Universidad del Valle de Guatemala
Sara María Paguaga - 20634
Guillermo Santos Barrios - 191517

Laboratorio 2.1 - Esquemas de detección y corrección de errores

Descripción de la práctica

La presente práctica facilita la comprensión de conceptos relacionados con la codificación, envío de mensajes binarios, detección y corrección de errores. Para ello se implementaron los siguientes algoritmos:

- Hamming: Este código genera tramas utilizando la siguiente premisa: $(m + r + 1) <= 2^r$, donde r es el número de bits redundantes y m es la cantidad de bits que contienen información. Los bits redundantes deben de tener posiciones que sean potencias de 2 (1,2,4,8,...). Por ejemplo, un código Hamming (11,7) convertirá la cadena 1001101 a la trama 01110010101. Para identificar si la trama es correcta se hace el proceso inverso. Primero obtenemos los bits de data de la trama recibida, calculamos los bits de paridad que debería de tener y se compara con los bits de paridad recibidos. Si estos no coinciden entonces sucede un error. Hamming permite corregir únicamente un error por la naturaleza en la que se calculan los bit de paridad (Tanenbaum, 2010).
- Checksum fletcher 8, 16, 32 y 64: Este código sirve únicamente para detectar errores. El principio es calcular una checksum, la cuál puede ser una palabra de 8, 16, 32 o 64 bits (usualmente). Esta checksum se agrega al final de la trama a enviar y posteriormente el receptor vuelve a calcular la suma y esta debe de coincidir con el valor recibido. Este algoritmo es más seguro que una checksum simple porque tiene un componente posicional (Tanenbaum, 2010).

Resultados

Prueba 1 - Hacer encoding y decoding de tramas

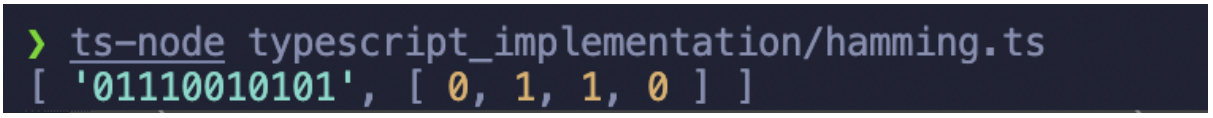
Hamming	Resultado Encoding Hamming	Fletcher16	Resultado Encoding Fletcher 16	Resultado Encoding Fletcher 32	Resultado Encoding Fletcher 64
1001101	01110010101	1001100	0100110001001100	00000000000000000100110001100110	
1111101	10111110101	1001110	0100111001001110	00000000000000000100111001100111	

				0	
1101100	00101011100	1001111	0100111101001111	0000000000000000100111110100111	

Tabla 1 - Tramas primer ejercicio

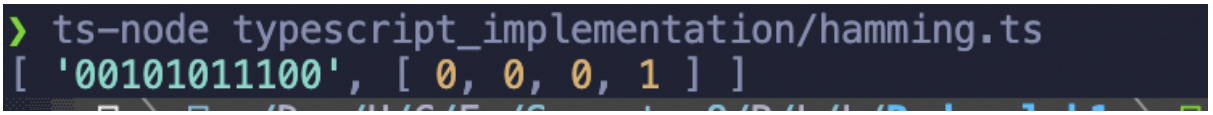
Encoding

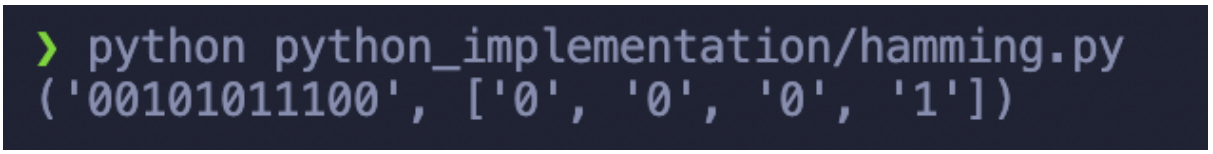
Hamming 1001101:



