UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería
Departamento de Ciencias de la Computación

Catedrático: Vinicio Paz

Laboratorio 2
Esquemas de detección y corrección de errores

Antonio Reyes 17273

Sección 10

Descripción de la práctica

La práctica del laboratorio consistió en 3 partes, Desarrollo de arquitectura para envío/recepción de mensajes, implementación de algoritmos y la prueba de dichos algoritmos. **Ilustración 1** presenta la arquitectura propuesta para la comunicación entre el emisor y receptor, el emisor se encarga de crear una cadena que debe pasarse a un formato binario, generar un ruido en dicha cadena y luego se enviará al receptor. El trabajo del Receptor es recibir dicha cadena con ruido y detectar (y dependiendo del algoritmo utilizado, corregir) el error en la cadena.

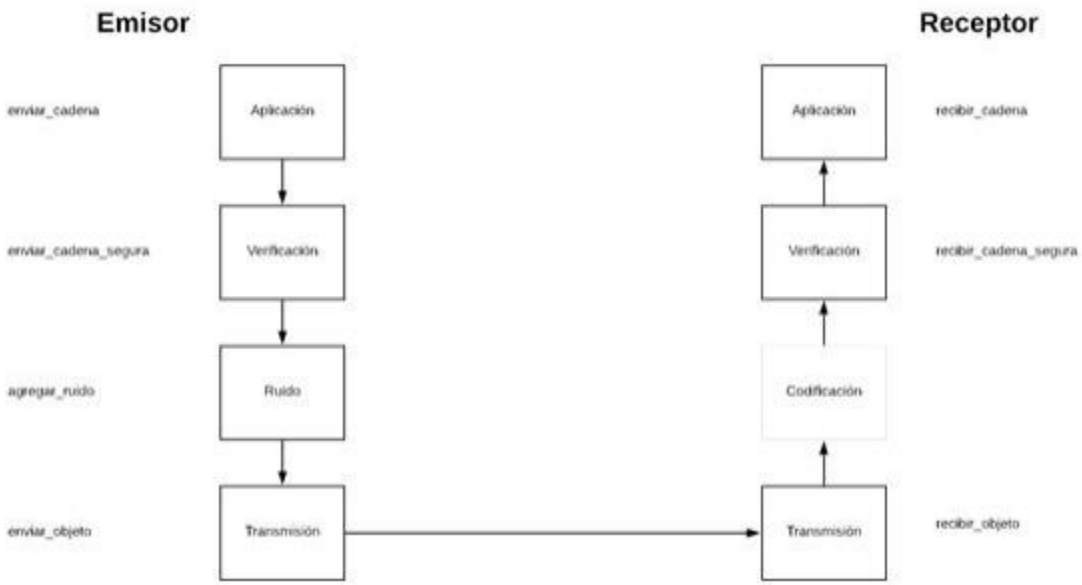

Ilustración 1: Arquitectura propuesta para arquitectura en capas

Se eligió implementar el Código de Hamming (algoritmo de corrección y detección) y CRC-32 (algoritmo de detección). El Código de Hamming se utiliza tanto para detectar un máximo de errores de un bits y corregir errores de un bit. La idea detrás de este código es introducir bits de redundancia y distribuirlos a lo largo de la cadena para poder detectar y solucionar errores si estos se encuentran. CRC-32 es una variante del CRC (Cyclic Redundancy Check), es un código de detección de errores utilizado en redes y dispositivos de almacenamiento para detectar cambios accidentales en datos sin procesar.

Resultados

Utilizando Python como el medio de comunicación entre emisor y receptor se realizó mediante sockets, se implementaron ambos algoritmos para enviar el mensaje. CRC-32 resultó tener un mejor funcionamiento. **Ilustración 2** muestra un ejemplo de la implementación del código de haming mostrando un resultado erróneo a causa de enviar un bit con múltiples bits cambiados.

```
D:\0\u\redes\laboratorio2\red-lab2>python utility.py
['01101000011011101011011000110000100100000011101100011011111011010110011111000111', 7]
switching  17  to  0
found at:  1
```

Ilustración 2: Error en Hamming

Por otro lado, **Ilustración 3** muestra una buena implementación, donde sólo se alteró un bit, se detectó el error y se corrigió para presentar el mensaje enviado.

```
CONEXION CON CLIENTE: ('192.168.99.1', 51945)
01101000011011110110110001100001001000000111010001101111011011001111001
hamming
01101000011011101011011000110000100100000011101100011011111011010110011111000111
The position of error is 12
Se recibio el mensaje:  hola tony
```

Ilustración 3: Implementación del Código de Hamming

Ilustración 4 muestra la implementación de CRC (Cyclic Redundancy Check). El algoritmo realiza una operación módulo sobre el mensaje codificado, este mensaje codificado viene adherido con una llave ya distribuida entre el emisor y receptor para poder evaluar si el mensaje tiene un error. Si el resultado del output, es cualquier valor que no sea 0, significa que hay un error/suciedad en el mensaje.

```
CONEXION CON CLIENTE: ('192.168.99.1', 51945)
01101000011011110110110001100001001000000111010001101111011011001111001
hamming
01101000011011101011011000110000100100000011101100011011111011010110011111000111
The position of error is 12
Se recibio el mensaje:  hola tony

CRC
100
```

Ilustración 4: Implementación del Código de Hamming seguido del output de CRC-32 confirmando que había suciedad en la cadena

Discusión

Después implementar ambos algoritmos, se decidió que CRC-32 tenía un mejor funcionamiento ya que, aunque sólo tiene el funcionamiento de detectar errores, este era capaz de reconocer más errores. El Código de Hamming está limitado a detectar y corregir errores de un bit debido a que se detectan errores por paridad. Esta consiste en añadir un bit de paridad que indica si el número de bits de valor 1 son pares o impares en la cadena, el error se detecta cuando se cambia la paridad. Como de demostró en **Ilustración 1**, el tener mucho ruido no le permite al código de Hamming a corregir la cadena y también puede fallar en la detección de error.

CRC-32 es más flexible para aceptar mayores tasas de errores debido a que sólo menciona si hay error en la cadena o no, no es tan específico respecto al error como lo es Hamming. Asumiendo que se está enviando información muy pesada, requeriría menos recursos utilizar algoritmos de detección de errores para evaluar la integridad de los mensajes enviados.

Comentario Personal

Sentí complicado este laboratorio, pero se debió principalmente a falta de tiempo debido trabajos de otras clases.

Conclusiones

- Algoritmos de detección son recomendables para evaluar grandes cantidades de información por requerir menor tiempo y recursos.

- CRC-32 es un algoritmo flexible porque puede tratar con cadenas de múltiples errores, pero termina siendo menos específico que el Código de Hamming.