De la demodulación a la descodificación: Hacia una comprensión e implementación completas de LoRa PHY

ZHENQIANG XU, SHUAI TONG, PENGJIN XIE y JILIANG WANG, Escuela de Software,
Universidad Tsinghua, China

LoRa, como representante de la tecnología de redes de área extensa de baja potencia, ha atraído una gran atención tanto del mundo académico como de la industria. Sin embargo, el conocimiento actual de LoRa dista mucho de ser completo, y las implementaciones presentan una gran brecha de rendimiento en SNR y velocidad de recepción de paquetes. Este artículo presenta una comprensión completa del protocolo de capa física (PHY) de LoRa y revela las razones fundamentales de la diferencia de rendimiento. Presentamos la primera implementación completa de LoRa PHY con una garantía de rendimiento demostrable. Mejoramos la demodulación para que funcione con una SNR extremadamente baja (20 dB) y validamos analíticamente el rendimiento, mientras que muchos trabajos existentes requieren una SNR> 0. Derivamos el orden y los parámetros de las operaciones de descodificación, incluyendo el blanqueamiento, la corrección de errores, el desentrelazado, etc., aprovechando las características de LoRa y la manipulación de paquetes. Implementamos un LoRa completo en tiempo real en la plataforma GNU Radio y llevamos a cabo amplios experimentos. Nuestro método puede lograr (1) una tasa de éxito de descodificación del 100%, mientras que los métodos existentes pueden soportar como máximo el 66,7%, (2) una sensibilidad de 142 dBm, que es la sensibilidad límite de la LoRa básica, y (3) un alcance de comunicación de 3.600 m en el área urbana, incluso mejor que la LoRa básica en el mismo entorno.

Conceptos CCS: - **Redes Protocolos de red**; **Análisis del rendimiento de redes**; **Redes de área extensa;**

Palabras y frases clave adicionales: Red de área extensa de baja potencia, LoRa, capa física

Formato de referencia ACM:
Zhenqiang Xu, Shuai Tong, Pengjin Xie y Jiliang Wang. 2022. From Demodulation to Decoding: Toward Complete LoRa PHY Understanding and Implementation. *ACM Trans. Sensor Netw.* 18, 4, Article 64 (Decem- ber 2022), 27 pages.
https://doi.org/10.1145/3546869

1 INTRODUCCIÓN

La tecnología **de red de área extensa de baja potencia (LPWAN)** ha demostrado ser muy prometedora para conectar millones de dispositivos en el **Internet de las cosas (IoT)** al proporcionar comunicación de larga distancia y baja potencia con una SNR muy baja. LoRa, como tecnología LPWAN representativa, ha sido adoptada recientemente de forma generalizada tanto en el mundo académico como en la industria [1-4]. Se ha utilizado en varias aplicaciones IoT, por ejemplo, ciudad inteligente, agricultura inteligente y logística inteligente [5-8]. También atrae a un gran

Este trabajo ha sido financiado en parte por NSFC No. 62172250, No. 61932013, Programa de Investigación Científica de la Iniciativa de la Universidad de Tsinghua.
Dirección de los autores: Z. Xu, S. Tong, P. Xie y J. Wang (autor correspondiente), School of Software, Tsinghua Univer- sity, Pekín, China; correos electrónicos: xu-zq17@mails.tsinghua.edu.cn, tl19@mails.tsinghua.edu.cn, xiepengjin@tsinghua.edu.cn, jiliangwang@tsinghua.edu.cn.

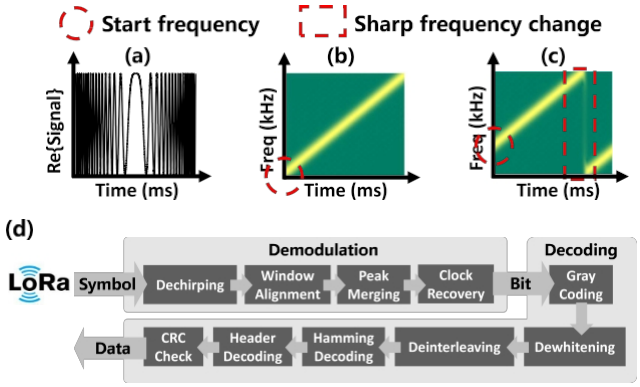

Fig. 1. (a) Parte real de un símbolo up-chirp base. (b) Símbolo up-chirp base. (c) Símbolo desplazado. (d) Procedimiento completo de LoRa PHY.

Varios trabajos de investigación incluyen la resolución de colisiones [9, 10], la retrodispersión de LoRa [11-13], la decodificación de señales débiles [14, 15], etcétera.

LoRa adopta el mecanismo de modulación **Chirp Spread Spectrum (CSS)**, que proporciona antiinterferencias y capacidad de comunicación de largo alcance. Como se muestra en la Figura 1(a) y (b), un símbolo base en **el protocolo de capa física (PHY)** LoRa es un chirp cuya frecuencia aumenta linealmente con el tiempo. La frecuencia inicial (f_{start}) de un símbolo representa la información codificada. Un símbolo LoRa tiene dos segmentos con una fuerte caída de frecuencia, como se muestra en la Figura 1(c). Aunque LoRa ha atraído mucha atención en el mundo académico y en la industria, los detalles de LoRa PHY, es decir, cómo LoRa demodula y decodifica la señal recibida, aún nos son desconocidos, porque LoRa PHY es un protocolo cerrado propiedad de Semtech Corporation. En este trabajo, pretendemos presentar una comprensión completa de la capa física de LoRa y proporcionar una herramienta mejor para analizar la red LoRa. Revelamos que la mayoría de los conocimientos existentes son incompletos e incluso incorrectos. Como resultado, incluso la implementación LoRa más utilizada**,** ***rpp0/gr-lora*** **(RPP0)** [16], tiene una gran diferencia de rendimiento con el hardware real, por ejemplo, **la tasa de recepción de paquetes (PRR)** de RPP0 es sólo del 20,0%, y RPP0 requiere una SNR mucho mayor que LoRa.

Hay dos pasos principales para entender LoRa PHY: demodulación y decodificación.[1] Demodu-
La demodulación traduce los símbolos en bits brutos, mientras que la descodificación traduce estos bits brutos en bytes de datos con sentido. El objetivo de la demodulación LoRa es derivar f_{start} de cada símbolo. La demodulación determina directamente el rendimiento del receptor. La demodulación LoRa adoptada actualmente traduce primero la señal a un tono único multiplicando cada símbolo por un chirp linealmente decreciente y extrae la frecuencia del tono único como frecuencia de inicio. Esta operación puede dar lugar a una gran pérdida de SNR debido a la pérdida de señal en la traducción y a la incapacidad de aprovechar las características de LoRa, lo que impide a LoRa la comunicación de largo alcance con baja SNR. Explicamos las razones exactas de la pérdida de señal en la sección 3.1.

Para la descodificación, los trabajos existentes [16-21] no pueden deducir el orden y los parámetros de las operaciones de descodificación. Así, tienen una PRR muy baja (por ejemplo, por debajo del 66,7%) incluso para una SNR alta debido a la falta de comprensión correcta del proceso de descodificación LoRa. Mientras tanto, suelen ser incompletos y exagerados; por ejemplo, los métodos de las referencias [16-18] afirman que pueden soportar todos los **factores de propagación (SF)**, pero nuestros experimentos muestran que no lo consiguen.

[1]Para el transmisor, la modulación y la codificación. Aquí sólo discutimos el comportamiento del receptor por simplicidad.

Tabla 1. Comparación de BastilleResearch/gr-lora (BR), RPP0 y tapparelj/gr-lora_sdr (TAPP) y nuestra implementación

	BR [17]	RPP0 [16]	TAPP [18]	Nuestra
Todos los SF	✗	✗	✗	✓
Todos los CR	✓	✓	✓	✓
CRC	✗	✗	✓	✓
Modo explícito	✗	✓	✓	✓
Modo implícito	✓	✗	✓	✓
Recuperación de reloj	✗	✗	✗	✓
Soporte SNR bajo	✗	✗	✗	✓
Tiempo real	✓	✓	✗	✓
Cobertura de paquetes	4.2%	20.0%	66.7%	100%

Este artículo proporciona una comprensión total y una completa pila LoRa PHY en tiempo real con una garantía de rendimiento demostrable. En la parte de demodulación, revelamos que la razón fundamental de la pérdida de SNR es la desalineación de fase debida a la desalineación de la ventana y al desfase interno del símbolo. En primer lugar, compensamos la desalineación de la ventana mediante una medición precisa del desfase de la ventana. Para abordar el desfase interno de los símbolos, a diferencia de los trabajos existentes con frecuencia de muestreo B (ancho de banda), sobremuestreamos la señal con 2B para calcular los picos de cada segmento por separado. A continuación, proponemos un método para fusionar esos dos picos con una pérdida de sensibilidad mínima. Demostramos teóricamente que la sensibilidad de nuestro método se aproxima mucho a la demodulación "perfecta". Además, proponemos un algoritmo dinámico de compensación de la deriva del reloj para dispositivos LoRa de bajo coste y paquetes LoRa relativamente largos.

En la parte de decodificación, mostramos cómo LoRa PHY traduce los bits de la parte de demodulación a paquetes con sentido. Según las observaciones de la fórmula oficial [22] para calcular el número de símbolos de un paquete LoRa, inferimos y verificamos la estructura del paquete de la caja negra LoRa manipulando el contenido del paquete. Basándonos en la estructura del paquete, analizamos los requisitos de cada proceso de descodificación y derivamos el orden de los procesos de descodificación para la codificación Gray, el desentrelazado, la descodificación Hamming y el blanqueamiento. A continuación, aprovechando el orden de descodificación inferido y la estructura de los paquetes, manipulamos los paquetes transmitidos para deducir los parámetros de configuración del proceso de descodificación, la estructura de la cabecera y el polinomio CRC.

Implementamos una **radio definida por software (SDR)** LoRa PHY en tiempo real, como se muestra en la Figura 1(d). La Tabla 1 muestra la comparación con los últimos avances. Nuestra implementación supera estos enfoques al (1) proporcionar decodificación en tiempo real con un requisito de SNR muy bajo, (2) soportar todos los modos y parámetros de LoRa, y (3) trabajar de forma robusta en dispositivos de bajo coste con desviación del reloj y paquetes largos. Demostramos teóricamente que nuestra implementación podría funcionar con SNR= 20 dB. El orden de las operaciones de descodificación también es demostrable, lo que puede verificarse mediante la secuencia aleatoria generada (Sección 4.3). En nuestros experimentos, el 100% de los paquetes se decodificaron correctamente. Por tanto, consideramos que la configuración es fiable.

Aportaciones.

- Mostramos la brecha de rendimiento entre las implementaciones LoRa existentes y el LoRa básico y revelamos las razones fundamentales. Presentamos una comprensión exhaustiva de la demodulación y descodificación de LoRa.
- Implementamos una PHY LoRa completa y en tiempo real en la plataforma GNU Radio SDR.[2] Basándonos en nuestro análisis, podemos incluso analizar y mejorar el rendimiento de los productos básicos

[2]https://github.com/jkadbear/LoRaPHY.

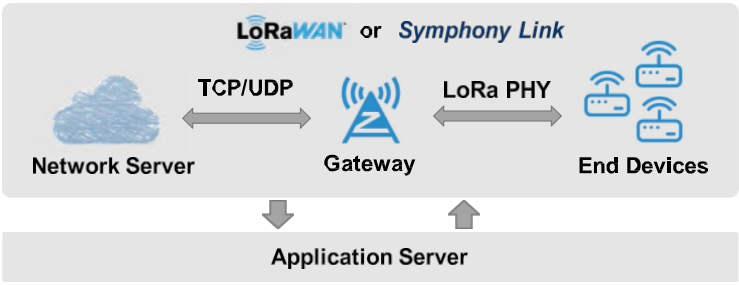

Fig. 2. Arquitectura de red LoRa. El protocolo LoRaWAN o Symphony Link controla el modo en que un servidor de red interactúa con los dispositivos finales.

LoRa. Por ejemplo, demostramos que el chip LoRa básico es vulnerable al "*ataque de desalineación de fase*" (sección 6.5) y cómo solucionar esta vulnerabilidad.

- Llevamos a cabo amplios experimentos para verificar el rendimiento. Los resultados muestran que la sensibilidad de nuestra implementación alcanza los 142 dBm, que es la sensibilidad límite de LoRa básico. Nuestra implementación tiene un alcance de comunicación efectivo de 3.600 m en el entorno urbano, mientras que las pasarelas RPI básicas (Sección 5) sólo alcanzan los máximo de 2.800 m. Nuestra implementación LoRa tiene una tasa de éxito de descodificación mucho mayor (100%) que los trabajos existentes (≤ 66,7%).

2 ANTECEDENTES

2.1 Hechos conocidos y desconocidos de LoRa

La figura 2 muestra la arquitectura principal de la red LoRa. El protocolo LoRaWAN [23] o Symphony Link [24] controla toda la red LoRa y proporciona una interfaz de datos al servidor de aplicaciones. Hay tres entidades de comunicación principales en la red LoRa: el servidor de red, la pasarela y el dispositivo final. El servidor de red se comunica con la pasarela utilizando una conexión tradicional a Internet, mientras que los dispositivos finales intercambian datos con la pasarela utilizando el protocolo inalámbrico LoRa PHY. Casi todos los componentes y protocolos utilizados en la Figura 2 tienen implementaciones completas de código abierto, excepto LoRa PHY. Aunque existen algunos trabajos de ingeniería inversa, la comprensión de LoRa PHY aún no es completa. A continuación, demostramos algunos conceptos básicos de LoRa PHY y discutimos los detalles sobre demodulación y decodificación en las secciones 3 y 4.

LoRa PHY emplea CSS para alcanzar una transmisión de largo alcance. La unidad básica de comunicación en LoRa PHY es un símbolo chirp. Como se muestra en la Figura 1(b), denominamos a este símbolo chirp no desplazado *chirp base*. Cuando la frecuencia aumenta con el tiempo, lo denominamos up-chirp y, en caso contrario, down-chirp. La novedad de la modulación LoRa consiste en utilizar la frecuencia de inicio de un símbolo desplazado cíclicamente para codificar datos. La figura 1(c) muestra un símbolo LoRa con dos segmentos chirp. En las especificaciones LoRa [22, 25, 26], el número de bits codificados por un símbolo es SF. El SF satisface $2^{SF} = B\,T$, donde B es el ancho de banda y T es el periodo chirp. Hay como máximo 2^{SF} símbolos up-chirp desplazados cíclicamente dado un SF específico. Un up-chirp puede representarse como

$$chirp\,(t\,; f_0) = Ae^{j\,2\pi\,(f_0 + \frac{B}{2T}t)t}\quad , \tag{1}$$

donde A es la amplitud y f_0 es la frecuencia inicial ($t = 0$).

