Universidad Del Valle de Guatemala — Departamento de Computación
Redes — Sección 20
Elías Alberto Alvarado Raxón — 21808
Adrian Fulladolsa Palma — 21592

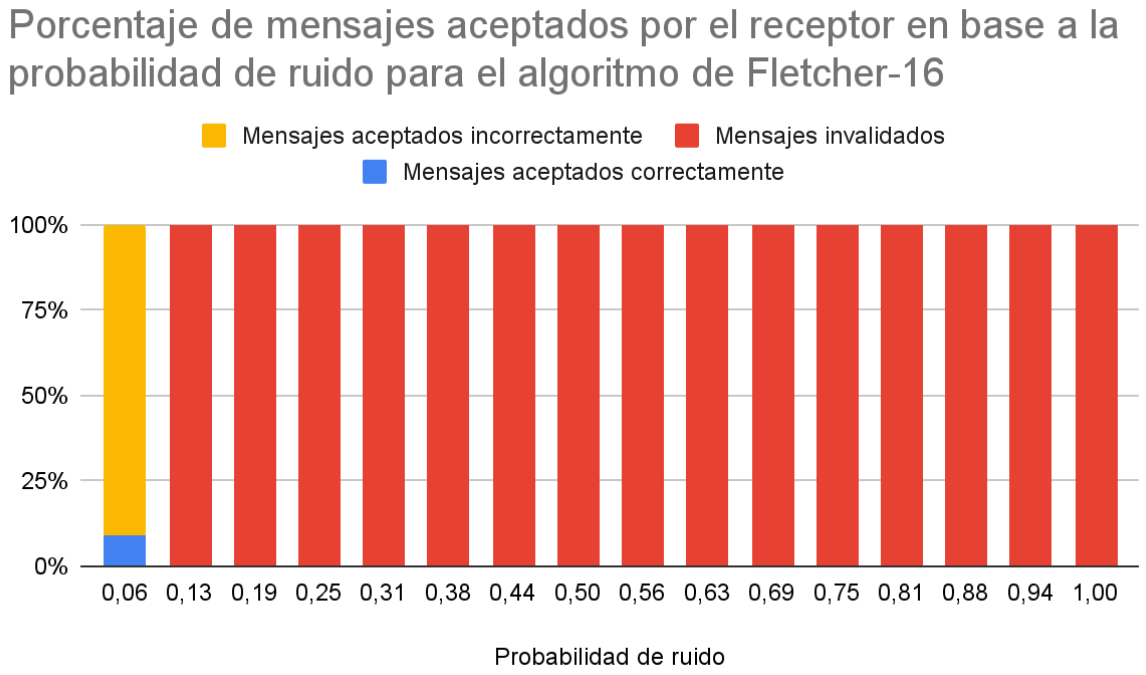
Laboratorio #2 - Parte 2

Esquemas de detección y corrección de errores

Esta práctica se basó en la implementación de dos algoritmos utilizados en el envío de mensajes. Hamming, un algoritmo de detección y corrección de errores. Fletcher Checksum 16, un algoritmo de detección de errores. Para este laboratorio se implementó un receptor y emisor para cada uno de los algoritmos, y se añadió la parte de comunicación por medio de sockets.

