UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

CC3085 – Redes

Sección 10

Ing. Miguel Novella Linare

Laboratorio 2

Samuel Argueta - 211024

Alejandro Martinez - 21430

GUATEMALA, 25 de julio de 2024

DESCRIPCIÓN DE LA PRÁCTICA

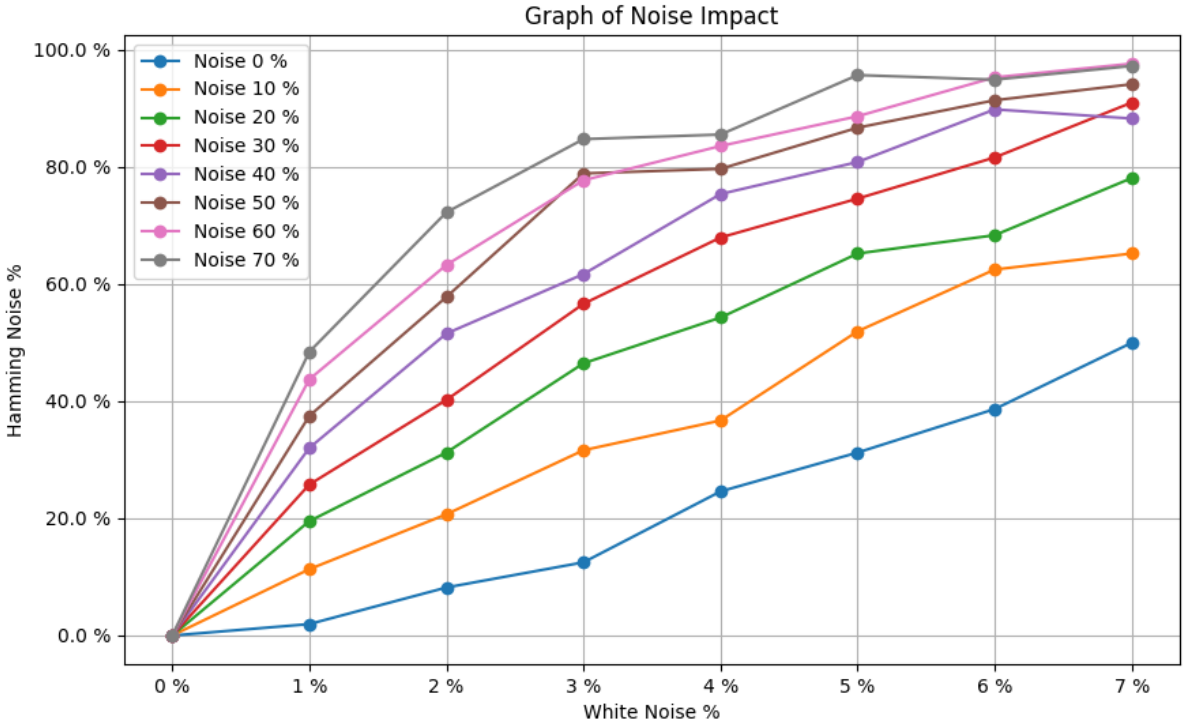
En este caso, el propósito del laboratorio es familiarizarse con el concepto de capas de una manera práctica enviando mensajes de manera local entre un “emisor” y un “receptor” emulado de manera local, en donde, los mensajes deberán ser traducidos a binario y, en algunos casos, poseer errores, justo como sucedería en un escenario real. Se debe solicitar un mensaje de entrada, que es lo que el Emisor desea transmitir, el cual deberá ser traducido a binario. El Receptor recibirá este código binario y deberá validar que esté bien, en caso de estarlo, mostrará el mensaje, en caso contrario, mostrará mensaje de error.

METODOLOGÍA

Utilización del concepto de Sockets para la comunicación entre clientes, esto con el fin de poder enviar y recibir mensajes de manera simulada entre emisor y receptor. En donde cada uno tendrá su implementación en lenguajes diferentes, con el fin de poder ser más realista la simulación. En este caso se tiene registro de cada uno de los envíos que se realizan del emisor a receptor mediante un archivo .log para constatar cada uno de los resultados.

