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

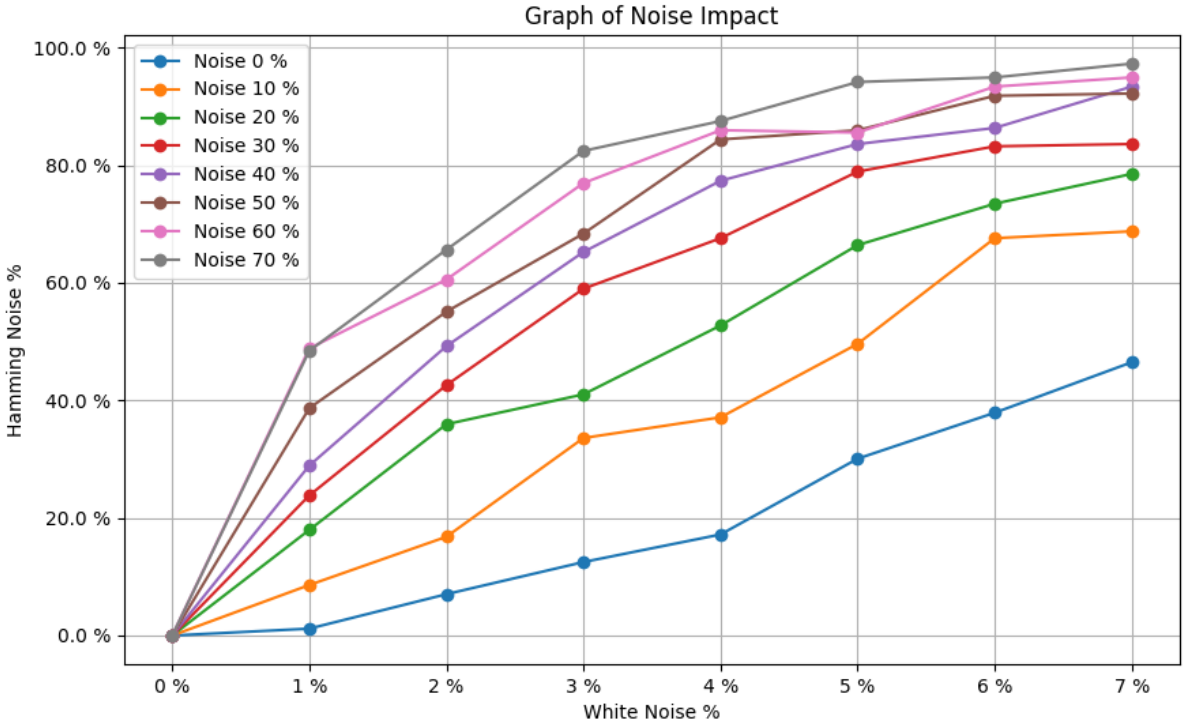
En este caso, el propósito del laboratorio es familiarizarse con el concepto de capas de una manera práctica enviando mensajes de manera local entre un “emisor” y un “receptor” emulado de manera local, en donde, los mensajes deberán ser traducidos a binario y, en algunos casos, poseer errores, justo 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

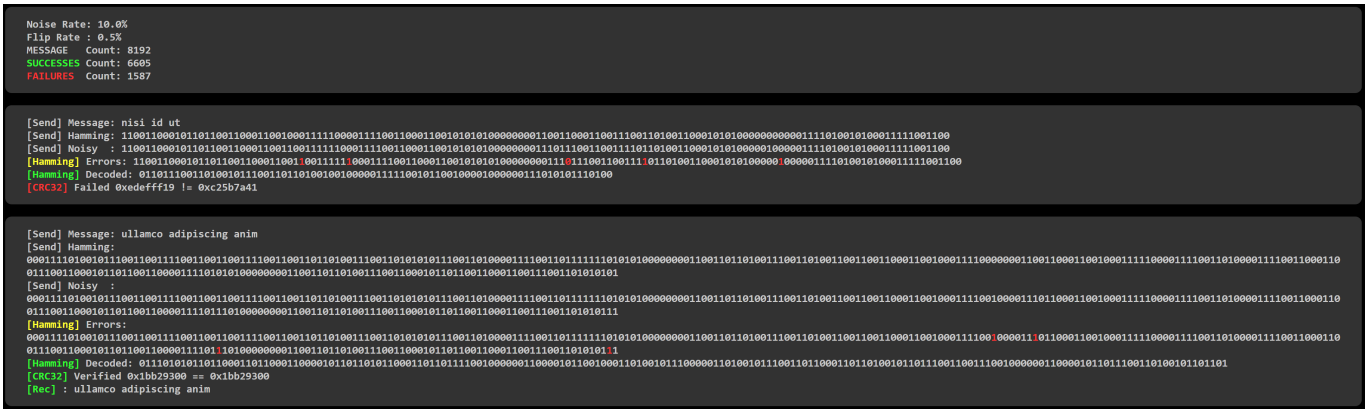
Utilización del concepto de Sockets para la comunicación entre clientes, esto con el fin de poder enviar y recibir mensajes de manera simulada entre emisor y receptor. En donde cada uno tendrá su implementación en lenguajes diferentes, con el fin de poder ser más realista la simulación. En este caso 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.
[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.
[Rec] : Mensaje original si se recibió correctamente.
```

```
[Send] Message: esse est mollit minim pariatur. Ut dolor
[Send] Hamming:
1100110010010100011111000011000111110000111100110010010101010100000000011001100100101000111110000110001111100011000101010000000011001101010101110011011
```

