Guatemala 01 de agosto de 2024
Sebastian Juarez - 21471
Sebastian Solorzano - 21826

Redes

Laboratorio No. 2 - Esquemas de detección y corrección

Descripción de la práctica

En este laboratorio, se implementó un sistema de transmisión de mensajes utilizando el algoritmo CRC32 para la detección de errores y el algoritmo Hamming para la corrección de errores. El objetivo fue crear un entorno donde se pudiera enviar, recibir y verificar la integridad de los mensajes mediante las aplicaciones de los algoritmos, y realizar pruebas para evaluar el rendimiento bajo diferentes condiciones.

Emisor: Este módulo se encarga de recibir un mensaje de entrada, convertirlo a su representación binaria, calcular el CRC32 o Hamming para el mensaje y simular la introducción de ruido en el mensaje mediante una probabilidad configurable de error. Luego, el mensaje ruidoso es enviado al receptor.

Receptor: Este módulo recibe el mensaje ruidoso del emisor, determina si el mensaje es correcto o contiene errores. El resultado de esta verificación es enviado de vuelta al proceso principal. Con el algoritmo de Hamming, si el mensaje contiene errores, los corrige y después envía el mensaje al proceso principal.

Proceso Principal: Este módulo coordina la comunicación entre el emisor y el receptor. Solicita el mensaje al usuario, envía el mensaje al emisor, recibe el resultado del receptor y muestra el resultado al usuario.

Automatización de Pruebas: Script para realizar pruebas automatizadas del sistema, variando el tamaño de los mensajes y la cantidad de mensajes enviados. Este script genera mensajes aleatorios, los envía a través del sistema y registra los resultados en un archivo JSON.

Análisis Estadístico y Gráficas: Este script analiza los resultados almacenados de las pruebas, calculando la tasa de éxito y errores, y generando gráficas que muestran la distribución de longitudes de los mensajes y la frecuencia de resultados correctos e incorrectos.

Arquitectura de capas

- Algoritmo de Corrección

```
Ingrese el mensaje a enviar: HOLA
Ingrese el algoritmo a utilizar (1 - Hamming, 2 - CRC32): 1
Resultado recibido: No se detectaron errores. H [1001000] No se detectaron errores. O [1001111] Error corregido en la posicion: 10. L [1001100] Error
 corregido en la posicion: 7. A [1000001]
```

```
Ingrese el mensaje a enviar: ADIOS
Ingrese el algoritmo a utilizar (1 - Hamming, 2 - CRC32): 1
Resultado recibido: Error corregido en la posicion: 6. A [1000001] Error corregido en la posicion: 2. D [1000100] No se detectaron errores. I [100100
 I [1001001] No se detectaron errores. O [1001111] Error corregido en la posicion: 7. S [1010011]
¿Desea enviar otro mensaje? (s/n): s
Ingrese el mensaje a enviar: Prueba
Ingrese el algoritmo a utilizar (1 - Hamming, 2 - CRC32): 1
Resultado recibido: No se detectaron errores. P [1010000] No se detectaron errores. r [1110010] No se detectaron errores. u [1110101] Error corregido
 en la posicion: 6. e [1100101] Error corregido en la posicion: 9. b [1100010] Error corregido en la posicion: 4. a [1100001]
```

Emisor:

```
Emisor Hamming escuchando en localhost:12345
Conexión aceptada desde ('127.0.0.1', 61206)
Mensaje recibido: HOLA
Conexión aceptada desde ('127.0.0.1', 61218)
Mensaje recibido: ADIOS
Conexión aceptada desde ('127.0.0.1', 61225)
Mensaje recibido: Prueba
```

