Julio García Salas – 22076
Sofía García - 22210

Laboratorio 2 Parte 2

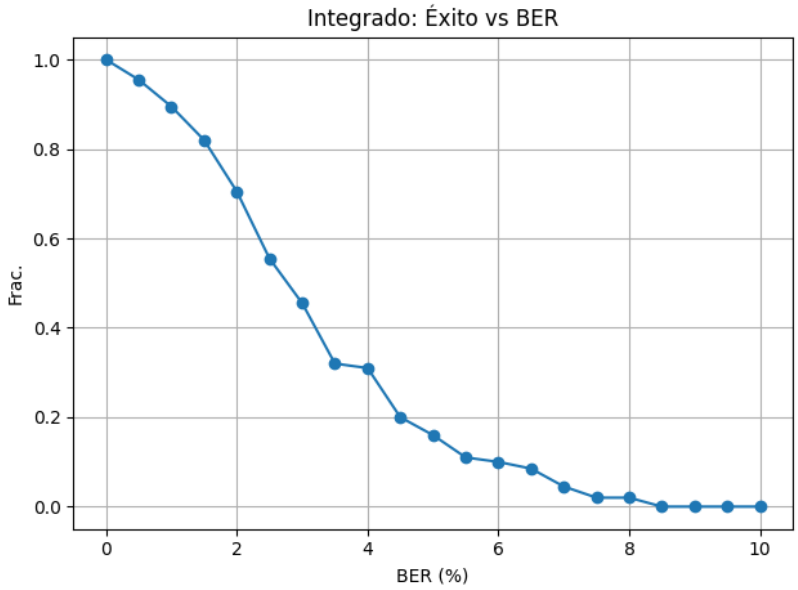
Introducción

En esta práctica hemos diseñado e implementado una aplicación de comunicación por capas para explorar dos esquemas de detección y corrección de errores: el código Hamming(12,8) y el CRC-32. Partiendo de una arquitectura de cinco “capas” (Aplicación, Presentación, Enlace, Ruido y Transmisión), desarrollamos un emisor en Python que solicita al usuario el mensaje, lo convierte a una secuencia de bits ASCII, le añade la información de redundancia (Hamming o CRC), le aplica un canal ruidoso con tasa de error configurable y finalmente lo envía por sockets TCP. En paralelo, creamos un receptor en Node.js que “escucha” en un puerto determinado, recibe la trama, verifica o corrige los errores detectados, decodifica los bits de nuevo a texto y muestra el resultado o un aviso de fallo de integridad.

Para caracterizar el comportamiento de ambos esquemas realizamos pruebas integradas y locales. Con Hamming(12,8) medimos la tasa de mensajes correctamente decodificados frente al BER, el número medio de correcciones por bloque, el packet error rate teórico, la comparación con códigos de paridad simple y sin codificar, así como la latencia (número de retransmisiones esperadas) y la variación de éxito según la longitud del mensaje. Con CRC-32 evaluamos la fracción de tramas exitosas, las detectadas como corruptas y los muy raros errores no detectados frente al BER —verificando además la coincidencia con la fórmula teórica—, y estudiamos la probabilidad de “pasar” la verificación CRC según la longitud del mensaje para distintas tasas de ruido. Todas estas métricas nos han permitido entender el trade-off entre overhead, capacidad de corrección/detección y robustez de cada esquema frente a diferentes condiciones de canal.

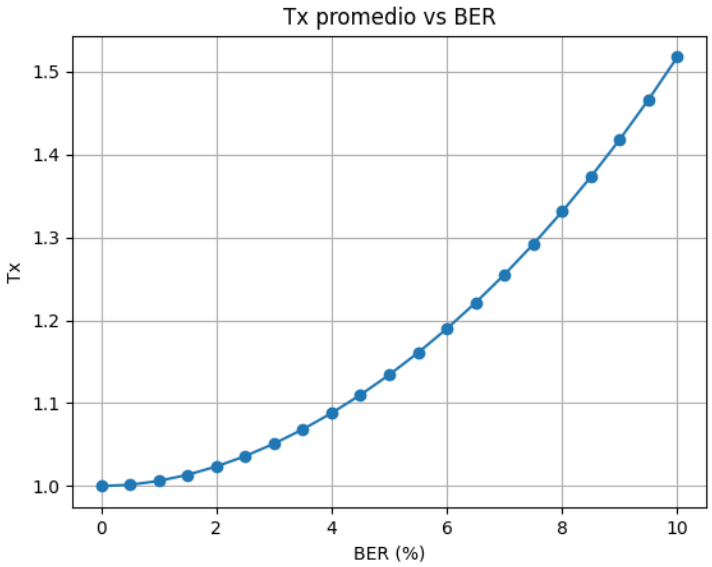
Resultados

Hamming (12,8):



