



Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 Redes

Jorge Caballeros 20009

Alejandra Guzmán 20262

Laboratorio 2:

Esquema de detección y corrección de errores

1. Descripción

La práctica consistió en un laboratorio orientado a la comprensión e implementación de algoritmos de detección y corrección de errores en sistemas de comunicación. El objetivo principal era brindar a los estudiantes una experiencia práctica en el manejo de fallas que pueden ocurrir durante la transmisión de datos y la aplicación de soluciones para detectar y corregir dichos errores. Esta práctica fue realizada en parejas.

La implementación de los algoritmos se llevó a cabo en dos lenguajes de programación distintos para demostrar la comunicación de estos. En el lado del emisor, se solicitaba una trama en binario y se calculaba la información adicional requerida para la detección/corrección de errores, concatenándola con el mensaje original. Por otro lado, en el lado del receptor, se recibía el mensaje con la información adicional y se realizaba la detección/corrección de errores según el algoritmo correspondiente.

En nuestro caso implementamos hamming y fletcher, adjuntando una leve descripción de los mismos:

Fletcher Checksum: El algoritmo funciona dividiendo los datos en bloques de tamaño fijo, generalmente de 8 o 16 bits. Luego, calcula dos valores de suma de comprobación, Fletcher-16 y Fletcher-32, en función del contenido de estos bloques. Estos valores de suma de comprobación se añaden a los datos y se envían junto con ellos. Al recibir los datos, el receptor vuelve a calcular las sumas de verificación y las compara con las sumas de verificación recibidas. Si coinciden, es probable que los datos se hayan recibido correctamente. Si las sumas de verificación no coinciden, se detecta un error y se pueden tomar las medidas adecuadas, como solicitar la retransmisión de los datos.



Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 Redes

Hamming: En código Hamming, los datos se dividen en bits de datos y bits de paridad. Los bits de paridad se colocan cuidadosamente para cubrir bits específicos de los datos, formando una estructura similar a una matriz. Estos bits adicionales aumentan la redundancia de los datos y permiten la corrección de errores. Cuando se transmiten los datos, el receptor verifica los bits de paridad y los bits de datos para determinar si ocurrió algún error. Si se detecta un error de un solo bit, el receptor puede localizar y corregir el bit erróneo.

Resultados:

Hamming

1. 11001001

```
66 | socketio.run(app, host='192.168.1.33', port=4000)
67 |

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

found 0 vulnerabilities
PS C:\Users\alegu\OneDrive\Documentos\PruebaHamming> node server.js
Trama con información adicional recibida: 1010000101
Se ha detectado y corregido un error en el bit 8
Trama corregida: 1010000111
PS C:\Users\alegu\OneDrive\Documentos\PruebaHamming> node server.js
Trama con información adicional recibida: 110001000111
Se ha detectado y corregido un error en el bit 18
Trama corregida: 110001000110
[]

PS C:\Users\alegu\OneDrive\Documentos\PruebaHamming> python client.py
Ingrese una trama en binario: 100001
PS C:\Users\alegu\OneDrive\Documentos\PruebaHamming> python client.py
Ingrese una trama en binario: 11001001
```



Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 Redes

2. 01011011

```
Trama corregida: 1010000111
PS C:\Users\alegu\OneDrive\Documentos\PruebaHamming> node server.js
Trama con información adicional recibida: 110001000111
Se ha detectado y corregido un error en el bit 18
Trama corregida: 1100010001110
PS C:\Users\alegu\OneDrive\Documentos\PruebaHamming> node server.js
Trama con información adicional recibida: 010101010111
Se ha detectado y corregido un error en el bit 18
Trama corregida: 0101010101110

PS C:\Users\alegu\OneDrive\Documentos\PruebaHamming> python client.py
Ingrese una trama en binario: 100001
PS C:\Users\alegu\OneDrive\Documentos\PruebaHamming> python client.py
Ingrese una trama en binario: 11001001
PS C:\Users\alegu\OneDrive\Documentos\PruebaHamming> python client.py
Ingrese una trama en binario: 01011011
```

