UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Redes – Jorge Yass

Laboratorio 2. Parte 2

Francis Aguilar – 22243

Angela García – 22869

Introducción

En cualquier sistema de comunicación digital, los errores de transmisión representan uno de los principales desafíos a enfrentar. Estos errores pueden surgir por múltiples factores como interferencias, ruido o fallos en el medio de transmisión, lo que compromete la integridad de los datos enviados. Por esta razón, a lo largo del tiempo y con la evolución del Internet, se han desarrollado diversos algoritmos y mecanismos diseñados específicamente para detectar y corregir errores durante la transmisión de datos.

Descripción de la práctica

Esta práctica de laboratorio tiene como objetivo el comprender de una manera profunda el funcionamiento de un modelo de capas y sus servicios, implementando sockets para la transmisión de información y lograr experimentar la transmisión de información expuesta en un canal no confiable usando los esquemas de detección y corrección de errores anteriormente usados en la práctica previa. Que son programas que simulan tanto el emisor como el receptor, y utilizando distintos lenguajes de programación para cada uno.

Resultados

Algoritmo de detección: fletcher_checksum

```
PS C:\Users\Francis\OneDrive - UVG\Francis\2025\Semestre 8\Redes\lab2-p2\deteccion> python cliente.py
Ingrese mensaje para enviar: h
Mensaje enviado: 01101000;0110100001101000011010000
```

```
Waiting for message
Recieved message:  01101000
Recieved checksum:  0110100001101000011010000
No se detectaron errores. Mensaje original: 01101000
 Mensaje decodificado: h
Waiting for message
```

```
Waiting for message
Recieved message:  0110100001101111011011000110000
Recieved checksum:  0110100001101111011011000110000101100001011000001
 Se detectaron errores en el mensaje. Mensaje descartado.
```

Algoritmo de corrección: Hamming

```
angel@galleta:~/redes/lab2_parte2_redes$ gcc correccion.c -o correccion
angel@galleta:~/redes/lab2_parte2_redes$ ./correccion

=== Bienvenido a mensajeria 3000 ===

--- Emisor ---
Ingrese el caracter a : R
Caracter ingresado: 'R'
Representacion binaria: 01010010
Longitud de la información: 8 bits
En decimal: 82
Bits de pariedad necesarios: 4
Total de bits en código Hamming: 12

---- Posicionamiento de bits: ----

 (leer de abajo para arriba)
Posicion 1: Paridad (se calculara)
Posicion 2: Paridad (se calculara)
Posicion 3: Dato 0 (bit 0 de datos)
Posicion 4: Paridad (se calculara)
Posicion 5: Dato 1 (bit 1 de datos)
Posicion 6: Dato 0 (bit 2 de datos)
Posicion 7: Dato 1 (bit 3 de datos)
Posicion 8: Paridad (se calculara)
Posicion 9: Dato 0 (bit 4 de datos)
Posicion 10: Dato 0 (bit 5 de datos)
Posicion 11: Dato 1 (bit 6 de datos)
Posicion 12: Dato 0 (bit 7 de datos)

---- Calculo de paridad: ----
P1 (posicion 1) cubre: 1 3 5 7 9 11 -> 3 unos, se pone P1=1
P2 (posicion 2) cubre: 2 3 6 7 10 11 -> 2 unos, se pone P2=0
P4 (posicion 4) cubre: 4 5 6 7 12 -> 2 unos, se pone P4=0
P8 (posicion 8) cubre: 8 9 10 11 12 -> 1 unos, se pone P8=1

Codigo Hamming final (12 bits):
100010110010
----Desglose por posiciones:
----
 (leer de abajo para arriba)
Pos 1: P1 = 1
Pos 2: P2 = 0
Pos 3: Dato = 0
Pos 4: P4 = 0
Pos 5: Dato = 1
Pos 6: Dato = 0
Pos 7: Dato = 1
Pos 8: P8 = 1
```

```
Codigo Hamming final (12 bits):
100010110010
----Desglose por posiciones:
----
 (leer de abajo para arriba)
Pos 1: P1 = 1
Pos 2: P2 = 0
Pos 3: Dato = 0
Pos 4: P4 = 0
Pos 5: Dato = 1
Pos 6: Dato = 0
Pos 7: Dato = 1
Pos 8: P8 = 1
Pos 9: Dato = 0
Pos 10: Dato = 0
Pos 11: Dato = 1
Pos 12: Dato = 0
Agregar ruido? (s/n): s
Cuantos errores introducir? (1-3): 1
Ingrese posicion del error 1 (1-based): 7
error 1 introducido en posicion 7
Ingrese cantidad de bits de datos originales: 8

--- Receptor  ---

--- DECODIFICACION ---
Codigo recibido (12 bits): 100010010010

Calculo de sindrome:
S1 (P1): 1 3 5 7 9 11 -> 3 unos (impar) -> S1=1
S2 (P2): 2 3 6 7 10 11 -> 1 unos (impar) -> S2=1
S4 (P4): 4 5 6 7 12 -> 1 unos (impar) -> S4=1
S8 (P8): 8 9 10 11 12 -> 2 unos (par) -> S8=0
Sindrome: 7
Error detectado - Sindrome: 7
Posicion indicada por sindrome: 7

/// ADVERTENCIA ///

El codigo Hamming solo puede:
1. Corregir 1 error de forma confiable
2. Detectar (pero NO corregir) 2 errores
3. Con 3+ errores: comportamiento impredecible y usualmente lo suele empeorar

Intentar correccion? (s/n): s
Correccion aplicada en posicion 7
Trama corregida: 01010010
```

