UNIVERSIDAD DEL VALLE DE GUATEMALA
CC3067 - Redes
Sección 20
Ing. Jorge Yass

Esquemas de detección y corrección de errores

Javier Alejandro Ramírez Heredia - 21600
Mario Andres Cristales Cardona - 21631

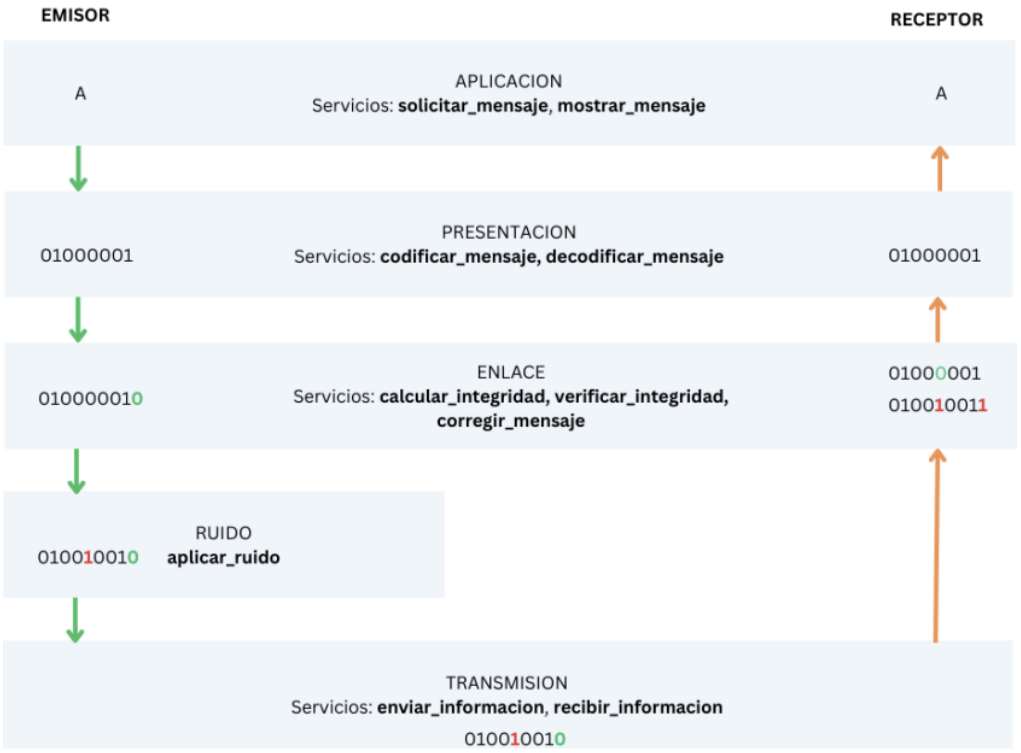
GUATEMALA, 01 de agosto de 2024

Descripción de la Práctica

En esta práctica, se implementaron dos algoritmos, uno para la transmisión y recepción de mensajes, en base a una arquitectura de capas con distintos servicios como se puede ver en la imagen de abajo. Los emisores se programaron en Python y los receptores en JavaScript. Los algoritmos utilizados fueron:

CRC-32: Este algoritmo de detección de errores utiliza la división polinomial de una secuencia de bits. En este caso, se empleó un polinomio de grado 32. El emisor divide el mensaje original por este polinomio y envía el resultado al receptor. El receptor realiza la misma operación y, según el residuo obtenido, detecta errores en el mensaje recibido.

Hamming (7-4): Este algoritmo de corrección de errores utiliza bits de paridad y potencias de 2. En la variante 7-4, el emisor empaqueta los mensajes en grupos de 4 bits, aplica el algoritmo de codificación y genera tramas de 7 bits. El receptor descompone la trama en sub-tramas de 7 bits para detectar y corregir errores.



