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

El ruido y 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

- Comprender a detalle 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

Existen dos grandes familias de algoritmos para el manejo de errores: de detección y de corrección. Ambos tienen sus ventajas y desventajas y son principalmente utilizados en distintos medios.

En este laboratorio se estará revisando al menos un algoritmo de cada uno de ellos, y cada estudiante trabajará la infraestructura para probarlo. 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.

a. 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 cada algoritmo en su versión del emisor (cálculo de la información necesaria para verificar la integridad) y del receptor (uso de la información proporcionada por el emisor para comprobar la integridad). Los algoritmos deben implementarse en dos lenguajes de programación distintos, de tal forma que un receptor en lenguaje B pueda validar un mensaje codificado (por el mismo algoritmo) en lenguaje A.

Lista de algoritmos sugeridos:

- 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 2:1 (por cada bit de entrada, salen dos). O sea la salida son *2k* bits.
- Detección de errores
  - Fletcher checksum
    - Para cualquier trama de longitud *k*, con bloques de 8, 16 o 32. *k* debe corresponder al bloque utilizado (mayor que el bloque, se agregan 0s de padding).
  - 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

En el caso del emisor se deben seguir los siguientes pasos generales:

1. Solicitar una trama en binario (i.e.: “110101”).
2. Calcular la información adicional que requiera el algoritmo seleccionado para detectar/corregir errores.
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:

2. Solicitar un mensaje en binario concatenado con la información adicional requerida por el algoritmo.
3. Realizar la detección/corrección de errores utilizando la información adicional proporcionada por el emisor.
4. 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++

- Solicita la trama 110101
- Se calcula el bit de paridad par que en este caso corresponde a 0.
- Se muestra la cadena 1101010 (trama + bit de paridad)

Receptor, implementado en Python

- Solicita la cadena mostrada 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.

d. Pruebas

Utilizando los algoritmos implementados realizar pruebas de detección/corrección:

- Utilizar mínimo tres tramas distintas sin manipular en el lado del receptor. Se debe evidenciar que el receptor no detecta errores y muestra la trama.
- Utilizar mínimo otras tres tramas distintas donde se modifique manualmente un bit al momento de pasarlas al receptor. Se debe evidenciar que el algoritmo detecta/errores los errores.
- Utilizar mínimo otras tres tramas distintas donde se modifique manualmente por los menos dos bits al momento de pasarlas al receptor. Se debe evidenciar que el algoritmo detecta/corrige los errores. Si el algoritmo no es capaz de detectar alguna de las tramas con errores, justificar la razón en la discusión del reporte.
- Utilizar una trama modificada especialmente para que el algoritmo del lado del receptor no sea capaz de detectar error e indicar por qué la trama no fue detectada en la discusión del reporte.

NOTAS

**Las mismas tramas se deben utilizar para ambos algoritmos

**En caso de errores no detectados, sólo pueden justificarse en caso de que sean por una debilidad del algoritmo, no por errores de implementación del algoritmo.

e. Reporte

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

- Nombres y carnés
- Título de la práctica
- Descripción de la práctica
  - Incluir explicación de los algoritmos utilizados
- Resultados
  - 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.
  - Evidenciar sus pruebas con capturas de pantalla, etc..
- Discusión
  - Análisis exhaustivo de los resultados, errores, algoritmos etc..
  - ¿Hubo casos donde no fue posible detectar errores? ¿Por qué?
- Comentario grupal sobre el tema (errores)
- Conclusiones
- Citas y Referencias

f.

g. 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/Discusión	Las explicaciones reflejan efectivamente lo realizado y aprendido en el laboratorio.	Las explicaciones omiten detalles importantes, pero en general expresan la idea central.	No es posible entender lo realizado en el laboratorio a partir de las explicaciones.
Formato del reporte/Calidad de conclusiones	Las conclusiones son reflejo del análisis preparado en la discusión. El reporte está ordenado, legible y con buen formato.	Las conclusiones no se encuentran totalmente fundamentadas en la discusión. El reporte cuenta con un formato y legibilidad aceptable.	Se concluye elementos de la teoría o no existe referencia alguna en la discusión. 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

h. Entregar en Canvas

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