UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Redes – Jorge Yaz

Lab 2

Daniel Alfredo Rayo Roldán, 22933

Irving Fabricio Morales Acosta, 22781

Descripción de la práctica

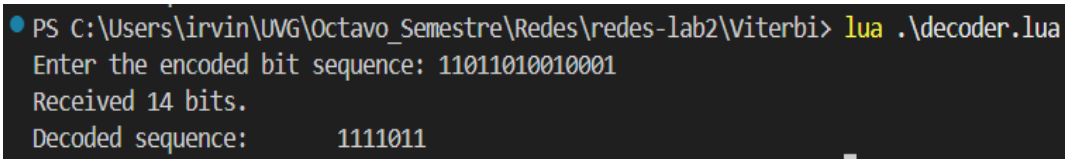
En esta práctica se implementa un algoritmo para detección de errores y el otro para corrección de los mismos. Para ello se implementaron emisores de mensajes y receptores en diferentes lenguajes.

Incluir las tramas utilizadas, las tramas devueltas por el emisor, indicar los bits cambiados de forma manual, y los mensajes del receptor para cada uno de los casos solicitados. Puede apoyarse de capturas.

Algoritmo de Viterbi : Corrección de errores

1. **Sin errores:**
    a. Trama enviada
    b. Trama devuelta por el emisor

    c. Respuesta del receptor



2. **Un error**
    a. Trama enviada
    b. Trama devuelta por el emisor

c. Respuesta del receptor

```
PS C:\Users\irvin\UVG\Octavo_Semestre\Redes\redes-lab2\Viterbi> lua .\decoder.lua
Enter the encoded bit sequence: 11111010010001
Received 14 bits.
Decoded sequence:       1111011
PS C:\Users\irvin\UVG\Octavo_Semestre\Redes\redes-lab2\Viterbi>
```

(se cambió el tercer dígito)

```
PS C:\Users\irvin\UVG\Octavo_Semestre\Redes\redes-lab2\Viterbi> lua .\decoder.lua
Enter the encoded bit sequence: 11011010010000
Received 14 bits.
Decoded sequence:       1111010
```

(se cambió el último dígito)

3. **¿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**.

   Sí, ya que Viterbi siempre devuelve la secuencia más probable de bits. Incluso cuando el número de errores supera su límite de corrección, por lo que no detecta esas fallas y simplemente corrige al camino de menor métrica.

   La capacidad máxima de corrección se define por la distancia libre del código. En este caso puedo corregir máximo dos errores sin riesgo de confundirse. Si introducimos más del límite, el algoritmo usará el camino más corto que no coincide con el original y no dará aviso de corregir.

```
PS C:\Users\irvin\UVG\Octavo_Semestre\Redes\redes-lab2\Viterbi> python .\viterbi.py
Enter the binary sequence to encode: 000
Error: Only '0' and '1' characters are allowed.
Enter the binary sequence to encode: 000
Encoded sequence: 000000
PS C:\Users\irvin\UVG\Octavo_Semestre\Redes\redes-lab2\Viterbi> lua .\decoder.lua
Enter the encoded bit sequence: 011000
Received 6 bits.
Decoded sequence:       101
PS C:\Users\irvin\UVG\Octavo_Semestre\Redes\redes-lab2\Viterbi>
```

4. **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. Durante las pruebas, en algunos casos el algoritmo no era capaz de detectar el error, esto es una desventaja, por ejemplo [...].**"

   El bit de paridad es tal vez el método más simple y sencillo para la detección de errores y consiste en añadir un bit extra por bloque de cualquier longitud, lo cual conduce a una redundancia mínima. Tiene una complejidad de O(n) por bloque a través de un XOR de todos los bits. Sin embargo, esta velocidad y eficiencia sacrifican la capacidad , ya que solo puede identificar errores en un número impar de bits. Tampoco ofrece capacidad de corrección. Su principal ventaja es el bajo overhead y facilidad de implementación.
   El algoritmo de Viterbi (en esta implementación) duplica la longitud de la trama, debido a la tasa ½) al utilizar polinomios generadores de 111 y 101 y un registro de desplazamiento de 2 bits. Esto nos da como resultado un overhead del 100% y un tiempo más lento de procesamiento, ya que por cada paso el decodificador necesita evaluar todas las ruta entre estados para calcular la métrica de Hamming y escoger la más probable. Gracias a todo este proceso Viterbi puede corregir automáticamente hasta dos errores por bloque, lo que mejora la fiabilidad en canales con ruido. Si los errores superan su capacidad de corrección, no emite alertas, solo devuelve la secuencia que más cree probable, aunque no sea la correcta.

   En general Viterbi es el algoritmo más configurable de los 3, en el que se puede decir la redundancia en los bits y cuantos bits pasados se tomarán en cuenta, pudiendo aumentar o disminuir la capacidad del algoritmo de detectar y corregir errores, cosa que no pasa

con los códigos de hamming que sin importar su configuración solo pueden corregir 1 errores y detectar 2. La principal debilidad de Viterbi, es que al usar redes markovianas, donde cada paso depende de pasos anteriores, no es resistente ante varios errores extremadamente juntos en distancia.