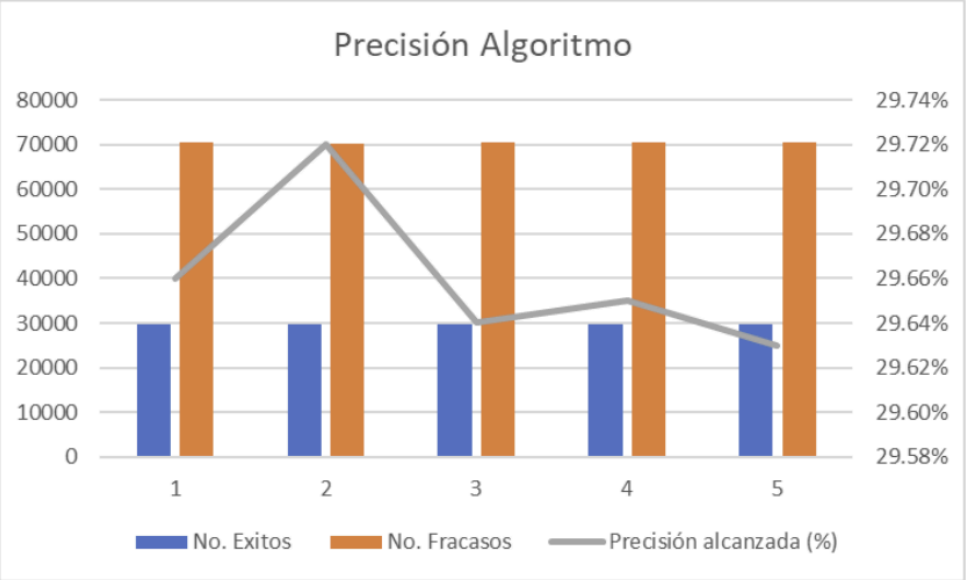
En este caso, el emisor recibe mensajes en caracteres comunes en la capa de aplicación, los transforma en cadenas de bits para ASCII extendido en la capa de presentación y aplica uno de los algoritmos mencionados en la capa de enlace. Luego, se agrega ruido al mensaje con una probabilidad de 0.01% por bit antes de enviarlo en la capa de transmisión, utilizando sockets en ambos lenguajes. Esta metodología permitió realizar pruebas con 100,000 palabras para comprobar la efectividad de los algoritmos y simular un entorno real.

Resultados

- **CRC-32**

Prueba	No. Exitos	No. Fracasos	Precisión alcanzada (%)
1	29,660	70,340	29.66%
2	29,724	70,276	29.72%
3	29,639	70,361	29.64%
4	29,649	70,351	29.65%
5	29,629	70,371	29.63%

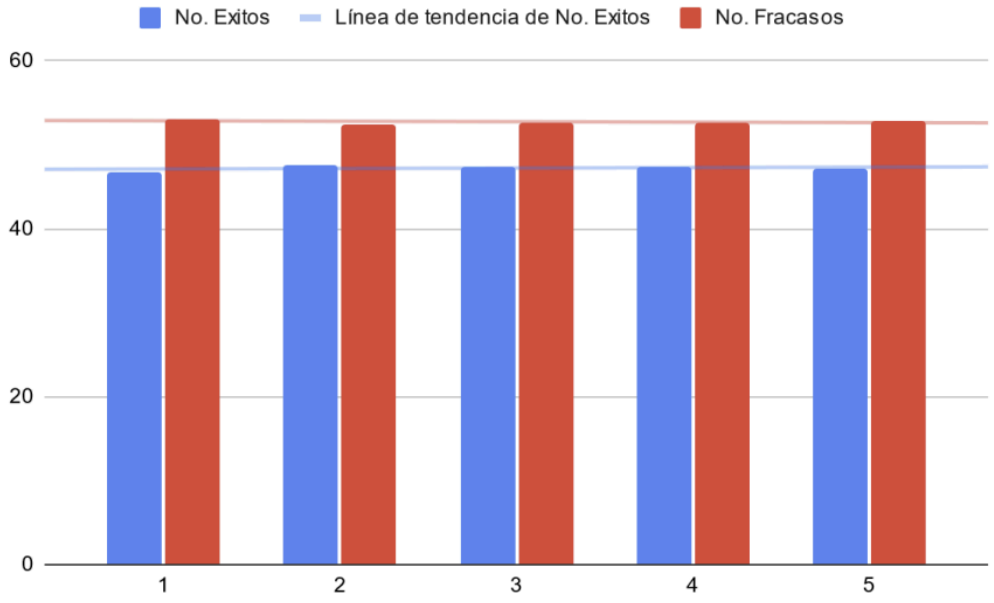
Cada prueba se realizó enviando 100,000 tramas con ruido. El promedio de éxito fue del 29.66%.