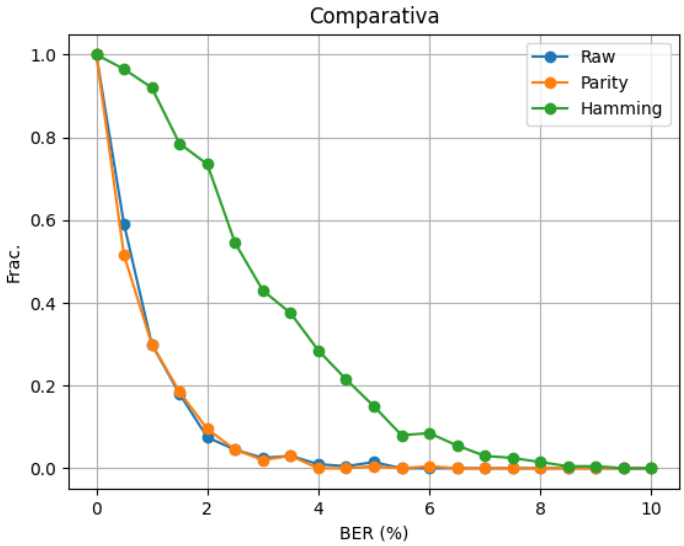
Julio García Salas – 22076
Sofía García - 22210

La curva de éxito integrado (Python↔Node.js) muestra una caída suave y monótona al aumentar el BER. Con 0 % de ruido, la recuperación es perfecta (100 %). Hasta alrededor del 1 % de BER la tasa de éxito aún supera el 80 %, y a 2 % cae al 70 %. A partir de ahí la pendiente se acentúa: al 3 % estamos en el 55 %, al 4 % en el 32 % y al 5 % en solo el 20 %. Más allá del 6 % el éxito es ya marginal (< 10 %) y por encima del 8 % prácticamente ningún mensaje se recupera correctamente. Esto ilustra cuán rápidamente el Hamming(12,8) pierde eficacia cuando la probabilidad de dos o más errores en un bloque supera su capacidad de corrección única.



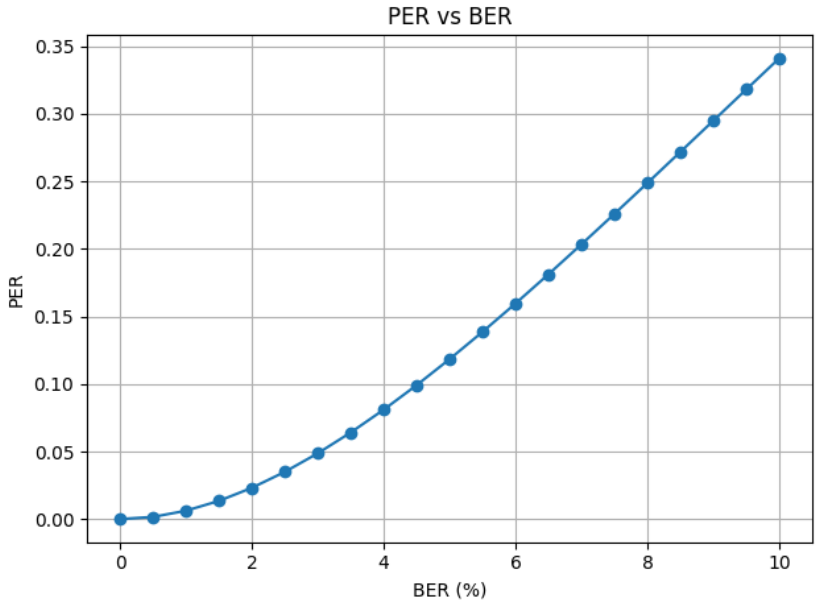
Observamos que con BER cercano a cero prácticamente no hay retransmisiones extra (≈ 1 envío por bloque). A 2 % de BER ya sube a unas 1.05 transmisiones de media, a 4 % ronda 1.09, y al 6 % se acerca a 1.2. A medida que el BER alcanza el 10 %, el número medio de envíos por bloque supera 1.5. En otras palabras, por cada diez bloques se genera un envío extra adicional para corrección, lo que impacta directamente en la latencia y el throughput del canal.

