

Universidad del Valle de Guatemala  
Facultad de Ingeniería



Redes  
Laboratorio #2

Esquemas de detección y corrección de errores  
Alejandro José Gómez Hernández 20347  
Juan Carlos Baján Castro 20109

### Descripción de la práctica:

La comunicación de información en la red es la base de todo el internet, cualquier modificación de los mensajes pueden resultar en incongruencias en la información y errores de comunicación. Para eso se han desarrollado gran cantidad de algoritmos que tienen como objetivo reducir, resolver y detectar estos inconvenientes. En el presente informe se presentan dos algoritmos, uno pretende resolver los errores en el mensaje y el otro pretende detectar dichos errores.

#### **Fletcher checksum (Algoritmo de detección de errores):**

Es un algoritmo de detección de errores que se utiliza para garantizar que los datos no se hayan dañado o alterado durante la transmisión. Es más eficiente que el checksum clásico porque reduce la cantidad de operaciones aritméticas necesarias.

El algoritmo básico del checksum de Fletcher de 16 bits es el siguiente:

1. El remitente divide los datos en bloques de 16 bits. Si los datos no se pueden dividir en bloques completos de 16 bits, se añaden ceros al final para completar el último bloque.
2. Se inicializan dos acumuladores a cero, que se llamarán suma1 y suma2.
3. Se añade cada bloque de 16 bits a suma1.
4. Después de cada adición a suma1, se añade suma1 a suma2.
5. Ambas sumas se reducen al módulo 65535.
6. Finalmente, suma1 y suma2 se combinan para formar el checksum. suma1 es la parte menos significativa del checksum y suma2 es la parte más significativa.

El receptor puede utilizar el mismo proceso para calcular el checksum del mensaje recibido. Si el checksum recibido coincide con el calculado, el mensaje se considera válido. Si no coinciden, el mensaje se considera dañado.

#### **Corrección de Errores:**

La corrección de errores se refiere a los métodos utilizados para detectar y corregir errores que pueden haber ocurrido durante la transmisión o el almacenamiento de datos. A diferencia de la simple detección de errores, como se describe en el algoritmo de Fletcher checksum, la corrección de errores puede identificar y corregir automáticamente los errores, sin necesidad de retransmitir los datos.

## **Código de Hamming:**

El código de Hamming es un código de corrección de errores. Desarrollado por Richard Hamming, es utilizado para detectar y corregir errores en los datos transmitidos. El funcionamiento de este, lo podemos definir de la siguiente forma:

1. Codificación: En Hamming (n, m), donde n es nuestra longitud total del mensaje codificado, se deben cumplir con  $m + r + 1 \leq 2^r$ . Ahora bien, para el proceso de codificación, se añaden r bits de paridad en posiciones específicas del mensaje original para codificarlo. Estos bits de paridad se calculan de manera que cada bit de paridad se relaciona con un subconjunto de bits de datos.
2. Transmisión: El mensaje codificado, que ahora cuenta con n bits, se transmite al receptor.
3. Detección de Errores: El receptor examina los bits de paridad para determinar si hay errores en la transmisión. Si los bits de paridad no coinciden con los bits de datos, se identifica un error.
4. Corrección de Errores: En caso se llegue a detectar un error, el código de Hamming identifica el bit que está dañado y lo arregla. Ahora bien, si solo hay un solo bit erróneo, el código de Hamming puede corregir el error sin que tenga que volver a transmitir los datos.
5. Decodificación: Se eliminan bits de paridad, y se obtiene el mensaje original.

## **Resultados:**

### **Fletcher checksum:**

Emisor:

tramas =

[

"110101100000101000101001",

"11010110000001001",

"100000101000101001",

"11010101001"

]

Output Emisor:

```
Trama enviada: 1101011000001010001010010000101010001010
Trama enviada: 110101100000010010000011101001011
Trama enviada: 1000001010001010010000011000110011
Trama enviada: 110101010010000011000101000
```

Fotografía 1 (resultados emisor)

Receptor:

tramas\_correctas (Tramas enviadas por el emisor) =

```
[
    "1101011000001010001010010000101010001010",
    "110101100000010010000011101001011", "1000001010001010010000011000110011",
    "110101010010000011000101000"
]
```

tramas\_errores =

```
[
    "1101011010001010001010010000101010011010",
    "110101100000010010000011101001010", "1000011010001010010000011000110011",
    "110101010110000011000101000"
]
```

Output Receptor

```
Tramas correctas
Mensaje recibido sin errores: 110101100000101000101001
Mensaje recibido sin errores: 11010110000001001
Mensaje recibido sin errores: 100000101000101001
Mensaje recibido sin errores: 11010101001
Tramas modificadas
La trama se descarta por detectar errores.
La trama se descarta por detectar errores.
La trama se descarta por detectar errores.
La trama se descarta por detectar errores.
```

