Understanding Long Range-Frequency Hopping Spread Spectrum (LR-FHSS) with Real-World Packet Traces

JUMANA BUKHARI, Computer Science, Florida State University, Tallahassee, United States and Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia
ZHENGHAO ZHANG, Computer Science, Florida State University, Tallahassee, United States

Long Range-Frequency Hopping Spread Spectrum (LR-FHSS) is a new physical layer option that has been recently added to the LoRa family with the promise of achieving much higher network capacity than the previous versions of LoRa. In this article, we present our evaluation of LR-FHSS based on real-world packet traces collected with an LR-FHSS device and a receiver we designed and implemented in software. We overcame challenges due to the lack of documentation of LR-FHSS, and our study is the first of its kind that processes signals transmitted by an actual LR-FHSS device with practical issues such as frequency error. Our results show that LR-FHSS meets its expectations in communication range and network capacity. We also propose customized methods for LR-FHSS that improve its performance significantly, allowing our receiver to achieve higher network capacity than those reported earlier.

CCS Concepts: • **Networks** → **Network experimentation**; **Network simulations;**

Additional Key Words and Phrases: Low Power Wide Area Networks, LR-FHSS, Network Capacity

ACM Reference Format:
Jumana Bukhari and Zhenghao Zhang. 2024. Understanding Long Range-Frequency Hopping Spread Spectrum (LR-FHSS) with Real-World Packet Traces. *ACM Trans. Sensor Netw.* 20, 6, Article 117 (October 2024), 30 pages. https://doi.org/10.1145/3694971

1 Introduction

LoRa [4] in recent years has emerged as one of the strongest wireless technologies for the connections of IoT devices to gateways potentially over long distances. In November 2020, Semtech, the company behind LoRa, announced a new **Physical layer (PHY)** option in the LoRa family, namely, **Long Range-Frequency Hopping Spread Spectrum (LR-FHSS)** [25]. LR-FHSS is completely different from the traditional **Chirp Spread Spectrum (CSS)** modulation in LoRa and is expected to achieve higher network capacity and support satellite communications in addition to terrestrial communications. Since the announcement, LR-FHSS has gained growing interest [14, 15, 22, 26, 37, 38, 45]. However, as LR-FHSS was introduced only recently, there is a lack of technical documentation and open-source resources. Most of the existing work thus relies on mathematical analysis or simulations with certain simplifying assumptions [15, 26, 37, 38, 45]; others list it as future work [14, 22]. Therefore, it would be beneficial to understand the complete details

Authors' Contact Information: Jumana Bukhari, Computer Science, Florida State University, Tallahassee, Florida, United States and Imam Abdulrahman Bin Faisal University, Dammam, Eastern, Saudi Arabia; e-mail: jbukhari@fsu.edu; Zhenghao Zhang, Computer Science, Florida State University, Tallahassee, Florida, United States; e-mail: zzhang@cs.fsu.edu.

ACM 1550-4859/2024/10-ART117
https://doi.org/10.1145/3694971

of LR-FHSS and conduct tests with signals from actual LR-FHSS devices so that the network performance can be evaluated when real-world issues, such as timing error, frequency estimation error, limits in error correction, and effectiveness of **Successive Interference Cancellation (SIC)**, are taken into account.

In this article, we present our study of LR-FHSS based on real-world packet traces. To the best of our knowledge, our study is the first of its kind that decodes actual signals transmitted by an LR-FHSS device. We used *SX1261* [32] as the transmitter, which is a commodity LR-FHSS development kit made available recently, and a USRP B210 [10] as the receiver. We collected traces of 1,000 randomly generated packets, where each trace consists of baseband samples taken by the USRP during the transmission of the packet. We also wrote software that can decode the trace correctly, i.e., convert the received baseband waveform into bits and pass the **Cyclic Redundancy Check (CRC)**. The design and implementation of our receiver have been demonstrated with real-world experiments in the POWDER wireless platform [16], which is an open platform with radios that can be controlled remotely. We used five radios as LR-FHSS nodes and one radio as the gateway and decoded almost all packets from the nodes when all nodes transmitted packets simultaneously almost non-stop. For a quantitative evaluation in networks of larger sizes, we relied on trace-driven simulations. For example, we added noise to the signal to find the **Packet Receiving Ratio (PRR)** as a function of the **Signal-to-Noise Ratio (SNR)**, from which the expected communication distance could be revealed. We also mixed the signals of a large number of packets into a synthesized trace, which emulates the actual signal received by the gateway when nodes in the network transmit packets at random times. From the number of packets that were decoded correctly by our receiver, the network capacity could be revealed. We have made our code and dataset publicly available at [17].

Our finding is, in short, that LR-FHSS mostly lives up to its expectations. That is, LR-FHSS signals can be decoded at low SNRs, such as −20 dB, which is comparable to the traditional CSS modulation [33] and therefore should achieve a similar communication distance. Also, as suggested earlier in [15, 33, 37], the network capacity with LR-FHSS is indeed much higher than that with CSS. It should be noted that the capacity with our receiver is actually higher than those reported in [15, 33, 37] mostly due to a few customized methods we designed for LR-FHSS. We also confirm that LR-FHSS achieves similar capacities in terrestrial and satellite networks.

We overcame a number of challenges in this study, which were mostly due to the lack of documentation of LR-FHSS. To decode the baseband waveform, complete information about signal modulation and packet format is needed. When we started this work, although various aspects of LR-FHSS have been discussed in sources such as [6, 9, 15], the information was not complete. It took us a bit of reverse engineering and some luck to be able to understand all details of LR-FHSS. The receiver design was also a challenge, because at the time, there was no available information on key issues such as how to detect a packet, perform symbol-level synchronization, and estimate the signal frequency. On the other hand, a poorly designed receiver results in poor performance and does not do justice for LR-FHSS. We were able to design every component of the receiver from scratch, as well as propose new methods, namely, customized error correction decoding and SIC, which were not mentioned in earlier studies but significantly improve the performance of LR-FHSS.

We are ready to admit that, to evaluate LR-FHSS, the ideal situation would be to deploy LR-FHSS nodes in a large area and record the signals from the nodes. While we have demonstrated our receiver design and implementation with a small network in the POWDER platform [16], our main approach was the trace-driven simulation due to a few practical constraints we could not overcome. First, as LR-FHSS achieves high capacity, to probe the actual limit of LR-FHSS, the number of nodes is large, i.e., over 60, which is beyond our current capability. Second, as LR-FHSS was designed to support satellite communications, to evaluate the performance LR-FHSS, the access of

at least one satellite is needed, which is even more challenging. Our trace-driven simulation does allow us to evaluate LR-FHSS when the real-world issues mentioned earlier are taken into account because the signals are from a real LR-FHSS transmitter. As our packet traces were collected under ideal conditions, i.e., with a strong line-of-sight path, the traces can be regarded as clean signals from an LR-FHSS transmitter, making them versatile for various kinds of purposes. For example, in our experiments in the POWDER platform [16], the radios could emulate LR-FHSS nodes because they could simply play the packet traces. Also, to evaluate LR-FHSS in various channels, the packet traces can be processed by widely accepted channel models to emulate the received signal in such channels, where the models can be the **Extended Typical Urban (ETU)** model [1, 2] for the terrestrial channel and the **Non-Terrestrial Networks (NTN)** model [11–13, 27, 41] for the satellite channel. As we have made our source code and dataset public for the community at [17], our packet traces will also facilitate future work on LR-FHSS by other researchers to suit their specific needs. It may be worth mentioning that the network capacity evaluation by Semtech itself [33] was also based on simulations.

To summarize, our contributions in this article include (1) complete documentation, source code, and packet traces of LR-FHSS with free access [17]; (2) a receiver design that achieves higher performance than those reported in earlier studies including that by Semtech itself [33]; and (3) an in-depth study of LR-FHSS that reveals the expected performance of LR-FHSS in terrestrial and satellite networks with real-world issues considered.

The rest of the article is organized as follows. Section 2 discusses related work. Section 3 gives a short description of LR-FHSS. Section 4 explains the trace collection. Section 5 explains our receiver design. Section 6 experimentally demonstrates our receiver. Section 7 evaluates LR-FHSS. Section 8 concludes the article.

2 Related Work

Many LPWAN technologies have emerged in recent years, such as LoRa [4], Sigfox [8], RPMA [7, 28], and NB-IoT [5, 18, 40], among which LoRa has attracted significant attention and has been deployed worldwide. The traditional physical layer of LoRa is based on the CSS, which has been extensively studied [19, 21, 23, 34–36, 39, 42–44]. LR-FHSS is completely different from CSS because LR-FHSS signals occupy a very narrow bandwidth, while CSS signals are chirps that occupy the entire system bandwidth. LR-FHSS has been branded by Semtech for supporting both satellite and terrestrial communications. Our work is different from studies on satellite communications about the traditional CSS modulation, such as [20, 24].

There has been increasing interest in LR-FHSS. In [15], the overall design of LR-FHSS was explained and the performance analyzed with simulations, where the results show that the network capacity with LR-FHSS is more than an order of magnitude higher than that with the traditional CSS modulation. In [45], the uplink of LoRa to satellite was studied, including the CSS and LR-FHSS modulations, where the results were obtained with simulations. In [37], LR-FHSS was studied theoretically and further verified with simulations. In [26], a closed-form solution of the outage probability of LR-FHSS in satellite-based IoT networks was given. In [38], connectivities of long-range satellite links, including LR-FHSS links, were studied with simulations. Due to the lack of information on LR-FHSS, certain simplifying assumptions were adopted in existing studies, such as perfect timing and frequency estimation, perfect error correction, and so forth. On the other hand, an actual LR-FHSS receiver may not estimate the timing and frequency perfectly and may not be able to decode a codeword correctly even if the number of symbols under collision is below a threshold. Further, SIC has not been considered for LR-FHSS in earlier work, which is actually one of the main factors in determining the network capacity. The need for an experimental or trace-driven study has been pointed out in [37]: “the presented results should be re-validated

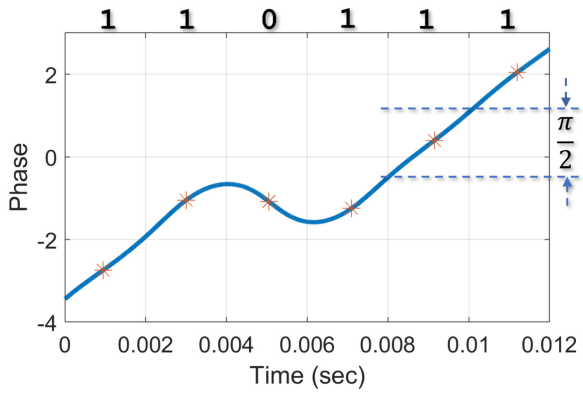


Fig. 1. An example of GMSK modulation.

and detailed by the follow-up studies, especially the experimental ones," which is the focus of our work.

