Guatemala 28 de Julio de 2022
Raúl Jimenez - 19017
Oscar Saravia - 19322
Donaldo Garica - 19683

REDES

LAB02

Link al repositorio en github

Descripción de la práctica

El laboratorio #2 tenía como objetivo principal identificar las ventajas y desventajas de los algoritmos para la detección y corrección de errores. De igual forma tenia como objetivo lograr entender los elementos de la arquitectura en capas de las redes. La práctica consistió en crear un programa en python en el cual existiera un servidor (emisor) y un receptor (cliente), el servidor iba a enviar mensajes simulando el proceso que se realiza en las redes. Para simular el concepto de redes se trabajó por medio de 4 capas en el emisor y 4 capas en el receptor. Las capas de emisor son:

- ➔ Aplicación: La capa de aplicación consiste en obtener el mensaje que el usuario va a enviar.
- ➔ Verificación: Esta capa iba a ser la encargada de pasar el mensaje a binario por código ASCII y luego ejecutar el algoritmo de detección o corrección.
- ➔ Ruido: La capa de ruido va a ser la encargada de crear “ruido” o pérdida de datos como puede ocurrir en eventos reales en la red.
- ➔ Transmisión: Por último la capa de transmisión va a ser la encargada de enviarla al cliente por medio de un socket de python.

Las capas del receptor son:

- ➔ Aplicación: Encargada de mostrar el mensaje recibido al usuario en texto normal.
- ➔ Verificación: Esta capa va a ser la encargada de pasar el mensaje obtenido de binario a texto normal.
- ➔ Codificación: Por medio de los algoritmos de detección y corrección de errores determinar si existe algún problema en el mensaje obtenido y que tan eficientes son al momento de corregir dichos errores en caso fuera necesario.
- ➔ Transmisión: Esta capa va a ser la encargada de recibir el mensaje por medio de los sockets en Python.

Una vez implementada esta arquitectura se buscaba implementar 3 algoritmos distintos. El algoritmo de Fletcher checksum en la parte de detección de errores, Paridad doble en la parte de corrección de errores y por último

Algoritmos

- ➔ **Fletcher checksum**: El algoritmo de fletcher checksum consiste en realizar una suma de ciertos bloques de bytes en el mensaje que se va a enviar para lograr determinar si al momento de realizar la misma suma del lado del receptor si tengamos el mismo mensaje. Lo primero que se debe realizar en este algoritmo es obtener el mensaje en binario, seguido de esto se procede a dividir el mensaje en cuatro bloques. Una vez se tengan estos cuatro bloques se procede a realizar una suma binaria de los bloques. En una red normal esta sumatoria sería enviada por medio del campo “Checksum” del UDP. Una vez el receptor tenga el mensaje y el valor del checksum procederá a realizar el mismo checksum solo que con el mensaje que se obtuvo y así poder determinar si se perdieron datos en el envío del mensaje. Si las sumatorias son distintas quiere decir que el mensaje que se obtuvo no es el mismo al que se envió (hubo pérdida de datos) y si son las mismas quiere decir que el mensaje enviado y recibido si son los mismos. (no hubo pérdida de datos)
- ➔ **Paridad simple**: es un algoritmo que permite detectar errores en la transmisión de mensajes. Este consiste en verificar el último bit del mensaje, si el valor del bit es 1, la cadena de bits del mensaje debería de tener un número impar de números 1, esto con el fin de que dicho número sea siempre par. Si al verificar la cuenta de bits con valor 1, estos son impares, significa que el mensaje fue modificado y necesita volver a ser enviado por parte del emisor. Este proceso puede ser susceptible a fallar, ya que si el mensaje es modificado de tal forma que la cantidad de bits con valor 1 siga siendo par, no sería considerado como un error en el mensaje.

- ➔ **Hamming**: el código Hamming es un conjunto de códigos de corrección de errores que se pueden usar para detectar y corregir los errores que pueden ocurrir cuando los datos se mueven o almacenan del remitente al receptor. Es una técnica desarrollada por R.W. Hamming para la corrección de errores. Para detectar el error calcula la paridad de los datos y por último convierte de binario a decimal.