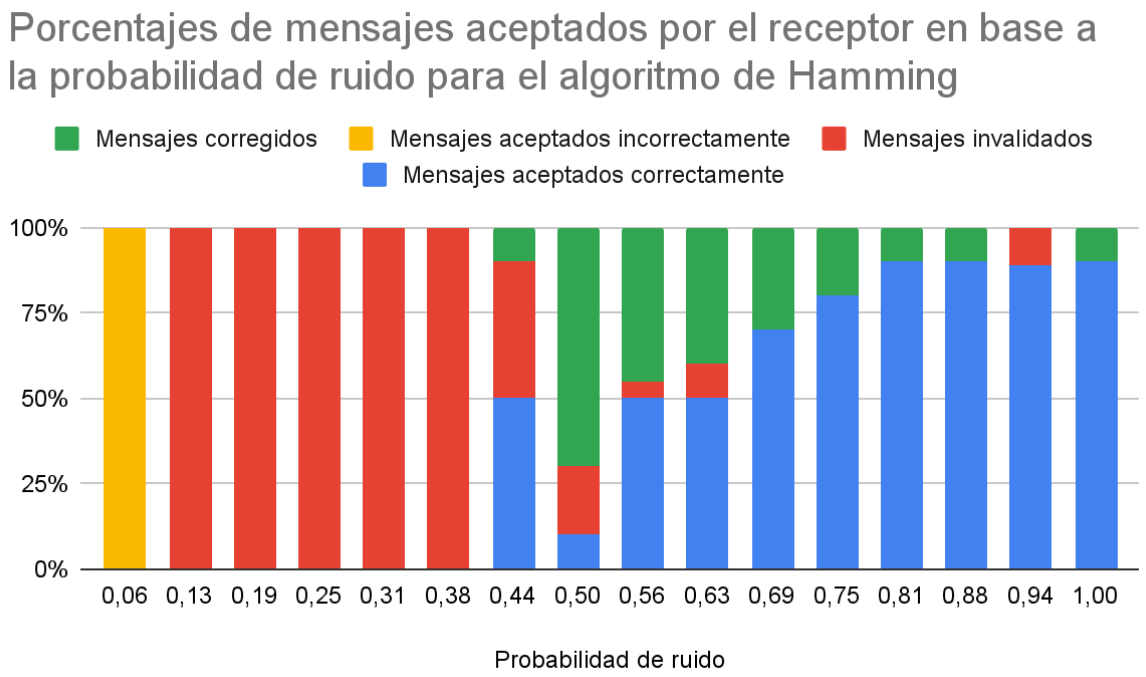
Pruebas

Fletcher-16 Checksum - Tasa de aceptación de error:



Gráfica 1. Análisis Fletcher-16

Hamming - Tasa de aceptación de error:



Gráfica 2. Análisis Hamming

Discusión

Dado que los algoritmos a analizar son para fines distintos, se intentará ser lo más objetivo en cuanto a las comparaciones.

En funcionamiento general, Hamming es sin lugar a dudas el mejor algoritmo. Pues entre sus ventajas existe la detección y corrección de errores, con limitaciones. Por otro lado, Fletcher solamente es capaz de detectarlos aunque con mayor precisión.

Hablando sobre la flexibilidad que tiene cada algoritmo para aceptar mayores tasas de errores. Hamming es el mejor para corregir errores cuando solamente fue un bit el cuál sufrió un swap, pero si se está intentando analizar un bloque el cuál tenía una probabilidad de cambio de 0.5 (50%) es muy probable que no pueda hacerlo correctamente. Pues, a pesar de que pueda detectar dos errores, es incapaz de corregirlos (incluso identificar el lugar donde están dichos errores). Por otra parte, Fletcher checksum tiene un mejor desempeño en escenarios como el mencionado anteriormente. Su única limitación es que es incapaz de corregir los errores, pero el hecho de que indica que los mensajes tienen un error cuando el ruido fue alto, es un punto a favor.

Todo esto lo podemos observar en los gráficos 1 y 2, donde claramente Fletcher demostró un mejor desempeño en la detección de errores a pesar que la probabilidad de cambio era alta, por lo tanto, el entrar en un escenario donde los bits de paridad cambiaran de tal modo que fuera muy difícil de detectar el error no fue problema para este algoritmo. Por otra parte, Hamming fue capaz de detectar y corregir errores cuando la tasa de error fue media; pero el hecho que no haya detectado errores cuando la tasa fue alta indica que el algoritmo fue incapaz de detectar los errores presentados. En otras palabras, el mensaje incurrió en un cambio donde sus bits de paridad fueron afectados, haciendo que el algoritmo no pudiera ver que el mensaje SI fue modificado.

Ahora, hablando sobre cuándo es mejor utilizar uno o el otro. La detección de errores es mejor cuando se trata de una aplicación en la que la retransmisión de datos se puede dar sin ningún problema, hasta por mano propia o bien, el mismo mensaje ya llega íntegro a su destino. Como es de esperar, la corrección de errores es mejor utilizada cuando el costo o la retransmisión en sí es difícil de acceder. Veamos escenarios como la transmisión de datos en vivo, donde es crucial acceder al mensaje original sin problemas. Por lo tanto, el uso dependerá, obviamente, de la forma en la que se maneja la transmisión de datos. Si ésta está atada a ciertos costos altos o simplemente no se pueda dar una retransmisión de datos.