La Figura 3 muestra la comprensión básica de un paquete LoRa estándar. En general, consta de tres partes: preámbulo, **delimitador de trama de inicio (SFD)** y símbolos de datos. El preámbulo es una serie de

Fig. 3. Un paquete LoRa en el plano tiempo-frecuencia.

up-chirps seguidos de dos up-chirps que indican el ID de red.[3]El SFD son 2,25 down-chirps que indican el inicio de los símbolos de datos. Los símbolos de datos incluyen la cabecera PHY, la carga útil y el CRC de la carga útil (la cabecera y el CRC son opcionales).

LoRa admite las siguientes configuraciones. **Tasa de código (CR)**: LoRa aplica código Hamming de tasa de codificación $\frac{4}{5}$, $\frac{4}{6}$, $\frac{4}{7}$, o $\frac{4}{8}$[4][22]. **Optimización de baja tasa de datos (LDRO)**: Cuando se envían paquetes largos, LoRa permite el modo LDRO para mejorar la estabilidad a costa de la velocidad de datos. Modo implícito/explícito: En el modo explícito, hay una cabecera PHY en el paquete, mientras que en el modo implícito, no hay cabecera PHY.

Desconocido:(4) (1) La demodulación ideal de LoRa, es decir, cómo un símbolo se traduce a bits, es todavía desconocida, que es la clave para que LoRa consiga una SNR baja y una comunicación a larga distancia.

(2) Ninguno de los trabajos existentes propone una comprensión completa de la estructura de los paquetes LoRa.

(3) Se desconocen los detalles de los modos específicos de LoRa, por ejemplo, el modo LDRO.

2.2 Implementaciones de código abierto más avanzadas

El espacio de la investigación previa en comunicación LoRa es bastante rico, incluyendo la decodificación de colisiones LoRa [1, 2, 4, 9, 10, 27], la retrodispersión basada en chirp LoRa [11-13, 28], la optimización de redes LoRa [15, 29], etcétera. Desgraciadamente, la mayoría de ellos no publican implementaciones accesibles al público. Teniendo en cuenta que la arquitectura de red LoRa es completamente nueva y que la comunidad no la conoce en su totalidad, cualquier nuevo esfuerzo de código abierto merece ser valorado. Por lo tanto, primero resumimos los proyectos de código abierto relacionados con LoRa disponibles públicamente.

Implementaciones de la capa superior. La capa superior de la red LoRa cuenta con numerosas implementaciones de código abierto. Para la pasarela LoRa y el dispositivo final, la corporación Semtech proporciona implementaciones integradas oficiales [30, 31]. Para los servidores de red, ChirpStack [32] es una implementación ampliamente utilizada y compatible con LoRaWAN. LoRaWAN-Server [33] ofrece una combinación compacta de servidor de red y servidor de aplicaciones. El código de más bajo nivel aquí es el controlador para los chips LoRa. Por lo tanto, no incluye el mecanismo de capa física de la comunicación LoRa.

Implementaciones de la capa física. Dado que LoRa PHY es un protocolo propietario perteneciente a la corporación Semtech, no existe ningún documento público que explique todos los detalles de LoRa PHY. Las investigaciones y sistemas existentes se basan principalmente en tres implementaciones SDR de código abierto: BR [17], RPP0
[16] y TAPP [18]. BR se centra en el modo de cabecera implícita del paquete LoRa. BR aplica demodulación basada en dechirp- ing. Ignora algunos detalles cruciales sobre el dechirping y no es fiable con una SNR baja. Discutiremos los detalles en la Sección 3. Además, la implementación de BR sólo admite la descodificación de paquetes LoRa con SF=8. Nuestras pruebas muestran que BR sólo puede descodificar paquetes LoRa con SF=8. Nuestras pruebas muestran que BR sólo puede descodificar los cuatro primeros bytes de un paquete y no consigue descodificar ningún paquete completo con más de cuatro bytes. RPP0 se centra en la descodificación explícita de paquetes LoRa en modo cabecera. Como se muestra en la Figura 1(c), RPP0 demodula los símbolos LoRa encontrando cambios bruscos de frecuencia y utilizándolos para estimar las frecuencias de inicio de los paquetes LoRa.

[3]Los dos símbolos no son típicamente up-chirps de base. Se utilizan como máscaras de red para separar diferentes redes. Por ejemplo, se establecen en 0x0304 para la red pública en LoRaWAN.

[4]Aunque los trabajos existentes han revelado muchos detalles sobre LoRa PHY, el rendimiento de demodulación no ideal y la tasa de éxito de descodificación no del
100% de éxito en la descodificación hacen que sus resultados sean incompletos.

símbolos. RPP0 se basa en información en el dominio del tiempo y no puede descodificar paquetes LoRa para SNR< 0. TAPP es una implementación reciente de LoRa. TAPP no está optimizado para la demodulación LoRa, lo que se traduce en una elevada sobrecarga computacional. La PRR de TAPP disminuye cuando el descodificador no puede consumir los datos de entrada a tiempo. Según nuestra evaluación, su pérdida de paquetes alcanza el 70% para una velocidad de datos de un paquete por segundo. El PRR medio de TAPP es sólo del 66,7%. En cuanto a la descodificación, ninguno de los trabajos existentes puede proporcionar una capacidad de descodificación completa, por lo que sólo pueden descodificar una parte de los paquetes LoRa. Además, hay otros trabajos relacionados con LoRa PHY que proporcionan implementaciones de código abierto. Charm [14] se centra en la descodificación de señales débiles aprovechando la estructura multipuerta de la red LoRa. Sin embargo, el decodificador Charm [34] sólo se implementa en modo offiine con MATLAB. Además, el autor ignora el procedimiento de descodificación y el problema de alineación de fase (véase la Sección 3). TinySDR [35] es una plataforma SDR adaptada a las redes de Internet de las Cosas. El autor implementa un modulador LoRa [36] en FPGA, que nos muestra un cómodo método de generación de chirp por hardware. Pero el proceso de demodulación suele ser mucho más complicado que el de modulación, y actualmente no existe ningún demodulador LoRa basado en FPGA de código abierto. Además, el proceso de decodificación tampoco se tiene en cuenta en TinySDR. Vemos que los trabajos existentes no han proporcionado una implementación totalmente utilizable y una comprensión completa de LoRa PHY.

Limitaciones fundamentales de los trabajos existentes. (1) Los trabajos existentes no pueden funcionar con una SNR baja, por lo que no tienen la ventaja crucial de LoRa. (2) Los trabajos existentes no pueden proporcionar una capacidad de descodificación completa para LoRa, y su rendimiento es significativamente inferior al del hardware LoRa básico. No pueden proporcionar soporte completo de LoRa, es decir, modo LDRO, todas las SF, etc.

Nuestro objetivo. Nuestro objetivo es proporcionar una implementación completa de LoRa PHY, incluyendo demodulación y decodificación, para permitir la recepción de señales con SNR extremadamente baja y todos los parámetros/modos de LoRa.

3 DEMODULACIÓN

En esta sección, revelamos el proceso de demodulación de LoRa y presentamos cómo romper las funda- mentales limitaciones de las implementaciones LoRa existentes.

3.1 Descodificación alineada en fase

El demodulador LoRa convierte los símbolos chirp en flujos de bits deschirpando primero la señal recibida. El proceso de dechirping consta principalmente de dos pasos: En primer lugar, multiplica cada chirp recibido por un chirp base descendente y, a continuación, realiza la transformación de Fourier en los resultados de la multiplicación, traduciendo los chirps a picos de energía en el dominio de la frecuencia. Idealmente, la multiplicación dará como resultado señales de tono único con fase continua, cuya energía puede acumularse en un único pico, como se muestra en la Figura 4(a3) (denominada IDEAL). Sin embargo, en la práctica se produce un desajuste de fase inevitable cuando la frecuencia del chirp desciende de su máximo a su mínimo. Este desajuste de fase está inducido principalmente por las inestabilidades de hardware de los dispositivos LoRa de bajo coste y se distribuye aleatoriamente entre 0 y 2π . Con la desalineación de fase, los picos de energía de la transformación de Fourier se distorsionan gravemente, como se muestra en la Figura 4(a4), lo que limita el rendimiento de la demodulación LoRa, especialmente con una SNR baja.

Los trabajos existentes no resuelven el problema de la desalineación de fase en la descirpitación. Por lo tanto, su rendimiento se deteriora drásticamente en escenarios de baja SNR, lejos de la sensibilidad ideal de la demodulación LoRa. Nuestro objetivo es acercarnos a la sensibilidad ideal de la demodulación LoRa bajo desalineación de fase. El diseño de nuestro demodulador LoRa consta principalmente de los siguientes componentes. **Alineación de ventanas.** Necesitamos alinear con precisión la ventana de demodulación (nivel de puntos de muestra) con cada símbolo para retener la energía de todo el símbolo para el proceso de demodulación. Los trabajos tradicionales utilizan la correlación del preámbulo para alinear la ventana. Sin embargo, el mero uso de la correlación preámbulo no permite lograr una alineación precisa debido a la **desviación de la frecuencia portadora (CFO)**. Aprovechamos la parte SFD del paquete LoRa para alinear la ventana, eliminando el impacto de la CFO.

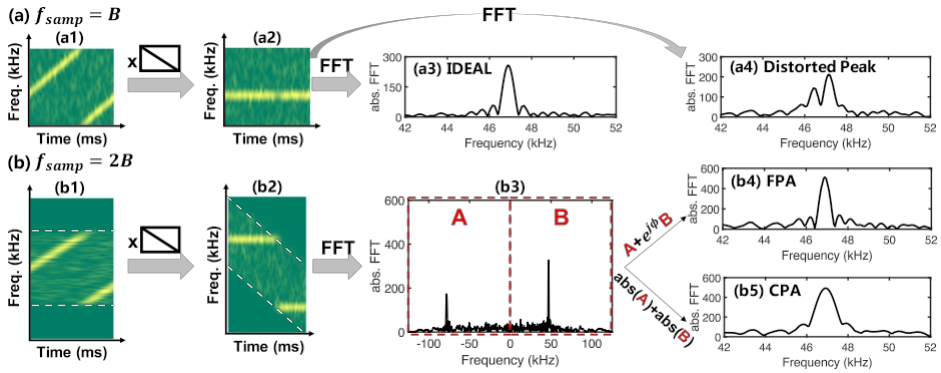

Fig. 4. El paso principal del dechirping es multiplicar un down-chirp y luego hacer la FFT. (a) Dechirping normal con pérdida de SNR. Para una situación ideal, obtenemos un pico como (a3). Pero en una situación no ideal, el pico de frecuencia se distorsionaría como (a4), lo que produciría resultados de demodulación erróneos. (b) Nuestra propuesta de dechirping alineado en fase. Utilizamos el filtro de paso bajo y el sobremuestreo para separar los dos segmentos de chirp en un símbolo LoRa y luego combinarlos. (b4) y (b5) muestran los resultados de dechirping alineado en fase fina y gruesa, respectivamente, que son estables en cualquier situación.

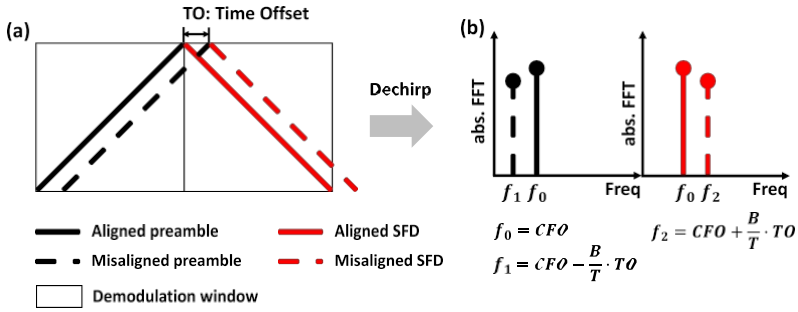

Fig. 5. Alineación de ventanas usando up-chirps y down-chirps de base.

CFO. La operación clave consiste en combinar los down-chirps en SFD junto con los up-chirps en el pre-amblaje. A ambos se les aplica el dechirping (el down-chirp se multiplica por un up-chirp base). Si la ventana de demodulación está perfectamente alineada con la señal, el pico del up-chirp y el pico del down-chirp aparecen en la misma frecuencia, independientemente de la CFO, como se muestra en la figura 5. En caso contrario, hay una diferencia significativa entre el pico del up-chirp y el pico del down-chirp. De lo contrario, existe una diferencia significativa en la frecuencia de los dos picos, dependiendo del desfase temporal entre el chirp y la ventana de demodulación. Por lo tanto, podemos conseguir una alineación precisa basándonos en la frecuencia de los picos correspondientes al preámbulo y a los SFD.

Sobremuestreo y fusión de picos. Anteriormente hemos revelado que el rea- son fundamental de la pérdida de SNR en la demodulación es la distorsión de picos debida al desalineamiento de fase cuando la frecuencia del chirp cae de su máximo a su mínimo. Para solucionar este problema, proponemos un enfoque basado en el sobremuestreo que recupera el pico de distorsión en caso de desalineación de fase y se aproxima a la sensibilidad ideal de la demodulación LoRa.

Como se ilustra en la Sección 2, LoRa modula las señales desplazando cíclicamente la frecuencia de un chirp ascendente base. Cada chirp modulado consiste en dos segmentos de chirp, uno con la frecuencia inicial de f_0 y el otro con f_0 B. Los trabajos existentes demodulan los símbolos chirp recibidos usando una frecuencia de muestreo igual al ancho de banda del chirp B. Así, después de multiplicar con la base

Cuando se produce un chirp descendente, ambos segmentos del chirp se convierten en señales monotono de la misma frecuencia debido al aliasing de frecuencia. Sin embargo, existe una desalineación de fase inevitable entre estos dos segmentos de chirp. Al realizar la transformación de Fourier en todo el símbolo, la energía de estos dos segmentos de chirp se suma de forma destructiva, lo que provoca una distorsión de picos y una pérdida de SNR para la demodulación LoRa. Los trabajos existentes no tienen ni idea del valor de la desalineación de fase. Por tanto, no pueden distinguir la señal de estos dos segmentos de chirp y no corrigen el pico de distorsión.

Proponemos una estrategia basada en el sobremuestreo para separar eficazmente los dos segmentos de chirp en el dominio de la frecuencia. En concreto, como las frecuencias iniciales de los dos segmentos de chirp son f_0y f_0B, cuando se sobremuestrea la señal con una frecuencia superior a 2B (es decir, F_s $2B$), el dechirping da lugar a dos picos distintos localizados por separado en f_0y F_s $B+ f_0$, como se muestra en la Figura 4(b3). Como no hay desalineación de fase dentro de cada segmento de chirp, estos dos picos están libres de distorsión. Esto nos da la oportunidad de concentrar perfectamente la energía de todo el chirp sumando coherentemente estos dos picos de segmentos de chirp y eliminando así la influencia de la desalineación de fase. En el resto de esta sección, ilustramos cómo fusionar estos dos picos de energía en el dominio de la frecuencia de forma eficiente con una pequeña pérdida de sensibilidad para la demodulación LoRa.

