Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
Cristian Laynez - 201281
Mario de León - 19019

Esquemas de detección y corrección de errores

La comunicación en las redes y la tecnología es esencial para la vida humana y la evolución, sin duda esto es esencial para el correcto funcionamiento y exitoso paso de datos del WIFI, internet, cpu, memoria, disquetes, CDs, DVDs, BRs, etc. Esto se logra por medio de una transmisión digital sobre un canal real de los niveles electrónicos de la señal. Suele suceder que a través de la comunicación exista ruido y errores el cuál puede llegar a cambiar la interpretación de la información enviada por medio de los bits. Por estos motivos existe mucha diversidad de protocolos de detección y corrección de errores que nos permiten no solamente sincronizar los mensajes sino también ordenar las tramas que se van enviando, definir los procesos a llevar a cabo y aparte identificar errores y corregirse en ese momento donde ha sido detectado.

Algoritmos utilizados:

- Código de Hamming:
  "El código de Hamming es un código lineal para la detección de errores que puede detectar hasta dos errores de bits simultáneos y es capaz de corregir errores de un solo bit. Se garantiza una comunicación confiable si la distancia entre el transmisor y el receptor es menor o igual a uno. El código de Hamming fue inventado por Richard Hamming en 1950. El método es útil para un cambio de un solo bit, que es más probable que dos o más cambios de bit." (un, 2023)

- Fletcher checksum:
  "El checksum de Fletcher es un algoritmo para implementar una suma de comprobación que permita detectar un error en alguno de los campos de un mensaje entre su emisión y recepción. Es muy común en protocolos de mensajería de bajo nivel, la máquina que genera el mensaje añade al final de la trama que envía otro campo con el CRC calculado con los bytes del cuerpo del mensaje. El receptor a su vez cuando recibe la trama calcula usando el mismo algoritmo el CRC del cuerpo del mensaje y lo compara con el campo que ha añadido el emisor, evidentemente si los resultados no son el mismo con toda probabilidad la trama se ha corrompido en algún momento de la transmisión." (Moran, 2023)

Resultados:

- ➢ Código de Hamming:
  - Emisor (Hecho en C++)
    Prueba con "1011001":

```
[clay@fedora Redes_Lab02]$ ./hamming/hamming 1011001
Input: 1011001
Output: 10101001110
r: 4
```

Prueba con “1001”:

```
[clay@fedora Redes_Lab02]$ ./hamming/hamming 1001
Input: 1001
Output: 1001100
r: 3
```

Prueba con “10110011”:

```
crislay@crislay-750-514:~/Documents/GitHub/Redes_Lab02$ ./hamming/hamming 10110011
Input: 10110011
Output: 101110010101
r: 4
```

- Receptor (Hecho en Java)

  Prueba con output “10101001110” obtenido de “1011001” (Sin errores)

```
[clay@fedora Redes_Lab02]$ java hamming/Hamming 10101001110 10101001110
Input: 10101001110, r = 4
Output: Todo bien 10101001110
Original: 1011001
CORRECTO
```

  Prueba con output “10101001110” obtenido de “1011001” (Con errores cambiando el 4to bit)

```
[clay@fedora Redes_Lab02]$ java hamming/Hamming 10111001110 10101001110
Input: 10111001110, r = 4
Output: Errores encontrados
10111001110
   |
Trama de bits Arreglado: 10101001110
Original: 1011001
CORRECTO
```

  Prueba con output “1001100” obtenido de “1001” (Sin errores)

```
[clay@fedora Redes_Lab02]$ java hamming/Hamming 1001100 1001100
Input: 1001100, r = 3
Output: Todo bien 1001100
Original: 1001
CORRECTO
```

  Prueba con output “1001100” obtenido de “1001” (Con errores cambiando el 4to digito)

```
[clay@fedora Redes_Lab02]$ java hamming/Hamming 1000100 1001100
Input: 1000100, r = 3
Output: Errores encontrados
1000100
   |
Trama de bits Arreglado: 1001100
Original: 1001
CORRECTO
```