Semtech itself has reported the performance of LR-FHSS in a recent application note [33], which includes the link packet loss ratio obtained with experiments and network capacity obtained with simulations. Our work serves as an independent study, which adds to the credibility of LR-FHSS. As will be discussed in Section 7, the network capacity achieved with our receiver is higher than that reported in [33], confirming our contribution in the receiver design.

3 Background

LR-FHSS was designed to support *uplink* communications from nodes to a gateway. The parameters of LR-FHSS depend on specific regions, such as the EU or the US; however, the difference is not fundamental. For example, the main difference between the EU and the US is the amount of system bandwidth, while the underlying signal modulations are the same. For simplicity, in this article, the focus is on the EU region, where two data rates, denoted as DR8 and DR9, are supported. The **Operating Channel Width (OCW)**, which is the total system bandwidth, is 137 kHz.

In LR-FHSS, a node transmits signals with a very narrow bandwidth of 488 Hz, which is referred to as **Occupied Band Width (OBW)**. The modulation is **Gaussian Minimum Shift Keying (GMSK)**, where each symbol modulates 1 bit. The symbol duration, denoted as T, is about 2 ms. Bit "0" is represented by a linear phase decrease of $\pi/2$ during the symbol time and "1" by a linear phase increase of $\pi/2$. The phase change is smoothed by a Gaussian filter so that the amount of phase change could be less than $\pi/2$ if 2 consecutive bits are different. Figure 1 shows an example of six symbols found in a packet trace, where the center of each symbol has been indicated with a marker and the symbol boundaries roughly align with the vertical lines.

A packet in LR-FHSS consists of headers and data fragments transmitted back to back on different frequencies, as shown in Figure 2. The header is always 0.233 seconds. For robustness, the header is repeated multiple times, which are 3 and 2 for DR8 and DR9, respectively, so that the gateway can detect the packet as long as one of the headers is received correctly. The data fragments are not repeated. There could be multiple fragments depending on the size of the payload, where each fragment is always 0.102 seconds except the last, which may be shorter because it may carry fewer bits.

A demodulated header and data fragment are shown in Figure 3. The header carries key information, such as the **Coding Rate (CR)**, payload length, and a 9-bit hop sequence ID, as well as an

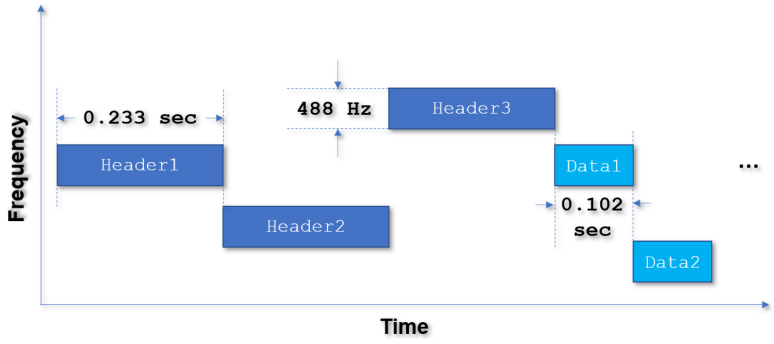


Fig. 2. A packet with DR8.

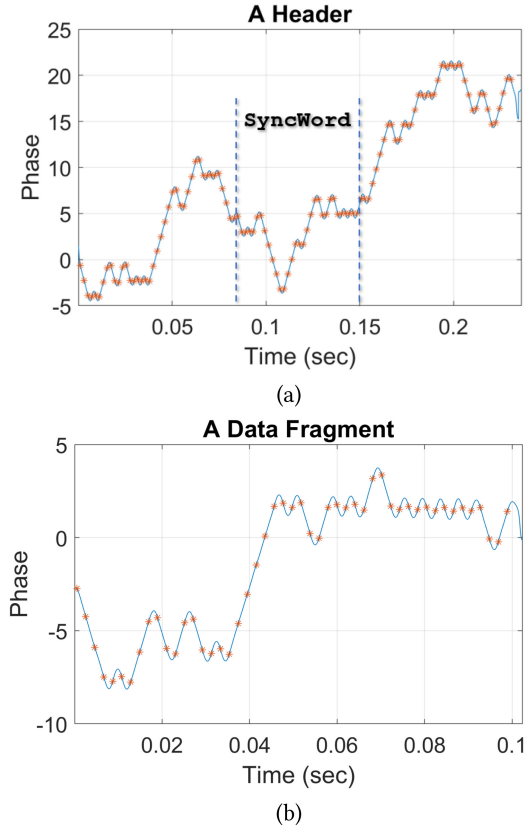


Fig. 3. The phase signals. (a) Header. (b) Data fragment.

8-bit CRC. With the hop sequence ID, the frequencies used by all headers and data fragments can be determined. The information and CRC are 40 bits in total, which are encoded into 80 bits by a convolutional code with rate 1/2 and memory depth 4. The 80-bit codeword is then interleaved. A constant binary vector of length 32, called the `SyncWord`, is then inserted into the middle of the codeword, which allows the receiver to estimate the symbol boundaries and frequency. The

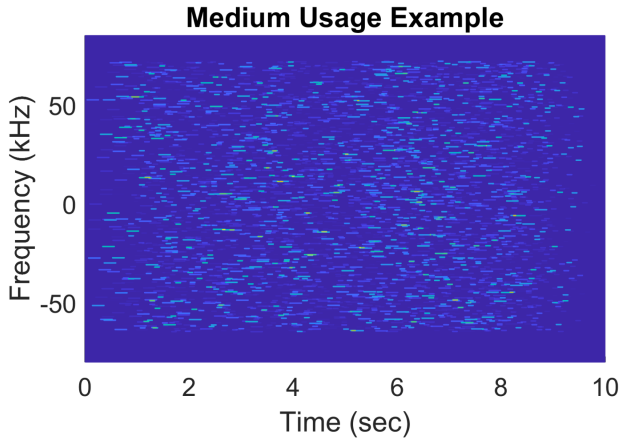


Fig. 4. The medium-usage map of a synthesized trace.

payload bits are first added with a 16-bit CRC, padded with six "0"s, then encoded by a convolutional code with memory depth 6, where the rates are 1/3 and 2/3 for DR8 and DR9, respectively. The codeword is then whitened, interleaved, and split into data fragments.

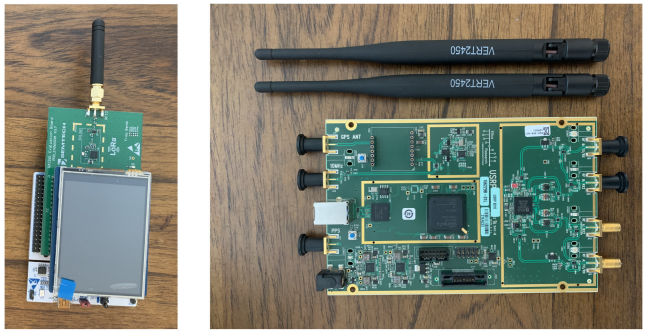
The system bandwidth supports 280 *channels*, where each channel is 488 Hz. Channels k, $k + 8$, $k + 16, \ldots, k + 272$ are in *group* k, where $1 \leq k \leq 8$. To transmit a packet, a node randomly selects a group and hops only among channels belonging to the same group. Figure 4 shows the medium usage of a synthesized trace containing 400 DR8 packets with payload size randomly distributed between 8 and 16 bytes. It can be seen that the medium is fairly busy in this trace; still, our receiver was able to decode over 90% of the packets.

LR-FHSS is very simple for the node, because a node only needs to encode the data and randomly select the transmission frequencies. The **Medium Access Control (MAC)** layer is basically ALOHA; i.e., a node can transmit a packet at any time of its choice. There is no need to maintain tight timing or frequency synchronization with the gateway, because packets can be transmitted at any time and frequency errors are equivalent to moving the signals to different frequency channels. Multiple access is supported by the physical layer due to the narrowness of the channel and random choices of frequency, which result in a low level of collision when the traffic load is not high. Long communication range is achieved also due to the narrowness of the channel because the noise power is proportional to the communication bandwidth.

4 Packet Trace Collection

We collected packet traces with SX1261 [32] as the transmitter and a USRP B210 [10] as the receiver. Figures 5(a) and 5(b) show the SX1261 and USRP, respectively.

We placed the transmitter and receiver close to each other with a line-of-sight path so that the signal from SX1261 was very strong and could be used to approximate clean signals from an LR-FHSS transmitter. Clean signals are preferred because, as mentioned earlier, they allow radios in the POWDER platform [16] to emulate LR-FHSS nodes, as well as facilitating trace-driven simulations. To elaborate, as the traffic is random, in a synthesized trace, the number of overlapping packets can fluctuate largely over time, e.g., between 20 and 60. As the signal and noise in all traces are added, if the packet traces are noisy, the actual SNR in the synthesized trace could also fluctuate, making it difficult to run an evaluation for a given SNR. In addition, the traces may be processed by different channel models. If the traces were collected in a strong multi-path environment, the



(a) (b)

Fig. 5. (a) SX1261. (b) USRP B210.