Resultados

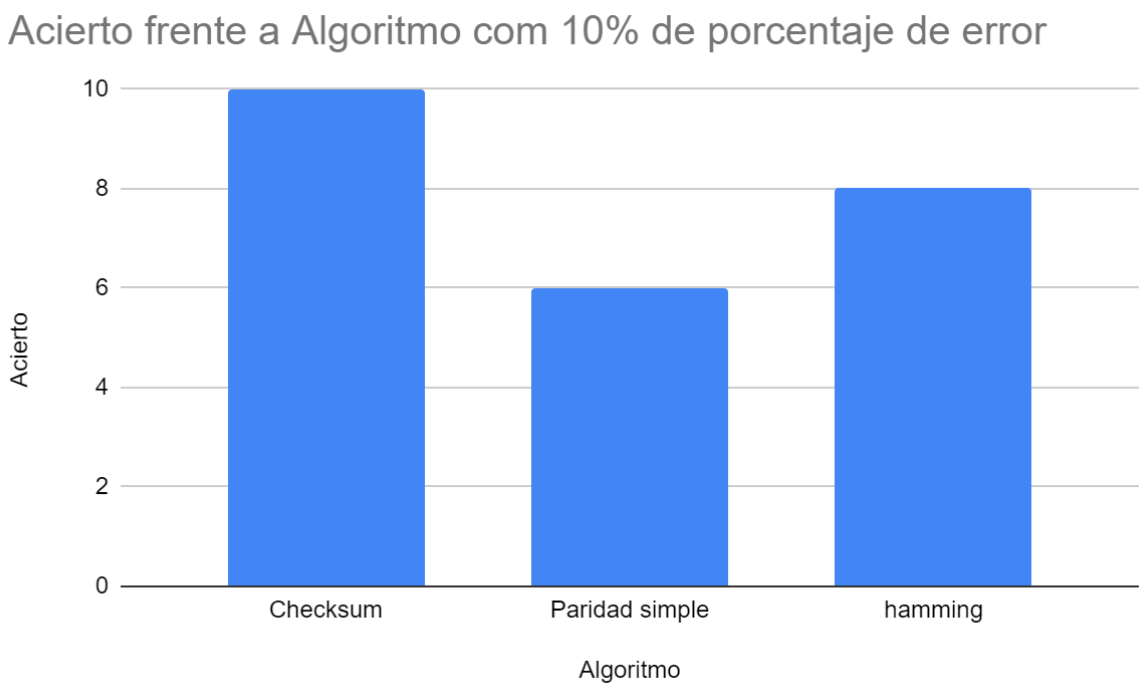

Figura 1. Gráfica de aciertos con 10% de porcentaje de error con un carácter.

