Universidad del valle de Guatemala
Redes

Diego Córdova: 20212
Andrea Lam: 20102

Laboratorio #2.2
Implementación de arquitectura de capas

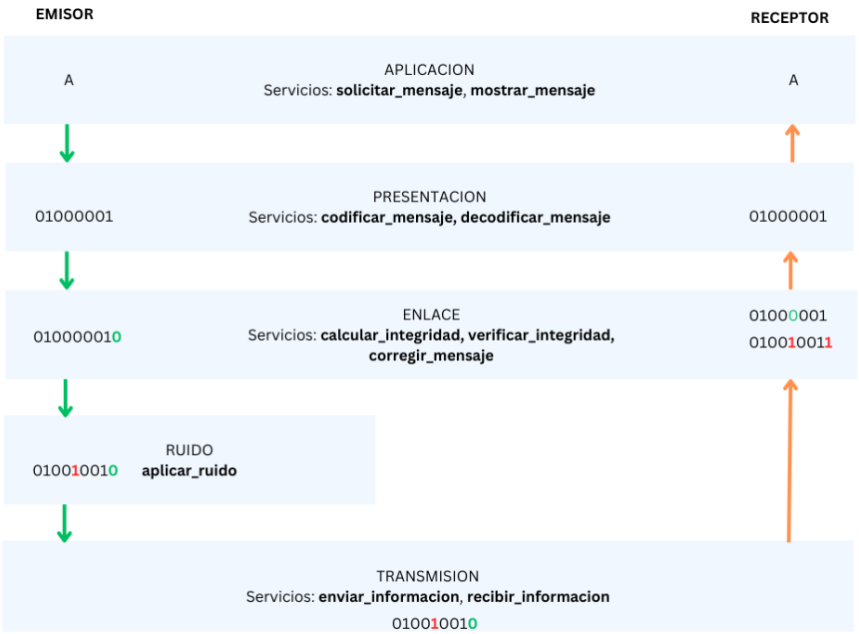
Descripción de la práctica

Esta práctica consistió en la implementación de 2 algoritmos, uno de corrección de errores y otro de detección de errores. Cabe destacar que se implementó un emisor y un receptor para cada algoritmo. En este caso los emisores fueron implementados en el lenguaje de programación *python* y los receptores en el lenguaje de programación *javascript*. A continuación se describen los algoritmos en cuestión:

- CRC-32
  Este algoritmo de detección de errores se basa en la división polinomial de una secuencia de bits. El largo de este polinomio depende de la implementación a realizar, en este caso se utilizó un algoritmo de grado 32 dado que se implementó el algoritmo *CRC-32*. En términos prácticos, el emisor divide el mensaje original por el polinomio de grado 32 y lo manda al receptor, este realiza el mismo procedimiento y dependiendo del residuo resultante realiza la detección de errores en el envío del mensaje.

- Hamming
  Este algoritmo de corrección de errores se basa en el concepto de los bits de paridad combinado con potencias de 2 para detectar y corregir errores en mensajes enviados. En este caso específico se implementó la variante 7-4, en donde el emisor empaqueta los mensajes en grupos de 4 para luego aplicar el algoritmo de codificación a las sub-tramas individuales y luego juntarlas para mandar el mensaje. Por la naturaleza del algoritmo las tramas resultantes de la codificación tienen un largo de 7 bits. Por tanto el emisor recibe la trama total y la procesa en sub-tramas de 7 bits para lograr la detección y corrección de errores.

Luego de esto se implementó una arquitectura de capas con la siguiente estructura.



En este caso el emisor recibe mensajes en caracteres comunes en la capa de *aplicación,* luego las transforma en cadenas de bits para ASCII extendido en la capa de *presentación* para luego aplicar alguno de los algoritmos detallados anteriormente dependiendo el caso en la capa de *enlace.* Por último se agrega ruido al mensaje con una probabilidad de 0.01% por cada bit para luego mandar la información en la capa de *transmisión.* En este caso la capa de transmisión se implementó utilizando sockets en los dos lenguajes utilizados.

Utilizando esta metodología se logró realizar pruebas con 100,000 palabras mensajes para comprobar la efectividad de los algoritmos utilizados y de está manera simular un entorno real.

