusión

En las gráficas #1 y #2 podemos observar que el algoritmo de Hamming es el que presenta más errores, pero esto no se debe específicamente a la capa de ruido o la implementación del algoritmo de Hamming, si no a las implicaciones que tiene el algoritmo en sí. Ya que el algoritmo de Hamming tiene la característica de sobrecorregir o fallar al momento de analizar extensiones de bit considerablemente extendidas (Ramón Invarato, 2016). Es por ello que en estas gráficas de comparación de concurrencia de errores entre algoritmos al transmitir los mensajes con ruido Hamming presenta una mayor cantidad de errores, debido a sus debilidades al analizar tramas extensas.

En lo que respecta a la capa de ruido en sí (*Noise Layer*), se podría argumentar que una implementación errónea de la misma daría paso a favorecer a alguno de los algoritmos, es por eso que se procedió a analizar las gráficas de recurrencia de errores entre cada 100 mensajes transmitidos para cada algoritmo. En estas gráficas se puede apreciar el comportamiento errático de la aparición de errores entre estos intervalos. Y esto se debe a que la capa de ruido tiene una implementación imparcial (La cantidad de mensajes transmitidos o el mensaje a transmitir en sí no afecta a su funcionamiento). La capa de ruido simplemente funciona como una probabilidad aplicada a cada bit de los mensajes, que cambiará el valor cuando esta se presente. Al no tomar en cuenta nada más, la aparición de errores es totalmente aleatoria (0.01% de cambiar en cada bit) lo cual se puede reflejar en el comportamiento errático de la aparición de errores en ambos algoritmos.

Comentario grupal sobre el tema

Tras realizar los procesos de transmisión entre los programas realizados, llegamos a la conclusión que es bastante importante el algoritmo utilizado para cada caso. En este caso, encontramos que el algoritmo de Hamming presentó un comportamiento extraño, pero después de estudiarlo comprobamos que este es bastante dependiente del largo de su trama analizada, por lo que vemos que cada algoritmo tiene un uso específico en donde puede resaltar más su eficacia. Además, la forma de detección y corrección de los algoritmos con este proyecto nos da una idea de cómo es que funciona en la actualidad la transmisión de paquetes para cada situación. De igual manera, aprendimos la forma de una estructura de capas, por lo que es bastante enriquecedor el hecho de poder haberlo puesto en práctica.

Conclusiones

- Hamming posee una mayor probabilidad de presentar errores cuando existe ruido ya que el ruido no solo se limita a 1 bit, al ser aplicado a todos los bits esto hace que la aparición de dos o más bits modificados sea alta, causando la falla de Hamming dado que la aparición de ruidos dobles o tramas extendidas no son uno de sus fuertes.
- CRC-32 posee una alta afinidad al momento de cifrar y descifrar los mensajes transmitidos independientemente de lo extendida que sea la trama, pero si el ruido está presente, la probabilidad de que la transmisión falle se eleva debido a la gran cantidad de información que agrega CRC-32 para poder efectuar su cifrado.
- Según los resultados obtenidos de este laboratorio, Hamming es mejor al momento de corregir tramas más cortas, debido a que por su forma de corrección respecto a los bits de paridad, este presenta errores más difíciles de trabajar después de cierta cantidad ingresada.

Referencias

Ramón Invarato. (2016, October 5). *Código de Hamming: Detección y Corrección de errores - Jarroba*. Jarroba. https://jarroba.com/codigo-de-hamming-deteccion-y-correccion-de-errores/

CRC32: Verificación de Redundancia Cíclica. (2019). Jc-Mouse.net. https://www.jc-mouse.net/java/crc32-verificacion-de-redundancia-ciclica