Prueba con output “101110010101” obtenido de “10110011” (Cuando todo esta en orden)

```
crislay@crislay-750-514:~/Documents/GitHub/Redes_Lab02$ java hamming/Hamming 101110010101 101110010101
Input: 101110010101
Todo ok
Output: 101110010101
CORRECTO
```

Prueba con output “101110010101” obtenido de “10110011” (Cuando un bit falla)

```
crislay@crislay-750-514:~/Documents/GitHub/Redes_Lab02$ java hamming/Hamming 101110010100 101110010101
Input: 101110010100
Errores encontrados
101110010100
           |
Trama de bits Arreglado:
Output: 101110010101
CORRECTO
```

Prueba con output “101110010101” obtenido de “10110011” (Error cuando hay más de un bit)

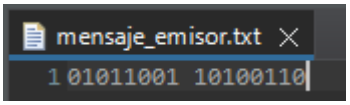
```
crislay@crislay-750-514:~/Documents/GitHub/Redes_Lab02$ java hamming/Hamming 101110010110 101110010101
Input: 101110010110
Errores encontrados
101110010110
           |
Trama de bits Arreglado:
Output: 101110010010
INCORRECTO
```

Prueba con output “101110010101” obtenido de “10110011” (Error cuando hay demasiados bits y no es capaz de corregirlo)

```
crislay@crislay-750-514:~/Documents/GitHub/Redes_Lab02$ java hamming/Hamming 101100011110 101110010101
Input: 101100011110
Este error no es posible de corregir
Output: -1
INCORRECTO
```

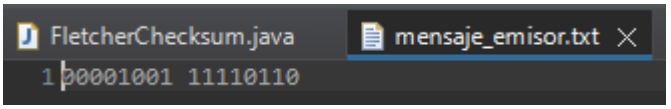
- ➢ Fletcher checksum:
  - Emisor (En Java):
    Prueba con “1011001”:

```
***EMISOR***
Bloques Concatenados: 01011001
Bloque 0: 01011001
Checksum Fletcher: 10100110
Mensaje Final: 0101100110100110
El mensaje final se ha escrito en el archivo: mensaje_emisor.txt
```



Prueba con “1001”:

```
***EMISOR***
Bloques Concatenados: 00001001
Bloque 0: 00001001
Checksum Fletcher: 11110110
Mensaje Final: 0000100111110110
El mensaje final se ha escrito en el archivo: mensaje_emisor.txt
```



- Receptor (En C++)

  Prueba con output de “1011001” (sin errores):

```
***RECEPTOR***
Checksum recibido: 10100110
Mensaje original: 01011001
El mensaje es valido.
Mensaje Original: 01011001
```

  Prueba con output obtenido de “1011001” (Con errores cambiando el 4to bit):

```
***RECEPTOR***
Checksum recibido: 10100110
Mensaje original: 01010001
El mensaje ha sido alterado durante la transmisión.
```

  Prueba con output obtenido de “1001” (sin errores):

```
***RECEPTOR***
Checksum recibido: 11110110
Mensaje original: 00001001
El mensaje es valido.
Mensaje Original: 00001001
```

  Prueba con output obtenido de “1001” (Con errores cambiando el 4to dígito):

```
***RECEPTOR***
Checksum recibido: 11110110
Mensaje original: 00000001
El mensaje ha sido alterado durante la transmisión.
```

Discusión:

Para el input "1011001":

El emisor calculó el checksum de la trama "1011001" y envió la trama y el checksum al receptor utilizando el algoritmo Fletcher Checksum. El receptor verificó el checksum al recibir la trama y no encontró errores. Esto demuestra que la trama no se modificó durante la transmisión y que la suma de comprobación proporcionada por el emisor coincide con el resultado calculado por el receptor. Como resultado, se puede deducir que la trama es confiable y no contiene errores.

