CC3067 - Redes
Sección 20
José Pablo Kiesling Lange - 21581
Erick Stiv Junior Guerra Muñoz - 21781

Laboratorio No.02 - Parte 1

Esquemas de detección y corrección de errores

Los algoritmos implementados para este laboratorio fueron **Fletcher checksum** y **Código de Hamming** para la detección y corrección de errores respectivamente.

Escenarios de pruebas

Los mensajes que se probarán serán:

- 11010110
- 101110101010
- 1111000010101100

Sin errores

- **Fletcher checksum**

```
PS C:\Users\TheKi\OneDrive - UVG\Semestre VIII\redes\redes_lab2\Fletcher> java JavaServer
Ingrese el mensaje binario a enviar: 11010110
Mensaje con checksum: 11010110df86
PS C:\Users\TheKi\OneDrive - UVG\Semestre VIII\redes\redes_lab2\Fletcher> java JavaServer
Ingrese el mensaje binario a enviar: 101110101010
Mensaje con checksum: 1011101010100000e80a
PS C:\Users\TheKi\OneDrive - UVG\Semestre VIII\redes\redes_lab2\Fletcher> java JavaServer
Ingrese el mensaje binario a enviar: 1111000010101100
Mensaje con checksum: 1111000010101100e80b
```

Figura 1. Mensajes enviados desde el emisor con su respectiva codificación

```
PS C:\Users\TheKi\OneDrive - UVG\Semestre VIII\redes\redes_lab2\Fletcher> python .\pythonScript.py
Mensaje recibido: 11010110df86
El mensaje es válido.
PS C:\Users\TheKi\OneDrive - UVG\Semestre VIII\redes\redes_lab2\Fletcher> python .\pythonScript.py
Mensaje recibido: 1011101010100000e80a
El mensaje es válido.
PS C:\Users\TheKi\OneDrive - UVG\Semestre VIII\redes\redes_lab2\Fletcher> python .\pythonScript.py
Mensaje recibido: 1111000010101100e80b
El mensaje es válido.
```

Figura 2. Mensajes recibidos desde el receptor con su respectiva verificación de errores.

- **Código de Hamming**

```
PS C:\Users\erick\OneDrive\Documentos\UVG\8vo. Semestre\Redes\Lab2\Hamming> java JavaServer
Ingrese el mensaje a enviar: 11010110
Mensaje codificado: 001010100110
PS C:\Users\erick\OneDrive\Documentos\UVG\8vo. Semestre\Redes\Lab2\Hamming> java JavaServer
Ingrese el mensaje a enviar: 101110101010
Mensaje codificado: 01100110101010100
PS C:\Users\erick\OneDrive\Documentos\UVG\8vo. Semestre\Redes\Lab2\Hamming> java JavaServer
Ingrese el mensaje a enviar: 1111000010101100
Mensaje codificado: 001111100000101001100
PS C:\Users\erick\OneDrive\Documentos\UVG\8vo. Semestre\Redes\Lab2\Hamming>
```

Figura 3. Mensajes enviados desde el emisor con su respectiva codificación

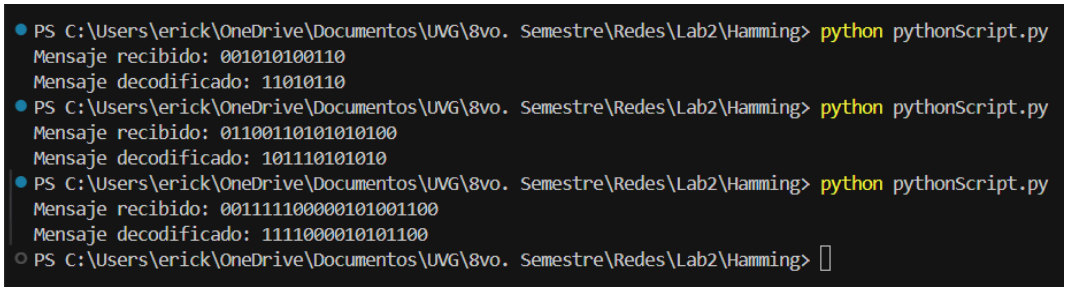

Figura 4. Mensajes recibidos desde el receptor con su respectiva verificación de errores.

Un error

Se modificó el código para que automáticamente se ingresaran errores en posiciones randoms. Se indica en consola la posición donde se inserta el error.

