UNIVERSIDAD DEL VALLE DE GUATEMALA
1IM4016 - Taller de Máquinas y Herramientas
Sección 10
Ing. Raúl Loarca

Laboratorio 2 - Segunda parte

Diego Valdez 21328

Sebastian Estrada 21405

GUATEMALA, 1 de agosto de 2024

Descripción de la práctica y metodología

Esta actividad se enfoca en aplicar esquemas de detección y corrección de errores durante la transmisión de datos a través de una arquitectura por capas. Se desarrolló una aplicación cliente-servidor utilizando sockets para simular el envío de mensajes a través de un canal no confiable. Se implementaron dos algoritmos principales: el código Hamming para la corrección de errores y CRC-32 para la detección de errores. La arquitectura incluye capas de aplicación, presentación, enlace y una capa de ruido para simular interferencias.

Resultados

Hamming

```
Ingrese una palabra:  hola
Datos con ruido: 10011000000,10101010000,10101011111,00101011100,10111001001
Datos enviados
Seleccione el tipo de codificación:
1. Corrección de errores con código de Hamming
2. Detección de errores con CRC-32
3. Salir
Opción:
```

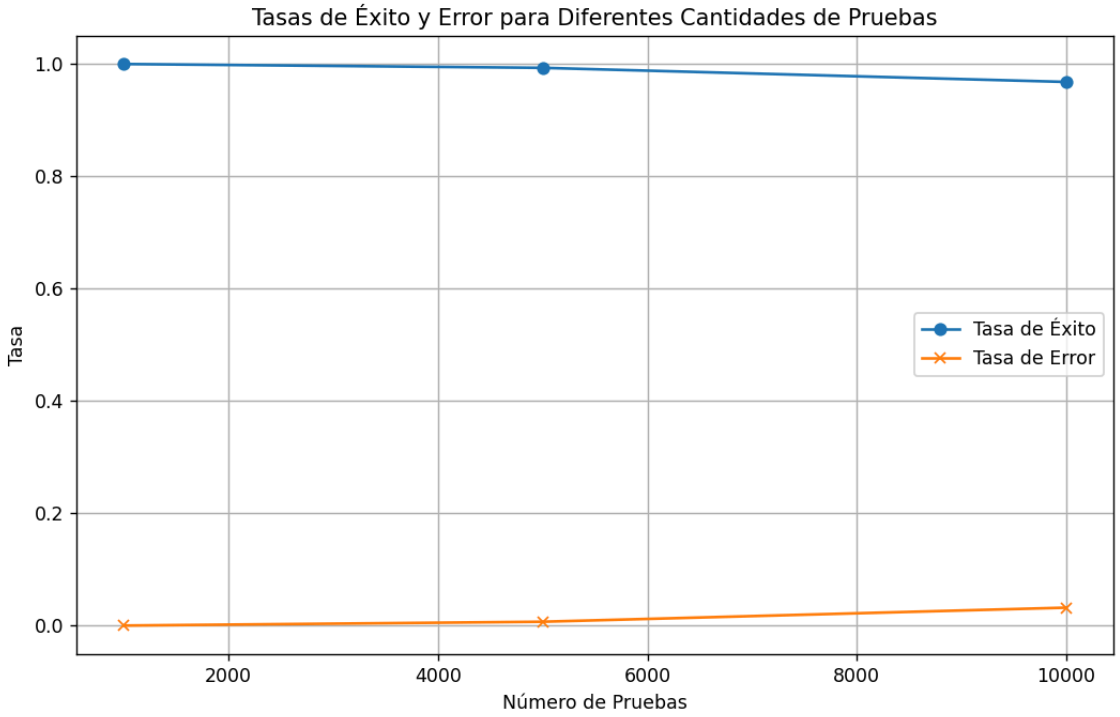
```
Datos recibidos: 10011000000,10101010000,10101011111,00101011100,10111001001
Bloque inválido: 0 10011000000
Bloque sin errores detectados: 10101010000
Bloque sin errores detectados: 10101011111
Bloque sin errores detectados: 00101011100
Bloque sin errores detectados: 10111001001
Mensaje decodificado: hola
```

CRC-32

```
Ingrese una palabra: hola
Mensaje: hola CRC: 01101111101000001111100110001000
Datos con ruido: 01101000011011110110110001100001 01101111101000001111100110001000
Datos enviados
```

```
Conectado por ('127.0.0.1', 57109)
Datos recibidos: 01101000011011110110110001100001
¿El mensaje es válido?: Sí
Mensaje recibido: 1 01101000011011110110110001100001 01101111101000001111100110001000
```

Pruebas



Discusión

Durante el laboratorio se probaron dos tipos de algoritmos distintos, uno es de detección que es el CRC-32 mientras que el algoritmo de corrección es algoritmo de Hamming. Durante el desarrollo de dichos algoritmos se encontraron distintos retos como el de hacer la paridad correspondiente, durante este laboratorio se agregó la funcionalidad de comunicación entre emisor y receptor mediante sockets. Tras que la comunicación funcionará correctamente se realizaron 3 pruebas, una con mil solicitudes, otra con cinco mil solicitudes y por último con diez mil solicitudes. El algoritmo que tuvo mejores respuestas fue el de Hamming esto se debe a que las cadenas binarias de una palabra son más pequeñas en Hamming haciendo que los errores sean más fáciles de detectar mientras que CRC al ser más larga la cadena la detección de errores es más complicada.

El algoritmo más adecuado para aceptar mayores tasas de errores es el de corrección de errores, como el de Hamming. Esto se debe a que, además de detectar la presencia de errores, también puede corregirlos, permitiendo manejar una mayor cantidad de errores sin necesidad de retransmitir la información. En cambio, los algoritmos de detección de errores, como el CRC-32, solo pueden identificar que hay un error, pero no corregirlo, lo que los hace menos efectivos cuando la tasa de errores es alta.

Es mejor utilizar un algoritmo de detección de errores en lugar de uno de corrección de errores cuando la probabilidad de errores es baja y se busca una mayor eficiencia en términos de procesamiento y ancho de banda. Los algoritmos de detección, como el CRC-32, suelen ser más simples y rápidos, consumiendo menos recursos. En situaciones donde los errores son raros, la capacidad de simplemente detectar y retransmitir puede ser más eficiente que la necesidad de corregir errores, lo que reduce la sobrecarga de procesamiento y comunicación

Conclusiones

- CRC es una herramienta eficiente que permite detectar errores debido a su alta fiabilidad a la hora de detectar errores, su implementación puede llegar a ser confusa por lo que puede llegar a ser una opción adecuada a implementar a la hora de enviar diversos datos. A pesar que no corrige los errores puede llegar a permitir una gran confianza en una red.
- El algoritmo de Hamming demostró ser más efectivo en pruebas con mil, cinco mil y diez mil solicitudes, ya que las cadenas binarias más cortas permiten una detección y corrección de errores más sencilla.
- El algoritmo de Hamming es ideal para situaciones con altas tasas de errores, ya que puede corregir errores sin necesidad de retransmitir la información, lo que hace el proceso mucho más eficiente.
- Elegir entre un algoritmo de detección o uno de corrección de errores depende de la tasa de errores esperada y de los recursos disponibles. La detección es más eficiente en entornos con errores esporádicos, mientras que la corrección resulta esencial en entornos con altas tasas de errores.

Referencias

https://github.com/Teviets/lab1-redes
(En la branch Diego-Dev se encuentran las pruebas)