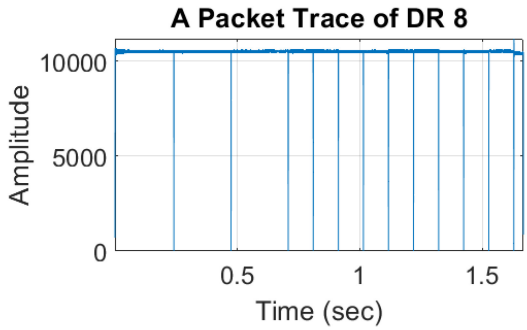


Fig. 6. A packet trace with DR8.

existing multi-path in the trace may interfere with the channel model. The trace of a packet with DR8 is shown in Figure 6, which has three header replicas and 10 data fragments. It can be seen that the signal is very clean and stable. There exist natural separations between the headers and data fragments because LR-FHSS inserts a small idle period between consecutive headers and data fragments.

For both DR8 and DR9, 500 packets were collected. The payload length of a packet was between 8 and 16 bytes, which does not include the 16-bit CRC. The number of packets of each size was either 55 or 56 so that the total number of packets of each data rate adds up to 500. Depending on the payload length, the number of data fragments with DR8 and DR9 were between 6 and 10 and 3 and 5, respectively. The transmission times of a packet with DR8 and DR9 were between 1.26 seconds and 1.67 seconds and 0.75 seconds and 0.95 seconds, respectively. The content of the payload and the hop sequence ID were randomly generated for each packet. Packet generation was reasonably convenient with SX1261, because SX1261 uses its driver to prepare the packet in software. As we had access to the driver code, in particular, `sx126x_lr_fhss_ping.c`, we could freely modify the content as well as system parameters of the packet, such as transmission power, carrier frequency, code rate, bandwidth, number of headers, and so forth.

During the trace collection, the system bandwidth was 137 kHz, the carrier frequency was 915 MHz, and the sampling rate was 500 ksps. For each raw trace, the start and end of the packets were found based on energy level and only signals containing the packets were kept. Each packet was then written to a trace file to be stored. As each sample is 4 bytes with 2 bytes for both the real and imaginary parts, a trace file is between 1.5 MB and 3.5 MB.

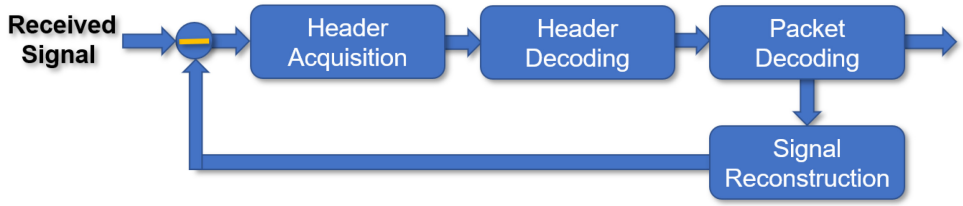


Fig. 7. The receiver structure.

5 Receiver Design

We designed and implemented a software LR-FHSS receiver from scratch, with which the performance of LR-FHSS can be evaluated. The receiver design is explained in this section.

5.1 Overview

A high-level structural view of our receiver is shown in Figure 7. The time-domain signal is first processed by the header acquisition component, which detects headers in the signal and finds the symbol boundaries and frequencies. The detected headers are then demodulated by the header decoding component. After this point, multiple headers belonging to the same packet are consolidated as one detected packet. Then, the detected packets are decoded by the packet decoding component. After a packet has been decoded, SIC is performed; i.e., the signal of the packet is reconstructed by the signal reconstruction component and subtracted from the received signal so that more packets can be detected and decoded.

5.2 Challenges and Contributions

When we started working on LR-FHSS, the complete information of LR-FHSS was not available. We got most help from an article [15], a document by Semtech [6], and the driver code of SX1261 [9]. However, the information was limited to the sender side at a high level, such as the error correction code, interleaving, whitening, and so forth, while no information could be found about the receiver on issues such as packet detection and signal demodulation.

We overcame the lack of documentation with a bit of reverse engineering and luck. One such example was related to `SyncWord`, which is needed to estimate the symbol boundary and frequency of the packet. We learned in [6] that the value of `SyncWord` is `0x2C0F7995`. We also knew that `SyncWord` is in the header. However, at the time, the exact location of `SyncWord` was unclear. Nevertheless, we started playing with the trace signal. It was fairly easy to identify the start of the header, which was a sharp rise of energy. Then, we manually tried different frequency offsets, because we knew that with the correct frequency offset, the phase signal should resemble those shown in Figure 3. Once that happened, we happily found that `SyncWord` matches the waveform in the middle of the header, which was a surprise to us because we thought it was at the beginning of the header like preambles in most other wireless networks. In fact, in Semtech's own document [6], `SyncWord` was shown at the beginning of the header structure in a figure, which has likely propagated to other documents and papers such as [15].

As mentioned earlier, the challenges also include designing the receiver to achieve good performance, as there was no existing literature on the receiver design for LR-FHSS. Therefore, our contribution also includes designing and implementing a functional receiver, as well as proposing novel solutions to improve the performance of LR-FHSS. To elaborate, some of the problems could be solved by standard library functions. For example, we were able to use the Matlab `lowpass` and `vitdec` functions to filter the signal and decode the convolutional code of the data, respectively.

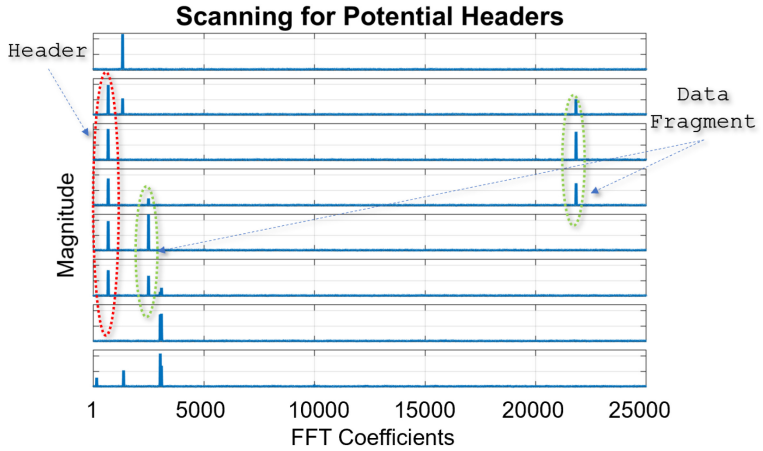


Fig. 8. The initial screening of headers.

However, most of the problems could not be solved in this manner. For example, header acquisition requires a customized solution because it is specific to LR-FHSS. Some of the other problems, such as demodulation and SIC, do not have good matching library functions, for which we had to design our own solutions. We also proposed a novel method, called **Interference Power-Aware Decoding (IPAD)**, which improves the error correction performance with little additional complexity.

5.3 Header Acquisition

Header acquisition refers to detecting a header and finding key parameters about the packet, including symbol boundary and frequency. Header acquisition is challenging because the header could start at any time on any frequency, and there could be many headers and data fragments transmitted at the same time.

5.3.1 Initial Screening. Our method starts with an initial screening step that spots potential *candidate headers* by looking for high-energy peaks in the frequency domain because the LR-FHSS signal is in a narrow band with energy concentrated in a small frequency range. One challenge is to separate headers from data fragments, because both produce such high peaks. Fortunately, a simple solution can be used because the lengths of header and data fragments are different. To elaborate, we divide the time-domain signal into segments of L seconds, where L is empirically chosen as 0.05, and perform FFT on each segment. As the header and data fragments are in five to six and two to three consecutive segments, respectively, they typically generate four to six and two to three observable peaks at the same or close locations in consecutive segments. Therefore, a candidate header can be identified if a *header pattern* is spotted, where header pattern is defined as peaks at very close locations in four or more consecutive segments. Note that a header sometimes generates only four peaks because it may overlap with six segments but the amount of overlap with the first and last segments is small and does not generate detectable peaks when the signal is weak. L is a system parameter. A larger L leads to better resilience against noise because more signal energy is collected in the segment; however, it may also blur the difference between the header and data fragments because it reduces the difference of the number of peaks. The value of 0.05 appeared to be the best based on empirical tests. Figure 8 shows the FFTs of eight consecutive segments in which a header pattern has been marked. Two data fragments have also been marked, which belong to the same packet.

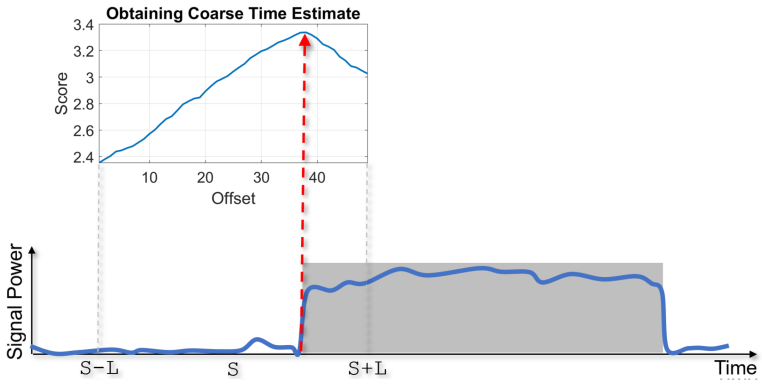


Fig. 9. Sliding window during coarse timing estimation.

When a header pattern is spotted, a candidate header is added to a list. The initial estimated start time of the header is simply the start time of the first segment where the header pattern is found. The initial estimated frequency can be calculated based on the location of the peak. The maximum timing error at this point is no more than L seconds, which may occur when the header starts at the end of a segment but is still identified because the signal is very strong. It should be noted that the frequency domain signal is always within the channel bandwidth but may exhibit peaks at random locations. A Gaussian filter is therefore used to smooth the frequency domain signal, which merges the energy within the signal bandwidth into one high peak at the center, so that the frequency error is typically within 100 Hz.