Ahora cambiando el 4to bit, el mensaje se mostró modificado e invalido por el receptor. El receptor verificó el checksum al recibir la trama, pero esta vez encontró un error en la trama. Los comprobantes no coincidieron, lo que indica que la trama se modificó durante la transmisión. El emisor y el receptor llegaron a la conclusión de que la trama estaba dañada y no era válida. El algoritmo Fletcher Checksum sólo se encarga de detectar errores, no corregirlos. Pudo detectar correctamente el error dado en el mensaje del emisor.

Para el input "1001":

El emisor calculó la suma de comprobación de la línea "1001" utilizando el algoritmo Fletcher Checksum y la envió al receptor junto con la suma de comprobación. El receptor verificó el checksum al recibir la trama, coincidiendo con la otra trama. Los comprobantes coincidieron, lo que indica que la trama no se modificó durante la transmisión. El emisor y el receptor llegaron a la conclusión de que la trama era válida.

Al modificar el 4to bit, la trama dio errores y se mostró que había sido alterada durante la transmisión. Pudo detectar correctamente el error dado en el mensaje del emisor.

Por el otro lado tenemos el algoritmo de Hamming en donde no solo se detectan los errores, sino que este es capaz de detectar los errores en base si el bit cambiado fue uno de paridad, en cambio si se detectan más errores en la paridad entonces ahí no siempre es capaz de corregir dichos errores porque se arregla en base a los bits. Si son demasiados errores los que tiene y no tienen que ver con los bits de paridad entonces fijo se habrá mandado un mensaje de forma incorrecta. Si este cuenta con un error y coincide con uno de los bits de paridad entonces este será capaz de corregirlo. Cuando se convierte de bits a decimales a la hora de detectar errores se obtiene el decimal en donde se encuentra dicho error, pero cuando hay mas de dos errores este ya no es capaz de arreglar el error como se espera, cuando se presenta dicho escenario es cuando no se puede solucionar ni corregir la trama.

Comentario grupal:

Sin duda la implementación de estos algoritmos fue un desafío muy interesante, ya que no solamente nos desafía a verificar que los algoritmos funcionen correctamente, sino también si se encuentra uno o más errores que estos sean capaces de corregir los bits erróneos sí es que eso es posible. Se entiende el porqué existen varios algoritmos y protocolos para transmitir información de un lugar a otro. En general, ambos algoritmos (Fletcher Checksum y Hamming) fueron efectivos para detectar errores en tramas de bits. Sin embargo, la aplicación en particular y el nivel de confiabilidad y corrección de errores requeridos determinarán el algoritmo más adecuado.

Conclusiones:

- La manera de mandar información es crucial para la comunicación correcta entre los medios para evitar mandar mensajes incorrectos o que existan incoherencias.
- Si existen demasiados errores en una trama el mensaje que se recibirá será completamente diferente al original.
- El algoritmo Fletcher Checksum fue efectivo en la detección de errores en las tramas de bits. Al comparar los checksums creados por el emisor y el receptor, pudo determinar correctamente cuando una trama fue alterada durante la transmisión.
- Aunque tenía ciertas limitaciones, el algoritmo de Hamming también tenía la capacidad de detectar errores. Debido a la distribución particular de bits de paridad, pudo detectar errores en algunas tramas, pero no en todas. Esto demuestra que su eficacia depende de dónde se encuentren los bits dañados.
- Ambos algoritmos hacen hincapié en la necesidad de mantener la integridad de los datos durante la transmisión. Los errores en la trama de bits pueden tener efectos significativos, especialmente en aplicaciones que requieren precisión.

Enlace Repositorio:
https://github.com/CrisLayB/Redes_Lab02.git

Citas y Referencias:

un. (2023). ¿Qué es un código de hamming? - definición de techopedia - Desarrollo 2023. Icy Science. https://es.theastrologypage.com/hamming-code

Moran, Marlon. (2023).Check Sum. https://es.scribd.com/document/420038536/Check-Sum#

https://www.geeksforgeeks.org/hamming-code-in-computer-network/