- **Hamming**

Prueba	No. Exitos	No. Fracasos	Precisión alcanzada (%)
1	46,848	53,152	46.85%
2	47,492	52,508	47.79%
3	47,353	52,647	47.35 %
4	47,285	52,715	47.28%
5	47,219	52,781	47.22%

Cada prueba se realizó enviando 100,000 tramas con ruido, obteniendo un promedio de éxito del 47.298%.



Discusión

En la sección de resultados, se observa que el algoritmo de Hamming tuvo un rendimiento superior con un promedio de 47.3%, mientras que el algoritmo CRC-32 alcanzó un 29.7% de éxito. Esta diferencia puede atribuirse a varios factores. El algoritmo de Hamming, al verificar cada bit de forma individual en secuencias de datos más cortas, puede detectar y corregir errores de un solo bit, lo que resulta en un análisis más detallado de bit a bit. Además, Hamming es más eficiente en términos de cálculo y almacenamiento para datos más cortos debido a la adición de bits de paridad, lo que facilita la detección de errores cuando se introduce ruido y se revisa bit por bit.

Por otro lado, el algoritmo CRC-32 es más flexible al manejar mayores tasas de error, ya que tiene una ventaja en términos de eficiencia con secuencias de datos más largas. Puede detectar una amplia variedad de patrones de errores, incluyendo aquellos en diferentes posiciones.

Para determinar qué tipo de algoritmo es más adecuado para la detección o corrección de errores, es crucial considerar la importancia de los datos y los recursos del sistema. Además de estos factores, también es importante evaluar la tolerancia a errores. Si los datos transmitidos o almacenados no son críticos y pueden retransmitirse fácilmente, un algoritmo de detección de errores es más adecuado. En contextos donde los errores son raros, los algoritmos de detección son preferibles. Sin embargo, si se espera que los errores ocurran con cierta frecuencia, un algoritmo de corrección de errores sería más apropiado.

Comentario grupal sobre el tema

Al principio, se esperaba que la automatización de las 100,000 pruebas fuera un proceso muy lento. Sin embargo, al implementarlo y ejecutarlo, descubrimos que la ejecución no toma más de un minuto para las 100,000 pruebas. Esto nos permitió realizar y optimizar todos los requerimientos de dichas pruebas de manera eficiente.

Conclusiones

- La arquitectura de capas, aunque compleja, facilita un mayor orden y encapsulamiento en la implementación de pipelines de datos.

- Simular un entorno real mediante la introducción de ruido en la transmisión de datos es crucial para evaluar la efectividad de los algoritmos en condiciones de producción.

- El algoritmo de Hamming demostró una mayor precisión en comparación con CRC-32 debido a su enfoque en la detección de errores bit a bit.

Referencias

- CRC-32. (s. f.). En Fuchsia. Recuperado de https://fuchsia.googlesource.com/third_party/wuffs/+/HEAD/std/crc32/README.md
- Invarato, R. (2017, 12 agosto). Código de Hamming: detección y corrección de errores - Jarroba. Jarroba. Recuperado de https://jarroba.com/codigo-de-hamming-deteccion-y-correccion-de-errores