Debido a la existencia de desalineación de fase, la fase de los dos picos de la figura 4(b3) está fuera de coherencia. Por lo tanto, la suma directa de estos dos picos complejos no aumentará la altura del pico o incluso se cancelará debido a la diferencia de fase de estos dos picos. Proponemos **la alineación de fase de grano fino (FPA)** para buscar todos los valores posibles de desalineación de fase $\frac{i\times 2\pi}{}$ 0 1 1 y compensarlo antes de sumar dos picos. Iteramos los posibles $\Delta\varphi = {}_{k}$ $(i = , , \dots , k -)$ k fase o ffset y seleccionar el $\Delta\varphi$ que genere el pico más alto. El resultado del FPA se muestra en la Figura 4(b4). Comparado con la Figura 4(a3), FPA puede acercarse al rendimiento de IDEAL al compensarse la diferencia de fase.

El proceso de búsqueda para determinar el valor de la desalineación de fase tiene una elevada complejidad computacional. Para reducir esta sobrecarga, proponemos **además la Alineación de Fase de Grano Grueso (CPA)**, que requiere mucha menos sobrecarga computacional y, por tanto, puede funcionar en hardware de bajo coste con recursos computacionales muy limitados. La observación clave es que la amplitud del pico es proporcional a la energía del segmento de chirp correspondiente. Dado que la energía de todo el símbolo es exactamente la suma de los dos segmentos del chirp, si sumamos directamente el valor absoluto (amplitud) de la parte A y la parte B, el pico resultante tendrá la misma altura que el pico IDEAL. El resultado del CPA se muestra en la figura 4(b5). Vemos que CPA cambia significativamente la forma del pico, es decir, aumenta la anchura del lóbulo principal. Además, CPA también eleva el nivel de ruido durante el proceso de suma, por lo que degrada ligeramente el rendimiento de demodulación en comparación con FPA. Dependiendo de la potencia de cálculo del receptor, podemos cambiar entre los métodos FPA y CPA. A continuación, analizamos teóricamente la **tasa de error de símbolo (SER)** con respecto a la dif-

de SNR para IDEAL, FPA y CPA.

Suponemos que la señal LoRa se transmite a través de un canal **con ruido gaussiano blanco aditivo (AWGN)**, en el que el ruido se modela siguiendo la distribución gaussiana compleja. Durante el proceso de dechirping, tanto los símbolos chirp como el ruido se transforman en picos de energía en el dominio de la frecuencia. Se produce un error de símbolo cuando alguno de los picos de ruido supera el pico del símbolo objetivo. Supongamos que la altura del pico del símbolo objetivo es h_d, y el pico máximo correspondiente al ruido es h_n; podemos derivar la tasa de error de símbolo esperada como

$$SER = P(h_d < h_n). \tag{2}$$

La SER es teóricamente una función de la SNR. Los detalles del cálculo de h_dy $h_n$para IDEAL, FPA y CPA se detallan en la Sección 8. El rendimiento teórico de IDEAL, FPA y CPA con SF8/12 se representa en

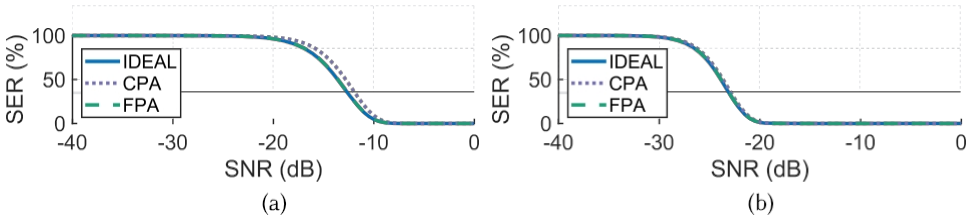

Fig. 6. Curva teórica SER-SNR para (a) SF8 y (b) SF12.

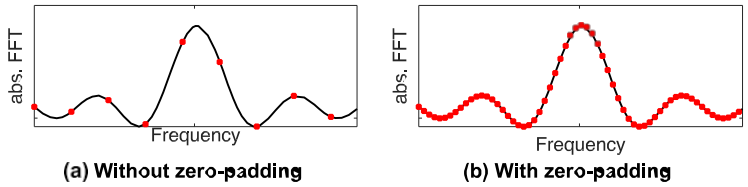

Fig. 7. Efectos de relleno cero. Los puntos rojos representan los resultados de la DFT tras el desescalonamiento. Las líneas negras representan los resultados de la DTFT tras el deschirpado.

Figura 6.[5]La longitud del paso de búsqueda en FPA (es decir, *1/k*) se establece empíricamente en 1/16 (un compromiso entre el rendimiento y la sobrecarga de cálculo). Vemos que tanto FPA como CPA tienen un rendimiento muy cercano a IDEAL para SF12, mientras que el SER teórico de CPA en SF8 es ligeramente mayor que el de los otros dos métodos. En resumen, el análisis teórico nos dice que FPA y CPA son dos métodos prácticos y eficaces en la demodulación LoRa, que se valida además en los resultados experimentales de la Sección 6. Nuestro análisis teórico también muestra que LoRa podría funcionar con una SNR extremadamente baja (es decir, SNR tan baja como 20 dB), como puede verse en la Figura 6(b).

Refinamiento de picos. Anteriormente hemos explicado cómo convertir los chirridos recibidos en picos en el dominio de la frecuencia. A continuación, necesitamos estimar la frecuencia de pico más alta para recuperar los bits de datos modulados. Sin embargo, en la práctica, la estimación de la altura de los picos está estrechamente relacionada con la res- olución de la frecuencia tras la transformación de Fourier. La línea negra de la figura 7(a) muestra el resultado de la **transformada de Fourier en tiempo discreto (DTFT)** ideal en la señal de destino, que es una función continua de la frecuencia. En la práctica, el receptor sólo realiza la **transformada discreta de Fourier (DFT)**, una versión muestreada de la DTFT en el dominio de la frecuencia, sobre la señal de destino. La salida de la DFT está marcada con puntos rojos en la figura 7. La DFT ingenua tiene una resolución de baja frecuencia. Por lo tanto, sus resultados se distribuyen escasamente sobre la curva DTFT ideal, lo que disminuye la altura del pico derivado y también afecta a la estimación del pico. Aplicamos un relleno cero a la señal en el dominio temporal para obtener una estimación precisa de los picos, equivalente a la interpolación en el dominio de la frecuencia. Como se muestra en la Figura 7(b), con zero-padding, la resolución de frecuencia de la DFT mejora significativamente, beneficiando nuestra estimación de picos. En la sección 6.3 se muestra que la cuadruplicación del zero-padding logra un equilibrio entre la sobrecarga de cálculo y la sensibilidad de demodulación de . El beneficio de más relleno cero es marginal.

3.2 Recuperación del reloj

Dado que LoRa se aplica habitualmente en dispositivos de bajo coste, es posible que los osciladores equipados no tengan una gran precisión, lo que da lugar a valores de bins imprecisos. La tabla 2 muestra los ocho primeros bins de pico desescalados de dos paquetes de un dispositivo LoRa con un CFO de unos 16 kHz. Observamos que la parte fraccional

[5]Los parámetros a nivel de bit, como CR y CRC, son irrelevantes para la SER a nivel de símbolo. Hay que tener en cuenta que el ancho de banda no aparece en la fórmula SER anterior, porque la SNR es función del ancho de banda (si aumentamos el ancho de banda sin aumentar la potencia de transmisión, la SNR en banda caerá).

Tabla 2. Deriva de pico en SF10 y SF12 (con ancho de banda de 125 kHz)

		First 8 Peak Bins							
SF10 (LDRO off)	Expected	677.0	333.0	861.0	93.0	853.0	149.0	789.0	597.0
	Received	677.2	333.2	861.3	93.3	853.4	149.4	789.4	597.5
SF12 (LDRO on)	Expected	1757.0 = 011011 011101	3453.0 = 110101 111101	673.0 = 001010 100001	3757.0 = 111010 101101	2409.0 = 100101 101001	809.0 = 001100 101001	2981.0 = 101110 100101	2545.0 = 100111 110001
	Received	1757.8 = 011011 011101 +0.8	3453.9 = 110101 111101 +0.9	674.0 = 001010 100010 +0.0	3758.2 = 111010 101110 +0.2	2410.3 = 100101 101010 +0.3	810.4 = 001100 101010 +0.4	2982.5 = 101110 100110 +0.5	2546.6 = 100111 110010 +0.6

Esperados: los bins ideales sin error de reloj; recibidos: los bins reales recibidos.

de los bins de pico sigue derivando con el tiempo. Si no cancelamos esa deriva, los bins fraccionarios se acumularán e incurrirán en errores en el bin entero. Estas derivas de los bins proceden principalmente de la **desviación de la frecuencia de muestreo (SFO)**. Debido a la diferencia en la frecuencia de muestreo, el símbolo LoRa real es τ más corto (o más largo) que el periodo de símbolo estándar T . Por lo tanto, la frecuencia de inicio del siguiente símbolo tiene una deriva adicional como

$$\Delta f = \frac{\tau}{T} - \frac{B}{} \tag{3}$$

Dado que la frecuencia portadora y la frecuencia de muestreo son generadas por el mismo oscilador, tenemos

$$\frac{\tau}{T} = \frac{SFO}{fsamp} = \frac{\Delta fosc}{fosc} = \frac{CFO}{fRF}, \tag{4}$$

donde f_{RF}es la frecuencia de la portadora de referencia, f_{osc}es la frecuencia del oscilador de referencia y Δf_{osc}es el sesgo de la frecuencia del oscilador. Tras la conversión a valores bin, la deriva bin estimada entre dos símbolos consecutivos es[6]

$$\Delta bin = \frac{\Delta f}{(2)^{(SF)}(B)(/)} = \frac{CFO}{fRF} 2^{SF}. \tag{5}$$

Por ejemplo, cuando CFO= 16 kHz, f_{RF}= 470 MHz, y SF= 10, la deriva de bin estimada Δbin 0,035, que coincide con la tendencia de la Tabla 2. Restamos Δbin acumulado de los bins de pico antes de la descodificación.

Otra cosa que tenemos que considerar aquí es el modo LDRO. Normalmente, cuando un periodo de chirp es demasiado largo (es decir, T> 16 ms), LDRO se activa automáticamente en los chips LoRa [22]. En el modo LDRO, como se ve en la parte inferior de la Tabla 2, los valores bin esperados siempre tienen la forma de $4n$+ 1, lo que significa que los dos **Bits Menos Significativos (LSB)** no codifican datos. Este diseño pretende proteger mejor los datos, ya que los bits más bajos son más vulnerables en la transmisión de paquetes largos. Para las señales LoRa de corta duración, los dos LSB son suficientemente estables para codificar datos (por ejemplo, SF10 y ancho de banda 125 kHz). No obstante, en el experimento comprobamos que los ocho primeros símbolos están siempre en modo LDRO para proteger la información de cabecera (Sección 4).

[6]Sin relleno cero, un bin entero en LoRa siempre representa una frecuencia de ($\frac{B)}{2^{SF}}$, independientemente del número de puntos FFT o la frecuencia de muestreo.

4 DECODING

4.1 Visión general

Esta sección sobre descodificación *no es una revisión* de trabajos anteriores. Nuestro objetivo es proporcionar una in- ferencia demostrable sobre la estructura de los paquetes LoRa, el orden de descodificación y las configuraciones. Por las hojas de datos oficiales de LoRa [22, 25, 37] y las patentes [26], conocemos la ecuación (6) para calcular el número de símbolos de un paquete. En cambio, desconocemos la estructura del paquete. También sabemos que hay cuatro procesos principales en la descodificación: Codificación Gray (G), desentrelazado (I), decodificación Hamming (H) y blanqueamiento (W). En cambio, se desconocen el orden de los procesos y los parámetros de configuración de cada uno de ellos. Para mostrar cómo LoRa PHY traduce los bits del módulo de demodulación a paquetes con sentido, analizamos la descodificación LoRa en tres pasos. En primer lugar, deducimos la estructura del paquete de la caja negra LoRa aprovechando la fórmula del número de símbolos. A continuación, revelamos el orden de los cuatro procesos principales de descodificación. Por último, deducimos los parámetros de configuración de cada proceso de descodificación de .

4.2 Inferencia de la estructura de paquetes

El documento oficial [22] proporciona una fórmula para calcular el número de símbolos de un paquete:

$$n_{sym} = 8 + \max\left(\left\lceil \frac{2PL + 4CRC - 5IH - SF + 7}{SF - 2DE} \right\rceil (4_{CR}), 0\right), \tag{6}$$

donde

PL es el número de bytes de carga útil, *SF* es el factor de propagación, *CRC* es 1 si la comprobación CRC está activada y 0 en caso contrario, *IH* es 1 si el modo de cabecera implícita (sin cabecera) está activado y 0 si el modo explícito (con cabecera) está activado, y *DE* es 1 si LDRO está activado y 0 en caso contrario.

De la ecuación (6) se deducen las siguientes *propiedades*:

- Un paquete LoRa tiene al menos ocho símbolos de datos, ya que $n_{sym} \geq 8$.
- $2PL + 4CRC$: El n_{sym} de un paquete con bytes de carga útil PL y comprobación CRC activada es igual al n_{sym} de un paquete con $PL + 2$ bytes de carga útil y comprobación CRC desactivada. Podemos deducir que la comprobación CRC ocupa 2 bytes en el paquete.
- $2PL + 4CRC - 5IH$: De forma similar al CRC, podemos deducir que la cabecera ocupa 2,5 bytes.
- $SF - 2DE$: SF es el número de bits de un símbolo de datos. La activación de LDRO provoca la reducción de dos bits por símbolo de datos.
- $\frac{4}{CR}$: El número de símbolos de datos aumenta con la unidad de $\frac{4}{CR}$ ($CR = 1, 2, 3, o\ 4$). Por ejemplo, $CR = 4$ significa que el paquete aplica el código Hamming (7,4), y n_{sym} tiene la forma $7n + 8$ ($n = 0, 1, 2, \ldots$).

Basándonos en estas propiedades, manipulamos los paquetes de datos para deducir su estructura. Primero aumentamos gradualmente el tamaño del paquete y observamos un aumento en escalera del número de símbolos de datos en las señales reales. Si $CR = 4$, entonces el número de símbolos n_{sym} aumentaría siguiendo una secuencia aritmética (es decir, 8, 15, 22, ...). También observamos que al aumentar continuamente el tamaño del paquete, el los bytes recién añadidos se codifican en los últimos $\frac{4}{CR}$ símbolos, y los primeros $\frac{4}{CR}(n - 1) + 8$ símbolos no cambian. Esto revela que los bytes de datos se codifican mediante un bloque de $\frac{(4)}{CR}$ símbolos.

