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

Laboratorio 2 - Primera parte

Esquemas de detección y corrección de errores

1 Antecedentes

Los errores de transmisión suceden en toda comunicación y es parte de los retos al momento de implementar este tipo de sistemas el manejar adecuadamente las fallas que puedan ocurrir. Por lo tanto, a lo largo de la evolución del Internet se han desarrollado distintos mecanismos que sirven tanto para la detección como para la corrección de errores.

2 Objetivos

- Analizar el funcionamiento de los algoritmos de detección y corrección.
- Implementar los algoritmos de detección y corrección de errores.
- Identificar las ventajas y desventajas de cada uno de los algoritmos.

3 Desarrollo

En clase estudiamos que entre los servicios que la capa de Enlace ofrece está la detección y corrección de errores, pues se asume que el medio en el cuál se transmite la data no es confiable. En este laboratorio se estarán implementado al menos un algoritmo de cada uno de ellos. El laboratorio será trabajado en parejas y un **único trío en caso de ser un número impar** de estudiantes. Los mismos grupos trabajarán la segunda parte del laboratorio.

Implementación de algoritmos

Para esta fase se deberán de implementar dos algoritmos, de los cuales uno debe de ser de corrección de errores y uno de detección de errores (tres en el caso del trio). Cada algoritmo debe ser implementado para el emisor y el receptor. Los algoritmos del lado del receptor deben implementarse en un lenguaje de programación distinto al utilizado para el emisor.

Lista de algoritmos sugeridos (entre otros):

- Corrección de errores
  - Códigos de Hamming para cualquier combinación (n,m) válida
  - Códigos convolucionales (Algoritmo de Viterbi)
- Detección de errores
  - Fletcher checksum

- CRC-32

EMISOR:

1. Solicitar una mensaje en binario.
2. Ejecutar el algoritmo seleccionado y generar la información necesaria para comprobar la integridad del mensaje.
3. Devolver el mensaje en binario concatenado dicha información.

RECEPTOR:

1. Solicitar un mensaje en binario concatenado con la información generada por el emisor.
2. Ejecutar el algoritmo seleccionado y comprobar la integridad del mensaje.
3. Devolver la siguiente información según corresponda:
    a. No se detectaron errores: mostrar el mensaje original (sin la información generada por el emisor)
    b. Se detectaron errores: indicar que el mensaje se descarta por detectar errores.
    c. Se detectaron y corrigieron errores: indicar que se corrigieron errores, indicar posición de los bits que se corrigieron y mostrar el mensaje corregido.

Nota: Los algoritmos no deben comunicarse de forma automática, será el estudiante quien deberá proveer los mensajes de entrada al momento de ejecutar cada algoritmo.

Ejemplo

Mensaje: 1001

Algoritmo: Bit de paridad par

Emisor, implementado en C++

- Solicita un mensaje, el estudiante proporciona 1001
- El algoritmo calcula el bit de paridad par que en este caso corresponde a 0.
- Se devuelve la cadena 10010 (trama + bit de paridad)

Receptor, implementado en Python

- Solicita un mensaje, el estudiante cambia el penúltimo bit y proporciona: 10000 (trama + bit de paridad)
- El algoritmo calcula el bit de paridad que en este caso corresponde a 1.
- Cómo el bit de paridad recibido (0) y el calculado (1) no coinciden se muestra un mensaje indicando que se detectaron errores y que el mensaje se descarta.
- Si el mensaje no hubiera sido manipulado, el receptor hubiera mostrado 1001

Escenarios de pruebas

Para los dos algoritmos realizar:

- Enviar un mensaje al emisor, copiar el mensaje generado por este y proporcionarlo tal cual al receptor, el cual debe mostrar el mensajes originales (ya que ningún bit sufrió un cambio). Realizar esto para tres mensajes distintos con distinta longitud.
- Enviar un mensaje al emisor, copiar el mensaje generado por este y cambiar un bit cualquiera antes de proporcionarlo al receptor. Si el algoritmo es de detección debe mostrar que se detecto un error y que se descarta el mensaje. Si el algoritmo es de corrección debe corregir el bit, indicar su posición y mostrar el mensaje original. Realizar esto para tres mensajes distintos con distinta longitud.
- Enviar un mensaje al emisor, copiar el mensaje generado por este y cambiar dos o más bits cualesquiera antes de proporcionarlo al receptor. Si el algoritmo es de detección debe mostrar que se detecto un error y que se descarta el mensaje. Si el algoritmo es de corrección y puede corregir más de un error, debe corregir los bits, indicar su posición y mostrar el mensaje original. Realizar esto para tres mensajes distintos con distinta longitud.
- ¿Es posible manipular los bits de tal forma que el algoritmo seleccionado no sea capaz de detectar el error? ¿Por qué si o por qué no? En caso afirmativo, demuestrelo con su implementación.
- 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, al modificar dos bits pude observar que en algunos casos el algoritmo no era capaz de detectar el error, esto es una desventaja con respecto a los otros métodos. Uno de estos casos fue el mensaje 1111 el cual cambio a 1100 pero el valor del bit de paridad fue el mismo*."

NOTAS

**Las mismos mensajes se deben utilizar para todos los algoritmos

**En caso de errores no detectados, solo pueden justificarse si es por una debilidad del algoritmo, no por errores de implementación.

Reporte

Al finalizar la actividad debe de realizarse un reporte **grupal** donde se incluyan las siguientes secciones:

- Nombres y carnés
- Título de la práctica
- Escenarios de pruebas
    - Incluir los mensajes utilizados, y screenshots de las respuestas del emisor y receptor para cada mensaje. En el caso de los mensajes con manipulación indicar los bits que sufrieron flip.
- Respuestas a las preguntas por cada algoritmo

3.1 Rúbrica de evaluación

Elemento	Excelente (1-0.9 pt.)	Aceptable con mejoras (0.9-0.5 pts.)	Inaceptable (0 pts)
Implementación	Los algoritmos funcionan de la manera esperada en todos los casos. Las tramas con errores no detectados corresponden a debilidades del algoritmo y no a errores de implementación.	Los algoritmos funcionan bien en la mayoría de casos, pero hay casos donde el algoritmo falla debido a errores de implementación.	Los algoritmos no fueron implementados, no compilan o la cantidad de fallas es muy alta.
Elemento	**Excelente (1 -0.9 pt.)**	**Aceptable con mejoras (0.9 - 0.5 pts.)**	**Inaceptable (0 pts)**
Reporte	Se muestra evidencia de todas las pruebas solicitadas, las respuestas a las preguntas se basan en las pruebas y están fundamentadas.	No hay evidencia de todas las pruebas realizadas, las respuestas se basan parcialmente en las pruebas, y/o el fundamento es débil.	No se incluye evidencia de las pruebas realizadas. Las respuestas no se basan en las pruebas, y/o carecen completamente de fundamento.

NOTAS

** La asistencia es obligatoria, una ausencia injustificada anula la nota del laboratorio

** Se debe cuidar el formato y ortografía

**El reporte debe ser incluido en el mismo repositorio de la implementación de los algoritmos. El repositorio debe ser público o privado con acceso al docente y los auxiliares del curso. Para la entrega se debe proporcionar el enlace al repositorio.