5.3.2 Coarse Timing Estimation. The next step obtains coarse estimates of the start times of the candidate headers. Suppose a candidate header has been spotted with an initial estimated start time of S. The idea is to slide a window of length D seconds, where D is the length of the header, starting from $S - L$ and stopping at $S + L$. When the window overlaps with the header exactly, maximum signal energy from the header is observed. Therefore, for a window that starts at $S - L + x$, where x is called the *offset*, the *score* is defined as the total amount of energy inside the window within the frequency bandwidth of the header. To reduce the timing error to the level of symbols, the window slides down T seconds at a time, where T denotes the symbol time. Figure 9 illustrates this process, where a header is shown at the bottom, which can be identified by the higher power than signals before and after. The score is shown at the top, which first rises then drops and is proportional to the amount of overlap of the sliding window and the header. The arrow in the figure points to the highest score, which is achieved when the window, shown as a shaded rectangle, completely overlaps with the header.

It should be mentioned that before the search is performed, the time-domain signal should be first down-converted then passed through a **Low-Pass Filter (LPF)** so that the energy of the header can be examined in the baseband without the interference and noise on other frequencies. The down-conversion is simply to divide the time-domain signal by a sinusoid whose the frequency is the initial estimated frequency of the header so that the header signal is moved to the baseband, i.e., within ±244 Hz. The LPF then rejects signals outside the baseband, which includes signals from other packets, as well as most of the noise. Removing noise outside the baseband gives the signal a huge SNR boost, because the noise power is proportional to the signal bandwidth. That is, as the OCW is 137 kHz while the OBW is only 488 Hz, after the LPF, the SNR is increased by 280 times, which is roughly 25 dB. As we could not find specifications about the LPF in LR-FHSS, we used a relatively "high-end" LPF in Matlab, namely, the `lowpass` function, which yields better

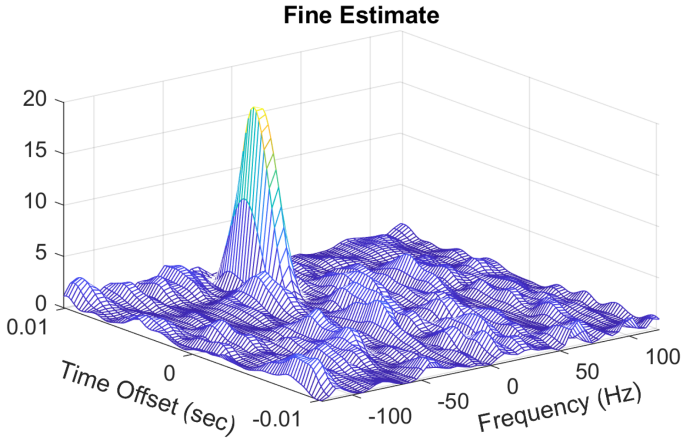


Fig. 10. The inner product with `SyncWord` during fine timing and frequency estimation.

performance than other LPFs we have tried. To be more specific, we used an `iir` filter where the cutoff frequency is 200 Hz, `StopbandAttenuation` is 100, and `Steepness` is 0.9. The cutoff frequency is slightly lower than half of OBW because of the transition band of the filter.

5.3.3 Fine Estimation. Lastly, for a candidate header, fine estimations of the symbol boundary and frequency are obtained. It is expected that, before this step, the timing and frequency errors have been reduced to within $5T$ seconds and 100 Hz, respectively. The objective of this step is to further reduce the errors to a level that allows for correct demodulation of the symbols, which, with our current implementation, are $T/10$ seconds and 5 Hz, respectively.

Unlike the initial screening and coarse timing estimation, fine estimation cannot be achieved based only on energy observations and must rely on the `SyncWord`. If the receiver has obtained the correct symbol boundary and frequency, it can take samples at the locations of the `SyncWord`, which should match `SyncWord`, i.e., produce a high *inner product* with `SyncWord`. Therefore, different combinations of timing and frequency offsets can be applied to the header signal and the one with the highest inner product can be used as the fine estimate.

To be more exact, denote a combination of timing and frequency offset as (t, f), where t and f are measured in seconds and Hertz, respectively. Let $V_{t,f}$ be the *test vector* of (t, f), which is obtained by taking 32 samples starting at t seconds with an interval of T seconds and then applying a frequency offset of f Hz. Let $\Psi_{t,f}$ be the *score* of (t, f), which is calculated according to

$$\Psi_{t,f} = | < V_{t,f}, \Theta > |^2, \tag{1}$$

where $<,>$ denotes the inner product calculation and Θ denotes the `SyncWord`. To achieve the desired level of accuracy, t and f increment a step of $T/10$ and 5, respectively, where t is within $\pm 5T$ and f is within ± 100. Figure 10 shows a typical search, where the peak, which corresponds to the best match, can clearly be seen.

In the implementation, the exhaustive search can be simplified with the help of FFT because FFT is much faster than naively calculating the inner products. Basically, FFT helps find the likely location of the peak, and the exhaustive search is only performed for points near the peak. To be more exact, let t_h be the h^{th} timing offset to be tested. Let matrix M be a matrix where row h is $|FFT(V_{t_h,0} \odot \bar{\Theta})|$, where $\odot$ denotes element-wise multiplication and $\bar{\Theta}$ denotes the conjugate of Θ. Suppose the maximum value of M is at row $\hat{h}$ and column $\hat{k}$. The best estimate of timing offset should be close to $t_{\hat{h}}$, while the best estimate of frequency offset should be close to $\hat{f}_{\hat{k}}$, where

$\hat{f}_k = k/(32T)$ Hz because the frequency peak is at location k when the sinusoid completes k cycles in $32T$ seconds. (t, f) is evaluated if $|t - t_{\hat{h}}| \leq 0.4T$ seconds and $|f - \hat{f}_k| \leq 20$ Hz, which reduces the number of inner product calculations from over 5,000 to no more than 81. The inner product calculation cannot be completely avoided because FFT can only evaluate frequency values that complete an integer number of cycles in 32 samples, while the actual frequency offset may not lead to an integer number of cycles.

5.4 Demodulation

The demodulation step converts the waveform of each symbol into a real number to be used by the error correction decoder. The information can be extracted from the slope of the phase signal, because a negative phase change should indicate "0" and a positive change "1." For simplicity, for each symbol, only two samples, denoted as m_1 and m_2, are used, which are to the left and right of the center of the symbol by $T/4$ seconds, respectively. The phase change from m_1 to m_2 is used as the estimate of the slope. Phase values near the boundaries of the symbol should be avoided because of the smoothing near the symbol boundary. It was found that using more than two samples only marginally improves the performance because the signal has already been through the low-pass filter and over-sampling does not further improve the SNR. As the error correction decoder needs a soft demodulation value, the output is the *normalized slope*, which is the estimated slope divided by the ideal slope when the bit is "1." For a receiver with multiple antennas, the slope is estimated for each antenna individually and then combined according to **Maximum Ratio Combining (MRC)**, where the weight of an antenna is proportional to its channel strength.

5.5 Error Correction with Interference Power-Aware Decoding (IPAD)

LR-FHSS uses different convolutional codes for the header and the data fragments. The details of the encoders can be found in the driver code of SX1261 [9]. In both cases, Viterbi decoders with soft decoding can be used. The encoder of the data always starts with initial state 0 and always pads "0"s to the end of the data so that the encoder returns to state 0 at the end. Therefore, we were able to use the standard `vitdec` function in Matlab to decode the data. The encoder of the header, on the other hand, is different, for which we had to write our own code. To elaborate, recall that the memory depth of the header code is 4. The encoder may start with any of the 16 states as the initial state and does not pad "0"s to the end. To decode the header, all 16 possible initial states are attempted. The one leading to the minimum cost is the estimated initial state, which will be used for signal reconstruction.

We further augment the decoder with a technique specifically designed for LR-FHSS, called IPAD. Often, a packet cannot be decoded correctly because part of it collides with another packet on a close frequency. Fortunately, in LR-FHSS, most of the collisions are known even before packets are being decoded because as long as one of the headers of a packet is decoded correctly, the start time, end time, and frequency of each header and data fragment of the packet are known. Therefore, if most packets have been detected correctly, the entire time-frequency occupancy map such as that in Figure 4 can be obtained, with which the collision conditions of every packet can be obtained to assist data decoding. For example, symbols under heavy collision should be discarded as they can be toxic. More generally, if the interference power of a symbol is higher, the soft value of the symbol should be given less confidence so that it will be less likely to misguide the decoder.

To estimate the interference power, suppose the data fragment to be decoded is centered at frequency 0. Suppose the number of interfering packets is U. Let the frequency and power per sample of packet u be F_u and P_u, respectively, where $1 \leq u \leq U$. Let $\Psi(f)$ be the interference power of a packet with unit power per sample with frequency f Hz. The estimated interference

power, denoted as ρ, is clearly

$$\rho = \begin{cases} 0 & \text{if } U = 0 \\ \sum_{u=1}^{U} P_u \Psi(F_u) & \text{if } U > 0 \end{cases}. \tag{2}$$

The power per sample of a packet is proportional to the highest score found during the fine estimation step of header acquisition averaged over all detected headers of this packet. $\Psi(f)$ is a constant and can be calculated offline by feeding a packet of unit power per sample centered at f Hz to the low-pass filter and measuring the amount of energy in the output. Let σ be the estimated noise power per sample after the low-pass filter, which is a constant in the system. Let P_0 be the estimated power per sample of the packet to be decoded. The **Signal-to-Interference and Noise Ratio (SINR)** and the SNR are therefore $P_0/(\sigma + \rho)$ and P_0/ρ, respectively.