Receptor:

```
Receptor Hamming listo y escuchando en el puerto 12349
Conexión aceptada desde /127.0.0.1:61207
Segmento recibido: '00110010000'
Decodificando el mensaje: 00110010000
Segmento recibido: '00110011111'
Decodificando el mensaje: 00110011111
Segmento recibido: '10110011110'
Decodificando el mensaje: 10110011110
Segmento recibido: '00100011001'
Decodificando el mensaje: 00100011001
Palabra completa decodificada: No se detectaron errores. H [1001000] No se detectaron errores. O [1001111] Error corregido en la posicion: 10. L [100
1100] Error corregido en la posicion: 7. A [1000001]
Palabra completa en Base64 enviada a main.py: Tm8gc2UgZGV0ZWN0YXJvbiBlcnJvcmVzLiBIIFsxMDAxMDAwXSBObyBzZSBkZXRlY3Rhcm9uIGVycm9yZXMuIE8gWzEwMDExMTFdIEV
ycm9yIGNvcnJlZ2lkbyBlbiBsYSBwb3NpY2lvbjogMTAuIEwgWzEwMDExMDBdIEVycm9yIGNvcnJlZ2lkbyBlbiBsYSBwb3NpY2lvbjogNy4gQSBbMTAwMDAwMV0g
```

- Algoritmo de Detección

```
PS C:\Users\50242\Documents\Universidad\CuartoAño\Redes\Lab2-Redes> python Main.py
Ingrese el mensaje a enviar: Hola
Ingrese el algoritmo a utilizar (1 - Hamming, 2 - CRC32): 2
Resultado recibido: El mensaje recibido es correcto. Mensaje: Hola

¿Desea enviar otro mensaje? (s/n): s
Ingrese el mensaje a enviar: Adios
Ingrese el algoritmo a utilizar (1 - Hamming, 2 - CRC32): 2
Resultado recibido: El mensaje recibido es correcto. Mensaje: Adios

¿Desea enviar otro mensaje? (s/n): s
Ingrese el mensaje a enviar: Prueba
Ingrese el algoritmo a utilizar (1 - Hamming, 2 - CRC32): 2
Resultado recibido: El mensaje recibido es incorrecto: Se detectaron errores.

¿Desea enviar otro mensaje? (s/n): n
PS C:\Users\50242\Documents\Universidad\CuartoAño\Redes\Lab2-Redes>
```

Emisor

```
PS C:\Users\50242\Documents\Universidad\CuartoAño\Redes\Lab2-Redes> python Emisor-CRC32.py
Esperando conexión...
Conexión establecida.

Mensaje recibido: Hola
Mensaje CRC32: 01001000011011110110110001100001110011111001001001010110101110110
Mensaje enviado: 01001000011011110110110001100001110011111001001001010110101110110
Conexión establecida.

Mensaje recibido: Adios
Mensaje CRC32: 01000001011001000110100101101111011100110111101110011010010000111110001
Mensaje enviado: 01000001011001000110100101101111011100110111101110011010010000111110001
Conexión establecida.

Mensaje recibido: Prueba
Mensaje CRC32: 01010000011100100111010101100101011000100110000101000001110111010001101101110101
Mensaje enviado: 01010000011100100111010101100101011000100110000101000001110111010001101101110101
```