Gráficos

Respuesta a las preguntas mencionadas

- ¿Qué algoritmo tuvo un mejor funcionamiento?
    - Hamming es fuerte en detección y corrección de errores simples, mientras que Fletcher es mejor para detección rápida y eficiente de errores múltiples sin corrección, por tanto, uno es mejor en entornos con necesidad de corrección automática y el otro en contextos con mayor volumen de datos y menor tolerancia al retraso.
- ¿Qué algoritmo es más flexible para aceptar mayores tasas de errores?
    - El algoritmo más flexible para aceptar mayores tasas de errores es el código de Hamming, ya que no solo detecta errores, sino que también puede corregir automáticamente errores de un solo bit y, en algunos casos, detectar errores de

dos bits. Esta capacidad de corrección lo hace más adecuado en entornos donde los errores son frecuentes o inevitables.

- ¿Cuándo es mejor utilizar un algoritmo de detección errores en lugar de uno de corrección de errores?
    - Es mejor utilizar un algoritmo de detección de errores en lugar de uno de corrección de errores cuando el sistema puede retransmitir los datos rápidamente y sin mucho costo. En estos casos, detectar que ocurrió un error y pedir una nueva transmisión es más eficiente que implementar lógica compleja para corregirlo automáticamente.

Discusión

Los sistemas de comunicación tienen una arquitectura de capas, estos sirven para organizar y estructurar el proceso de transmisión de datos de una manera modular, escalable y eficiente. Cada capa tienen una función específica que permite que se pueda trabajar en una capa sin afectar a los demás o actualizar una capa sin tener que rehacer el programa. Además, una aplicación no necesita saber cómo es que se envían los bits, el usuario solo se encarga de enviar y recibir mensajes.

Las capas que se tienen son las siguientes: aplicación, presentación, enlace, ruido y transmisión. Cada una de estas tiene servicios que se implementaron en los algoritmos por parte del emisor y receptor. Entonces, se realizaron pruebas para evaluar el comportamiento de cada uno de los algoritmos. Empezando con el algoritmo de ...

Comentario grupal sobre el tema

Es interesante conocer un poco de todo lo que llevan los sistemas de comunicación y cómo es que existen varios algoritmos para la detección y corrección de errores. Y cómo es que las capas son muy importantes para la modularidad, abstracción, reutilización, estandarización y escalabilidad. Considerando todos los servicios que conllevan, tanto de parte del emisor y receptor.

Conclusiones

- Se comprendió el funcionamiento de modelos de capas y sus servicios, implementando emisor y receptor con cada una de sus capas correspondientes.
- Se experimentó la transmisión de información por medio de un canal no confiable y se usó sockets para enviar y recibir la trama de indormación por medio del puerto elegido.
- Se analizó el funcionamiento de los esquemas de detección y corrección, siendo el algoritmo de ...

Citas y referencias

Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2), 147–160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x

GeeksforGeeks. (2021, July 28). *Hamming Code - Error detection and error correction*. https://www.geeksforgeeks.org/hamming-code-error-detection-and-error-correction/

Sanjay Raghavendra (2020) Implementation of Fletcher CheckSum [Video]. YouTube https://www.youtube.com/watch?v=e4nuUYiQE-A