Además, descubrimos que los ocho primeros símbolos reciben un tratamiento especial. Ya sea de forma explícita o implícita modo cabecera, los valores de los ocho primeros símbolos aparecen siempre en forma de $4k + 1$ ($k = 0, 1, 2, \ldots$), lo que significa que se descartan los dos últimos bits de datos, y hay $SF - 2$ bits de datos en cada símbolo. Podemos deducir que los ocho primeros símbolos están en modo LDRO, es decir, $DE = 1$. LDRO ofrece mejor protección, y la cabecera puede estar en los ocho primeros símbolos. Sabemos que LoRa utiliza la tasa de codificación CR para proteger la carga útil, y los bytes de carga útil están codificados por un bloque de $\frac{(4)}{CR}$ símbolos. Del mismo modo, la primeros ocho símbolos forman un bloque de ocho símbolos, y suponemos que utiliza una tasa de codificación $\frac{4}{8}$ para dar a la a la cabecera la mayor protección (4 bits de paridad para un nibble). También observamos que los ocho primeros

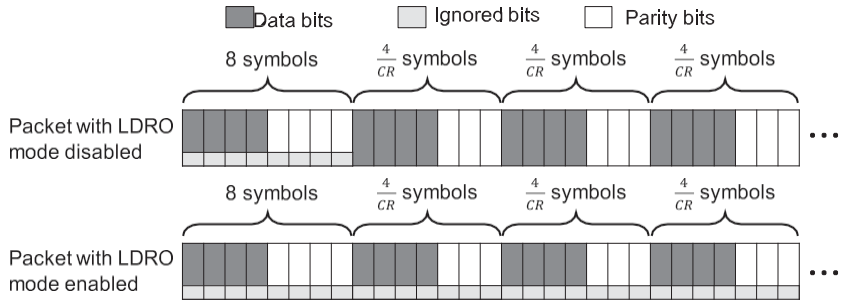

Fig. 8. La estructura inferida del paquete LoRa para el modo LDRO desactivado/activado. Cuando el modo LDRO está desactivado, sólo los ocho primeros símbolos tienen bits ignorados (protegiendo la información de cabecera). Cuando el modo LDRO está activado, se ignoran los dos bits menos significativos de todos los símbolos.

Los símbolos codifican bits de carga útil en modo de cabecera implícita. Este diseño puede reducir la complejidad del hardware de descodificación, ya que el modo de cabecera explícito e implícito puede procesarse ahora con el mismo procedimiento.

La figura 8 muestra la estructura inferida de un paquete con el modo LDRO desactivado/activado. En el modo LDRO, se descartan los dos últimos bits de cada símbolo. Independientemente de si el modo LDRO está habilitado o deshabilitado, los primeros ocho símbolos están en modo LDRO y utilizan la tasa de codificación 4. Los siguientes están en bloques que contienen $(\frac{4}{CR})$ símbolos. Los cuatro primeros símbolos codifican bits de datos, mientras que los resto de símbolos codifican bits de paridad.

Verificación de la estructura de paquetes. Verificamos la estructura de paquetes anterior comparando la ecuación (6) y el número de símbolos calculado mediante la estructura de paquetes inferida.

(6) y el número de símbolos calculado utilizando la estructura de paquetes inferida. Para $SF \geq 7$, el

Los ocho primeros símbolos contienen $8 - (SF - 2)\frac{4}{8} = 4SF - 8 \geq 20$ bits de datos. Por lo tanto, los ocho primeros símbolos pueden incluir siempre toda la información de cabecera. Excepto la cabecera, los ocho primeros símbolos pueden contener $4(SF - 2) - 20(1 - IH) = 4SF + 20IH - 28$ bits de carga útil. Supongamos que hay PL bytes ($8PL$ bits) de carga útil en total. Si el CRC está activado, se necesitan 16 bits adicionales. Excepto los ocho primeros símbolos, el número total de bits de datos necesarios para el resto de símbolos es $8PL + 16CRC - 20IH - 4SF + 28$. El blanqueamiento y la decodificación Gray no afectan al número total de bits ni al número total de símbolos. La operación de intercalado sólo nos obliga a rellenar los bytes redundantes cuando el resto de bytes de carga útil no pueden llenar un bloque, por lo que no afecta al número total de bloques de símbolos. Cada bloque tiene $(\frac{4}{CR})$ símbolos, y cada símbolo del bloque podría codificar $SF - 2DE$ bits. Debido a la existencia de paridad y bits, sólo $4(SF - 2DE)$ bits de un bloque son bits de datos reales. Entonces, el número total de símbolos del paquete es

$$8 + \max\left(\left\lceil \frac{8PL + 16CRC - 20IH - 4SF + 28}{4(SF - 2DE)} \right\rceil (CR + 4), 0\right). \tag{7}$$

Es obvio que la fórmula (7) es la misma que la ecuación (6), lo que significa que la estructura de paquetes inferida es correcta.

4.3 Orden de las operaciones de descodificación

En esta sección, intentamos desvelar el orden de los cuatro procesos principales en la descodificación LoRa: Codificación Gray (G), desentrelazado (I), decodificación Hamming (H) y blanqueamiento (W). Es fácil saber que la codificación Gray debe ser el primer paso, porque pretende resolver el problema de la deriva adyacente de símbolos (véase Codificación Gray en el apartado 4.4). La descodificación Hamming opera sobre un flujo de bytes mientras que un símbolo LoRa

contiene $SF - 2DE$ bits. Por tanto, se basa en el flujo de bytes transformado tras el desentrelazado. Así, el orden de "G, I, H" debería ser "G I→→ H". Por lo tanto, el desentrelazado tiene tres posiciones posibles, es decir, después de

"G", "I" o "H". Las implementaciones de BR, RPP0 y TAPP asumen tres posiciones diferentes de blanqueamiento, respectivamente.

Demostramos que la posición del blanqueamiento no afecta a los resultados de descodificación. Como el orden del proceso de codificación es inverso al de descodificación, utilizamos la operación en la codificación para mostrar la influencia de la posición de "W". Si consideramos los datos como un vector de bits D, el entrelazado es una operación que reordena la posición de los bits. Podemos representar el entrelazado como una matriz I con cada fila o columna conteniendo sólo un "1". La codificación Hamming, como método de codificación lineal, puede representarse como una matriz H . Tenemos $H\,D = [P\ D\ D]$, donde P es la matriz de paridad. El blanqueamiento es $W\,D$, donde W es un vector de bits aleatorio y es exclusivo-o (XOR). A continuación se indican los tres posibles órdenes de descodificación y las correspondientes operaciones de codificación:

$$\text{"W I}\rightarrow\rightarrow\text{ H"}: \quad W_{(1)} \oplus (I \cdot [P \cdot D\ D]), \tag{8a}$$

$$\text{"I W}\rightarrow\rightarrow\text{ H"}: \quad I \cdot (W_2 \oplus [P \cdot D\ D]), \tag{8b}$$

$$\text{"I H}\rightarrow\rightarrow\text{ W"}: \quad I \cdot [P \cdot (W_3 \oplus D)\ (W_3 \oplus D)], \tag{8c}$$

donde W_1, W_2, y W_3son tres vectores de bits aleatorios diferentes. La fórmula (8b) puede reescribirse como

$$(I \cdot W_2) \oplus (I \cdot [P \cdot D\ D]). \tag{9}$$

Por lo tanto, la fórmula (8a) y la fórmula (9) son equivalentes (sea $W_1 = I\ W_2$) . La fórmula (8c) puede reescribirse como

$$I \cdot ([P \cdot W_{(3)\,W(3)}] \oplus [P \cdot D\ D]). \tag{10}$$

La fórmula (10) es un caso especial de la fórmula (8b) (sea $W_2 = [P\ W_{(3)\,W(3)}]$). Por lo tanto, tanto "I W H" como "I H W" pueden representarse por "W I H", lo que significa que la posición de "W" no afecta a los resultados finales de descodificación. Mostraremos la posición correcta de "W" en la sección 4.4.

Para facilitar el análisis, *supondremos primero que el orden de descodificación es* "G W I H". Como la paridad par de todos los ceros es cero y el entrelazado en LoRa es sólo una realineación diagonal de todos los bits, las palabras clave de salida después de "H" e "I" son ceros cuando los bits de entrada son todos ceros. Aquí suponemos que LoRa adopta la comprobación de paridad comúnmente utilizada, y los resultados finales muestran que esta suposición es correcta. "W" es el XOR de los valores de datos y una secuencia pseudoaleatoria. Cuando el orden de descodificación es "G W I H", como $x\ 0 = x$, si ponemos todos los bits de transmisión a ceros, entonces los valores de salida después de "G" son la secuencia de blanqueo. Entonces, podríamos utilizar la secuencia de blanqueamiento derivada para recuperar los resultados temporales antes de "I" y "H" para su posterior análisis.

4.4 Configuración de cada operación

Esta sección revela las configuraciones clave de las cuatro operaciones de descodificación LoRa: Codificación Gray, desentrelazado, decodificación Hamming y blanqueamiento. También mostramos cómo revelar la estructura de la cabecera y el polinomio CRC.

Codificación Gray. La codificación Gray se utiliza ampliamente en muchos sistemas de comunicación inalámbricos, y consiste en pasar de un vector de bits a una representación binaria. Las representaciones adyacentes de la codificación Gray sólo tienen una diferencia de un bit. En los sistemas de comunicación inalámbricos, es más probable que identifiquemos erróneamente un símbolo con su símbolo adyacente que con otro símbolo aleatorio. Por ejemplo, en la modulación LoRa, la deriva de bin adyacente es habitual, tal y como se describe en el apartado 3.2. Con la codificación Gray, el error de bit causado por la identificación errónea adyacente se reduce a un bit por símbolo, que tiene una alta posibilidad de ser corregido por el mecanismo de corrección de errores. La codificación Gray estándar puede expresarse como

$$v = v_0 \oplus (v_0 >> 1), \tag{11}$$

donde $v_0$representa el valor bin demodulado en bruto,>> es la operación de desplazamiento a la derecha del bit, y v es el resultado de la codificación Gray. Sin embargo, cuando continuamos nuestro análisis basado en el símbolo natural

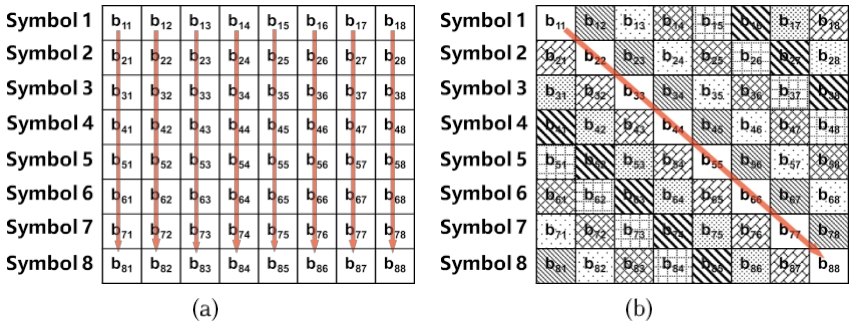

Fig. 9. Ejemplo de bloque de desentrelazado para 8 símbolos SF8 (CR=4). Cada fila del bloque es una representación de 8 bits de un símbolo LoRa después de la codificación Gray. $b_{(i)(,)(j)}$ representa el j-ésimo bit del símbolo i-ésimo. Por ejemplo, $b_{(i)(,)(1)}$ es el LSB del símbolo i. (a) Intercalado fila-línea de orden columna-mayor, la enésima palabra de código se compone de bits $b_{(i)(,)(n)}$ (i= 1, … , 8). (b) Entrelazado diagonal utilizado en LoRa, la enésima palabra de código se compone de bits $b_{i,(i+n-1)\%8+1}$ (i= 1, … , 8).

y la codificación Gray estándar, no podemos descodificar los paquetes con una tasa de éxito del 100%. El proyecto RPP0 sufre este problema y aún contiene un error no corregido.[7] Para entender la razón, recordemos el modo LDRO de LoRa. cuando LDRO está activado, el codificador pondrá bytes de datos en los 2 bits SF altos de un símbolo y luego añadirá "1" para reducir la influencia de la deriva de bin. Desde la perspectiva del hardware, es razonable añadir "1" en cualquier condición, porque podemos ahorrar el coste del circuito para juzgar si LDRO está habilitado. Suponemos que todos los símbolos de salida del codificador tienen un desplazamiento de bin. Si es cierto, antes de aplicar la codificación Gray, deberíamos restar "1" a los resultados de la demodulación. Afortunadamente, los resultados de la descodificación confirman nuestra suposición. El proceso de "G" en la descodificación LoRa podría modificarse como

$$v = (v_0 - 1) \oplus ((v_0 - 1) >> 1). \tag{12}$$

Obsérvese que TAPP también aplica a la codificación Gray operaciones similares a las nuestras, pero lo explican como un mecanismo de codificación Gray diferente y utilizan un algoritmo de fuerza bruta para derivar el "uno" adicional. Sin embargo, parece poco natural utilizar una codificación Gray no estándar. En cambio, nuestra explicación es más razonable.

Desentrelazado. El entrelazado se utiliza para reducir el impacto de los errores de ráfaga. Con el entrelazado, los errores pueden distribuirse en varios grupos de bits y corregirse mediante **la corrección de errores hacia delante (FEC)**. Se menciona [26] que LoRa aplica el entrelazado diagonal en lugar del entrelazado convencional fila-línea. La Figura 9(a) muestra el entrelazado de línea de fila de orden columna-mayor de ocho símbolos. Para el entrelazado de filas, los LSB ($b_{(i)(,)(1)}$) de los ocho símbolos se ensamblan en un byte. De la Sección 3 se desprende que los LSB de un símbolo son más frágiles que **los bits más significativos (MSB)** del símbolo. Desde la perspectiva de la FEC, es un mal diseño agrupar los bits frágiles. La Figura 9(b) muestra el entrelazado diagonal utilizado en LoRa. El entrelazado diagonal distribuye los LSB frágiles en bytes diferentes y es más robusto. A continuación, manipulamos los paquetes transmitidos para derivar el mapeado diagonal detallado. Como ejemplo, enviamos paquetes con SF= 8 y CR=4 en modo de cabecera implícita. Así, el bloque intercalado es un bloque de 8 8 como se muestra en la Figura 9(b). En primer lugar, suponemos que el FEC utilizado en CR=4 es el código Hamming estándar (7, 4) con una extensión de un bit. Por lo tanto, después de "G" y "W", la palabra clave para el nibble "0000" es "00000000", y la palabra clave para "1111" es "11111111".[8] Supongamos que los bytes de envío son todos ceros excepto que el cuarto byte es 0x0F, observamos que

[7] https://github.com/rpp0/gr-lora/issues/99.