Let x_0 be the original soft value of the demodulated symbol based on the slope. The soft value sent to the Viterbi decoder, denoted as x, is shrunken from x_0 by the ratio of the SINR to SNR:

$$x = x_0 \frac{(\frac{P_0}{\sigma+\rho})}{(\frac{P_0}{\sigma})} = x_0 \frac{\sigma}{\sigma + \rho}. \tag{3}$$

Intuitively, according to Equation (3), the soft value is "attenuated" more when ρ is higher. To elaborate, note that the Viterbi algorithm basically finds the binary values of the transmitted symbols to minimize a total cost. When the soft value of an individual symbol is x_0, assuming there is no interference, the cost of the symbol is proportional to $(x_0 - v)^2/\sigma$ for $v \in \{-1, 1\}$ because the noise is assumed to be white Gaussian. The cost should be adjusted if it is known that some symbols are under interference. Approximating the summation of noise and interference still as white Gaussian, the effect of the interference can be treated as increasing the noise power from σ to $\sigma + \rho$. As a result, the cost of a symbol under interference should be $(x_0 - v)^2/(\sigma + \rho)$. One method to implement this is to feed the Viterbi decoder $(x_0, \sigma + \rho)$ for each symbol if the Viterbi decoder understands that the second value is to be treated as a weight factor. However, the `vitdec` function in Matlab does not have an option to accept a weight for each symbol, which, in effect, is to treat the cost as $(x_0 - v)^2/\sigma$, i.e., to assume all symbols have the same noise level. Equation (3) is therefore used instead because

$$\frac{(\frac{x_0\sigma}{\sigma+\rho} - v)^2}{\sigma} = \frac{x_0^2\sigma}{(\sigma+\rho)^2} - 2v\frac{x_0}{\sigma+\rho} + \frac{1}{\sigma}, \tag{4}$$

while the actual cost should be

$$\frac{(x_0 - v)^2}{\sigma + \rho} = \frac{x_0^2}{\sigma+\rho} - 2v\frac{x_0}{\sigma+\rho} + \frac{1}{\sigma+\rho}. \tag{5}$$

Note that in both Equation (4) and Equation (5), the first and third terms are irrelevant to the choice of v and therefore do not affect the decoding result, while the second terms in Equation (4) and Equation (5) are identical.

A further modification was made for the following reason. When the interference is much stronger than the noise, according to Equation (3), the soft value is shrunken significantly, which can actually be counterproductive when the SINR of the symbol is still high. In fact, the soft value need not be adjusted if the SINR is already high. Therefore, the soft value is adjusted only when the SINR is less than ζ, which is empirically chosen as 5 for DR8 and 10 for DR9. When the soft value is adjusted, the SNR is replaced with ζ if the SNR is higher than ζ. This modification also improves the robustness against errors due to various approximations that have to be adopted in the process.

5.6 Successive Interference Cancellation (SIC)

SIC is a widely adopted method to reduce interference and improve network performance. The idea is that, after a packet has been decoded, its waveform can be regenerated and subtracted from the received signal so that the interference it causes to other packets can be removed. Note that it is not difficult to regenerate the waveform to match the transmitted waveform, which is simply to pass the decoded packet through the modulation process. The challenge is to match the *received* waveform, which has been modified by the wireless channel and is subject to residual errors of symbol timing and frequency estimation. Such residual errors may not bother data demodulation but can lead to significant errors during signal reconstruction.

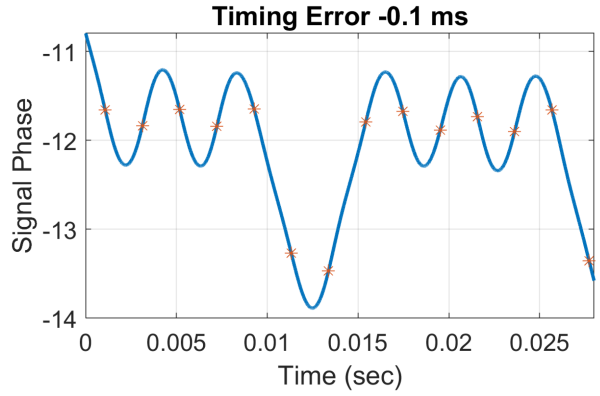
We designed a customized SIC method for LR-FHSS that can be applied to both headers and data fragments. In the following, it is explained for data fragments. The input consists of the data bits as well as Γ, which is the received waveform of the fragment. Γ should have been down-converted and passed through the low-pass filter. Given the data bits in this fragment, the ideal waveform, denoted as Γ^*, can be obtained by linearly increasing or decreasing the phase according to the value of each bit. The core of the problem is to convert Γ^* to match the signal in Γ but not the noise.

For simplicity, first, consider a single antenna system. The differences between Γ and Γ^* are mainly caused by three factors, namely, channel distortion, timing error, and frequency error. Channel distortion exists because the wireless channel always modifies the amplitude and phase of the signal. Fortunately, owing to the narrow bandwidth, the channel is flat, so the channel state can be represented by a single complex number denoted as $ae^{i\theta}$; i.e., the received signal is basically the transmitted signal multiplied by $ae^{i\theta}$. As a reasonable estimate of a is the magnitude of the received signal, the only parameter that needs to be estimated is the channel phase θ. The timing error is denoted as τ and is defined as the difference between the actual start time of a symbol and the estimated start time of the same symbol. The residual frequency error is denoted as δ. As the data fragment has been decoded correctly, both τ and δ are small.

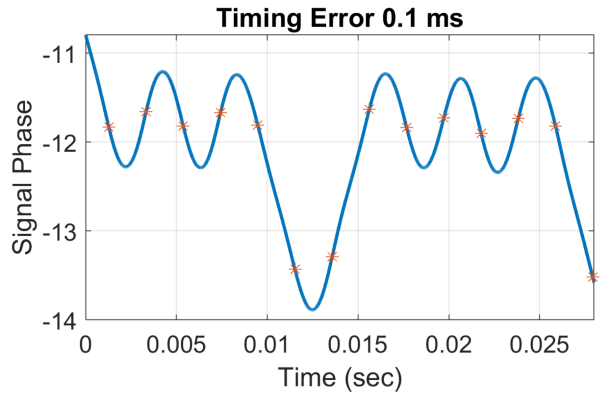
Our method is a two-step process, where τ is estimated during Step 1, while δ and θ are estimated during Step 2. We first explain Step 1, after which it should be clear why τ can be estimated without δ and θ. The estimation is based on transitions in the data from “0” to “1” and “1” to "0," because the phase errors found in such transitions are proportional to τ. To elaborate, define the phase of a symbol as the phase in the middle of the symbol. When $\tau = 0$, the phases of two symbols in a transition should be the same, because the phase has been increased and decreased by the same amount. However, as shown in Figure 11, if the estimated sample time is too early, i.e., $\tau < 0$, the phase difference is $-\pi\tau/T$ for a “0” to “1” transition and $\pi\tau/T$ for a “1” to “0” transition; similarly, if the estimated sample time is too late, i.e., $\tau > 0$, the amount of difference stays the same, but the signs are flipped. Therefore, based on the phase difference of the transitions, τ can be estimated. Note that θ is canceled during the phase difference calculation, while δ does not lead to a significant bias because it is small.

In Step 2, first, the sample time is adjusted based on τ. If there is no more timing error, $\Gamma = \Gamma^* ae^{i\theta} \odot W_\delta + n$, where W_δ denotes a sinusoid with frequency δ, n denotes noise, and $\odot$ denotes element-wise multiplication of two vectors. Therefore, to estimate δ and θ is to find the phase difference of Γ and Γ^* and fit the difference with a line, where the slope and y-intercept are δ and θ, respectively. To accommodate phase jitters in certain channels, Step 2 is actually performed every 10 symbols, called a *recon-segment*. As an example, Figure 12 shows a typical case, where the reconstructed signal matches the actual signal very well.

Next, consider multiple antenna systems with A antennas where $A > 1$. τ can be estimated independently for each antenna and combined according to MRC. δ should still be the same across antennas, but the channel phase, denoted as θ_1, θ_2,. . ., θ_A, could be different. Therefore, they should be jointly estimated. To be more specific, let the number of samples used in the



(a)



(b)

Fig. 11. Phase error due to timing errors. (a) Negative error. (b) Positive error.

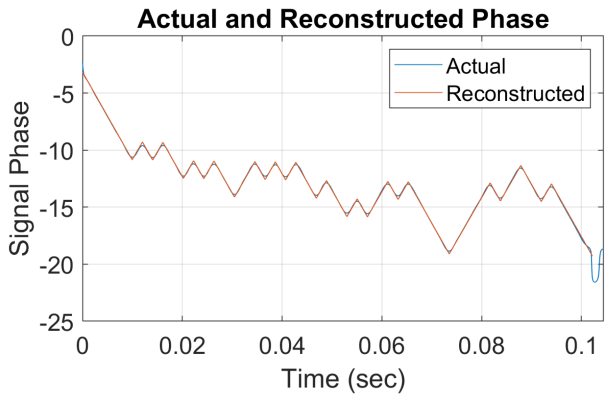


Fig. 12. Actual and reconstructed phase signal of a data fragment.

recon-segment be N, which is currently 40. Let the difference of the received phase signal and the ideal phase be a matrix denoted as Θ; i.e., $\Theta_{a,t}$ is the phase difference of the signal from antenna a at sample t, where $1 \leq a \leq A$ and $1 \leq t \leq N$. As the signal strengths of different antennas are

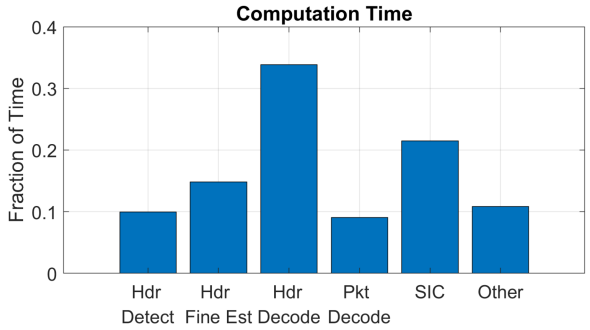


Fig. 13. Computation time breakdown in one typical case.