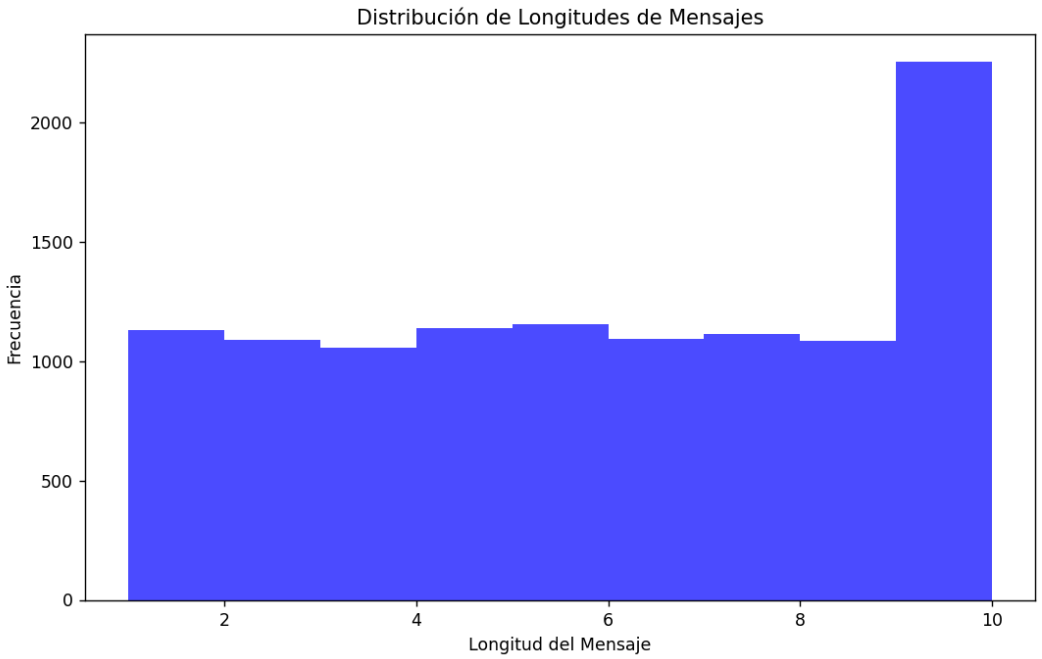
Receptor

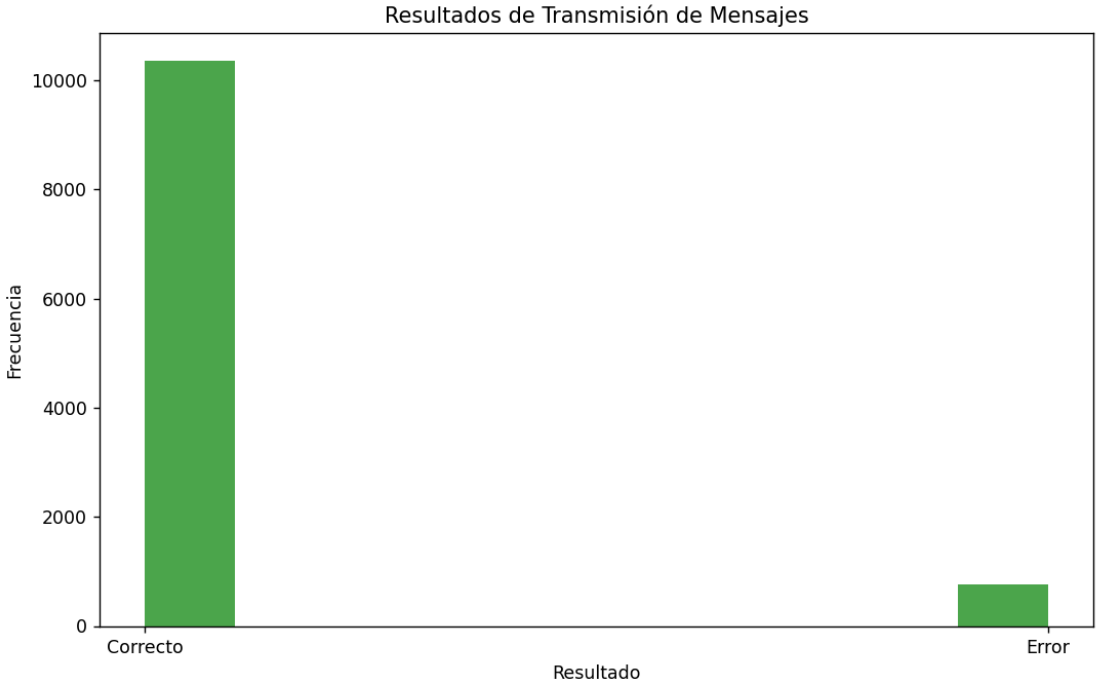
```
PS C:\Users\50242\Documents\Universidad\CuartoAño\Redes\Lab2-Redes>  & 'C:\Program Files\Java\jdk-18\bin\java.exe' '
ionMessages' '-cp' 'C:\Users\50242\AppData\Roaming\Code\User\workspaceStorage\6742784438f5184f256a7788c751976d\redha
f5976\bin' 'CRC32'

Esperando conexión...
Conexión establecida.

Mensaje recibido: 01001000011011110110110001100001
Mensaje decodificado: Hola
Enviando resultado...
Conexión establecida.

Mensaje recibido: 0100000101100100011010010110111101110011
Mensaje decodificado: Adios
Enviando resultado...
Conexión establecida.

Mensaje recibido: 01010000011100100111010101100101011000100110000101
Enviando resultado...
```