[8] Obsérvese que aquí hemos eliminado la influencia del blanqueamiento.

$b_{11} = b_{22} = \cdots = b_{88} = 1$ en la Figura 9(b). Por lo tanto, la diagonal principal representa el cuarto byte 0x0F. El problema del desplazamiento de un bin mencionado en la codificación Gray refleja aquí que no siempre podemos obtener ocho unos en un bloque si aplicamos directamente la codificación Gray estándar. Desplazar el mapeo en uno resuelve este problema y se ajusta perfectamente a nuestro siguiente proceso de descodificación. Cambiando los bits de datos todo-1 en el paquete transmitido, podemos derivar el mapeado completo para el intercalado, como se muestra en la Figura 9(b). Para otros parámetros, el proceso de desentrelazado es similar. La única diferencia es que

el tamaño de bloque pasa a ser $\frac{4}{CR} \times (SF - 2DE)$. Como resultado, resumimos el proceso de desentrelazado como

$$c_{i,j} = b_{j,(i+j-1)\%(SF-2DE)+1}, \tag{13}$$

donde $c_{(i),(j)}$ es *el j-ésimo* bit de la *i-ésima* palabra de código después del desentrelazado, $b_{(j),(i)}$ es el *i-ésimo* bit del j-ésimo símbolo después de la codificación Gray, e $i \in \{1, 2, \ldots, \frac{4}{CR}\}, j \in \{1, 2, \ldots, SF - 2DE\}$.

Decodificación Hamming. Tras el desentrelazado, obtenemos los datos codificados en forma de palabras clave, pero aún se desconoce la posición de los bits de datos y de paridad en una codificación. LoRa proporciona cuatro valores CR válidos ($\frac{4}{5}, \frac{4}{6}, \frac{4}{7}, \frac{4}{8}$), que determinan la longitud de la palabra de código y, por tanto, la fuerza FEC. Cuando CR se establece en $\frac{4}{5}$ o $\frac{4}{6}$, LoRa aplica el código Hamming(7, 4) o el código Hamming ampliado con un bit de paridad adicional. Cuando CR está en $\frac{4}{5}$ o $\frac{4}{6}$, LoRa puede detectar errores de bit pero no puede corregirlos. En primer lugar, suponemos que un nibble con tasa de código $\frac{4}{8}$ está protegido por el código Hamming(8, 4) estándar. Pero los resultados de nuestro análisis final muestran que esta suposición requiere algunas modificaciones. Para determinar la posición de cada bit de datos/paridad en la palabra de código, podríamos variar los bytes de envío pero mantener fijo un bit específico. Por ejemplo, para comprobar la posición del LSB del cuarto byte, fijamos los bytes de transmisión como 0x01, 0x03, 0x05, ..., 0x0F.[9] Entonces, el bit que siempre es 1 en el bloque intercalado es nuestro objetivo, es decir, el LSB en este caso. Encontramos que los cuatro LSBs en el codeword son bits de datos mientras que los cuatro MSBs son bits de paridad, lo que difiere del código estándar Hamming(8, 4) $p_{(1)}\, p(2)\, d(1)\, p(3)\, d(2)\, d(3)\, d(4)\, p4$, donde d_i es el *i-ésimo* bit de datos y p_i es el *i-ésimo* bit de paridad. El conjunto de ecuaciones (14) muestra la relación entre los bits de datos y los bits de paridad en el código Hamming(8, 4) estándar,

$$\begin{aligned} p_1 &= d_1 \oplus d_2 \oplus d_4 \\ p_2 &= d_1 \oplus d_3 \oplus d_4 \\ p_3 &= d_2 \oplus d_3 \oplus d_4 \\ p_4 &= d_1 \oplus d_2 \oplus d_3 \oplus d_4. \end{aligned} \tag{14}$$

Denotamos los cuatro LSB como d_1, d_2, d_3, d_4 secuencialmente. Para averiguar qué bit de los MSB es p_i, variamos los bits de datos para mantener $p_i = 1$. Por ejemplo, seleccionando las palabras clave con $d_1 \oplus d_2 \oplus d_4 = 1$, el bit de paridad que siempre es igual a "1" es el bit de paridad p_1. Del mismo modo, se derivan las posiciones de p_2 y p_3. Sin embargo, el bit de paridad restante no encaja en la definición de p_4. Tras una cuidadosa observación, descubrimos que se trata de una paridad que cubre d_1, d_2 y d_3, es decir

$$p_5 = d_1 \oplus d_2 \oplus d_3. \tag{15}$$

Para otras tasas de código, podemos deducir la posición del bit de forma similar. Cuando se utiliza $CR = \frac{4}{7}$, el bit de paridad p_1 se abandona. Cuando se utiliza $CR = \frac{4}{6}$, los bits de paridad p_1 y $p_{(2)}$ se abandonan. Cuando se utiliza $CR = \frac{4}{5}$, sólo hay un bit de paridad y es natural utilizar p_4 para cubrir todos los bits. La conclusión no cambia cuando varía SF. La figura 10 muestra las posiciones de los bits para diferentes tasas de código. Nótese que LoRa aplica un código Hamming no estándar, el número de denominación de la paridad es arbitrario y nuestra denominación es sólo un tipo de ellos.

Desenfoque. En la sección 4.3, asumimos que la operación de blanqueamiento ocurre después de "G" y demostramos que la posición de blanqueamiento no afecta a los resultados finales. Aquí discutiremos la posición "cor- rect" para el desentrelazado. El proceso de desentrelazado y descodificación Hamming ha sido

[9] Obsérvese que no podemos controlar directamente los bits de paridad.

Fig. 10. Posición de los bits de paridad y los bits de datos en una codificación con CR=4, 4, 4, 4.
8 7 6 5

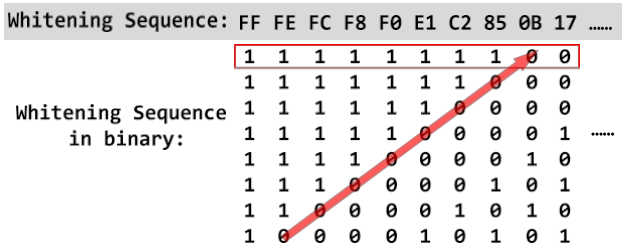

Fig. 11. Secuencia de blanqueamiento de "G I H→→→ W.".

interpretado claramente más arriba. Movemos la posición de "W" a las otras dos posiciones y enviamos bytes todo-cero para derivar la secuencia de blanqueamiento bajo diferentes selecciones de orden. Encontramos que "G W I H" da diferentes secuencias de blanqueamiento para varias combinaciones de SF y CR mientras que la secuencia de blanqueamiento de "G I W H" y "G I H W" se mantienen iguales. Utilizar la misma secuencia para todos los paquetes es más razonable. Por tanto, primero excluimos "G W I H". Luego tenemos que elegir el orden correcto entre "G I W H" y "G I H W". La hoja de datos del chip LoRa [22] menciona que la secuencia de blanqueamiento en modo FSK se genera mediante un **registro de desplazamiento de realimentación lineal (LFSR)**. Suponemos que también existe un LFSR que genera la secuencia de blanqueamiento para el modo LoRa. Aplicamos el algoritmo Berlekamp-Massey [38] a las secuencias de blanqueamiento de los dos órdenes de descodificación para obtener el LFSR correspondiente. Para "G I W H", no importa cómo cambiemos el orden de entrada de los bits (por ejemplo, primero el LSB o primero el MSB), el tamaño mínimo del LFSR que obtenemos del algoritmo Berlekamp- Massey es de al menos 64 bits. Pero sabemos que siempre se puede construir una secuencia de 2n bits a partir de un LFSR de n bits. Cualquier secuencia que contenga 128 bits podría representarse como la salida de un LFSR de 64 bits. El LFSR de la secuencia "G I W H" es demasiado largo. Comprobando cuidadosamente la secuencia de blanqueo de "G I H W" como se muestra en la Figura 11, encontramos que no es como los típicos bits aleatorios, y cada byte parece ser el estado de un LFSR. Recogemos los MSB de los bytes de la secuencia, como se muestra en el recuadro rojo de la Figura 11, para ejecutar el algoritmo de Berlekamp-Massey. El polinomio LFSR para derivar dicha secuencia es $x^8+ x^6+ x^5+ x^4+ 1$. Puede verse en la literatura [39] que es el polinomio de máxima longitud para un registro de desplazamiento de 8 bits. La figura 12 muestra la estructura del LFSR, y podemos ver que los ocho bits del registro componen un byte de la secuencia de blanqueamiento, y cada bit del registro se utiliza para blanquear datos de un bit. Dado que la secuencia de blanqueamiento de "G I H W" puede ser generada por un LFSR con una longitud mucho menor, consideramos que es el orden correcto.

4.5 CRC

Tras cuatro pasos de codificación Gray, desentrelazado, codificación Hamming y desblanqueo, se muestran los paquetes LoRa PHY sin procesar. En secciones anteriores, enviamos paquetes en modo de cabecera implícita y desactivamos la comprobación CRC. A continuación tratamos de analizar el algoritmo CRC utilizado en la carga útil LoRa. Según lo observado en la sección 4.2, la suma de comprobación CRC ocupa 16 bits en la cola de un paquete. Por tanto, primero

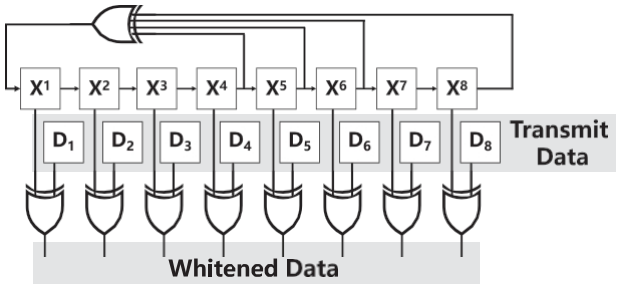

Fig. 12. LFSR $x^8+x^6+x^5+x^4+1$ utilizado en LoRa.

enviar paquetes de cabecera explícita y cabecera implícita con el mismo contenido de datos para comprobar la cobertura del CRC. Los resultados muestran que el CRC sólo se realiza sobre la carga útil. Calcular la suma de comprobación CRC de una serie de bits, en teoría, es considerarla como un gran número binario y calcular el resto con respecto a un "divisor". El divisor suele representarse como un polinomio basado en *GF* (2) (campo de Galois de dos elementos), por ejemplo, x^3+x+1. El análisis de la parte CRC de LoRa PHY consiste en encontrar el polinomio utilizado. Una característica del CRC es que para un polinomio de grado n, si el dividendo original es 1, entonces el resto es el propio polinomio. Si construimos un paquete sólo con el último bit igual a 1, entonces la suma de comprobación CRC en los dos últimos bytes nos dice exactamente qué polinomio se utiliza. Mientras tanto, el polinomio CRC no puede seleccionarse al azar; debe seguir algunos principios para garantizar algunos requisitos específicos. Por lo tanto, es natural elegir el polinomio entre los polinomios CRC estándar. Si comparamos el polinomio que obtenemos y los conjuntos de polinomios estándar, no nos costará mucho encontrar el polinomio que realmente se utiliza. Basándonos en el análisis anterior, activamos el CRC de la carga útil en el modo de cabecera implícita y establecemos los dos últimos bytes de la carga útil como 0x0001, 0x0080, 0x0100 y 0x8000, respectivamente (los demás bytes se ponen a cero). Como no estamos seguros del orden endian y LSB-MSB en las implementaciones de hardware CRC, probamos las cuatro combinaciones posibles. El resultado, sin embargo, sorprendentemente, está más allá de nuestras expectativas. Independientemente de lo que establezcamos en los dos últimos bytes de carga útil, la suma de comprobación CRC es exactamente la misma que la de los dos últimos bytes de carga útil. En la implementación común de CRC, antes de hacer el cálculo del resto, se rellenan n bits cero después de los datos para asegurar que los últimos n bits también están bajo protección CRC completa.[10] El fenómeno que observamos significa que el paso de relleno cero no está implementado en LoRa PHY. Por lo tanto, enviamos 0x0001, 0x0080, 0x0100, 0x8000 en el tercer y cuarto bytes desde el último para derivar el polinomio. Los bytes CRC recibidos son 0x1021, 0x9188, 0x3331 y 0x1B98, respectivamente. 0x1021 se refiere al polinomio denominado CCITT-16, siendo $x^{16}+x^{12}+x^5+1$. Aplicamos la comprobación CRC con este polinomio y probamos paquetes con bytes aleatorios. Las sumas de comprobación CRC calculadas por nuestra implementación coinciden con los bytes CRC en todas las muestras. Debido a las características del CRC, es difícil que una implementación CRC errónea tenga la misma suma de comprobación que la correcta, incluso en un pequeño grupo de muestras, lo que significa que nuestro resultado es correcto.

4.6 Cabecera

La parte más desconocida de LoRa PHY es la cabecera. En la sección 4.2 ya hemos sabido que la cabecera en modo explícito tiene 20 bits. Nuestro objetivo es encontrar la organización y el significado de los 20 bits. Se dice que la longitud **de la carga útil (PL)** está codificada en la cabecera y ocupa al menos 8 bits, ya que la longitud máxima de la carga útil es 255. Variamos la longitud de la carga útil e intentamos descodificar el paquete utilizando el mismo proceso de descodificación en modo de cabecera implícita suponiendo que la cabecera también está blanqueada . Desafortunadamente, no somos capaces de recuperar los bytes de datos en modo de cabecera implícita. El

[10]Aunque la implementación a nivel de código no contenga explícitamente el paso de añadir cero [40].

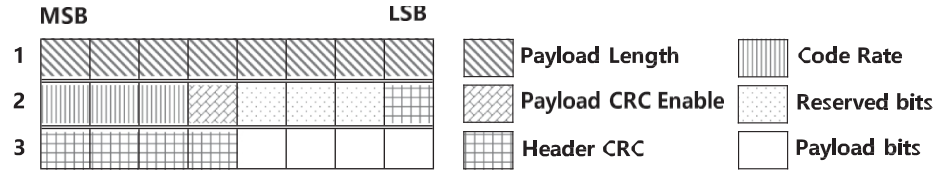

Fig. 13. Estructura de la cabecera de LoRa en los tres primeros bytes.

problema proviene de la secuencia de blanqueo utilizada para la cabecera. No obstante, observando los bits que cambian con la longitud de la carga útil, podemos identificar la posición de PL. Obsérvese que en los primeros 2,5 bytes, sólo cambian los bits de PL y CRC de cabecera al variar la longitud de la carga útil del paquete. Observamos que los bits de las posiciones 1-8 y 16-20 son posibles bits de PL. Además, los resultados intermedios nos muestran que la cabecera no está blanqueada y que el primer byte de una cabecera antes del blanqueamiento es exactamente PL. Nuestro LFSR derivado sólo puede generar 255 posibles bytes de blanqueamiento. No hay más bytes de blanqueamiento adicionales para el blanqueamiento de la cabecera. Por lo tanto, es razonable no blanquear la cabecera del paquete. Nuestras siguientes pruebas para encontrar otros bits refuerzan esta suposición. Por lo tanto, en lo sucesivo no aplicaremos la operación de blanqueamiento a la cabecera. Cambiando un parámetro y fijando otros parámetros, la posición de la tasa de código y el bit de habilitación de la carga útil-CRC se determinan de forma similar. Los bits de tasa de código son 001, 010, 011, 100 para $CR{=}4, 4, 4, 4$, respectivamente. El bit de habilitación de carga útil-CRC es $_{5\,6\,7\,8}$
se pone a 1 si se activa la carga útil CRC. En nuestros experimentos, hay cinco bits que cambian rápidamente al establecer diferentes parámetros. Consideramos estos cinco bits como CRC de cabecera. Hay tres bits que se mantienen a cero sea cual sea la configuración de los parámetros, y los consideramos bits reservados. La figura 13 ilustra la estructura de la cabecera en los tres primeros bytes.