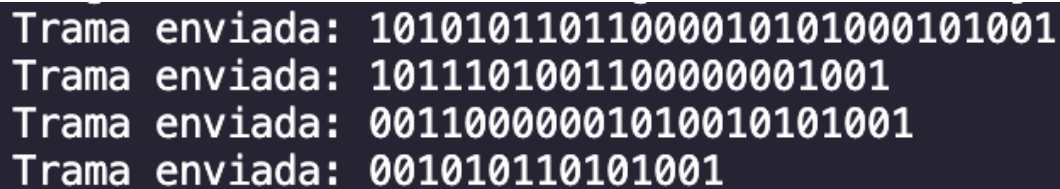
Fotografía 2 (resultados receptor)

### Código de Hamming:

Emisor:

tramas =

```
[  
"110101100000101000101001",  
"11010110000001001",  
"100000101000101001",  
"11010101001"  
]
```



```
Trama enviada: 10101011011000010101000101001  
Trama enviada: 1011101001100000001001  
Trama enviada: 00110000001010010101001  
Trama enviada: 001010110101001
```

Fotografía 3 (resultados emisor)

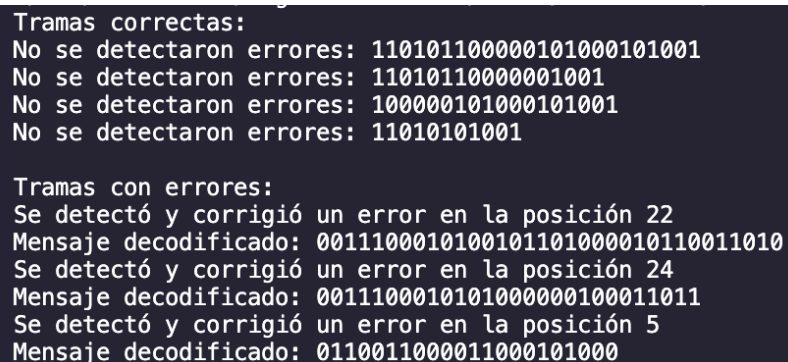
Receptor:

tramas\_correctas = [

```
"10101011011000010101000101001",  
"1011101001100000001001",  
"00110000001010010101001",  
"001010110101001",  
]
```

tramas\_errores =

```
[  
"11010110100010100010100100001010011010",  
"110101100000010010000011101001010", "11010101010000011000101000"  
]
```



```
Tramas correctas:  
No se detectaron errores: 110101100000101000101001  
No se detectaron errores: 11010110000001001  
No se detectaron errores: 100000101000101001  
No se detectaron errores: 11010101001  
  
Tramas con errores:  
Se detectó y corrigió un error en la posición 22  
Mensaje decodificado: 0011100010100101101000010110011010  
Se detectó y corrigió un error en la posición 24  
Mensaje decodificado: 0011100010101000000100011011  
Se detectó y corrigió un error en la posición 5  
Mensaje decodificado: 0110011000011000101000
```

Discusión:

En el caso del algoritmo de Fletcher los errores fueron detectados exitosamente pues cada cambio representó influencia suficiente para que se encontraran las fallas. Todas las tramas procesadas se utilizaron de forma congruente tanto en el emisor como el receptor para poder entender el flujo de la información y los posibles cambios. Como se puede observar en la fotografía 1, luego de utilizar el algoritmo para procesar el mensaje inicial, el resultado fue replicado en el receptor. En la fotografía 2 se puede observar que las tramas sin haber obtenido ningún tipo de modificación resultaron en éxito (mensaje recibido sin errores) demostrando la efectividad del algoritmo. Por otro lado, luego de haberse modificado las tramas se puede observar que los errores fueron detectados exitosamente y se descarta. De esta manera se demuestra que el algoritmo de Fletcher Checksum es eficiente. No se detectaron modificaciones que no pudieran ser procesadas por el algoritmo.

En contraste con el algoritmo de Fletcher, el código de Hamming proporciona una funcionalidad adicional de no solo detectar errores sino también corregirlos. La técnica de Hamming introduce bits de paridad en posiciones específicas dentro de la trama, lo que permite identificar y corregir un error de un solo bit en la trama recibida. Esta capacidad de corrección añade una capa adicional de robustez a la transmisión de datos. En la implementación discutida, se pudo observar que los errores en posiciones específicas fueron detectados y corregidos con éxito. Sin embargo, el proceso requiere una coordinación meticulosa entre el emisor y el receptor, ya que los bits de paridad deben ser manejados de manera idéntica en ambos extremos de la comunicación.

Para el algoritmo de Fletcher, se recomienda tener bastante presente la aplicación consciente de su capacidad para detectar errores. Aunque es eficiente en la detección de cambios y fallas en la trama, no proporciona la corrección de errores. Por lo tanto, es adecuado en escenarios donde la detección es suficiente y la corrección puede ser manejada de otra manera. El uso de Fletcher en sistemas donde la sobrecarga debe minimizarse podría ser una buena aplicación de este algoritmo, debido a que demostró ser bastante simple y eficiente.