Pruebas

- Algoritmo de Corrección

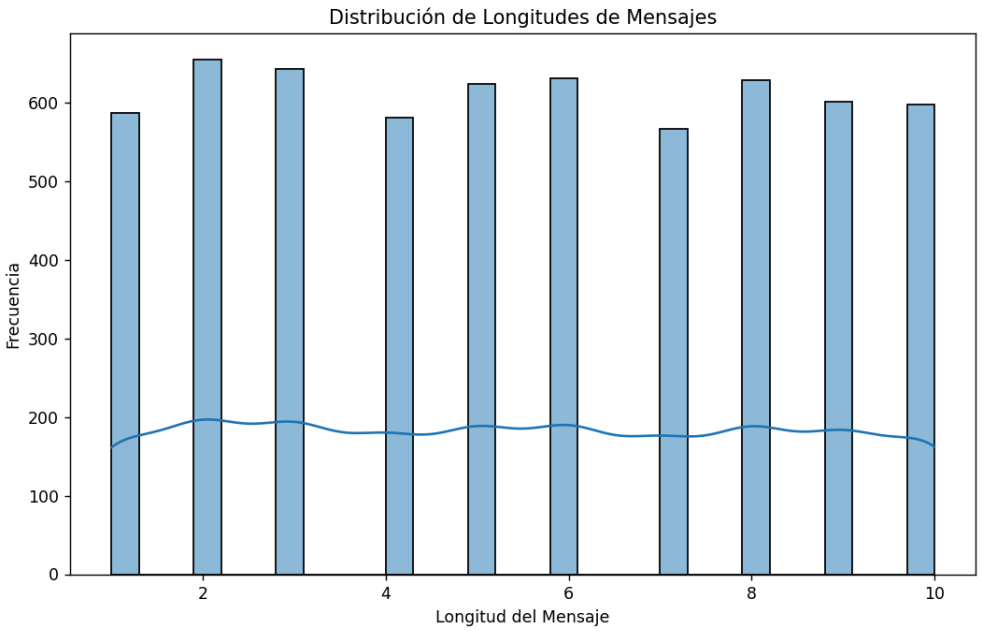
```
Total de mensajes: 11110
Mensajes correctos: 3634
Mensajes con errores: 7476
Porcentaje de correccion de errores: 83%
```