No logramos encontrar un polinomio CRC5 estándar que satisficiera los resultados del CRC de cabecera utilizando el método de la sección 4.5. Denotamos 12 bits de parámetro (PL, CR y bit de habilitación de carga útil-CRC) como un vector de bits v_1. Denotamos los cinco bits CRC de cabecera como un *vector* de bits v_2. Nuestro objetivo es encontrar una matriz M que satisfaga $v_2 = M v_1$. Dado que el valor de $v_1$está bajo nuestro control, podemos diseñar una serie de v_1, digamos, $v_1^{((0)(1)(0))}, v_1^{((0)(2)(0))}, \ldots, v_1^{((0)(12)(0))}$) . Forman una matriz V_1. La serie relativa v_2 es $v_2^{((0)(1)(0))}, v_2^{((0)(2)(0))}, \ldots, v_2^{((0)(12)(0))}$) . Forman la matriz V_2. Por lo tanto , $V_2 = M \cdot V_1$. Si V_1es una matriz identidad, entonces $M = V_2$.

¿Cómo hacer una matriz de identidad V_1? A pesar de que podemos controlar el contenido de la cabecera, no podemos controlar finamente el estado de cada bit. En nuestro caso, parece imposible enviar un paquete con $v_{1(1)}$ que contenga un solo "1". Sin embargo, para nuestra sorpresa, el chip LoRa admite inesperadamente el *envío de un paquete con una longitud de carga útil cero*. Los ocho bits PL se convierten en ceros. Si desactivamos el CRC de carga útil y ponemos $CR{=}4$o $CR{=}4$o $CR{=}4$, entonces sólo un bit de tasa de código en $v_1$es 1. ¿Es posible que sólo el bit de habilitación de carga útil-CRC sea "1"? La respuesta vuelve a ser la linealidad de CRC. Supongamos que enviamos dos paquetes con carga útil cero y $CR{=}4$. El paquete A tiene carga útil CRC pero el paquete B no. Entonces los vectores de bits correspondientes son $v_1^A = 000000000011$, $v_2^A = 01100$, $v_1^B = 000000000010$, $v_2^B = 00111$. Por lo tanto,

$$\begin{aligned} M \cdot (000000000001) &= M \cdot v_1^A \oplus v_1^B \\ &= M \cdot v_1^A \oplus M \cdot v_1^B \\ &= v_2^A \oplus v_2^B \\ &= 01011. \end{aligned} \tag{16}$$

(11) Aquí ignoramos los bits reservados.

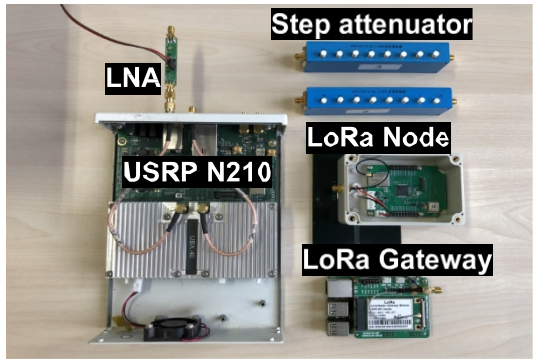

Fig. 14. Dispositivos de nuestros experimentos: un dispositivo SDR USRP N210 con LNA adicional, aNenuadores de paso para medir la sensibilidad en entornos cableados, nodos LoRa básicos y una pasarela LoRa básica.

Se puede aplicar un proceso similar para los bits PL. Por último, resumimos el cálculo del CRC de cabecera como $v_2 = M - v_1$, donde

$$M = \begin{bmatrix} 1\,1\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,0\,0\,0\,1\,1\,1\,0 \\ 0\,0\,1\,0\,0\,1\,0\,0\,1\,1\,0\,1\,0 \\ 0\,0\,1\,0\,0\,1\,0\,1 \\ 0\,0\,0\,1\,0\,0\,1\,0 \end{bmatrix} \quad \begin{bmatrix} 0\,0\,0\,1 \\ 1\,1\,0\,1 \\ 1\,1\,1\,1 \end{bmatrix} \tag{17}$$

5 APLICACIÓN

Implementamos todo el LoRa PHY en C++ en la plataforma de Radio Definida por Software GNU Radio. También proporcionamos un código en versión MATLAB para fines de simulación. Diseñamos el receptor como dos bloques separados: demodulador y decodificador (para el transmisor: modulador y codificador). Así, en el futuro podrán integrarse en nuestra implementación otros nuevos mecanismos de demodulación sin cambiar el bloque decodificador. Por ejemplo, si sustituimos el demodulador LoRa por un demodulador de colisión, podremos descodificar directamente los paquetes de colisión sin cambiar la lógica del bloque descodificador. Todos los experimentos de la sección 6 se llevan a cabo en el USRP N210 en tiempo real (no guardamos las señales de banda base sin procesar para procesarlas fuera de línea). El USRP está conectado a un portátil con Ubuntu 19.10, CPU i5-7200U y 8 G de RAM. Utilizamos dispositivos LoRa básicos para evaluar el rendimiento de nuestra LoRa PHY implementada, incluidos nodos finales LoRa con SX1268 [25]/SX1278 [22] y pasarelas LoRa Raspberry-Pi 3B+ (RPI) con SX1301 [37]. La figura 14 muestra los dispositivos utilizados en nuestros experimentos. El atenuador de paso se utiliza en los experimentos con cable para medir la sensibilidad.

6 EVALUACIÓN

6.1 Tasa de éxito en la descodificación

Verificamos la eficacia de nuestro LoRa PHY implementado probando si puede descodificar los paquetes LoRa transmitidos desde los dispositivos LoRa básicos. Configuramos un dispositivo LoRa básico para que transmita repetidamente paquetes aleatorios a nuestro receptor LoRa implementado. La verdad sobre el terreno de los bytes transmitidos se registra a través del puerto serie. Para una evaluación exhaustiva, ajustamos el transmisor LoRa con distintos parámetros, incluidos seis SF (7~ 12), cuatro CR ($\frac{4}{5}, \frac{4}{6}, \frac{4}{7}, \frac{4}{8}$), y dos modos de encabezamiento (explícito/implícito), y con/sin CRC, es decir, 6× 4× 2× 2= 96 combinaciones. Utilizamos una frecuencia portadora de 475 MHz y un ancho de banda de 250 kHz.[12] La longitud de la carga útil se fija en

[12]La frecuencia portadora y el ancho de banda no afectan a la lógica de descodificación.

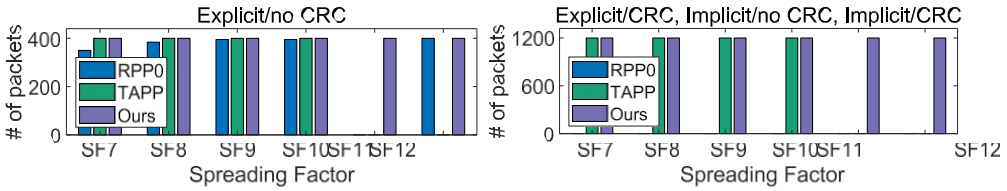

Fig. 15. Prueba de la tasa de éxito de descodificación. Se envían un total de 9.600 paquetes y se prueban todas las tasas de código (^{4}, ^{4}, ^{4}, ^{4}). RPP0 no puede descodificar ningún paquete SF11. TAPP no puede descodificar ningún paquete SF11/12. Nuestro descodificador puede descodificar todos los paquetes. (a) Número de paquetes recibidos correctamente de tres implementaciones en modo de encabezado explícito con CRC desactivado. (b) RPP0 no admite la configuración de {modo de encabezado explícito con CRC activado, modo de encabezado implícito con CRC desactivado, modo de encabezado implícito con CRC activado} y, por tanto, no recibe ninguno.

16 bytes. Para cada configuración, la transmisión se repite 100 veces. Colocamos el transmisor LoRa cerca del receptor USRP, por lo que todas las señales se reciben con una SNR alta. Dado que el entorno de comunicación es ideal, cualquier paquete perdido refleja que la implementación está incompleta.

En la figura 15 se muestra el resultado del experimento. Entre todas las configuraciones, BR sólo funciona en SF8 con cabecera implícita, mientras que RPP0 sólo admite la cabecera explícita. Ninguno de estos dos métodos admite CRC. Como los procesos de descodificación de BR y TAPP son costosos desde el punto de vista computacional, no funcionan con paquetes de gran longitud a una velocidad de transmisión elevada. En concreto, BR no puede descodificar ningún paquete de más de 5 bytes, y TAPP empieza a perder paquetes cuando el ciclo de trabajo del transmisor es superior a 0,5. Debido a todas estas limitaciones, el PRR global de BR sólo alcanza el 4,2%. Además, podemos ver que TAPP no consigue descodificar ningún paquete SF11/SF12, y RPP0 no puede resolver paquetes SF11. En resumen, RPP0 cubre 1.924/9.600 paquetes al 20,0% y TAPP cubre 6.400/9.600 paquetes al 66,7%. Por el contrario, nuestro descodificador puede descodificar todos los paquetes bajo cualquier configuración LoRa.

6.2 Sensibilidad

En esta sección comprobamos la sensibilidad de nuestro descodificador LoRa. Conectamos un transmisor LoRa básico a nuestro descodificador a través de un cable de RF de 20 m con dos *atenuadores de paso*. Así, podemos controlar con precisión la atenuación de la señal ajustando el valor de estos dos *atenuadores*.

Utilizar el valor de los *atenuadores de paso* como atenuación del canal no es exacto, porque los enlaces cableados, incluidos los cables y conectores de RF, también introducen atenuación, lo que provoca errores de estimación de la sensibilidad. Por lo tanto, antes del experimento, utilizamos un analizador de espectro, Keysight N9322C, para estimar y calibrar con precisión la atenuación del enlace. A continuación, ajustamos la potencia de transmisión al mínimo, es decir, 7,9 dBm, para evitar fugas en el canal inalámbrico. Experimentamos con la configuración de SF8 y un ancho de banda de 250 kHz. Dado que BR no descodifica ningún paquete de más de cuatro bytes, no comparamos su rendimiento en el resto de esta sección. La Figura 16(a) muestra los resultados de sensibilidad de cuatro descodificadores SDR. El criterio de sensibilidad en LoRa se define como el RSSI mínimo que alcanza el PRR> 90% cuando la carga útil es de 32 bytes. Como RPP0 y TAPP no aprovechan las características LoRa para la demodulación, su PRR cae rápidamente a medida que baja el RSSI de la señal recibida. Las sensibilidades de RPP0 y TAPP son 106 y 108 dBm, respectivamente. En cambio, el rendimiento de nuestro decodificador implementado se mantiene estable y alcanza una sensibilidad de 126 dBm.

Además, realizamos un experimento con un SF mayor, es decir, SF12. El resultado se muestra en la Figura 16(b). Nuestra sensibilidad medida es 142 dBm, que se aproxima a la sensibilidad óptima de LoRa. Curiosamente, observamos que la sensibilidad de la pasarela RPI es de sólo 139,5 dBm. Esto se debe a que la pasarela RPI no adopta el diseño óptimo de la pasarela LoRa. Por tanto, nuestro descodificador tiene mejores prestaciones que ella.

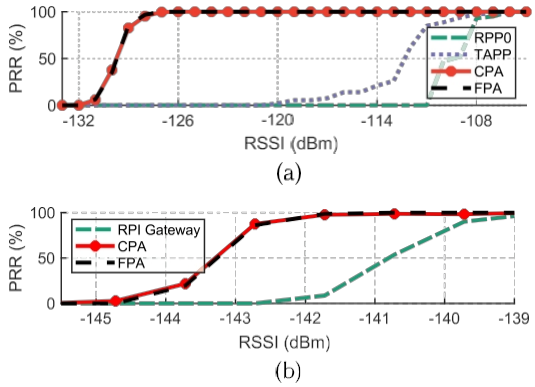

Fig. 16. Pruebas de sensibilidad de distintos descodificadores con (a) carga útil de 10 bytes, SF8, BW = 250 kHz y (b) carga útil de 32 bytes, SF12, BW= 125 kHz.

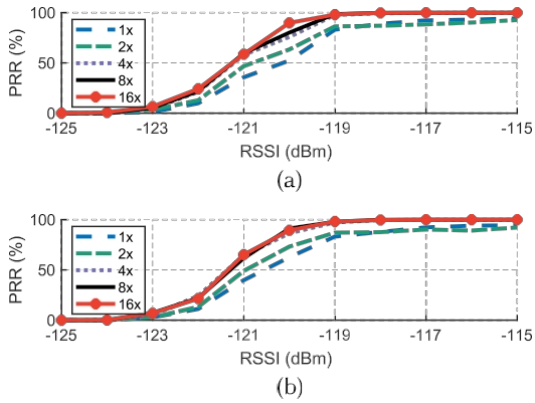

Fig. 17. Influencia de la relación de relleno cero en (a) CPA y (b) FPA con k= 8.

6.3 Influencia del acolchado cero

A continuación, verificamos la validez de la estrategia de refinamiento de picos propuesta, que mejora la resolución de frecuencias mediante el relleno cero. Utilizamos la misma configuración experimental que en la sección 6.2, evaluando los métodos FPA y CPA. La Figura 17 muestra el rendimiento del receptor bajo diferentes niveles de RSSI y ratios de zero-padding. Tanto para el FPA como para el CPA, vemos que el PRR aumenta con una mayor relación de relleno *cero r* . La mejora del rendimiento para r de 1 a 4 es observable, mientras que para $r \geq 4$ la mejora del rendimiento se vuelve insignificante.