different, a *weight* is assigned to each antenna depending on the signal strength, which is denoted as w_a for antenna a and normalized so that $\sum_{a=1}^{A} w_a = 1$. Mathematically, the problem can be formalized as finding δ and $\theta_1, \theta_2, \ldots, \theta_A$ so that

$$G = \sum_{a=1}^{A} \sum_{t=1}^{N} w_a (\Theta_{a,t} - \delta t - \theta_a)^2 \tag{6}$$

is minimized. Taking the partial derivative of G with respect to θ_a and setting it 0, it can be found that

$$\theta_a = \frac{\sum_{t=1}^{N} \Theta_{a,t}}{N} - \frac{(N+1)\delta}{2}. \tag{7}$$

Similarly, taking the partial derivative of G with respect to δ and setting it 0, it can be found that

$$\delta \sum_{a=1}^{A} w_a \sum_{t=1}^{N} t^2 = \sum_{a=1}^{A} \sum_{t=1}^{N} w_a \Theta_{a,t} t - \frac{N(N+1)}{2} \sum_{a=1}^{A} w_a \theta_a, \tag{8}$$

which leads to

$$\delta = \frac{\sum_{a=1}^{A} \sum_{t=1}^{N} w_a \Theta_{a,t} t - (N+1)/2 \sum_{a=1}^{A} \sum_{t=1}^{N} w_a \Theta_{a,t}}{N(N+1)(2N+1)/6 - N(N+1)^2/4}. \tag{9}$$

5.7 Computation Time

Our receiver is currently implemented in software and is mainly intended for enabling the analysis of LR-FHSS and revealing its performance. The computation complexity of our receiver is reasonable for offline processing, e.g., about 13 minutes to process a trace of 10 seconds with 400 DR8 packets on a machine with 2.9 GHz CPU and 32 GB memory. The complexity is roughly proportional to the number of packets in the trace because heavy computation is needed only for detected packets.

More details can be found in the example shown in Figure 13, which is a breakdown of the fraction of time spent on each task when processing a trace with 400 DR8 packets among which 91% were received correctly. About 25% of the time was spent on header acquisition, which includes detecting the potential headers and obtaining fine estimates of timing and frequency offset. About 34% of the time was spent on decoding the header signal into bits, which is the highest among all tasks. Header acquisition and decoding are time-consuming because the current priority is not to miss packets; as a result, some fake headers must be processed. About 9% of the time was spent on packet decoding, which is because the Matlab `vitdec` function is highly efficient. On the other hand, the Viterbi decoder for the header was written by us and is less efficient than the built-in function of Matlab. Moreover, as explained earlier, the header decoding must be repeated 16 times

because the initial state is unknown. About 22% of the time was spent on SIC, which is expected because signal reconstruction involves some heavy computation.

If packets are to be detected and decoded in real time, the receiver should be implemented in hardware. Certain tasks that are very time-consuming in software can be significantly sped up with hardware. For example, in the example discussed in Figure 13, over 49% of time was actually spent in one function that is in charge of down-converting the signal to baseband and low-pass filtering. Inside this function, low-pass filtering takes over 70% of the time. In a hardware implementation, such tasks can be completed at line speed and will not become the main bottleneck of the system. To further speed up the receiver, parallel processing can be adopted. For example, to remove the bottleneck caused by header processing and SIC, multiple units can be used to process multiple headers simultaneously or to reconstruct the signals of multiple packets simultaneously. It is straightforward to parallelize such tasks because header processing and signal reconstruction of different headers or packets are independent of each other.

6 Experimental Validation

We demonstrate our receiver design and implementation with real-world experiments in the POWDER wireless platform [16] and use this opportunity to compare LR-FHSS with the CSS modulation.

6.1 Methodology

We set up radios in POWDER as nodes or the gateway to test LR-FHSS. The radios in POWDER are **Universal Software Radio Peripheral (USRP)** devices [31] that can transmit packets by simply playing packet trace files. The gateway can simply receive the signal from all nodes and write the signal to a file to be processed. In our experiments, the sampling rate was always 500 kHz and the carrier frequency was selected between 3.4 and 3.55 GHz depending on the availability and the interference level of the spectrum. Each experiment always lasted 10 seconds where each node transmitted randomly selected packets with small random gaps between consecutive packets. The total number of packets transmitted by a node in one experiment was always 108, with nine packets on each size between 8 and 16 bytes. All packets from the same node were transmitted with the same power. DR9 was used rather than DR8 to maximize the traffic load. As the packet traces were collected with the same frequency group, a random group was selected for each packet by applying a frequency offset to the packet trace.

CSS experiments were run in the same manner as LR-FHSS. For fairness, the parameters of CSS were chosen to best match the data rate of DR9, which is 325 bps. That is, the **Spreading Factor (SF)** of CSS was 10, the **Coding Rate (CR)** was 4, and the bandwidth was 62.5 kHz. As a result, the data rate of CSS was 305 bps. Packet traces were collected in a similar manner as LR-FHSS. That is, a CSS device referred to as Feather [3] (Adafruit Feather M0 with RFM95 LoRa Radio 900 MHz) was used to transmit packets with sizes between 8 and 16 bytes. The packets were received by the USRP at close range and written to trace files. The CSS packets were decoded by TnB [30], the open-source implementation of which can be downloaded at [29]. TnB was used because it performs better than other open-source LoRa receivers in part because it exploits the block structure of the symbols and can correct more bit errors than the default Hamming decoder. As TnB prefers an **Over-Sampling Factor (OSF)** of 8, and the traces were collected with OSF 8; i.e., for each transmitted element of the chirp, eight samples were taken.

6.2 Communication Range and Robustness in the Real World

We first conduct a link test in which packets were transmitted from one POWDER radio to another with different transmission gains of the USRP. As a result, the PRR, which is a function of the signal

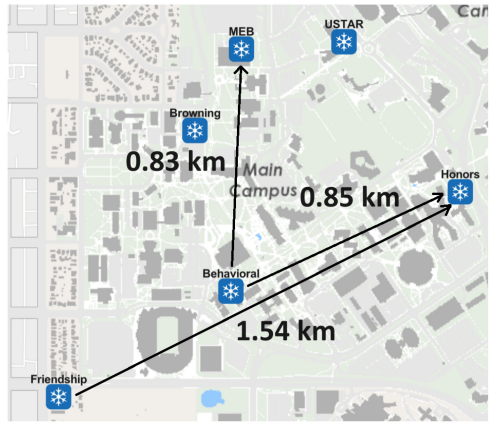


Fig. 14. Locations of radios used in the communication range test.

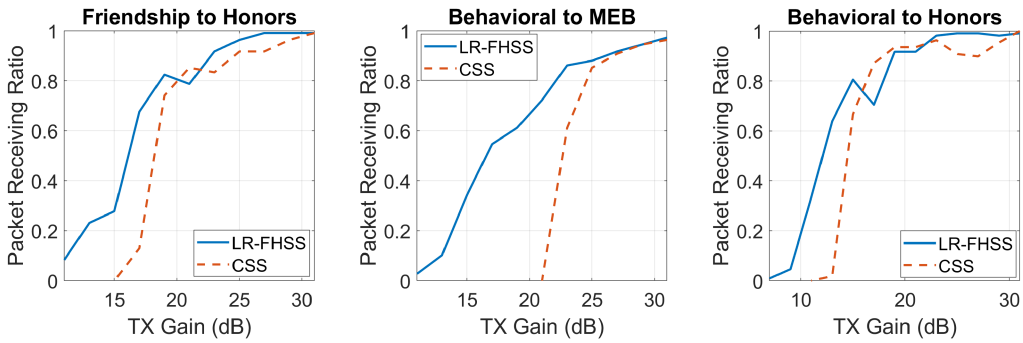


Fig. 15. Result of the communication range test.

power, changed from close to 1 to close to 0. A total of three links were tested, where the locations of the transmitter and receiver, as well as the distances of the links, are shown in Figure 14.

6.2.1 Communication Range. The PRRs of LR-FHSS and CSS are shown in Figure 15 as a function of the transmission gain for all three links. It can be seen that for the longest link, i.e., the link from "Friendship" to "Honors," which is 1.54 km, the PRR of LH-FHSS is above 0.9 when the transmission gain is 23 dB or above. As the maximum transmission gain of the USRP is 31.5 dB, a communication range of 1.54 km was achieved with only 15% of the transmission power of the USRP. Therefore, our experiment suggests that LR-FHSS is capable of supporting long communication range. It can also be seen that for all three links, LR-FHSS performed slightly better than CSS when the transmission gain was high; however, LR-FHSS performed much better when the transmission gain was low. This suggests that LR-FHSS is slightly better than CSS in communication range because the communication range should be defined as the length of the link when the link is operational with PRR around 0.9 or above; however, LR-FHSS likely survives better in more challenging situations with weak links.

6.2.2 Robustness in the Real World. With the signals collected in our experiments, we were also able to study the robustness of LR-FHSS and CSS in the real world. Specifically, we could study the robustness against two important factors, namely, interference and frequency drift.

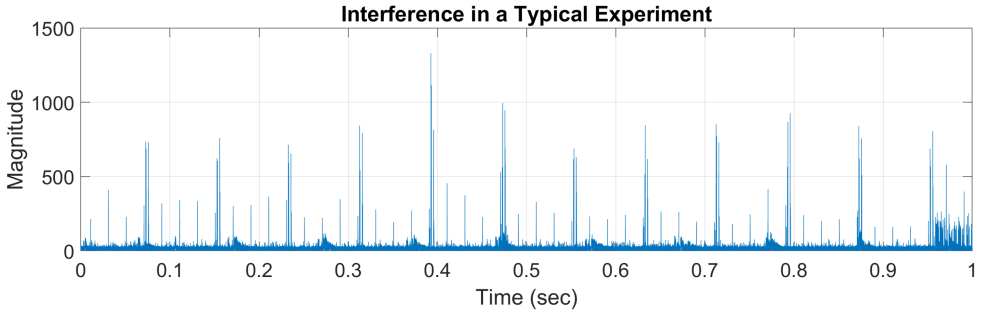


Fig. 16. Received time-domain signal in one typical experiment with heavy interference.