3. 101010101010

```
PS C:\Users\alegu\OneDrive\Documentos\PruebaHamming> node server.js
Trama con información adicional recibida: 11010101011010011
Se ha detectado y corregido un error en el bit 26
Trama corregida: 110101010110100110

PS C:\Users\alegu\OneDrive\Documentos\PruebaHamming> python client.py
Ingrese una trama en binario: 101010101010
```

Mensajes correctos:



Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 Redes

1.

```
PS C:\Users\alegu\OneDrive\Documents\PruebaHamming> node server.js
Trama con información adicional recibida: 00110011100
No se detectaron errores
Trama: 001100111000
█

PS C:\Users\alegu\OneDrive\Documents\PruebaHamming> python client.py
Ingrese una trama en binario: 1010011
PS C:\Users\alegu\OneDrive\Documents\PruebaHamming> python client.py
Ingrese una trama en binario: 1010011
█
```

2.

```
PS C:\Users\alegu\OneDrive\Documents\PruebaHamming> node server.js
Trama con información adicional recibida: 00110011100
No se detectaron errores
Trama: 001100111000
○ PS C:\Users\alegu\OneDrive\Documents\PruebaHamming> node server.js
Trama con información adicional recibida: 11110110101
No se detectaron errores
Trama: 11110111101
█

PS C:\Users\alegu\OneDrive\Documents\PruebaHamming> python client.py
Ingrese una trama en binario: 1010011
PS C:\Users\alegu\OneDrive\Documents\PruebaHamming> python client.py
Ingrese una trama en binario: 1010011
○ PS C:\Users\alegu\OneDrive\Documents\PruebaHamming> python client.py
Ingrese una trama en binario: 1111011
█
```

3.

```
PS C:\Users\alegu\OneDrive\Documents\PruebaHamming> node server.js
Trama con información adicional recibida: 01110101001
No se detectaron errores
Trama: 01111101001
█

PS C:\Users\alegu\OneDrive\Documents\PruebaHamming> python client.py
Ingrese una trama en binario: 1111011
PS C:\Users\alegu\OneDrive\Documents\PruebaHamming> python client.py
Ingrese una trama en binario: 0100100
█
```

Fletcher:

Mensajes correctos



Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 Redes

1. 01001000 01001111 01001100 01000001

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL AZURE
launcher: '54832' '-' 'C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes\Fletcher\server.py'
Servidor corriendo en *:3000
Un cliente se ha conectado
El cliente se ha desconectado
Un cliente se ha conectado
('Mensaje recibido: , 01001000 01001111 01001100 01000001\nChecksum recibido: , 59429\nChecksum calculado: , 59429\nMensaje correcto\nMensaje en ASCII:HOLA',)
PS C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes\Fletcher> node .\sender.js
Conectado al servidor!
> HOLA
Mensaje enviado: {"message":"01001000 01001111 01001100 01000001","fletcherCheck":59429}
>
```

Mensajes adaptados y con errores

2. 01100010 01110101 01100101 01101110 01100001 01110011

```
PS C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes> c:; cd 'c:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes'; & 'C:\Program Files\Python311\python.exe' 'c:\Users\jorge\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '55241' '-.-' 'C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes\Fletcher\server.py'
Servidor corriendo en *:3000
Un cliente se ha conectado
('Mensaje recibido: , 01100010 01110101 01100101 01101110 01100001 01110011\nChecksum recibido: , 45184\nChecksum calculado: , 45184\nMensaje correcto\nMensaje en ASCII:buenas',)
PS C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes\Fletcher> node .\sender.js
Conectado al servidor!
> buenas
Mensaje enviado: {"message":"01100010 01110101 01100101 01101110 01100001 01110011","fletcherCheck":45184}
>
```

3. 01110111 01100101 01101110 01100001 01110011

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL AZURE
PS C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes> c:; cd 'c:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes'; & 'C:\Program Files\Python311\python.exe' 'c:\Users\jorge\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '55287' '-.-' 'C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes\Fletcher\server.py'
Servidor corriendo en *:3000
Un cliente se ha conectado
('Mensaje recibido: , 01110111 01100101 01101110 01100001 01110011\nChecksum recibido: , 27680\nChecksum calculado: , 27680\nMensaje correcto\nMensaje en ASCII:wenas',)
PS C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes\Fletcher> node .\sender.js
Conectado al servidor!
> wenas
Mensaje enviado: {"message":"01110111 01100101 01101110 01100001 01110011","fletcherCheck":27680}
>
```



Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 Redes

Mensajes alterados/incorrectos :

Al introducir show_error en la consola se nos dan mensajes con error aleatorios.