6.4 Prueba de codificador

En esta sección presentamos la evaluación de nuestro codificador LoRa implementado, que es una versión inversa del decodificador LoRa. La única diferencia en la implementación del codificador con respecto al decodificador son los bytes redundantes para rellenar los últimos $(^{4)}$ símbolos. Aunque estos bytes parecen fijos en

implementación del chip LoRa, no tienen sentido o $_{(C)\ nR}$ en el lado del receptor. En nuestra implementación rellenamos con ceros los bytes de la izquierda. Cuando el codificador está activo, abre un puerto UDP y espera datos. Enviamos 16 bytes al proceso de codificación a través de un script python. Después, la señal LoRa generada se envía desde el USRP a un dispositivo final LoRa. Comprobamos las salidas del puerto serie del dispositivo para ver si los datos se transfieren correctamente. Utilizamos diferentes parámetros (es decir, diferentes SF, CR y

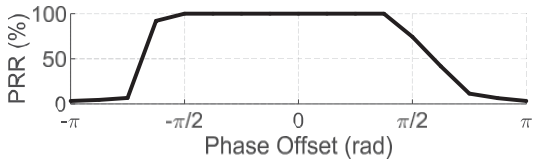

Fig. 18. PRR del chip LoRa al recibir señales LoRa especialmente construidas con diferentes valores de desalineación de fase.

ancho de banda) para el envío de datos. Además, para cada ajuste de parámetros, la transmisión se repite 100 veces. Los resultados finales muestran que 9.600 paquetes enviados desde nuestro codificador pudieron ser todos decodificados con éxito , lo que desde otro punto de vista revela que nuestro análisis e implementación son fiables.

6.5 Influencia de la desalineación de fase

En la Sección 3, proponemos FPA y CPA para la demodulación LoRa. Demostramos que el rendimiento de los dos métodos se aproxima teóricamente a la demodulación IDEAL. Esto se debe a que son resistentes a la fluctuación de fase causada por un hardware no perfecto y a la multitrayectoria. Entonces, ¿cuál es el método de demodulación real implementado en el chip LoRa básico? ¿Adopta FPA y CPA? Curiosamente, descubrimos que el algoritmo de demodulación utilizado en el chip LoRa no es tan bueno como FPA o CPA. Construimos paquetes LoRa que contienen cinco bytes especialmente diseñados. Como resultado, los cuatro símbolos de datos consecutivos son iguales con $f_{start} = 0$, es decir, un símbolo con una fuerte caída de frecuencia en el medio. Este tipo de símbolo es el más vulnerable a la desalineación de fase entre los dos segmentos de chirp. Antes de enviar el paquete por SDR, añadimos manualmente un desfase en los dos segmentos. Nuestro receptor que utiliza FPA y CPA es resistente a la desalineación de fase y su PPR es del 100%. Sin embargo, descubrimos que la pasarela LoRa se enfrenta a frecuentes pérdidas de paquetes cuando recibe paquetes con una desalineación de fase significativa. La figura 18 muestra el PRR con respecto al desalineamiento de fase. Su PRR es de casi el 100%.
cuando el desfase está entre $(-\frac{\pi}{2}, \frac{\pi}{2})$. Pero disminuye rápidamente cuando el desfase se convierte en
grande. El resultado significa que el chip LoRa adopta un método de demodulación más débil que nuestros métodos. En otras palabras, el chip LoRa básico es vulnerable al "*ataque de desalineación de fase*". A pesar de que este tipo de "ataque" es difícil de realizar, cualquier pequeño defecto es digno de mención en el ámbito de la seguridad. Conjeturamos que el chip LoRa puede compensar un desfase fijo $\Delta\varphi$ durante el proceso de demodulación, por ejemplo, estima $\Delta\varphi$ a partir del preámbulo y aplica $\Delta\varphi$ a los siguientes símbolos de datos. Si $\Delta\varphi$ cambia, entonces el chip LoRa no consigue demodular el paquete correctamente. Así, el PRR cae cuando cambia el desfase. Por supuesto, no conocemos la implementación interna en el chip LoRa a menos que Semtech publique los documentos. Pero, en resumen, la implementación de demodulación del chip LoRa es más débil que nuestro método FPA y CPA.

6.6 Prueba en exteriores

Por último, comparamos nuestro descodificador con una pasarela RPI básica en un entorno real. Colocamos el receptor fuera de una ventana en el segundo piso. Seleccionamos cuidadosamente las ubicaciones de los transmisores, asegurándonos de que no haya una trayectoria de línea de visión entre el transmisor y los receptores, lo que refleja la situación habitual de la comunicación LoRa. El transmisor envía repetidamente un paquete de 32 bytes en modo de cabecera explícita con ajuste SF12, BW= 125 kHz, CR=$\frac{4}{5}$, y CRC activado. Elevamos el transmisor a 1,8 m de altura con un trípode. La figura 19 muestra la configuración del experimento mencionada anteriormente. La figura 20 muestra la PRR en diferentes distancias de comunicación. Si fijamos PRR = 95% como barra, nuestro descodificador admite un rango de comunicación de hasta 3.600 m, mientras que la pasarela RPI básica sólo alcanza el máximo de 2.800 m. Un detalle interesante es la fluctuación de PRR de la pasarela RPI en 2.400 m. Suponemos que el motivo es el desfase causado por el complejo

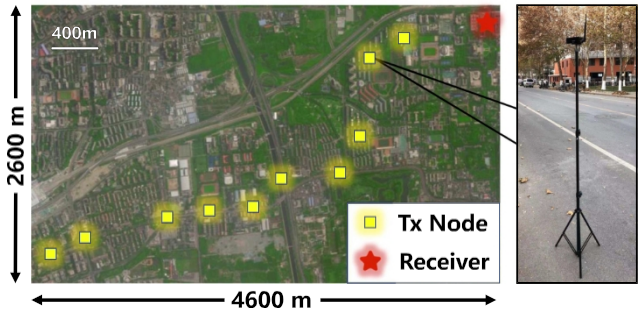

Fig. 19. Despliegue del transmisor y del receptor. El transmisor está colocado sobre un trípode de 1,8 m.

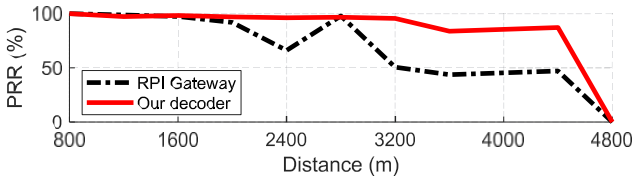

Fig. 20. Experimentos de alcance de comunicación en exteriores.

entorno. Como se menciona en la sección 6.5, el chip LoRa básico es vulnerable al problema de desalineación de fase. Por lo tanto, nuestro decodificador parece más fiable en comunicaciones al aire libre.

7 TRABAJO RELACIONADO

Ingeniería inversa de protocolos inalámbricos. El rápido desarrollo de Internet de las Cosas trae consigo una pluralidad de protocolos propietarios. El impulso endógeno para desarrollar protocolos cerrados en lugar de utilizar los protocolos estándar existentes se debe a muchas razones. Por ejemplo, un protocolo propietario podría ahorrar costes de licencia a los fabricantes de la Internet de las Cosas, o el nuevo protocolo proporciona funciones específicas dirigidas a aplicaciones y dispositivos integrados concretos. Sin embargo, desviarse de las especificaciones estándar a veces allana el camino a los atacantes, ya que los fabricantes podrían introducir algunos diseños inseguros. Mientras tanto, estos protocolos suelen estar indocumentados y cerrados al público, lo que atrae a los investigadores de redes a realizar ingeniería inversa sobre ellos, incluyendo la RFID criptográfica [41, 42, 53], la red inalámbrica en el coche [43], el protocolo ad hoc Apple Wireless Direct Link [44, 45], etcétera. La ingeniería inversa de estos protocolos profundiza nuestra comprensión de las redes existentes y revela fallos en el diseño de los protocolos. Nuestro trabajo también pretende ofrecer a otros investigadores una comprensión más completa de la capa física de LoRa y proporcionar una herramienta mejor para analizar la red LoRa.

Trabajos relacionados con LoRa PHY. LoRa proporciona una modulación única que ofrece largo alcance, lo que resulta atractivo en muchas aplicaciones. La tecnología tradicional de retrodispersión adolece del alcance de la comunicación, que se limita al nivel del metro. Trabajos recientes sobre retrodispersión que combinan LoRa [12, 47, 52] mejoran el alcance hasta el nivel del kilómetro. Mientras tanto, adoptando una modulación similar a LoRa, Netcatter [13] soporta 256 transmisiones backscatter actuales. Debido a la baja velocidad de transmisión de datos y a las características de gran cobertura de LPWAN, la capacidad de la red es relativamente pequeña [46], lo que provoca problemas cruciales de colisión. Se han realizado muchos esfuerzos para mejorar el rendimiento de la red LoRa. Otros trabajos representativos son la decodificación de colisiones [1-4, 9, 10, 27, 49] y la decodificación de señales débiles [50, 51]. Se basan principalmente en la unicidad de la demodulación LoRa PHY para

separar los chirps en paquetes de diferencia. Nuestro trabajo proporciona una comprensión más profunda de LoRa PHY e impulsará la investigación relacionada con LoRa PHY.

8 CONCLUSIÓN

Este artículo presenta una comprensión exhaustiva de la demodulación y decodificación de LoRa y revela las razones fundamentales de la brecha de rendimiento entre los trabajos existentes y LoRa básico. Este trabajo es la primera implementación completa de LoRa PHY con una garantía de rendimiento demostrable para el chip LoRa de caja negra. Mejoramos la demodulación para lograr una decodificación con SNR extremadamente baja (20 dB) con una garantía teórica de rendimiento. Además, derivamos el orden y los parámetros de las operaciones de descodificación, incluyendo el blanqueamiento, la corrección de errores, el desentrelazado, etc., aprovechando hechos oficialmente conocidos de LoRa y la manipulación de paquetes. Además, implementamos la primera PHY LoRa SDR completa en tiempo real en la plataforma GNU Radio. La evaluación muestra que nuestro método puede lograr (1) una tasa de éxito de descodificación del 100%, mientras que los métodos existentes pueden soportar como máximo el 66,7%; (2) sensibilidad de 142-dBm, que es el límite del LoRa básico; y (3) un alcance de comunicación de 3.600 m en la zona urbana, similar al de LoRa básico en el mismo entorno. configuración.

APÉNDICE

Modelo de cálculo de SER. Supongamos que la señal se transmite a través de un canal AWGN, en el que el ruido sigue la distribución gaussiana compleja, es decir, $\mathcal{CN}(0, \sigma^2)$. A continuación, tras la deschirpación, la distribución del ruido pasa a ser $\mathcal{CN}(0, M\sigma^2)$, donde M es el número de muestras dentro de la ventana de demodulación . La altura máxima del ruido tiene una varianza pequeña y puede aproximarse como $c\sigma$, donde c es una constante. Supongamos que la altura del pico de datos h_d sigue la distribución gaussiana $\mathcal{N}(\mu_d, \sigma_d^2)$. Tenemos

$$SER = P(h_d < c\sigma) = Q\left(\frac{c\sigma - \mu_d}{\sigma_d}\right), \tag{18}$$

donde Q es la función de cola de la distribución normal estándar. Denotemos SNR como $\Gamma = \frac{A^2}{\sigma^2}$, donde A es la amplitud de la señal.

IDEAL. Dado que la multiplicación y la FFT son lineales, el pico de datos complejos en IDEAL sigue la compleja distribución gaussiana compleja $\mathcal{CN}(h, M\sigma^2)$, donde $h = MA$. La altura del pico sigue la distribución de Rice, que puede aproximarse como distribución gaussiana $\mathcal{N}(\mu_1, \sigma_1^2)$, donde $\mu_1 = \sigma\sqrt{\frac{M\pi}{2}} L_1\left(-\frac{h^2}{2M\sigma^2}\right)$ y $\sigma_1^2 = 2M\sigma^2 + h^2 - \mu_1^2$. $L_1(\cdot)$ es el polinomio de Laguerre. Según la referencia [47], la altura máxima de otra altura de ruido M - 1 es aproximadamente $\sqrt{2M\sigma^2 \ln(M-1)} = c_1\sigma$, donde $H_l = \sum_{i=1}^{l} \frac{1}{i}$ representa el l-ésimo número armónico. Tomando μ_1 , σ_1 , y c_1 en la ecuación (18), obtenemos el SER de IDEAL como sigue;

$$SER = Q\left(\frac{\sqrt{2H_{(M)}(\Gamma)} - L_1\left(-\frac{1}{2}\right)\sqrt{\frac{\pi}{2}}}{\sqrt{M + \frac{\pi}{2} - \frac{\pi}{2}L_1^2\left(-\frac{1}{2}\right)}}\sqrt{M\Gamma}\right). \tag{19}$$

FPA. Si la compensación de fase $\theta = 0$, entonces el dominio de la frecuencia después de la adición es igual a IDEAL. Desplazar el ruido en θ no afecta a la distribución de ruido de media cero. Por lo tanto, la distribución de ruido después de la adición todavía se puede considerar como $\mathcal{CN}(0, M\sigma^2)$, lo que significa que la altura máxima de ruido es $c_2\sigma$, donde $c_2 = c_1$. El pico de datos en FPA sigue $\mathcal{CN}(h_1 + e^{j\theta}h_2, M\sigma^2)$, donde $h_1(h_2)$ es la primera (segunda) altura del segmento chirp y $h_1 + h_2 = h$. Sea $h_3 = |h_1 + e^{j\theta}h_2|$. La altura del pico sigue distribución de Rice, que puede aproximarse como $\mathcal{N}(\mu, \sigma^2)$, donde $\mu = \sigma\sqrt{\frac{M\pi}{2}} L_{\frac{1}{2}}\left(-\frac{h^2}{2M\sigma^2}\right)$, $\sigma^2 = M\sigma^2 + h_3^2 - \mu_2^2$. El SER del FPA está relacionado con el símbolo transmitido, y aquí simplificamos el envío símbolo con $h_1 = h_2 = \frac{h}{2}$. El mejor caso es que $\theta = 0$ y $h_3 = h$. El peor caso es que $|\theta| = \frac{\pi}{k}$ y

$h_3 = h\cos(\frac{\pi}{2k})$. Consideramos la media de ellas $h - \frac{1+(\cos)((\pi)(\cdot))}{2} = \frac{h\cos\frac{2}{2}(\frac{\pi}{4k})}{2}$ como h final . Por lo tanto, la SER del FPA puede aproximarse como

$$SER_2 = Q\left(\sqrt{\frac{2H_{(M)(\cdot)}(1) - (\cdot, \frac{\lambda}{2}L_1 - \frac{(M)(\Gamma)}{2}\cos^{(4)}(\frac{\pi}{4k})}{2 + M\Gamma\frac{\cos 4\pi}{4k} - \frac{\pi}{2}L^2 - \frac{(M)(\Gamma)\cos 4}{2}\frac{(\pi)\,t}{4k}}}\right). \quad (20)$$