RESULTADOS

Stats **Log**



Ejemplos de un fallo y una corrección:

```
[Send] Message : Indica el mensaje original.
[Send] Hamming : Mensaje, en bits, despues del algoritmo de Hamming.
[Send] Noisy :  Mensaje, en bits, despues del algoritmo de Hammingr Ruido.
    En amarillo [ERROR RATE]: Bits con Ruido Blanco
    En verde    [NOISE FACTOR]: Bits restringidos a un solo flip cada 7
bits (cada chunk de hamming)
[Hamming] Errors  : Indica la posición de los bits con errores.
[Hamming] Decoded : Mensaje en bits despues de ser decodificado.
[CRC32] o [CRC32] y los hashes si el mensaje se computa correctamente.
[Received] : Mensaje original si se recibió correctamente.
```

```
[Send] Message: est fugiat eiusmod
[Send] Hamming:
1100110010010100011111000011000111110011000101010000000011001101100110000
1111010010111001100001111111001100011001110011011010010001111100110001010100
0000001100110010010111001100011001000111101001010001111100001111001101010
0111001101111111110011010011100
[Send] Noisy  :
1100110010010100011111000011000111110011000101010000000011001101100010000
1111010011111001100001111111001100011001110011111100100011111001100010101000
0000001100110000010111001100011001010111010010101011111000011110011010101
0111001101110111111011010011100
[Hamming] Errors:
1100110010010100011111000011000111110011000101010000000011001101100010000
1111010011111001100001111111001100011001110011111100100011111001100010101000
```

```
0000001100110000010111001100011001010111101001010101111100001111001101010101
011100110111011111101101001100
[Hamming] Decoded:
0110010101110011011101000010000001100110011101010110011101101001011000010101
11010000100000011001010110100101110101011100110110110101101111101100100
[CRC32] Verified 0x47a97694 == 0x47a97694
[Success] : est fugiat eiusmod
```

```
[Send] Message: aliqua. laborum.
[Send] Hamming:
11001101101001110011001111001100110001100100011111101001000111101001011100
11011010010101010001011001010100000000110011001111001100110110100111001100
101010110011011111111000111101010100001111010010111001101010101010101000101
10
[Send] Noisy  :
11001101101001111011011111001100110001100100011111111001000111101001011000
11011010010101010000011001010100000000110011001111001100110110100111001100
101010110011011101110001111010101001010011010010111001101010101010101000101
10
[Hamming] Errors:
11001101101001111011011111001100110001100100011111111001000111101001011000
11011010010101010000011001010100000000110011001111001100110110100111001100
10101011001101110111000111101010100101011010010111001101010101010101000101
10
[Hamming] Decoded:
01100001011011000110100101110001011101010110000100101110001000000110110001
100001011000100110111101110010001001010110110100101110
[CRC32] Failed 0x5b16ee51 != 0x5fbf871b
[Failure]
```

DISCUSIÓN

CRC32 es altamente fiable para la detección de errores en los datos transmitidos, pero carece en la corrección de los mismos. Este no corrige los errores, pero es extremadamente eficaz para detectarlos. Su poder está en que puede detectar rafagas de errores, esto se entiende por varios bits incorrectos en la transmisión pero, si la tasa de errores es alta, lo detecta, solo que se debe realizar una retransmisión de datos o el uso de un algoritmo adicional.

En contraparte, Hamming no solo detecta errores, sino que también los corrige. Este es mas flexible para manejar y corregir errores de 1 solo bit, pero su capacidad de corrección disminuye drásticamente cuando aumenta la tasa de errores. No puede corregir errores en rafaga, es decir, múltiples bits con problemas en la transmisión.

Cada uno tiene su escenario de uso. Hamming es útil en situaciones donde la corrección de errores es crucial; CRC32 es eficiente y rápido en la detección de errores, lo que lo hace adecuado para entornos donde la corrección de errores no es crítica, pero la detección rápida es esencial.

Algoritmos de detección de errores (CRC32)	Algoritmos de Corrección de errores (Hamming Code)
Alta tasa de errores, esto es cuando la probabilidad de errores es alta y la corrección en el lugar no es factible o eficiente.	Baja tasa de errores, donde los errores son raros y se pueden manejar de manera eficiente en el lugar.
Retransmisión de datos, donde es viable transmitir datos, como en redes de computadoras, se puede utilizar para detectar errores y solicitar una retransmisión si se detecta un error.	Crítica de corrección en tiempo real, aplicaciones críticas donde la corrección en tiempo real es esencial, como en sistemas de memoria de computadoras y comunicaciones espaciales.
Simplicidad y eficiencia, cuando se necesita un método simple y eficiente para verificar la integridad de los datos sin la sobrecarga de la corrección de errores.	Limitación de retransmisión, Cuando la retransmisión de datos no es posible o deseable debido a restricciones de tiempo o ancho de banda.

CONCLUSIONES

Cada algoritmo es útil según las necesidades. CRC32 es más adecuado para la detección rápida y eficiente de errores en sistemas donde la retransmisión es una opción. Hamming Code es ideal para entornos donde la corrección de errores en tiempo real es crucial y la tasa de errores es baja.

El Algoritmo de Hamming nos permite corregir errores, dado que solo exista un error en el chunk/segmento definido. Adicionalmente de este ruido controlado, al usar ruido blaco se pueden introducir errores que no se pueden corregir.

Dado este hecho, se utiliza el Algoritmo / hashing CRC32 para verificar si existen errores que hamming no pudo corregir, lo cual descarta ese mensaje.

REFERENCIAS

Ritter, C. (2016, March 6). A painless guide to CRC error detection algorithms. Command Line Fanatic. https://commandlinefanatic.com/cgi-bin/showarticle.cgi?article=art008

GeeksforGeeks. (n.d.). Modulo-2 binary division. Retrieved August 1, 2024, from https://www.geeksforgeeks.org/modulo-2-binary-division/

GeeksforGeeks. (n.d.). Hamming code in computer network. Retrieved August 1, 2024, from https://www.geeksforgeeks.org/hamming-code-in-computer-network/

TechTarget. (n.d.). Hamming code. Retrieved August 1, 2024, from https://www.techtarget.com/whatis/definition/Hamming-code