- **Fletcher checksum**

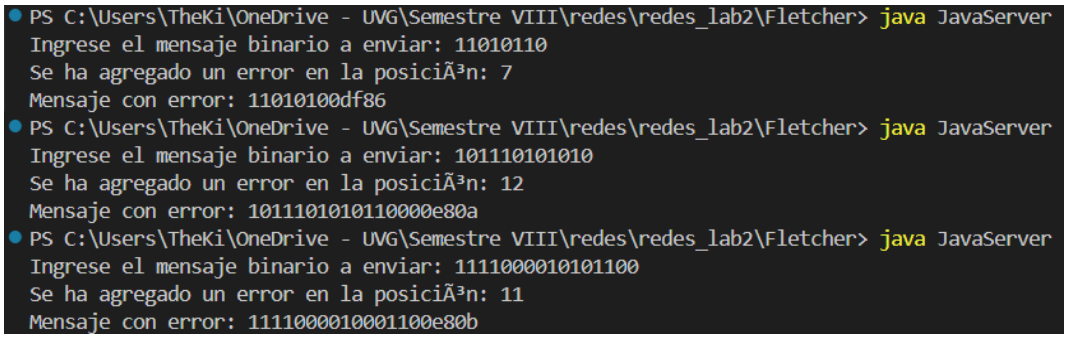

Figura 5. Mensajes enviados desde el emisor con un error con su respectiva codificación

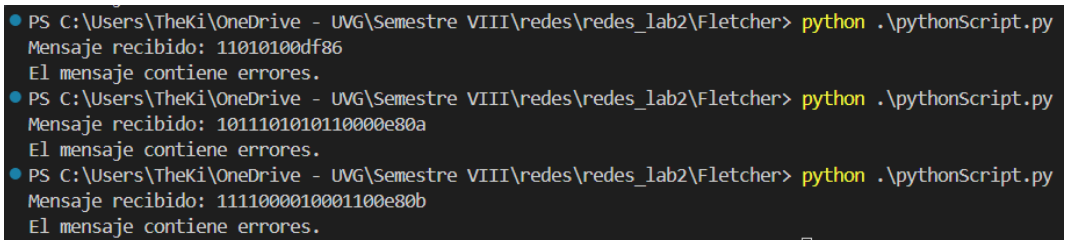

Figura 6. Mensajes recibidos desde el receptor con un error detectado.

- **Código de Hamming**

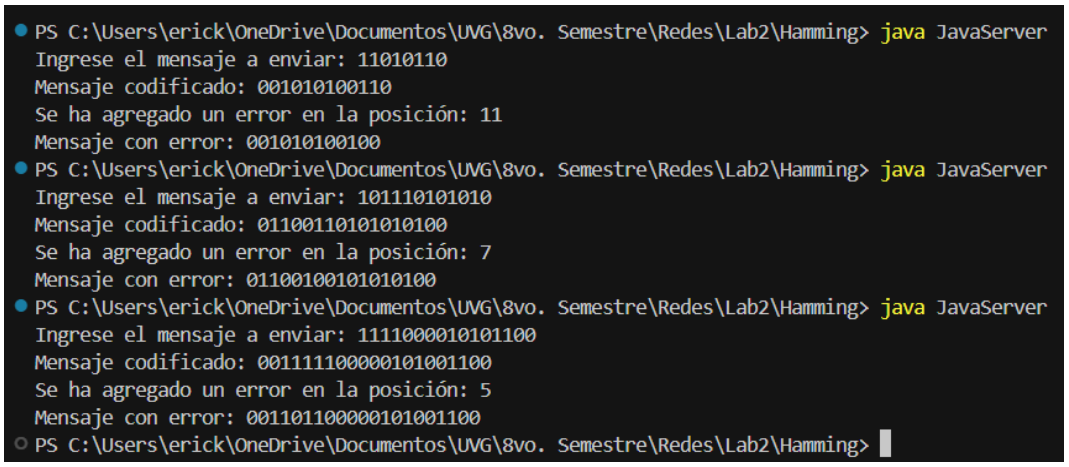

Figura 7. Mensajes enviados desde el emisor con un error con su respectiva codificación

