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 en la segunda parte del laboratorio.

Implementación de algoritmos

Para esta fase se deberán de implementar mínimo dos algoritmos (uno por cada miembro, o tres en caso del trío). De estos algoritmos, como mínimo, uno debe de ser de corrección de errores y otro de detección de errores. Se debe implementar tanto el emisor como el receptor para cada algoritmo, en un programa que se ejecute en la terminal. Los algoritmos del lado del receptor deben implementarse en un lenguaje de programación distinto al utilizado para el emisor.

Lista de algoritmos sugeridos (pueden implementar otros):

- Corrección de errores
  - Código de Hamming
    - Para cualquier *código(n,m)* que cumpla *(m + r + 1) <= 2^r*
  - Códigos convolucionales (Algoritmo de Viterbi)
    - Para cualquier trama de longitud *k*. La tasa de código es m:1 (por cada bit de entrada, salen m bits de salida).
- Detección de errores
  - Fletcher checksum
    - Para cualquier trama de longitud *k*, con bloques de 8, 16 o 32 (las 3 opciones, configurable). *k* debe corresponder al bloque utilizado (mayor que el bloque, se agregan 0s de padding en caso el mensaje sea menor).
  - CRC-32
    - Para cualquier trama de longitud *n*, $M_n(x)$, y el polinomio estándar para CRC-32 (uno de 32 bits, investigar cual es), donde n>32 (o padding si es menor a eso).

Nota: Los algoritmos no deben comunicarse de forma automática; eso sera la siguiente parte en donde conectaremos y el emisor y el receptor. Por ahora nos enfocaremos en la implementación y prueba de los algoritmos como tal. Por ello sus pruebas por ahora serán llamando directamente la función de emisor y pasando eso como input de la función receptor.

Ejemplo (algoritmo bit de paridad par). En el caso del emisor se deben seguir los siguientes pasos generales:

1. Solicitar una trama en binario (i.e.: “110101”).
2. Ejecutar el algoritmo y obtener la información adicional que se requiera para comprobar la integridad del mensaje (i.e.: “como hay un número par de 1s, el bit de paridad a agregar es 0”).
3. Devolver el mensaje en binario concatenado con la información adicional requerida para la detección/corrección de errores (i.e.: “110101**0**”).

Del lado del receptor se deben seguir los siguientes pasos generales:

1. Recibe como input un mensaje (la salida del emisor) que sigue un protocolo específico (i.e.: “110100”, note que tiene error).
2. Realizar la detección/corrección de errores (“el bit de paridad es 0, pero vemos un número impar de 1s, por lo que hubo error”).
3. Devolver la siguiente información correspondiente a cada caso:
   a. No se detectaron errores: mostrar la trama recibida
   b. Se detectaron errores: indicar que la trama 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 la trama corregida.

Ejemplo

Trama: 110101 - *Algoritmo*: Bit de paridad par.

Emisor, implementado en C++

- Input: la trama a enviar, 110101
- Se calcula el bit de paridad par que en este caso corresponde a 0.
- Output: la trama con el bit de paridad, 1101010.

Receptor, implementado en Python

- Input: la cadena generada por el emisor: 1101000 (trama + bit de paridad, se modifica manualmente el último bit de la trama original)
- Se calcula el bit de paridad que en este caso corresponde a 1.
- Cómo el bit de paridad recibido y el calculado no coinciden, se detectaron errores y la trama se descarta. Se muestra o devuelve información del error.

Pruebas

Para cada algoritmo implementado, realizar pruebas de detección/corrección:

- (sin errores): 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.
- (un error): 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 detectó 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.
- (dos+ errores): 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 detectó 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 sí 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, en algunos casos el algoritmo no era capaz de detectar el error, esto es una desventaja, por ejemplo [...]."*

**Los mismos mensajes se deben utilizar para ambos algoritmos (para tener una base y compararlos)

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

Reporte

Al finalizar la actividad debe de realizarse un reporte **grupal** donde se incluya por lo menos lo siguiente:

- Nombres y carnés, encabezado, descripción de la práctica, etc. Formato adecuado.
- Escenarios de pruebas
    - 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.
- Respuestas a las preguntas, por cada algoritmo

4 Rúbrica de evaluación

Elemento	Excelente (1-0.9 pt.)	Aceptable con mejoras (0.9-0.5 pts.)	Inaceptable (0.5-0 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 (0.5)**	**Aceptable con mejoras (0.25 pts.)**	**Inaceptable (0 pts)**
Resultados y Escenarios	Las explicaciones reflejan efectivamente lo realizado y aprendido en el laboratorio. Evidencian sus pruebas.	Las explicaciones omiten detalles importantes, pero en general expresan la idea central. No evidencian todas sus pruebas.	No es posible entender lo realizado en el laboratorio a partir de las explicaciones. No hay evidencias ni pruebas.
Formato del reporte y Respuestas	Las conclusiones son reflejo del análisis preparado en la discusión. El reporte está ordenado, legible y con buen formato.	Las respuestas no reflejan análisis, y su fundamento es débil. El reporte cuenta con un formato y legibilidad aceptable.	Las respuestas carecen de fundamento o análisis. El reporte no sigue un formato y no es legible.

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

** Se debe cuidar el formato y ortografía del reporte

** El reporte debe incluirse en su repositorio también, el cual deben asegurarse que el docente y auxiliares tengan acceso para poder revisar. Entregar el enlace al repo en Canvas.

d. Entregar en Canvas

- **Reporte** en formato PDF
- **Todo el código** involucrado y cualquier elemento para su compilación (makefiles, etc).
- **Link a su repositorio**
    - El **repositorio también debe tener el Reporte** PDF.