```
1111111100110011110011001100111100110011000110010001111100110001010100000000
0011001101010101110011000110011100110001011011001100011001110011010101010101
0101000000000000111100000000110011011010010001111010101011001100011001110011
0110100100011111001100000111101001010001111010101001010100010110010101010000
0000010010101001010001111100110001010100000000011001101001100110011011111111
1100110011110011001101111111100011110101010
[Send] Noisy  :
1100110010110100011111000011000111110000110100110011010101110100001 0001100
1100100100000111110000110001111100110001010100000001001001101110101110011 01
1111111100111011110011001101111110011001100011001001111100011000101010000000
0011001111010101110011000110011100110001011001001100011011110011010101010101
1101000000000001101000000001100110110100100011110101010110011000110011100 11
0110100100011111001100000110101001000001111011101001010100010110010101010000
0000010011101001010001111000110111010100000000011001101001100110011011111111
1100100001110011001101111111100010110111010
[Hamming] Errors:
1100110010110100011111000011000111110000110100110011010101110100001 0001100
1100100100000111110000110001111100110001010100000001001001101110101110011 01
1111111100111011110011001101111110011001100011001001111100011000101010000000
0011001111010101110011000110011100110001011001001100011011110011010101010101
1101000000000001101000000001100110110100100011110101010110011000110011100 11
0110100100011111001100000110101001000001111011101001010100010110010101010000
0000010011101001010001111000110111010100000000011001101001100110011011111111
1100100001110011001101111111100010110111010
[Hamming] Decoded:
0110010101110011011100110110010100100000011001010111001101110100001000000 1
1011010110111101101100011011000110100101110100001000000110110101101001011 0
1110011010010110110100100000011100000110000101110010011010010110000101110 1
0001110101011100100010111000100000010101010111011100100000011001000110111 1
0110110001101111011100 10
[CRC32] Failed  0x1ac4f92 !=  0xac57970
```

```
[Send] Message: commodo incididunt cillum exercitation reprehenderit
proident, ullamco
[Send] Hamming:
1100110100001111001101111111111001101010101110011010101011100110111111111100
1101001100110011011111111010101000000000110011000110011100110001011011001101
0000111100110001100111001101001100110011000110011100110100110000011110100 1
0111001100010110000111110011000101010000000001100110100001111001100011001 11
0011001111001100110011110000011110100101110011010101010101010000000001100 11
0010010100011111110000110011001001010001111010101011001101000011110011000 1
1001000111110011001100110110100100011111001100110011000110011100110111111 1
1100110001011001010100000000000011110101010110011001001010001111000000000 01
1110101010110011001001011100110111000011001100100101110011000101101100110 1
0011001100110010010100011110101010110011000110010001111100110001010100000 00
0000011110000000000111101010101100110111111111100110001100111001101001100 11
0011001001011100110001011000011111001100010101001111000101010000000000001 11
1010010111001100111100110011001111001100110110100111001101010101110011010 0
0011110011011111111
[Send] Noisy  :
1100110100001111001101111111111001101010001110011010101011100110111111111100
```

```
1101101100110011011111010101000000011001100010001110011000101101100110100
0000111100111001000111001101001100110011000110001100110010100110000011110100
0101001100010110000111110011000101010001000011001101010011110011000110011100
0001001111001100110011100000011110000101110111010101010101010000000011001100
0010010100011111110100110011001001000011111000101011001101000001110011000100
0001100111110011001100110110000100111111001000110011000110011100110111111100
1100111000011001010100000000000111101010101100110010000100011110100000100100
1110101011110011000001011100110111000011011100100101110111000100101100110100
0011101100110010010100011110101010110011000110010001111100110001010110000000
0000011110000100000111101010100100110111111111001100011001110111010011000011
0011001001011101110001011001011111001000010101001111000101010000001000011100
1010010111001100111001100110011110011001101101001110011010101011000110100000
0011110011011111110
[Hamming] Errors:
1100110100001111001101111111110011010100111001101010101110011011111111110000
1101101100110011011111010101010000000111001100010001100111001100010110110011010
0000111100111001000111001101001100110011000110001100111010011000001111010010
0101001100010110000111110011000101010001000011001101010011110011000110011100
0001001111001100110011100000011110000101110111010101010101010000000011001100
0010010100011111110100110011001001000111110001010110011010000011100110001
0001100111110011001100110110000100111111001000110011000110011100110111111100
1100111000011001010100000000000111101010101100110010000100011110100000100100
1110101011110011000001011100110111000011011100100101110111000100101100110100
0011101100110010010100011110101010110011000110010001111100110001010110000000
0000011110000100000111101010100100110111111111001100011001110111010011000011
0011001001011101110001011001011111001000010101001111000101010000001000011100
1010010111001100111001100110011110011001101101001110011010101011000110100000
0011110011011111110
[Hamming] Decoded:
0110001101101111011011010110110101101111011001000110111100100000011010010 1
1011100110001101101001011001000110100101100100011101010110111001110100001 0
0000011000110110100101101100011011000111010101101101001000000110010101111 0
0001100101011100100110001101101001011101000110000101110100011010010110111 1
0110111000100000011100100110010101110000011100100110010101101000011001010 1
1011100110010001100101011100100110100101110100001000000111000001110010011 0
1111011010010110010001100101011011100111010000101100001000000111010101101 1
0001101100011000010110110101100011011011 11
[CRC32] Verified 0xd59d764f == 0xd59d764f
[Rec] : commodo incididunt cillum exercitation reprehenderit proident,
ullamco
```

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