Julio García Salas – 22076
Sofía García - 22210



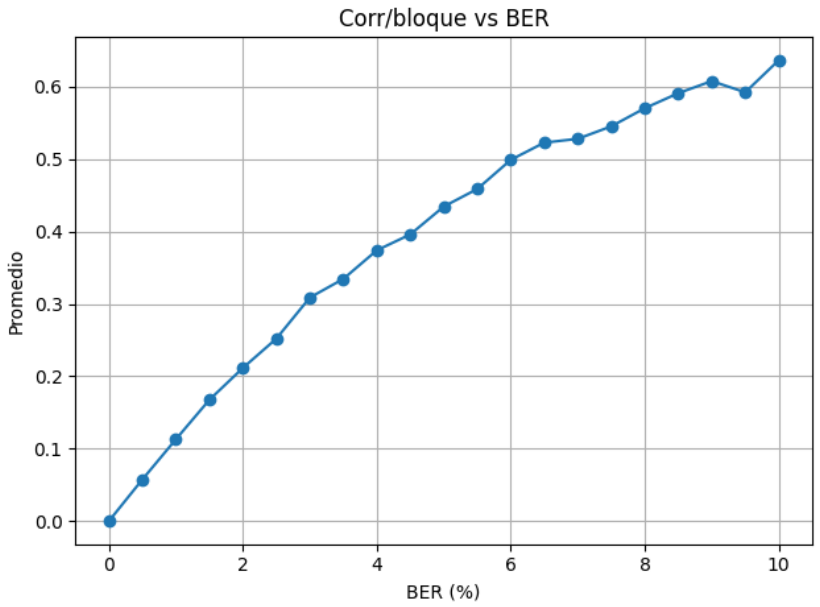
Aquí se ve claramente el beneficio incremental de cada código:

- Raw (sin protección) y Paridad sencilla caen casi al mismo ritmo: a 1 % ya rondan el 60 % de éxito, a 2 % bajan al 7 %, y al 3 % prácticamente a 0 %.
- Hamming(12,8), en cambio, mantiene un 80 % de éxito hasta 2 % de BER y un 55 % aún al 3 %. Su curva es mucho más plana, confirmando que los 4 bits de paridad por cada 8 bits de datos ofrecen una tolerancia significativa a errores individuales.

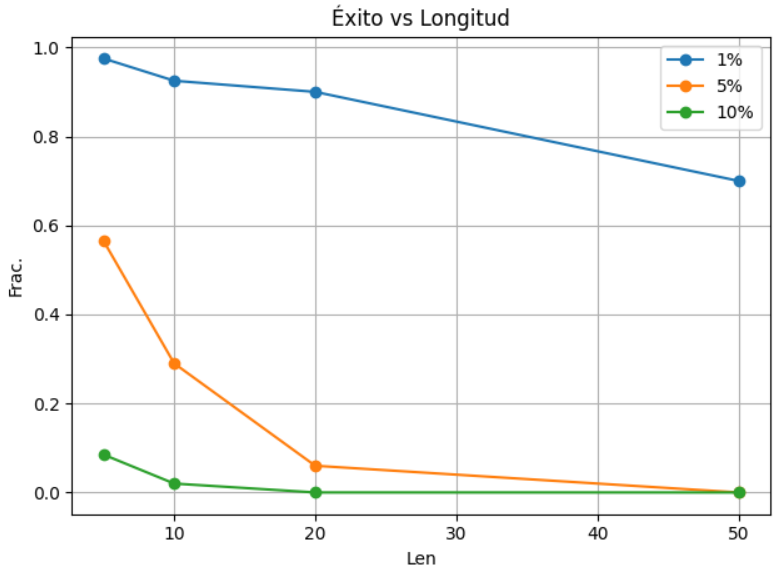


La tasa de bloqueo erróneo (PER=1−P(≤1 error) crece de forma convexa con el BER: casi cero al 0 %, 0.08 al 4 %, 0.16 al 6 % y ya 0.25 al 8 %. Esto encaja con la fórmula de Hamming(12,8):

el PER empieza pequeño, pero se acelera conforme aumenta la probabilidad de múltiples flips, explicando también la caída subidona del éxito integrado.