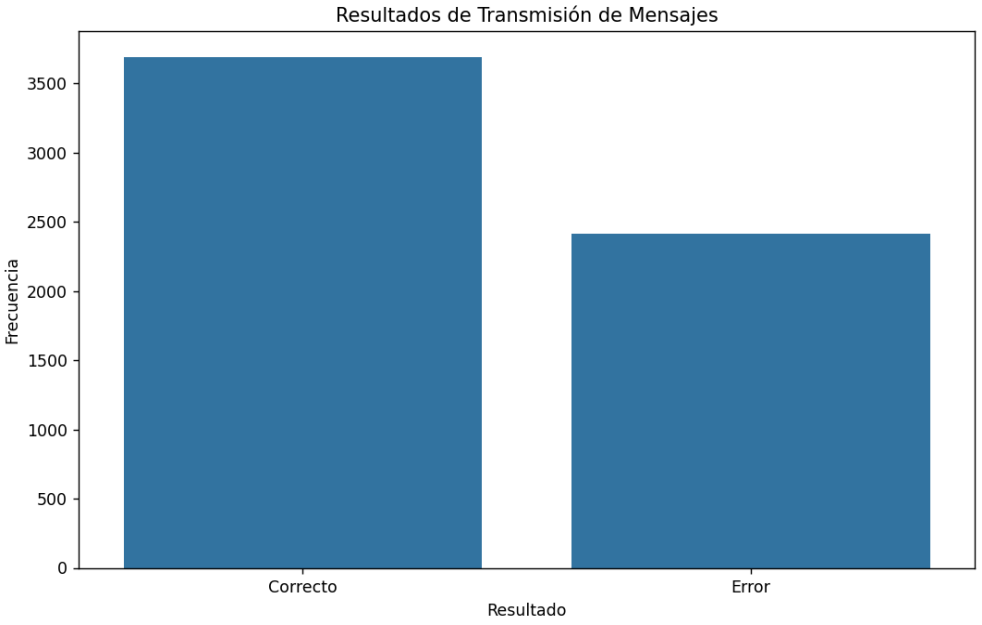


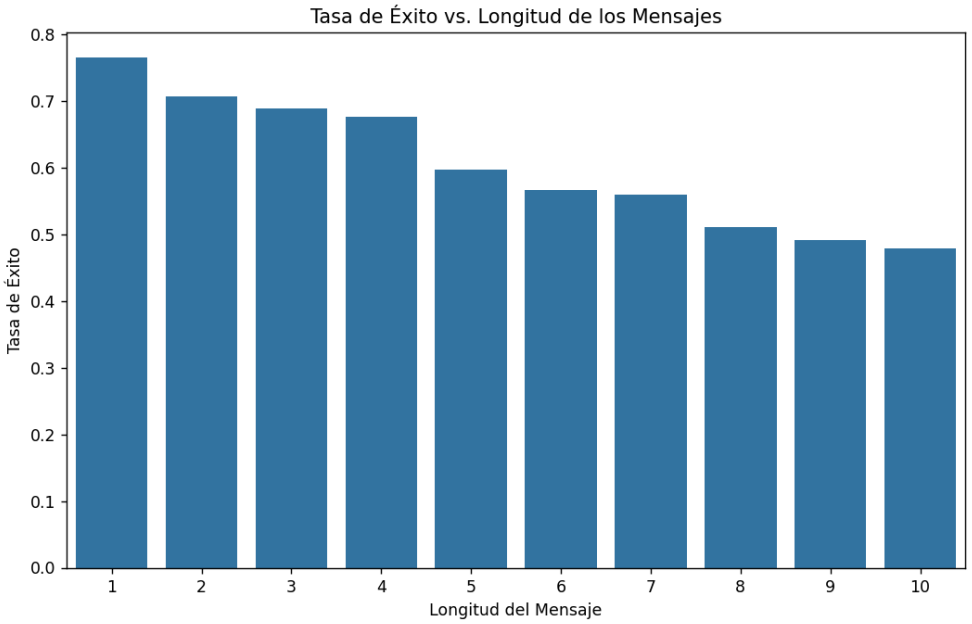


- Algoritmo de Detección

```
PS C:\Users\50242\Documents\Universidad\CuartoAño\Redes\Lab2-Redes> python Analisis.py
Total de mensajes: 6110
Mensajes correctos: 3694
Mensajes con errores: 2416
Tasa de éxito: 60.46%
```


Distribución de Longitudes de Mensajes
Frecuencia
0
100
200
300
400
500
600
2
4
6
8
10
Longitud del Mensaje


Resultados de Transmisión de Mensajes
Frecuencia
0
500
1000
1500
2000
2500
3000
3500
Correcto
Error
Resultado



Discusión

En el análisis del algoritmo de Hamming, observamos variaciones significativas en la tasa de éxito en la corrección de errores. Este indicador cuantifica tanto los errores que el código logra corregir como los casos en los que no se detectan errores inicialmente. Los resultados demuestran que el código de Hamming es efectivo aproximadamente el 83% del tiempo, logrando corregir correctamente los errores y presentar el mensaje recibido junto con su representación binaria en ASCII.