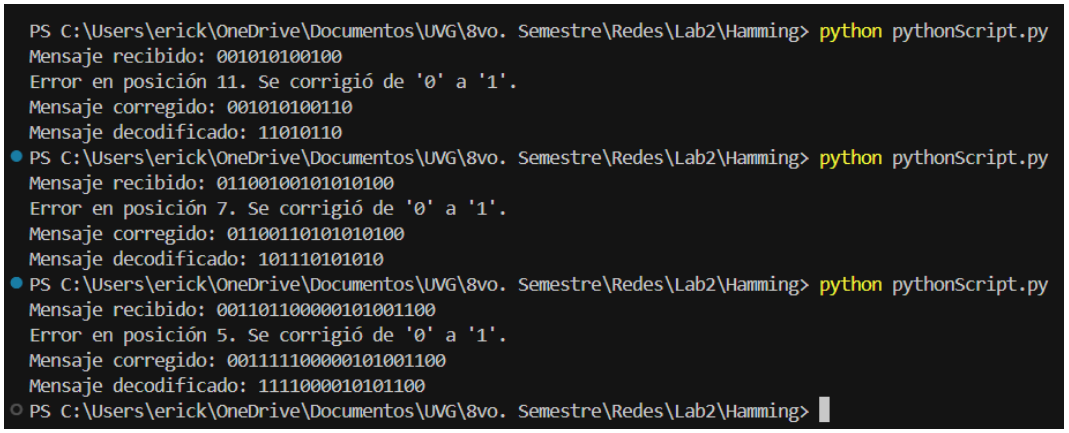

Figura 8. Mensajes recibidos desde el receptor con un error detectado.

Dos o más errores

- **Fletcher checksum**

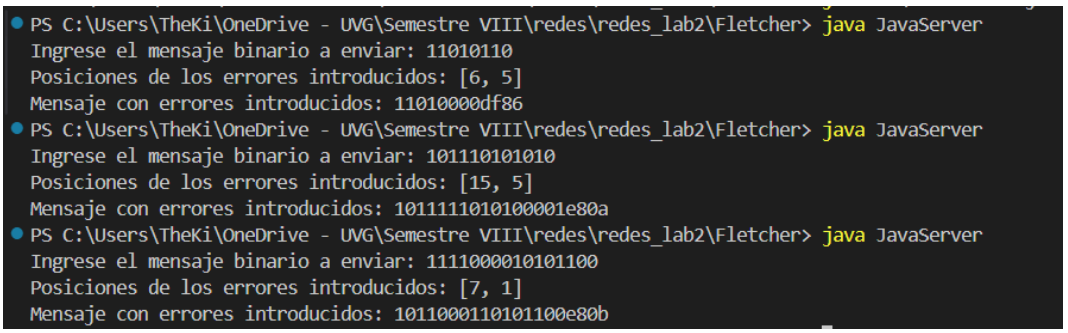

Figura 9. Mensajes enviados desde el emisor con dos errores con su respectiva codificación

```
PS C:\Users\TheKi\OneDrive - UVG\Semestre VIII\redes\redes_lab2\Fletcher> python .\pythonScript.py
Mensaje recibido: 11010000df86
El mensaje contiene errores.
PS C:\Users\TheKi\OneDrive - UVG\Semestre VIII\redes\redes_lab2\Fletcher> python .\pythonScript.py
Mensaje recibido: 10111110101000001e80a
El mensaje contiene errores.
PS C:\Users\TheKi\OneDrive - UVG\Semestre VIII\redes\redes_lab2\Fletcher> python .\pythonScript.py
Mensaje recibido: 10110001101011100e80b
El mensaje contiene errores.
```

Figura 10. Mensajes recibidos desde el receptor con dos errores detectados.

- **Código de Hamming**

```
PS C:\Users\erick\OneDrive\Documentos\UVG\8vo. Semestre\Redes\Lab2\Hamming> java JavaServer
Ingrese el mensaje a enviar: 11010110
Mensaje codificado: 001010100110
Se ha agregado un error en la posición: 11
Mensaje con error: 001010100100
Se ha agregado un error en la posición: 10
Mensaje con error: 001010100000
PS C:\Users\erick\OneDrive\Documentos\UVG\8vo. Semestre\Redes\Lab2\Hamming> java JavaServer
Ingrese el mensaje a enviar: 101110101010
Mensaje codificado: 01100110101010100
Se ha agregado un error en la posición: 7
Mensaje con error: 01100100101010100
Se ha agregado un error en la posición: 8
Mensaje con error: 01100101101010100
PS C:\Users\erick\OneDrive\Documentos\UVG\8vo. Semestre\Redes\Lab2\Hamming> java JavaServer
Ingrese el mensaje a enviar: 1111000010101100
Mensaje codificado: 001111100000101001100
Se ha agregado un error en la posición: 4
Mensaje con error: 001011100000101001100
Se ha agregado un error en la posición: 15
Mensaje con error: 001011100000100001100
```