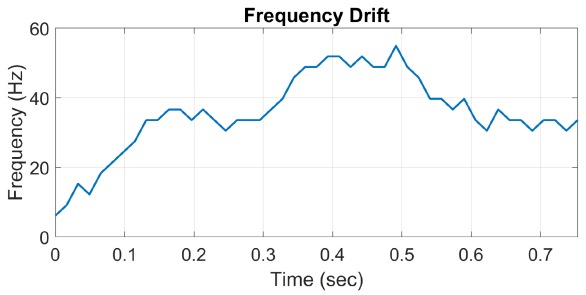


Fig. 17. Frequency drift during the reception of one packet.

There existed heavy interference from unknown sources when the experiments were conducted, which was out of our control. For example, Figure 16 shows the magnitude of the received time-domain signal of 1 second in a typical case. It can be seen that there were periodical transmissions with power significantly above the noise floor, some of which had a magnitude around 1,000. Such high-power samples were not due to our transmission because the SNR of our received signal was below 0 dB; i.e., our signal was weaker than noise. Frequency drift refers to the change of the carrier frequency, which can be caused by many factors, such as temperature change and Doppler shift, where the latter is more severe if the carrier frequency is higher. In our experiments, the amount of drift could be 50 Hz in 1 second, while the duration of a packet was also about 1 second. For example, Figure 17 shows the estimated frequency drift during the reception of one packet.

We took measures to suppress the interference and track the frequency drift. The specific type of interference encountered in our experiments was highly bursty, i.e., was present only for a small fraction of the time. Therefore, to reduce the impact of the interference, the high-magnitude samples can be masked, i.e., set to 0. The results in Figure 15 were obtained by masking all time-domain samples with magnitude above the 95th percentile. As the original TnB implementation [29] estimates the CFO only with the preamble at the beginning of the packet and does not track frequency drift during reception of a packet, we added a frequency tracking component to TnB. The results of CSS in Figure 15 were obtained with the frequency drift tracking option on. We did not add any code to track frequency drift for LR-FHSS because LR-FHSS is highly resilient to the amount of frequency drift encountered in our experiments.

Figure 18 reveals the robustness of LR-FHSS and CSS with the results from “Friendship” to “Honors.” To be more specific, in the figure, “LR-FHSS” refers to our LR-FHSS receiver with interference suppression on; “LR-FHSS_0” refers to our LR-FHSS receiver with interference suppression off;

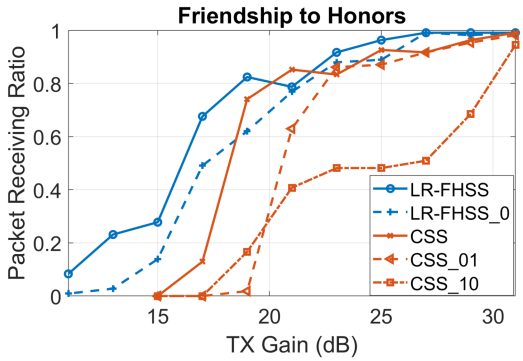


Fig. 18. Robustness against interference and frequency drift.

"CSS" refers to TnB with both interference suppression and frequency drift tracking on; "CSS_01" refers to TnB with interference suppression off but frequency drift tracking on; and "CSS_10" refers to TnB with interference suppression on but frequency drift tracking off. It can be seen that both schemes are resilient to interference when the signal is strong. For example, LR-FHSS and LR-FHSS_0 are very close when the transmission gain is 21 dB or above. As expected, when the signal becomes weaker, interference starts to cause more failures to packet reception. CSS is highly sensitive to frequency drift because the difference between CSS and CSS_10 is large.

Frequency drift turns out to be a major issue for CSS in our experiments because a CSS symbol is very long, i.e., 0.016 seconds, so that even a frequency drift of 50 Hz can cause demodulation errors because it can change the peak location by 1. Such drift is likely of a lesser concern when the data rate is higher because the symbol is shorter. For link tests, we track the frequency drift by probing nearby frequencies and move to the frequency that best matches the current symbol, i.e., produces the highest peak, which is an acceptable approach because only one packet is in the air at any time so that the highest peak must be generated by the packet we wish to receive. Frequency drift tracking is more challenging in a network scenario when multiple packets are in the air simultaneously, because it is not simple to determine the *owner* of a peak, where the owner refers to the packet that generated the peak. We adopt a best-effort approach by using the choice of TnB to determine the owners of peaks.

LR-FHSS is highly resilient to frequency drift around 50 Hz because it only leads to a small change of the phase slope and introduces a small bias to the soft demodulated values, which does not significantly affect the Viterbi decoder. It is possible to detect the bias, but we found it does not change the result in our experiments.

6.3 Network Test

We next compare LR-FHSS and CSS in a network setting by transmitting packets from five nodes to the gateway simultaneously. Figure 19 shows the locations of the radios and the SNR of the links measured in decibels estimated by TnB based on the received packets. The link SNRs were not proportional to the link distances in part because different transmitter gains were used for the nodes.

Figure 20 shows 10 seconds of the signal received by the gateway and the estimated signal from each individual node in a typical LR-FHSS experiment, where the estimated signal from a node is the reconstructed signal of all decoded packets from this node calculated by the SIC component of the LR-FHSS receiver. It can be seen that the nodes were transmitting packets almost non-stop

Fig. 19. Locations of the radios in the network test.

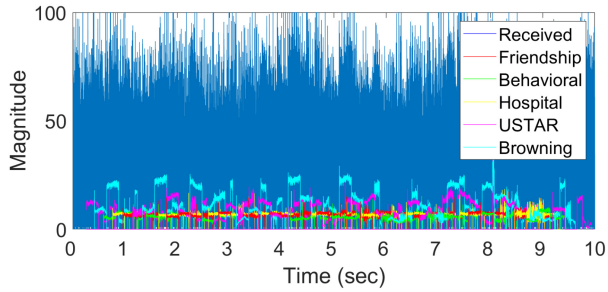


Fig. 20. The received signal and estimated signals from individual nodes.

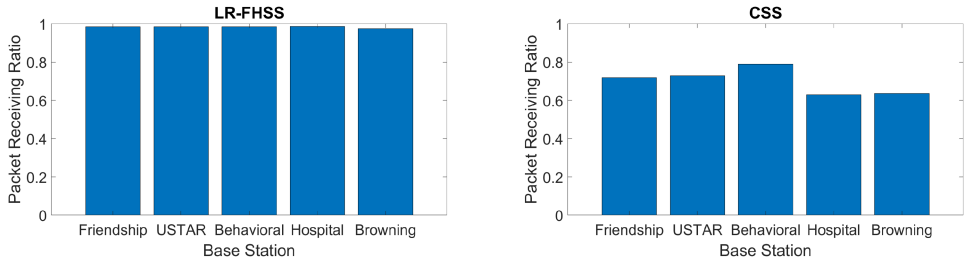


Fig. 21. The PRRs of the nodes in the experiment.

and had different signal powers. The signal power of the same node may also differ in different header and data fragments because of frequency-selective fading.

Figure 21 shows the PRR of all nodes, which is the average of three runs. It can be seen that the gateway was able to decode almost all LR-FHSS packets from the nodes, which confirms that our receiver is capable of decoding real-world signals over long-distance links in an environment filled with interference. Similar tests were conducted for the CSS modulation and the results are also shown in Figure 21. It can be seen that the PRR of CSS is much lower, which is because the CSS modulation is not originally designed to support simultaneous transmissions from multiple nodes.

6.4 Summary and Discussions

Our link experiments confirm that LR-FHSS indeed supports long communication range and is resilient to both interference and frequency drift encountered in our experiments. Our results suggest that LR-FHSS achieves slightly better communication range than the CSS modulation, which echoes those by Semtech in [33]. Our network test experiments demonstrate that our LR-FHSS receiver is capable of processing signals in the real world from multiple nodes with random packet arrival times, with random frequency offsets, and under heavy interference. It is also clear that LR-FHSS is capable of supporting much higher network capacity than CSS. It is, however, beyond our capability to set up an even larger network to probe the limit of the network capacity of LR-FHSS. Also, currently, testbeds with real satellite links are not available. We therefore further evaluate LR-FHSS with trace-driven simulations, as discussed in the next section.

7 Evaluation of LR-FHSS

In this section, we report our findings of the performance of LR-FHSS obtained with trace-driven simulations.

7.1 Channel Models

Three types of channels were tested, namely, the **Additive White Gaussian Noise (AWGN)** channel, the ETU channel [1, 2], and the NTN channel [11–13, 27, 41]. The AWGN channel is the simplest where only white Gaussian noise was added to the signal. The ETU channel represents challenging terrestrial channels with strong multi-paths, large channel fluctuation, and delay spread around 5 μs. The NTN channel represents non-terrestrial channels of satellite links. White Gaussian noise was also added to the signal in ETU and NTN channels. Different seeds were used for different nodes that lead to different channel parameters such as path delay and path gain. The number of antennas at the receiver was one for the AWGN channel and two for other channels, because the AWGN channel does not have antenna diversity. For the NTN channel, the carrier frequency was 10 GHz and the satellite speed was 7,562.2 m/s, which were selected according to typical conditions of **Low Earth Orbit (LEO)** satellite links. The delay profile was `NTN-TDL-D`, which has a strong line-of-sight path, because most users will likely set up the satellite link with a clear view of the sky. It was also assumed that the node could estimate the Doppler frequency shift and approximate it as a constant, which can be achieved by analyzing downlink signals from the satellite, so that it could cancel most of the Doppler shift before transmitting any packet.