1.

Original:01110011 01101000 01101111 01110111 01011111 01100101 01110010
01110010 01101111 01110010

Erróneo: 01110011 01101000 01101111 01110111 01011111 01100101 01110010
01110010 01101111 00110010

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL AZURE
Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes'; & 'C:\Program Files\Python311\python.exe' 'c:\Users\jorge\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '55843' '- -' 'C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes\Fletcher\server.py'
Servidor corriendo en *:3000
Un cliente se ha conectado
Error en el mensaje
Mensaje recibido: , 01110011 01101000 01101111 01110111 01011111 01100101 01110010
10 01110010 01101111 00110010
Checksum recibido: , 38734
Checksum calculado: , 2286
Mensaje incorrecto
Mensaje en ASCII:show_error2
█
```

```
PS C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes\Fletcher> node .\sender.js
Conectado al servidor!
> show error
Mensaje enviado: {"message":"01110011 01101000 01101111 01110111 01011111 01100101 01110010 01101111 00110010","fletcher":38734}
> █
```

2.

Original:01110011 01101000 01101111 01110111 01011111 01100101 01110010
01110010 01101111 01110010

Erróneo: 01110011 01101000 01101111 11110111 01011111 01100101 01110010
01110010 01101111 01110010

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL AZURE
Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes'; & 'C:\Program Files\Python311\python.exe' 'c:\Users\jorge\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '55914' '- -' 'C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes\Fletcher\server.py'
Servidor corriendo en *:3000
Un cliente se ha conectado
Error en el mensaje
Mensaje recibido: , 01110011 01101000 01101111 11110111 01011111 01100101 01110010
10 01110010 01101111 01110010
Checksum recibido: , 38734
Checksum calculado: , 7118
Mensaje incorrecto
Mensaje en ASCII:show_error
█
```

```
PS C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes\Fletcher> node .\sender.js
Conectado al servidor!
> show error
Mensaje enviado: {"message":"01110011 01101000 01101111 01110111 01011111 01100101 01110010 01101111 01110010","fletcher":38734}
> █
```



Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 Redes

3. **Original:**01110011 01101000 01101111 01110111 01011111 01100101 01110010
01110010 01101111 01110010
Erróneo: 01110011 01101000 01101111 01110111 01011111 01100101 01110110
01110010 01101111 01110010

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL AZURE
Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes'; & 'C:\Program Files\Python311\python.exe' 'c:\Users\jorge\.vscode\extensions\ms-python.python-2023.14.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '55994' '- -' 'C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes\Fletcher\server.py'
Servidor corriendo en *:3000
Un cliente se ha conectado
Error en el mensaje
Mensaje recibido: , 01110011 01101000 01101111 01110111 01011111 01100101 01110110 01110010 01101111 01110010
Checksum recibido: , 38734
Checksum calculado: , 42834
Mensaje incorrecto
Mensaje en ASCII:show_error

PS C:\Users\jorge\OneDrive\Documentos\GitHub\Laboratorio-2.1-Redes\Fletcher> node .\sender.js
Conectado al servidor!
> show_error
Mensaje enviado: {"message":"01110011 01101000 01101111 01110111 01011111 01100101 01110010 01101111 01110010","fletcher":38734}
>
```



Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 Redes

COMENTARIO GRUPAL

El código de Hamming detecta y corrige un solo bit de error, pero no puede corregir más de dos bits de error. Por otro lado, el código de Fletcher detecta errores en cualquier posición, pero no los corrige. Ambos métodos pueden generar falsos positivos o negativos y requieren cálculos adicionales que aumentan el coste computacional y el ancho de banda necesario para la transmisión de datos. La elección entre ellos depende de los requisitos específicos del sistema. Como grupo fue una actividad interesante aunque algo desafiante porque no entendíamos de todo el funcionamiento de los algoritmos y entender en su totalidad como funcionaban y como el ruido afectaba a los mismos.