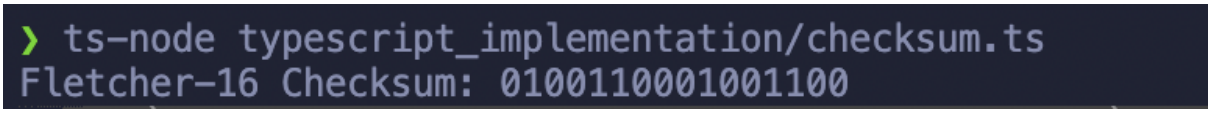
Hamming 1111101:

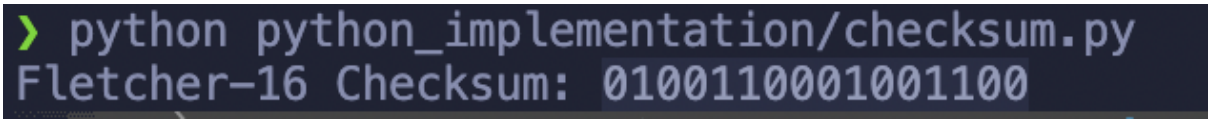
Hamming 1101100:





Fletcher16 1001100:





Fletcher16 1001110:

```
❯ ts-node typescript_implementation/checksum.ts
Fletcher-16 Checksum: 0100111001001110
```

```
❯ python python_implementation/checksum.py
Fletcher-16 Checksum: 0100111001001110
```

Fletcher16 1001111:

```
❯ ts-node typescript_implementation/checksum.ts
Fletcher-16 Checksum: 0100111101001111
```

```
❯ python python_implementation/checksum.py
Fletcher-16 Checksum: 0100111101001111
```

Fletcher32 1001100:

```
❯ python python_implementation/checksum.py
Fletcher-32 Checksum: 00000000000000000100110001001100
```

```
❯ ts-node typescript_implementation/checksum.ts
Fletcher-32 Checksum: 00000000000000000100110001001100
```

Fletcher32 1001110:

```
❯ python python_implementation/checksum.py
Fletcher-32 Checksum: 00000000000000000100111001001110
```

```
❯ ts-node typescript_implementation/checksum.ts
Fletcher-32 Checksum: 00000000000000000100111001001110
```

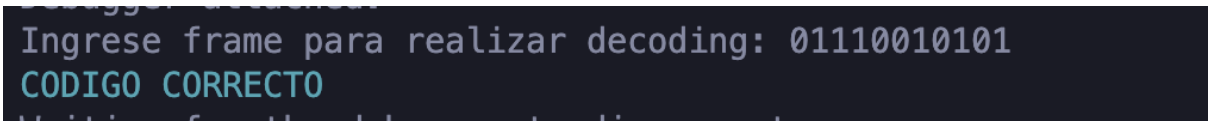
Fletcher32 1001111:

```
❯ python python_implementation/checksum.py
Fletcher-32 Checksum: 00000000000000000100111101001111
```

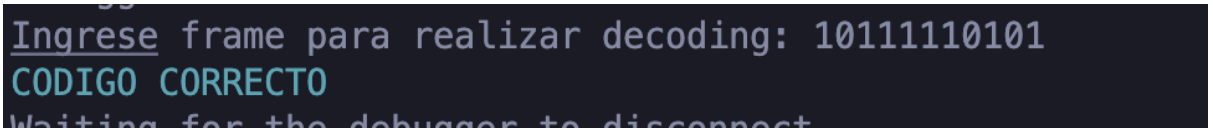
```
❯ ts-node typescript_implementation/checksum.ts
Fletcher-32 Checksum: 00000000000000000100111101001111
```

Decoding

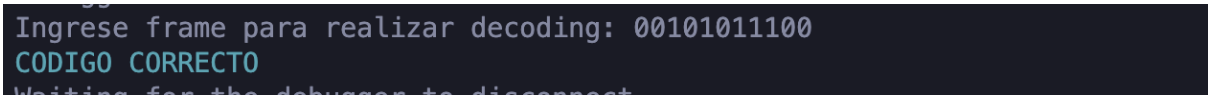
Hamming 01110010101:



Hamming 10111110101:

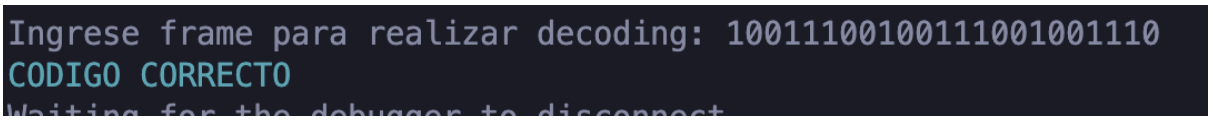


Hamming 00101011100:

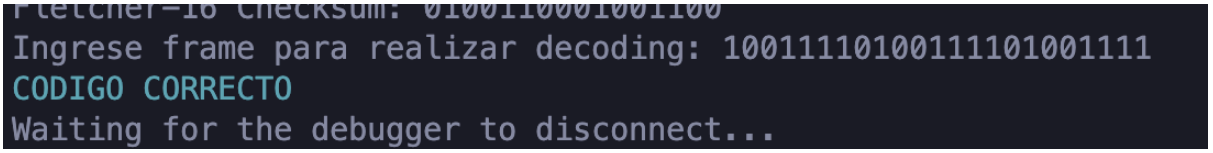


Fletcher 1001100010011000100100:

Fletcher 1001110010011100100110:



Fletcher 1001111010011110100111:



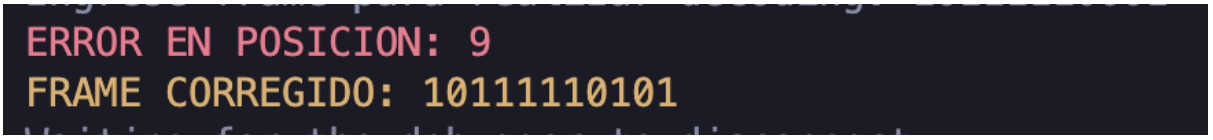
Prueba 2 - Corrección y detección de error

Hamming	Hamming con error	Fletcher16	Fletcher con error
1010101	11110100100	1001100	0100110001001101
1111101	10111110001	1001110	0100111001001111
1101100	00101111100	1001111	0100111111001111

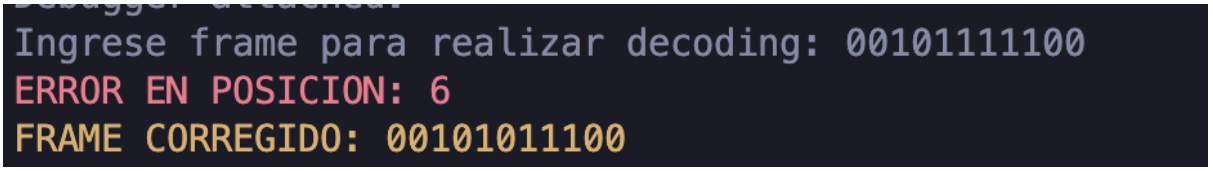
Tabla 2 - Tramas segundo ejercicio

Hamming 11110100101:

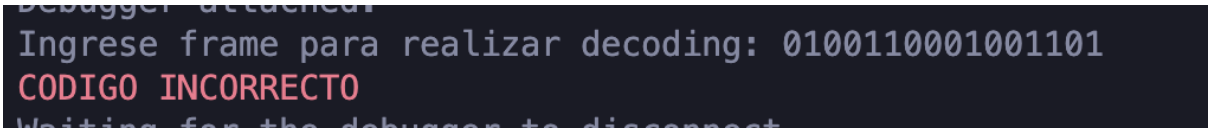
Hamming 10111110001:



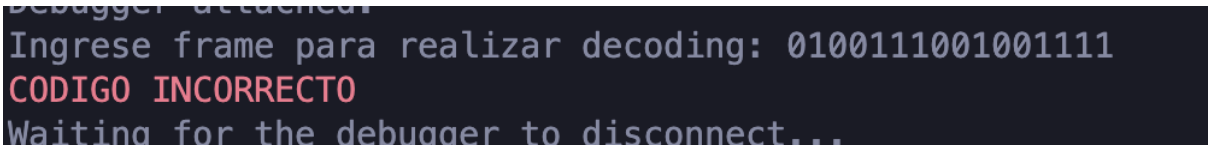
Hamming 00101111100:



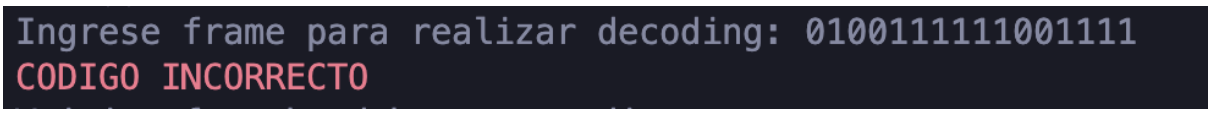
Fletcher 0100110001001101:



Fletcher 0100111001001111:



Fletcher 0100111111001111:



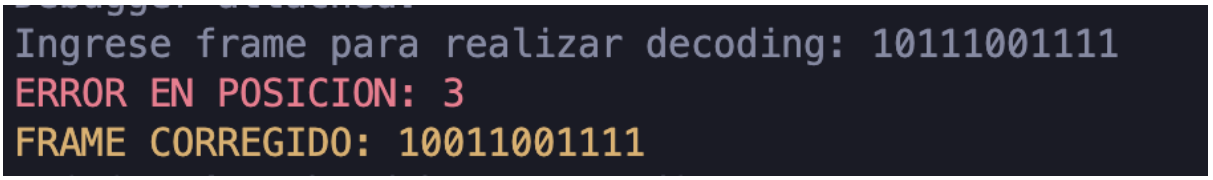
Prueba 3 - Corrección y detección de errores

Esta prueba busca identificar si los algoritmos pueden identificar más de un error. Ambos algoritmos detectan que hay errores. Sin embargo, el algoritmo de Hamming pierde la capacidad de indicar la posición de los errores. Esto se debe a que solo es capaz de detectar un error en la trama. En las capturas de pantalla se puede observar que erróneamente indica una posición de error cuando estos se encuentran en otras posiciones.

Hamming	Hamming con errores	Fletcher16	Fletcher con error
1010111	1011101001111	1011100	10100000101110001011100
1010101	111101111101	1101110	11011111110111001101110
1101101	001011111100	1000111	11101110100011101000111

Tabla 3 - Tramas tercer ejercicio

Hamming 1011101001111:

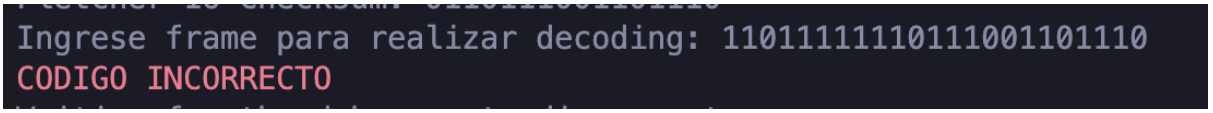


Hamming 1111011101:

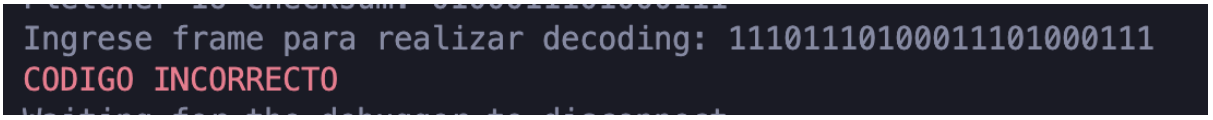
Hamming 0010111100:

Fletcher 10100000101110001011100:

Fletcher 11011111110111001101110:

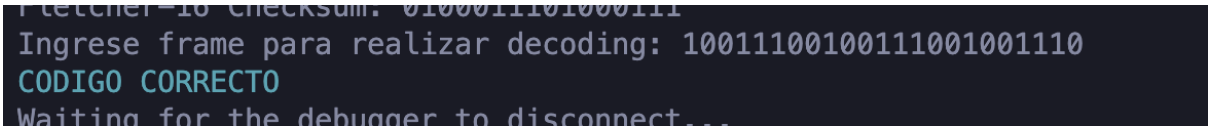


Fletcher 11101110100011101000111:



Prueba 4 - Utilizar trama con error que no pueda ser detectado por receptor

Utilizando el receptor con el algoritmo de Fletcher16 y la siguiente trama: 10011100100111001001110, se procede a agregar un cero a la izquierda 010011100100111001001110. En las dos capturas de pantalla que se muestran se puede observar que el algoritmo afirma que ambos códigos son correctos, a pesar de el cero que se agregó en el segundo.



Discusión

El primer paso es validar si los algoritmos son capaces de hacer tramas y validar si las mismas son correctas o incorrectas. En la Tabla 1 se listan las tramas que fueron usadas para esta prueba, así como el resultado tras pasarlas por los algoritmos elegidos. Posteriormente se adjuntan capturas de pantalla que demuestran que el receptor es capaz de categorizar las tramas emitidas por el emisor como correctas. El receptor únicamente necesita la trama original para identificar si es correcta o no.

Posteriormente se realiza una segunda prueba para encontrar la capacidad del algoritmo de detección y corrección de errores (Hamming) con tramas incorrectas. También se utiliza el algoritmo de Fletcher como identificador de errores. En las capturas de pantalla se puede observar que el algoritmo correctamente identifica errores y en el caso de

Hamming es capaz de mostrar en qué posición se encuentra el mismo. Todas las tramas de la Tabla 2 fueron identificadas correctamente por los algoritmos, en el caso de Hamming se observa que correctamente identifica la posición del error y retorna la trama correcta. Cabe destacar que esto solo puede suceder si hay un error, de lo contrario el algoritmo implementado de Hamming no funciona correctamente. Esto se intentó en el ejercicio 3 (Tabla 3) donde se envían tramas con dos errores y se puede observar que el algoritmo de Hamming logra identificar correctamente que hay errores, pero ya no los corrige correctamente.

Por último en la Prueba 4 se intentó buscar una manera de "pasar por alto" el algoritmo de detección de errores. En este caso se agrega un cero a la izquierda de la trama original. Esto provoca que ambas tramas sean aceptadas.

Comentario Grupal

Esta práctica nos permitió conocer la complejidad de asegurar la integridad de la información cuando ésta viaja por una red. Esta integridad puede verse afectada por la calidad del medio, capacidad del receptor, etc. Asimismo se descubrió la importancia de conocer las razones por la cuál utilizar un algoritmo que detecta errores y uno que corrige errores. Por otra poner a prueba los algoritmos de Hamming y Checksum fletcher nos ayudó a conocer tanto sus ventajas y desventajas como puntos débiles. Todo este proceso es beneficio para lograr entender a mayor profundidad la codificación y decodificación de mensajes para luego ser capaces de enviar de forma eficiente mensajes en futuras etapas.

Conclusiones

- El uso de algoritmos de corrección de errores debe ser justificado por el uso que tengan los datos y la calidad del medio.
- La detección de errores puede ser una medida de seguridad suficientemente aceptable para el uso cotidiano que se le dan a las redes dado que el emisor puede enviar nuevamente paquetes con errores.
- Si se utiliza un enfoque orientado a la conexión es más efectivo utilizar un algoritmo de corrección de errores.
- La eficacia del algoritmo de Hamming se ve afectada cuando hay múltiples errores en la trama y puede indicar falsos errores.
- Los algoritmos tanto de Hamming como de Checksum Fletcher pueden ser burlados por técnicas como colocar ceros al principio de la cadena de bits y dar falsos positivos.

Referencias

- Tanenbaum, A. S., & Wetherall, D. J. (2010). Computer Networks (5th ed.). Pearson.
- https://www.youtube.com/watch?v=gQK9nROFX20&t=375s&ab_channel=UniversitatPolit%C3%A8cnicadeVal%C3%A8ncia-UPV
- https://en.wikipedia.org/wiki/Fletcher%27s_checksum

Repositorio de github:
https://github.com/MGonza20/Redes_lab1