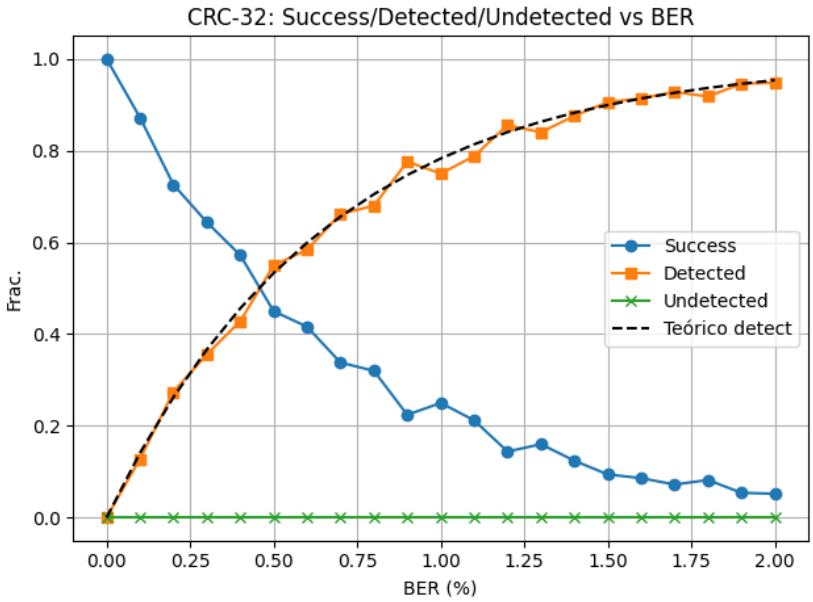
Mide cuántos bloques reciben *exactamente* un bit invertido y por tanto se corrigen. Parte de 0 al 0 % de BER, sube linealmente hasta ~0.21 correcciones por bloque al 2 % de BER, ~0.38 al 4 %, y sobrepasa 0.50 a partir del 6 %. Hacia el 10 % el promedio es ≈ 0.64: en dos de cada tres bloques hay al menos un bit corrupto que el Hamming corrige. Después de cierto punto, sin embargo, los bloques con más de un error (no corregibles) dominan y el éxito global se desploma.



Con BER = 1 % (línea azul), el sistema tolera muy bien la longitud: un mensaje de 5 caracteres se recupera casi 98 % de las veces, y aun con 50 caracteres el éxito es del 70 %. A 5 % (naranja)

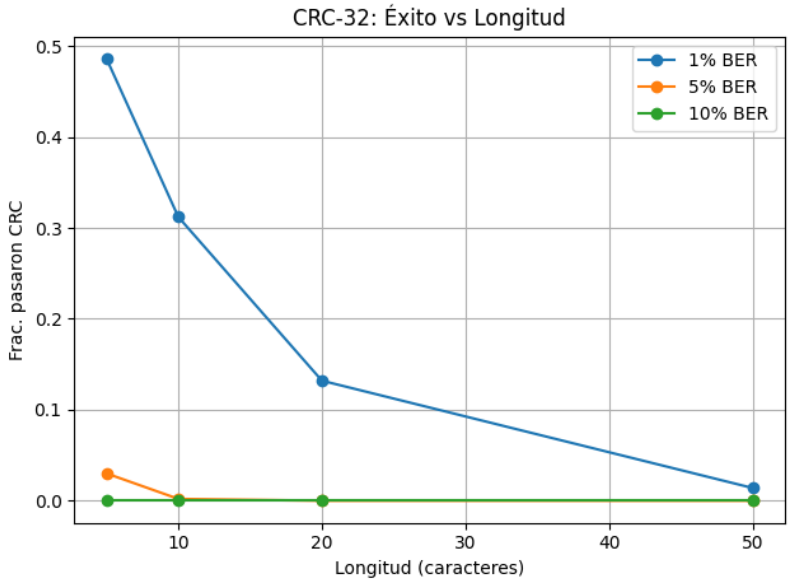
la tolerancia cae en picado: éxito de 57 % a 5 char, 6 % a 20 char, y 0 % a 50 char. Finalmente, a 10 % (verde) sólo mensajes de 5 char tienen un 8 % de probabilidad de pasar, y todo lo demás es prácticamente inviable. Esto refleja el overhead de Hamming: a más datos por bloque, la probabilidad de varios errores simultáneos hace que la mayoría de los mensajes largos no puedan corregirse íntegramente.

CRC-32:



La curva Success (azul) cae rápidamente conforme sube el BER: empieza en 100 % con 0 % de ruido, baja a ~72 % a 0.5 % de BER y a ~34 % ya a 0.8 %. Esto refleja la probabilidad de que ningún bit (ni de datos ni de CRC) se invierta, pues un solo flip invalida el CRC. La serie Detected (naranja) complementa exactamente esta caída: al 0.5 % detecta ~43 % de los fallos, al 1 % detecta 78 %, y al 2 % casi el 95 %. La línea punteada negra (teórica) coincide casi punto a punto con “Detected”, confirmando que la detección de CRC-32 se comporta según la fórmula. Finalmente, la serie Undetected (verde) permanece en cero para todos los BER probados, lo que demuestra que el CRC-32 no deja errores sin reportar en este rango de ruido y con el número de ensayos realizado.