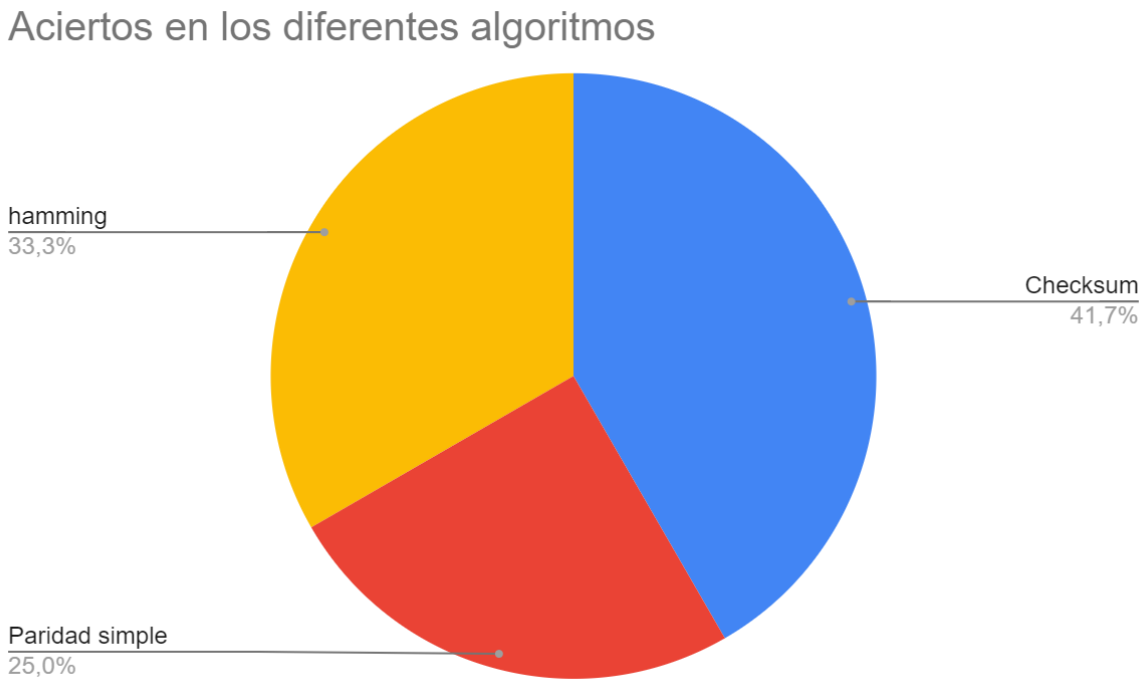

Figura 2. Gráfica de aciertos totales de algoritmos

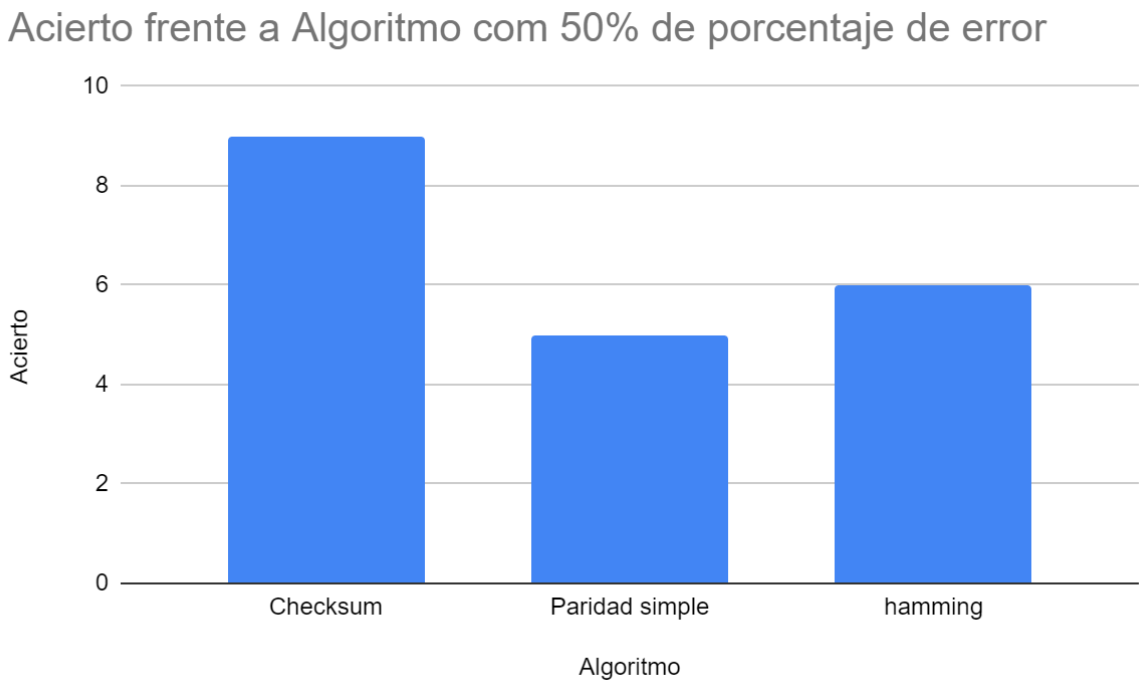

Figura 3. Gráfica de aciertos con 50% de porcentaje de error con un carácter

Discusión

En la práctica se realizaron tres distintos algoritmos, dos de detección de errores y uno de corrección de errores como se mencionó anteriormente. Para probar que tan efectivos son estos algoritmos lo que se realizó fue un set de distintos tests. Primero se probaron los algoritmos con un porcentaje de error de 10% y enviando un carácter para determinar si estos acertaba cuando el mensaje enviado era distinto a lo recibido. En este caso como se puede observar la figura 1 nos muestra que el algoritmo que tuvo mejores resultados fue el de checksum. Luego de esto se probaron los algoritmos con un porcentaje de error de 50% igual con un carácter. Como se puede observar la figura 2 que el algoritmo checksum tuvo tres aciertos más en comparación del de Hamming y cuatro aciertos más que el de paridad. Con estas dos gráficas ya podemos determinar que el algoritmo de checksum es más eficiente al momento de detectar errores en la transferencia de datos.

El más flexible para aceptar mayor tasa de errores fue el de checksum, como se puede observar en las gráficas anteriores, al aumentar el ruido en las cadenas de bits, el algoritmo checksum fue el más flexible a la hora de detectar fallos en los mensajes, ya

que a comparación con los otros algoritmos utilizados, este utiliza un procedimiento más complejo, el cual es más eficaz a la hora de detectar una cadena modificada.

Por un lado, un código detector incorpora información adicional junto con los datos transmitidos de manera que se pueda determinar si se produjo o no un error durante la transmisión. Por otro lado, un código corrector incorpora más información que uno detector, ya que la idea es, además de detectar si se produjo un error, tener la certeza de en qué lugar se produjo a fin de poder corregirlo sin requerir la retransmisión del dato en cuestión.

Conclusiones

- ➔ En base a las gráficas podemos concluir que el algoritmo de checksum tiene un mayor porcentaje de aciertos.
- ➔ En base a las gráficas se pudo determinar que el algoritmo de corrección de errores de paridad es el menos efectivo entre los 3 utilizados.
- ➔ Utilizar el número de paridad para verificar la validez de un mensaje no es confiable, ya que es susceptible a que se cambien dos bits, haciendo que el algoritmo no detecte el fallo.
- ➔ Se puede determinar que el algoritmo de checksum es más efectivo debido a que realiza el mismo cálculo de caracteres binarios para ser comparado y determinar si hay errores.