Figura 11. Mensajes enviados desde el emisor con dos errores con su respectiva codificación

```
PS C:\Users\erick\OneDrive\Documentos\UVG\8vo. Semestre\Redes\Lab2\Hamming> python pythonScript.py
Mensaje recibido: 001010100000
Error en posición 1. Se corrigió de '0' a '1'.
Mensaje corregido: 101010100000
Mensaje decodificado: 11010000
PS C:\Users\erick\OneDrive\Documentos\UVG\8vo. Semestre\Redes\Lab2\Hamming> python pythonScript.py
Mensaje recibido: 01100101101010100
Error en posición 15. Se corrigió de '1' a '0'.
Mensaje corregido: 01100101101010000
Mensaje decodificado: 101010101000
PS C:\Users\erick\OneDrive\Documentos\UVG\8vo. Semestre\Redes\Lab2\Hamming> python pythonScript.py
Mensaje recibido: 001011100000100001100
Error en posición 11. Se corrigió de '0' a '1'.
Mensaje corregido: 001011100010100001100
Mensaje decodificado: 1111001010001100
```

Figura 12. Mensajes recibidos desde el receptor con dos errores detectados.

Discusión

¿Es posible manipular los bits de tal forma que el algoritmo seleccionado no sea capaz de detectar el error? ¿Por qué sí o por qué no? En caso afirmativo, demuestrelo con su implementación.

En el caso del algoritmo de detección Fletcher checksum se investigó y se encontró que se podían introducir errores compensados. Esto es por ejemplo, si en una posición se cambia un 0 a un 1 (aumentando el valor), y en otra se cambia un 1 a un 0 (disminuyendo el valor), el efecto neto no alteraría el checksum. Sin embargo, los errores introducidos han sido lo suficientemente significativos como para ser detectados por el receptor, y por ende, los errores han podido ser manejados.

Por su parte, el algoritmo de Hamming es capaz de corregir únicamente un error introducido en el mensaje original. Como se puede apreciar en las imágenes 11 y 12, al agregar más de un único error a la cadena, el algoritmo no es capaz de detectar correctamente ni siquiera la posición en la que hay error, puesto que se alteran las paridades de todos los bits de verificación. Al ser capaz de detectar solamente un error en la cadena, el algoritmo empieza a verificar la paridad en los bits de verificación, por lo que puede que detecte un error en la cadena o no, pero este error será distinto a los dos inyectados inicialmente. En conclusión, al ingresar más de un único error en Hamming, este podría detectarlo como un único error en alguna posición que puede o no ser una de las posiciones de error correctas, lo que producirá que al corregir y decodificar el mensaje, se detecte como un mensaje erróneo.

En base a las pruebas que realizó, ¿qué ventajas y desventajas posee cada algoritmo con respecto a los otros dos? Tome en cuenta complejidad, velocidad, redundancia (overhead), etc. Ejemplo: “En la implementación del bit de paridad par, me di cuenta que comparado con otros métodos, la redundancia es la mínima (1 bit extra). Otra ventaja es la facilidad de implementación y la velocidad de ejecución, ya que se puede obtener la paridad aplicando un XOR entre todos los bits. [...].”

Por parte del algoritmo de Fletcher checksum, posee simplicidad de implementación y de ejecución, lo cual puede ser de utilidad para sistemas con recursos limitados. A su vez, en términos de complejidad algorítmica es bastante baja ya que solo tiene sumas y operaciones módulo. Por último, como se discutió anteriormente, la introducción de errores forzados y que en teoría podrían pasar sin detectarse (errores compensados) se detectaron por parte del receptor, lo que le da alta validez al algoritmo. Sin embargo, la mayor desventaja que presenta en comparación al otro algoritmo es que solo detecta errores, sin poderlos corregir.

En el caso del algoritmo de Hamming, la principal ventaja es que no solo es capaz de detectar el error sino que aplica la corrección del mismo para poder ser interpretado de forma correcta por el receptor. A su vez, la inserción de bits de paridad añade complejidad al mensaje y evita que pueda ser fácilmente interceptado si un tercero no tiene en cuenta que se está utilizando este algoritmo dentro del sistema. Sin embargo, su mayor dificultad tiene que ver también con su mayor ventaja, puesto que no es capaz de detectar errores complejos que involucren más de un único bit. Además, la complejidad del algoritmo es mayor en comparación a Fletcher, por lo que se requiere una cantidad más amplia de recursos para ser llevado a cabo en situaciones más complejas.