CPA. Las dos alturas de pico de datos separadas en CPA siguen la distribución de Rice, respectivamente. El suma de ellos tienen la distribución aproximada $W(\mu_3, \sigma^2)$, donde $\mu_3 = \mu_3^J + \mu_3^{JJ}$, $\sigma^2 = 2M\sigma^{(2)} + h_1^2 + h^{(2)} - (\mu_{(3)}^J)(2) - (\mu_{(3)}^{JJ})2$, $\mu_{(3)}^J = \sigma_{2h}\frac{M\pi h)(1)}{2M\sigma}L_1(-\frac{(h)(1)(h)(2))}{2}$, $\mu_{(3)}^{JJ} = \sigma_{2h}\frac{M\pi h)(2)}{2M\sigma}L_1(-\frac{(h)(2)(h)(2))}{2}$. El pico de ruido

La altura sigue la distribución de Rayleigh. Denotemos el sumatorio en valor absoluto de dos partes de ruido como $Y_i (i = 1, 2, \ldots, M - 1)$. Y_i sigue aproximadamente a $W(\mu_Y, \sigma^{2)}$, donde

$$\mu_{(Y)} = h_1 + h_2\frac{M\pi}{2}\sigma, \sigma^2 = \frac{(4)(-)\pi}{2h}M\sigma^2. \quad (21)$$

Sea $S = \max_{i=1,2,\ldots}(,)(M)(-)(1)Y^{(i)}$. Por la desigualdad de Jensen [48]

$$e^{t(E)(S)} \le E\,e^{tS} = E\max_i e^{tY_i} \le \sum_{i=1}^{M-1} E\,e^{tY} = (M-1)e^{\mu_Y t + \frac{1}{2}\sigma_Y^2 t^2}, \quad (22)$$

donde la última igualdad se deduce de la definición de la función generadora de momentos gaussiana. Tomando el logaritmo a ambos lados de la desigualdad (22), tenemos

$$E(S) \le \frac{\ln(M-1)}{t} + \mu_Y + \frac{\sigma^2 t}{2}. \quad (23)$$

Para $t > 0$, según la desigualdad de medias aritméticas y geométricas, sabemos que

$$E(S) \le \mu_Y + \sqrt{2\ln(M-1)}\sigma_Y = c_3\sigma, \quad (24)$$

donde $c_3 = (\frac{h_1 + h_2}{2h}, \sqrt{\frac{M\pi}{2} + (4-\pi)M\ln(M-1)})$. Utilizamos la cota superior de $E(S)$ para estimar la max en el método CPA. Consideremos la situación $h_1 = h_2 = \frac{h}{2}$; tenemos

$$SER_3 = Q\left(\frac{(*)(\Gamma)}{\sqrt{2 + (M_2^{\lambda(\Gamma)(\cdot)})_2(\pi_2^{\lambda(L)})(\frac{2}{4})((-)^{(M)(\Gamma)(\cdot)})}}\right). \quad (25)$$

REFERENCIAS

[1] Shuai Tong, Zhenqiang Xu y Jiliang Wang. 2020. CoLoRa: Enabling multi-packet reception in LoRa. En *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'20)*. IEEE, 2303-2311.

[2] Xianjin Xia, Yuanqing Zheng, Tao Gu. 2020. FTrack: Decodificación paralela para transmisiones LoRa. *IEEE/ACM Trans. Netw.* 28, 6 (2020), 2573-2586. https://doi.org/10.1109/TNET.2020.3018020

[3] Zhenqiang Xu, Shuai Tong, Pengjin Xie y Jiliang Wang. 2020. FlipLoRa: Resolución de colisiones con cuasi-ortogonalidad arriba-abajo. En *Proceedings of the IEEE International Conference on Sensing, Communication and Networking (SECON'20)*.

[4] Zhe Wang, Linghe Kong, Kangjie Xu, Liang He, Kaishun Wu y Guihai Chen. 2020. Online concurrent transmissions at LoRa gateway. En *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'20)*. IEEE, 2331-2340.

[5] Wenju Zhao, Shengwei Lin, Jiwen Han, Rongtao Xu y Lu Hou. 2017. Diseño e implementación de un sistema de riego inteligente basado en LoRa. En *Actas de los talleres IEEE Globecom (GC Wkshps'17)*. IEEE, 1-6.

[6] Irfan Fachrudin Priyanta, Frank Golatowski, Thorsten Schulz, y Dirk Timmermann. 2019. Evaluation of LoRa tech- nology for vehicle and asset tracking in smart harbors (Evaluación de la tecnología LoRa para el seguimiento de vehículos y activos en puertos inteligentes). En *Proceedings of the 45th Annual Conference of the IEEE Indus- trial Electronics Society (IECON'19)*, Vol. 1. IEEE, 4221-4228.

[7] Davide Magrin, Marco Centenaro y Lorenzo Vangelista. 2017. Evaluación del rendimiento de las redes LoRa en un escenario de ciudad inteligente. En *Actas de la Conferencia Internacional del IEEE sobre Comunicaciones (ICC'17)*. IEEE, 1-7.

[8] Umber Noreen, Ahcène Bounceur y Laurent Clavier. 2017. A study of LoRa low power and wide area network technology. En *Actas de la Conferencia Internacional sobre Tecnologías Avanzadas para el Procesamiento de Señales e Imágenes (ATSIP'17)*. IEEE, 1-6.

[9] Rashad Eletreby, Diana Zhang, Swarun Kumar y Osman Yağan. 2017. Empowering low-power wide area net- works in urban settings. En *Actas de la Conferencia ACM del Grupo de Interés Especial en Comunicación de Datos (SIGCOMM'17)*.

[10] Shuai Tong, Jiliang Wang y Yunhao Liu. 2020. Combating packet collisions using non-stationary signal scaling in LPWANs. En *Proceedings of the ACM International Conference on Mobile Systems, Applications, and Services (MobiSys'20)*.

[11] Yao Peng, Longfei Shangguan, Yue Hu, Yujie Qian, Xianshang Lin, Xiaojiang Chen, Dingyi Fang y Kyle Jamieson. 2018. PLoRa: Una red pasiva de datos de largo alcance a partir de transmisiones LoRa ambientales. En *Proceedings of the ACM Con- ference of the Special Interest Group on Data Communication (SIGCOMM'18)*.

[12] Vamsi Talla, Mehrdad Hessar, Bryce Kellogg, Ali Najafi, Joshua R. Smith y Shyamnath Gollakota. 2017. LoRa backscatter: Enabling the vision of ubiquitous connectivity. *Proc. ACM Interact. Mobile Wear. Ubiq. Technol.* 1, 3 (2017), 1-24.

[13] Mehrdad Hessar, Ali Najafi y Shyamnath Gollakota. 2019. Netscatter: Habilitación de redes de retrodispersión a gran escala. En *Actas del simposio USENIX sobre diseño e implementación de sistemas en red (NSDI'19)*. 271-284.

[14] Adwait Dongare, Revathy Narayanan, Akshay Gadre, Anh Luong, Artur Balanuta, Swarun Kumar, Bob Iannucci y Anthony Rowe. 2018. Charm: Exploiting geographical diversity through coherent combining in low-power wide-area networks. En *Actas de la 17ª Conferencia Internacional ACM/IEEE sobre Procesamiento de la Información en Redes de Sensores (IPSN'18)*. IEEE, 60-71.

[15] Akshay Gadre, Revathy Narayanan, Anh Luong, Anthony Rowe, Bob Iannucci y Swarun Kumar. 2020. Configuración de frecuencia para redes de área extensa de baja potencia en un latido. En *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'20)*.

[16] Pieter Robyns, Peter Quax, Wim Lamotte y William Thenaers. 2018. Un decodificador de software multicanal para el esquema de modulación LoRa. En *Actas de la Conferencia Internacional sobre Internet de las Cosas, Big Data y Seguridad (IoTBDS'18)*.

[17] Matthew Knight y Balint Seeber. 2016. Decodificación de LoRa: realización de una LPWAN moderna con SDR. En *Actas de la Conferencia GNU Radio*.

[18] Joachim Tapparel, Orion Afisiadis, Paul Mayoraz, Alexios Balatsoukas-Stimming y Andreas Burg. 2020. An open- source LoRa physical layer prototype on GNU radio. En *Proceedings of the IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC'20)*. 1-5. DOI:http://dx.doi.org/10.1109/SPAWC48557.2020. 9154273

[19] Josh Blum. 2016. Módem LoRa con LimeSDR. Obtenido de https://myriadrf.org/news/lora-modem-limesdr/.

[20] RevSpace. DecodingLoRa. Obtenido de https://revspace.nl/DecodingLora.

[21] Alexandre Marquet, Nicolas Montavont y Georgios Z. Papadopoulos. 2019. Investigating theoretical performance y técnicas de demodulación para LoRa. En *Actas del simposio internacional del IEEE sobre un mundo de redes inalámbricas, móviles y multimedia (WoWMoM'19)*. IEEE, 1-6.

[22] Semtech. 2020. Hoja de datos SX1276/77/78/79, Rev. 7.

[23] Especificación LoRaWAN v1.1. Obtenido de https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/.

[24] Enlace Symphony. Obtenido de https://www.link-labs.com/symphony.

[25] Semtech. 2019. Hoja de datos SX1268, Rev. 1.1.

[26] Olivier Bernard, André Seller y Nicolas Sornin. 2014. Transmisor de largo alcance de baja potencia. Patente EP 2 763 321 A1.

[27] Xiong Wang, Linghe Kong, Liang He y Guihai Chen. 2019. mLoRa: Un protocolo de recepción multipaquete en redes LoRa. En *Actas de la conferencia internacional del IEEE sobre protocolos de red (ICNP'19)*. IEEE, 1-11.

[28] Rajalakshmi Nandakumar, Vikram Iyer, y Shyamnath Gollakota. 2018. Localización 3D para dispositivos de tamaño subcentimétrico. En *Actas de la Conferencia ACM sobre sistemas de sensores en red integrados (SenSys'18)*.

[29] Yinghui Li, Jing Yang y Jiliang Wang. 2020. DyLoRa: Towards energy efficient dynamic LoRa transmission control. En *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'20)*. IEEE, 2312-2320.

[30] Nodo LoRaMac. Obtenido de https://github.com/Lora-net/LoRaMac-node.

[31] Proyecto de pasarela LoRa SX1302. Obtenido de https://github.com/Lora-net/sx1302_hal.

[32] ChirpStack. Obtenido de https://www.chirpstack.io/.

[33] Servidor LoRaWAN. Obtenido de https://github.com/gotthardp/lorawan-server.

[34] Decodificador Charm. Obtenido de https://github.com/WiseLabCMU/charm-decoder.
[35] Mehrdad Hessar, Ali Najafi, Vikram Iyer y Shyamnath Gollakota. 2020. TinySDR: Low-power SDR platform for over-the-air programmable IoT testbeds. En *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'20)*. 1031-1046.
[36] TinySDR. Obtenido de https://github.com/uw-x/tinysdr.
[37] Semtech. 2017. Hoja de datos SX1301, V2.4.
[38] Nadia Ben Atti, Gema M. Díaz-Toca y Henri Lombardi. 2006. El algoritmo berlekamp-massey revisitado. *Applic. Algebr. Eng. Commun. Comput.* 17, 1 (2006), 75-82.
[39] Wikipedia. LFSR. https://en.wikipedia.org/wiki/Linear-feedback_shift_register.
[40] Ross Williams et al. 1993. A painless guide to CRC error detection algorithms. https://zlib.net/crc_v3.txt.
[41] P. Fraga-Lamas y T. M. Fernández-Caramés. 2017. Ingeniería inversa del protocolo de comunicaciones de una tarjeta RFID de transporte público. En *Actas de la Conferencia Internacional del IEEE sobre RFID (RFID'17)*. 30-35.
[42] Karsten Nohl, David Evans, Starbug Starbug y Henryk Plötz. 2008. Reverse-engineering a cryptographic RFID tag.. En *Actas del Simposio de Seguridad USENIX*, Vol. 28.
[43] Ishtiaq Rouf, Robert D. Miller, Hossen A. Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe e Ivan Seskar. 2010. Security and privacy vulnerabilities of in-car wireless networks: A tire pressure moni- toring system case study. En *Proceedings of the USENIX Security Symposium*, Vol. 10.
[44] Milan Stute, David Kreitschmann y Matthias Hollick. 2018. La salsa secreta de mil millones de manzanas: Receta para el protocolo ad hoc de enlace directo inalámbrico de Apple . En *Actas de la 24ª Conferencia Internacional Anual sobre Informática Móvil y Redes*. 529-543.
[45] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir y Matthias Hollick. 2019. Mil millones de interfaces abiertas para Eve y Mallory: MitM, DoS y ataques de rastreo en iOS y macOS a través del enlace directo inalámbrico de Apple. En *Actas del simposio de seguridad de USENIX*. 37-54.
[46] Branden Ghena, Joshua Adkins, Longfei Shangguan, Kyle Jamieson, Philip Levis y Prabal Dutta. 2019. Desafío: Las LPWAN sin licencia aún no son el camino hacia la conectividad ubicua. En *Actas de la conferencia internacional anual de la ACM sobre informática móvil y redes (MobiCom'19)*. 1-12.
[47] Tallal Elshabrawy y Joerg Robert. 2018. Aproximación de forma cerrada del rendimiento BER de la modulación LoRa. *IEEE Commun. Lett.* (2018).
[48] Pascal Massart. 2007. *Concentration Inequalities and Model* Selection, Vol. 6. Springer.
[49] Qian Chen y Jiliang Wang. 2021. AlignTrack: Push the limit of LoRa collision decoding. *IEEE 29th International Conference on Network Protocols (ICNP'21)*, 1-11. DOI:10.1109/ICNP52444.2021.9651985
[50] Chenning Li, Hanqing Guo, Shuai Tong, Xiao Zeng, Zhichao Cao, Mi Zhang, Qiben Yan, Li Xiao, Jiliang Wang y Yunhao Liu. 2021. NELoRa: Towards ultra-low SNR LoRa communication with neural-enhanced demodulation. En *Proceedings of the 19th ACM Conference on Embedded Networked Sensor* Systems, 56-68.
[51] Shuai Tong, Zilin Shen, Yunhao Liu y Jiliang Wang. 2021. Combating link dynamics for reliable lora connection in urban settings. En *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 642-655.
[52] Jinyan Jiang, Zhenqiang Xu, Fan Dang y Jiliang Wang. 2021. Long-range ambient LoRa backscatter with parallel decoding. En *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 684-696.
[53] Qianwen Miao, Fu Xiao, Haiping Huang, Lijuan Sun y Ruchuan Wang. 2019. Sistema de asistencia inteligente basado en algoritmo de distribución de frecuencia con etiquetas RFID pasivas. *Ciencia y tecnología de Tsinghua* 25, 2 (2019), 217-226.

Recibido el 24 de febrero de 2022; revisado el 19 de mayo de 2022; aceptado el 5 de junio de 2022.