Resultados:

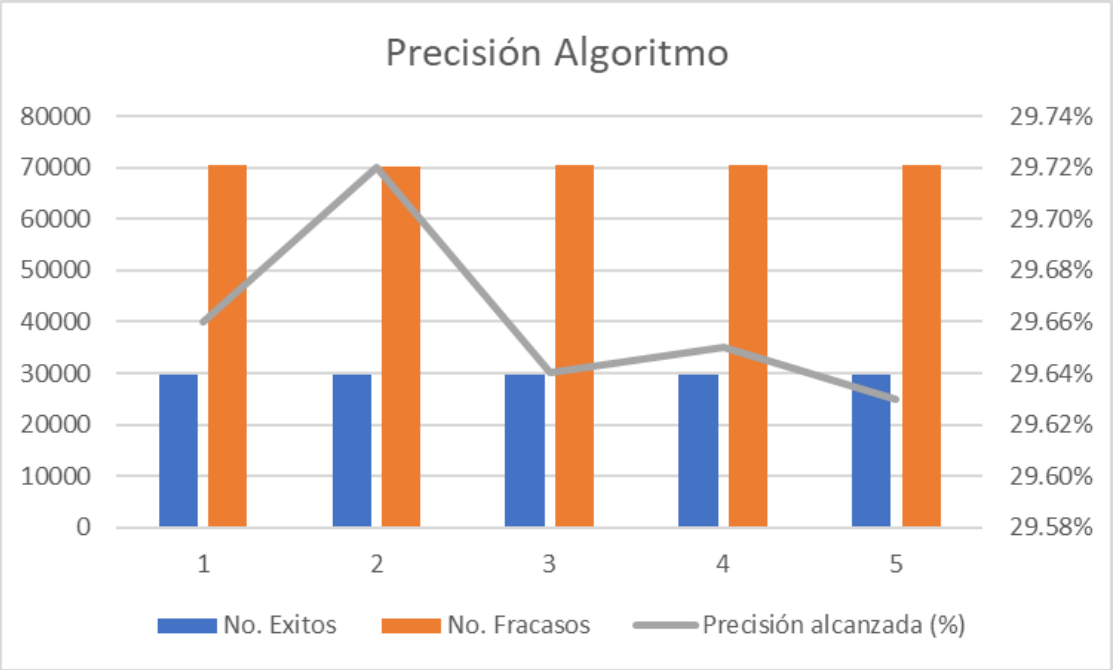
Pruebas Realizadas:

CRC-32

Prueba	No. Exitos	No. Fracasos	Precisión alcanzada (%)
1	29660	70340	29.66%
2	29724	70276	29.72%
3	29639	70361	29.64%
4	29649	70351	29.65%
5	29629	70371	29.63%

Cada una de las pruebas se realizó con 100,000 tramas enviadas con ruido.