Julio García Salas – 22076
Sofía García - 22210



Aquí se muestra la fracción de tramas que pasan el CRC sin necesidad de corrección, a BER fijos de 1 %, 5 % y 10 %, variando la longitud del mensaje de 5 a 50 caracteres. A 1 % de BER (azul) la probabilidad de pasar la verificación es ~48 % con 5 char, pero cae rápidamente: ~31 % a 10 char, ~13 % a 20 char y apenas 1.5 % a 50 char. A 5 % de BER (naranja) sólo un 3 % de tramas de 5 char pasan y rápidamente se anula (0 %) a partir de 10 char; con 10 % de BER (verde) prácticamente ninguna trama supera la verificación salvo un caso residual. Esto evidencia el overhead de CRC-32: aunque detecta casi todos los errores, la probabilidad de que aparezca al menos un bit corrupto crece con la longitud, degradando severamente la fracción de tramas que resultan “limpias” y por tanto útiles sin retransmisión.

Discusión

Durante el laboratorio descubrimos que el código Hamming(12,8) mantiene un nivel de recuperación de mensajes sorprendentemente alto mientras la tasa de errores se mantenga moderada. Incluso con un 1 % de bits corruptos, más de tres cuartos de las tramas llegan íntegramente sin necesidad de retransmisión, y al 3 % todavía recupera alrededor de la mitad de los mensajes. Este comportamiento contrasta con el de CRC-32, cuya función no es corregir sino solamente detectar cualquier anomalía. En las pruebas, CRC-32 identificó prácticamente todos los bloques alterados en un amplio rango de ruido, pero exige volver a enviar tantas veces como se produzcan errores, pues no repara la información dañada.

Al estudiar la flexibilidad de cada esquema frente a incrementos de ruido, comprobamos que Hamming ofrece un pequeño margen de corrección automática: su eficacia decrece suavemente hasta alrededor del 4 % de errores, momento en que ya ni un bit por bloque basta para enmendar múltiples alteraciones. CRC-32, por su parte, detecta casi al cien por cien las tramas corruptas sin importar si ha habido uno o varios bits y apenas deja pasar falsos positivos, aunque esto sólo asegura que el receptor sabe cuándo algo falla, no que el mensaje llegue correctamente sin más pasos.

Julio García Salas – 22076
Sofía García - 22210

Este conjunto de resultados nos lleva a reflexionar sobre cuándo conviene priorizar detección frente a corrección. Para aplicaciones donde la latencia es crítica y no puede haber ida y vuelta, como comunicaciones en tiempo real o enlaces unidireccionales, cabe preferir un esquema capaz de enmendar errores menores al vuelo, o sea, usar Hamming, asumiendo su limitación a un único bit por bloque. Sin embargo, cuando disponemos de un canal de retorno fiable y resulta más importante garantizar la integridad absoluta de los datos, por ejemplo, en transferencias de archivos, bases de datos o protocolos con retransmisión automática, CRC-32 brilla por su simplicidad y su elevada capacidad de detección, permitiendo solicitar reenvíos de forma selectiva y confiable.

Conclusiones

- El código Hamming(12,8) recupera eficazmente la mayoría de los mensajes mientras la tasa de errores se mantenga por debajo del 3 %.
- CRC-32 detecta casi todas las alteraciones sin dejar errores no detectados, aunque requiere retransmisiones para corregir.
- Con BER bajos, Hamming reduce notablemente las retransmisiones en comparación con un esquema de detección pura.
- La tasa prácticamente nula de falsos negativos de CRC-32 lo hace ideal cuando la integridad de los datos es prioritaria.
- La elección entre detección y corrección depende del equilibrio entre latencia (retrasmisiones) y necesidad de integridad sin interacción.

Referencias

Hamming, R. W. (1950). Error detecting and error correcting codes. Bell System Technical Journal, 29(2), 147–160.

International Organization for Standardization. (2002). ISO/IEC 13239: Information technology—High-level data link control (HDLC) procedures. ISO.

Tanenbaum, A. S., & Wetherall, D. J. (2011). Computer Networks (5th ed.). Pearson.

Stallings, W. (2013). Data and Computer Communications (10th ed.). Pearson.

zlib Development Team. (1995). zlib 1.2.11 manual. Retrieved from https://zlib.net/manual.html