Ahora bien, en cuanto al código de Hamming, su capacidad para corregir errores de un solo bit lo hace valioso en contextos donde la integridad de los datos es crítica. Sin embargo, es importante destacar que su implementación exige una coordinación y comprensión meticulosas de los bits de paridad y cómo se colocan dentro de la trama. Debido a que pueden ocurrir varios errores si no se tienen los fundamentos de los bits de paridad, así como también las definiciones claras de los algoritmos. Se recomienda su uso en aplicaciones donde la corrección de errores es vital y donde los recursos para su implementación precisa están disponibles. La elección entre Fletcher y Hamming deberá considerarse

cuidadosamente, ponderando las necesidades específicas de detección frente a corrección y los recursos disponibles para la implementación.

También, como ejercicio de análisis, como equipo se tomó la decisión de plantearse casos en los que no fuera posible detectar errores. Para esto, se planteó distintos escenarios para cada algoritmo. En el caso de Fletcher, por ejemplo en caso existan cambios compensatorios en la trama que no llegasen a alterar el resultado del checksum, podría ser que el error pase desapercibido. Sin embargo, durante el proceso de programación y experimentación, no se encontró este escenario. Ahora, respecto a Hamming, podría llegar a existir el escenario que ocurran múltiples errores en la trama que llegasen a afectar más de un bit y no se pueda corregir o detectar, en caso no se haya programado de una manera robusta y defensiva. Sin embargo, este no fue el caso encontrado durante el proceso de programación y experimentación.

#### Comentario grupal:

Alejandro Gómez, conocido como "Ale" llevó a cabo el algoritmo de Hamming. Cabe destacar que es un método intrigante que no sólo detecta, sino también corrige los errores de un solo bit. Sin embargo, la implementación inicial presentó ciertos desafíos, específicamente en la identificación correcta de la posición del error. Fue un detalle pequeño, pero crucial, similar a una nota desafinada en una pieza musical, que nos llevó a revisar y ajustar el algoritmo. Tras una cuidadosa revisión, se hicieron ajustes para mejorar la precisión de la detección y corrección de errores. Esto involucró una comprensión más profunda de los cálculos de los bits de paridad y cómo estos interactúan con las tramas de datos.

Juan Carlos Baján, conocido como "Juanca", llevó a cabo el algoritmo de Fletcher con gran destreza. Este algoritmo, aunque no corrige errores, ha demostrado ser un detector de errores muy completo. El enfoque meticuloso por parte de Juanca, y la investigación realizada respecto al algoritmo fue fundamental para la comprensión de cómo diferentes métodos pueden abordar el mismo problema.

Al comparar y contrastar estos dos métodos, pudimos apreciar cómo Hamming y Fletcher cumplen diferentes roles en la transmisión de datos. Ambos tienen su lugar en diferentes contextos y aplicaciones, algo similar a cómo diferentes instrumentos se seleccionan para lograr distintos tonos en la música.

Fue una experiencia enriquecedora que ha ampliado nuestra comprensión de cómo se pueden detectar y manejar los errores en la transmisión de datos. La colaboración, la experimentación y las iteraciones nos han llevado a una implementación efectiva y precisa.

### Conclusiones:

1. El algoritmo de Fletcher demostró ser eficiente en la detección de errores en la transmisión de información, ya que todos los cambios realizados en las tramas fueron identificados exitosamente. Esto indica que el algoritmo es sensible a las variaciones en las tramas, lo que contribuye a su utilidad en la detección de errores.
2. El código de Hamming va un paso más allá y permite la corrección de errores de un solo bit, lo cual puede ser de suma importancia en ambientes donde la integridad de datos sea crucial.
3. El uso congruente del algoritmo tanto en el emisor como en el receptor permitió un procesamiento efectivo de las tramas y una precisa identificación de los cambios. Esto subraya la importancia de la consistencia en la utilización del algoritmo en ambos extremos de la transmisión de datos.
4. En los casos en los que las tramas no fueron modificadas, el algoritmo de Fletcher confirmó la integridad de los datos, mostrando así su efectividad no sólo en la detección de errores sino también en la confirmación de la correcta transmisión de la información. En los casos en que las tramas se modificaron, los errores se identificaron y se descartaron las tramas modificadas, lo que confirma la robustez del algoritmo de Fletcher en la detección y manejo de errores.
5. Los dos algoritmos presentados tienen distintas aplicaciones en distintos contextos, y elegir entre ambos depende de los requisitos específicos de detección y/o corrección de errores en una situación dada.
6. La implementación precisa y coherente del código de Hamming en el emisor y el receptor es de suma importancia para que funcione como se espera. Esto puede añadir complejidad en comparación con algoritmos de detección de errores más sencillos.



## Referencias:

Fast Computation of Fletcher Checksums. (2016). Intel.  
<https://www.intel.com/content/www/us/en/developer/articles/technical/fast-computation-of-fletcher-checksums.html>

Wikipedia Contributors. (2023, July 1). Fletcher's checksum. Wikipedia; Wikimedia Foundation.  
[https://en.wikipedia.org/wiki/Fletcher%27s\\_checksum](https://en.wikipedia.org/wiki/Fletcher%27s_checksum)

Hamming, R. W. (1950). "Error Detecting and Error Correcting Codes." Bell System Technical Journal, 29(2), 147-160.

Morelos-Zaragoza, R. (2006). "The Art of Error Correcting Coding." Wiley.