Promedio de éxito: 29.66%

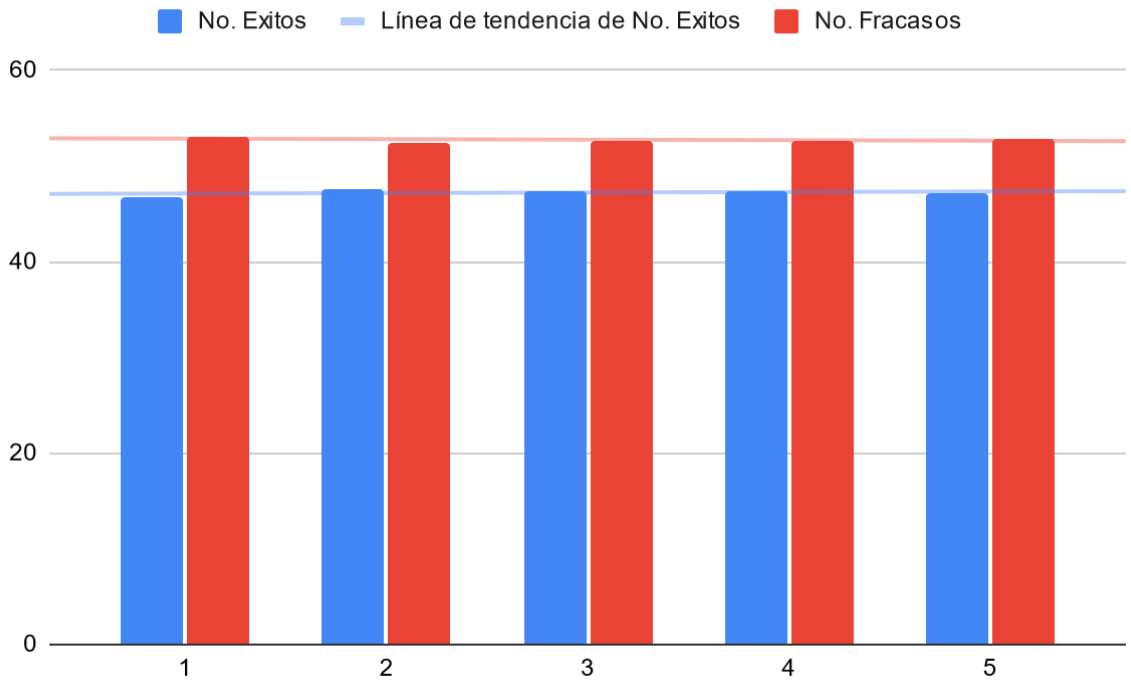


Hamming

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 una de las pruebas se realizó con 100,000 tramas enviadas con ruido.

Promedio de éxito: 47.298%



Discusión

Como se puede observar en la sección de resultados, el algoritmo que obtuvo un mejor rendimiento fue el de Hamming, obteniendo un promedio de 47.3%. Mientras que el algoritmo de CRC-32 obtuvo un porcentaje de éxito del 29.7%. Esto se puede dar por diferentes factores; Una de las posibles razones es que el algoritmo de Hamming al verificar por bit individual en secuencias de datos más cortas, puede detectar y corregir errores de un solo bit, pero da un mejor análisis de bit a bit. Asimismo el algoritmo de Hamming es más eficiente en términos de cálculos y almacenamiento para datos más cortos, ya que de por sí agrega bits de paridad adicionales, al momento de agregar ruido y tener que revisar bit por bit, ayuda a la detección de errores.

Generalmente el algoritmo de CRC-32 es más flexible al momento de aceptar mayores tasas de error. Esto se debe a que al momento de poder manejar secuencias de datos más largas, el CRC-32 tiene más ventaja en términos de eficiencia, este puede detectar una amplia variedad de patrones de errores, contando con errores en diferentes posiciones.

Una parte muy importante para determinar qué tipo de algoritmo es mejor para la detección de errores en lugar de uno de corrección de errores es la importancia de los datos a utilizar y los recursos disponibles del sistema que se está utilizando. Además de esos factores principales, existen otros factores como la tolerancia a errores, ya que si los datos transmitidos o almacenados no son críticos y pueden ser retransmitidos con facilidad, un algoritmo de detección de errores es mejor. También para el caso, que si se habla en el ámbito de la naturaleza de errores, donde si los errores son raros o es poco probable que ocurran en un contexto de la aplicación, los algoritmos de detección son mejores. Normalmente es mejor un algoritmo de corrección de errores si se espera que ocurran errores con cierta regularidad.

Comentario grupal sobre el tema (errores)

En un principio se esperaba que la automatización de las 100,000 pruebas fuera un proceso sumamente tardado. Sin embargo, al momento de implementarlo y ejecutarlo nos dimos cuenta que la ejecución no supera el minuto para las 100,000 pruebas por lo que fue posible realizar y optimizar todos los requerimientos de dichas pruebas.

Conclusiones

- Una arquitectura de capas aunque brinda complejidad en cuanto a implementación, tiene la ventaja de un mayor orden y encapsulamiento al momento de implementar pipelines complejos para el empaquetado, envío y desempaquetado de datos.

- Por medio de la introducción de ruido en la transmisión de datos se logró simular un entorno real y así comprobar la efectividad de los algoritmos en un entorno de producción.

- El algoritmo de hamming tuvo una precisión máxima de 47.79% y una precisión mínima de 46.85%.

- Al algoritmo crc-32 tuvo una precisión de tuvo una precisión máxima de 29.72% y una precisión mínima de 29.63%

- El algoritmo de Hamming logró una mayor precisión por su detección de errores bit a bit.

Citas y Referencias

CRC-32. (s. f.). 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. https://jarroba.com/codigo-de-hamming-deteccion-y-correccion-de-errores/