Durante la evaluación, que incluyó más de 10,000 mensajes, identificamos errores. Algunos caracteres pueden confundirse debido a similitudes en sus representaciones binarias, especialmente cuando los patrones binarios son parecidos.

La primera gráfica muestra la distribución de los mensajes analizados según su longitud, con una prevalencia notable de mensajes de diez caracteres. Este parámetro fue medido para evaluar la complejidad que el algoritmo debe manejar y el tiempo requerido para procesar todas las pruebas.

En la segunda gráfica, validamos la tasa de errores y aciertos; de los 11,110 mensajes procesados, el 83% se manejó correctamente, analizando y corrigiendo los errores como se esperaba. El 17% restante presentó problemas en la detección y corrección de caracteres.

Adicionalmente, las pruebas revelaron la mayoría de los errores. En el mecanismo del emisor, se introdujo un bit de ruido por cada 20 bits procesados. En algunos casos, estos bits erróneos se traducían en letras que coincidían con caracteres ASCII válidos,

resultando en mensajes con errores no detectados, como "HOLK", donde la 'K' era un bit de ruido malinterpretado.

Los resultados obtenidos con el algoritmo CRC32 indican que, el sistema tuvo una tasa de éxito del 60.46%, lo que significa que más de la mitad de los mensajes enviados fueron recibidos correctamente sin errores. Sin embargo, también revela que aproximadamente el 39.54% de los mensajes presentaron errores al ser recibidos.

La primera gráfica muestra la distribución de las longitudes de los mensajes enviados. Se puede observar que la mayoría de los mensajes tienen longitudes que oscilan entre 1 y 10 caracteres. Aunque hay una ligera variación en la frecuencia de las longitudes, la diferencia es muy pequeña, siendo más frecuente la longitud 2 y menos frecuente la longitud 7. Esto indica que el sistema se probó con una variedad de tamaños de mensajes, lo que permite evaluar su desempeño en diferentes condiciones.

La segunda gráfica, que representa el porcentaje de mensajes correctos e incorrectos, confirma que una parte significativa de los mensajes sufrió errores durante la transmisión. Con un 60.46% de mensajes correctos y un 39.54% de mensajes incorrectos, queda claro que hay margen para mejorar la robustez del sistema frente a errores.

La tercera gráfica muestra la relación entre la longitud de los mensajes y la tasa de éxito en la transmisión. Aquí, se observa una tendencia clara, mientras menor es la longitud del mensaje, mayor es la tasa de éxito, y mientras mayor es la longitud del mensaje, menor es la tasa de éxito. Esto puede deberse a que los mensajes más largos tienen más bits que pueden ser alterados por el ruido, aumentando la probabilidad de errores durante la transmisión. Por lo tanto, el sistema muestra una mayor robustez con mensajes más cortos.

Conclusiones

- El algoritmo CRC32 ha mostrado ser eficiente, con una tasa de éxito del 60.46%. Sin embargo, el 39.54% de mensajes con errores sugiere que hay factores externos que afectan el rendimiento del sistema.
- La longitud del mensaje tiene un impacto en la tasa de éxito. Mensajes más cortos son menos propensos a errores, mientras que mensajes más largos tienen mayores probabilidades de ser alterados durante la transmisión.
- El algoritmo de Hamming es relativamente bueno para poder manejar errores en los bits codificados.

- Dependiendo de la complejidad del mensaje, como el uso de mayúsculas y minúsculas dentro de este, puede que el algoritmo lo detecte como otro carácter con un número binario parecido.

Repositorio

https://github.com/Sebas021210/Lab2-Redes