7.2 SNR Threshold of One-to-one Links

The SNR threshold is defined as the minimum SNR to maintain the PRR at 0.9 or above in a one-to-one link where packet loss is not due to collision, which is an important metric that reveals the capability to deal with weak signals. It may be worth mentioning that the SNR threshold has not been reported in earlier work except in the document by Semtech [33], because the measurement requires the decoding of actual LR-FHSS packets.

We synthesized signal traces by adding only one packet to the trace and scaled it according to the SNR level. Figure 22 shows the results of DR8 and DR9 in all three types of channels. It can be seen that the SNR thresholds of DR8 are −19 dB, −19 dB, and −21 dB for the AWGN, ETU, and NTN channels, respectively, while those of DR9 are −17 dB, −16 dB, and −18 dB, respectively. The ETU channel is the most challenging because of strong multi-path and large channel fluctuations. It is somewhat unexpected that the performance is the best in the NTN channel, which is because the `NTN-TDL-D` delay profile is dominated by a strong line-of-sight path and at the same time can benefit from antenna diversity.

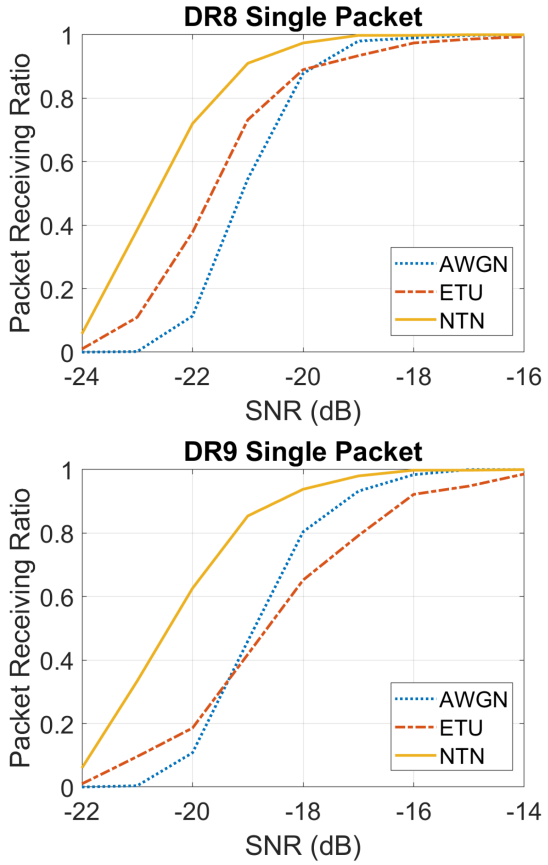


Fig. 22. Receiving ratio of packets in one-to-one links.

7.3 Network Capacity

Network capacity is defined as the total number of data bits received correctly by the gateway per second when the PRR is 0.9. LR-FHSS was proposed mainly to improve the network capacity.

Our receiver is capable of processing composite signals from multiple nodes. We synthesized traces that contained multiple packets by adding signals of individual packets in the time domain. To be more specific, a trace lasted 10 seconds and was initially only noise of unit power. Then, depending on the traffic load, a certain number of packets were randomly selected and added to the trace at random times. For each packet, first, a frequency group was randomly selected, according to which a frequency offset was applied to the packet. The packet was then passed through the channel, scaled according to its SNR, then added to the trace. The SNR was randomly selected within $[-17, 3]$ dB and $[-13, 7]$ dB for DR8 and DR9 packets, respectively. There was a range of 20 dB for both DR8 and DR9 to model errors in transmission power control. To focus on network-wide issues, the minimum SNR was a few decibels higher than the SNR threshold found in Section 7.2, so that packet loss was mainly caused by collision. The traffic load in all tests ranged from 0.48 kbps to 4.8 kbps at a step of 0.48 kbps. Basically, the lowest and highest loads corresponded to 50 and 500 packets in 10 seconds, respectively. When the traffic consisted of packets with one data rate, at the highest load, the average concurrency, i.e., the average number

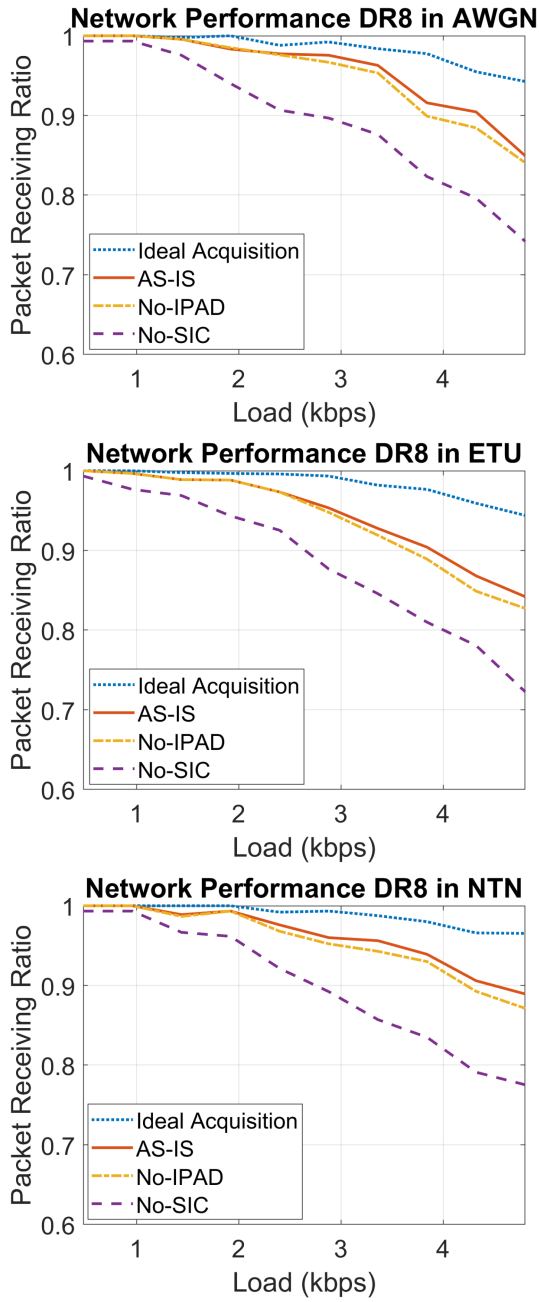


Fig. 23. Network capacity with DR8.

of packets transmitted simultaneously at any given time, was over 73 and 42 for DR8 and DR9, respectively.

7.3.1 DR8 or CR9 Only. First, we synthesized traces consisting of packets with only one data rate. Figure 23 and Figure 24 show the performance of DR8 and DR9, respectively. In the figures,

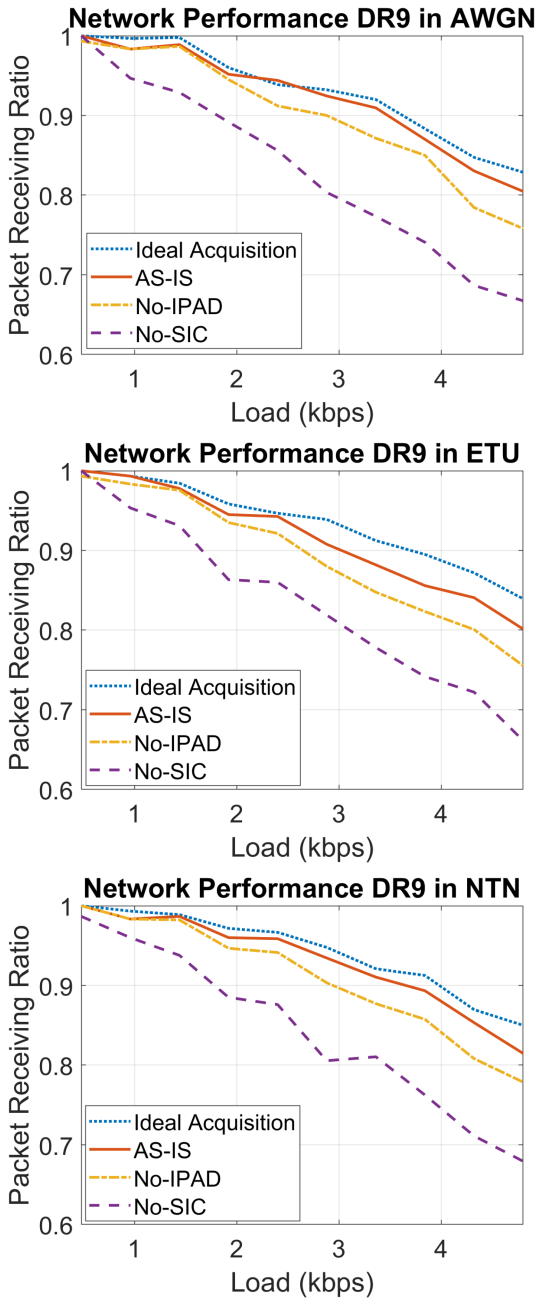


Fig. 24. Network capacity with DR9.

"AS-IS" refers to the actual receiver we designed, while others refer to the actual receiver with certain variations for comparison. To be more exact, "Ideal Acquisition" refers to "AS-IS" with perfect header acquisition; i.e., the header acquisition step is bypassed and the receiver is directly given the start time and frequencies of the packet and only needs to decode the data fragments; "No-IPAD" refers to "AS-IS" with IPAD disabled; and "No-SIC" refers to "AS-IS" without SIC.

It can be seen that the network capacities of DR8 are 3.92 kbps, 3.51 kbps, and 4.04 kbps in the AWGN, ETU, and NTN channels, respectively, while those of DR9 are 3.13 kbps, 2.72 kbps, and 3.29 kbps, respectively. The network capacities for the same data rate are similar in all channels because packet loss is mainly caused by collisions if the SNR is sufficiently high. In [15], it was reported that the maximum network goodput was about 2 kbps. The results in [37] suggested that the network capacity was about 2 kbps. In [33], Semtech reported that the network capacity is 700,000 pkt/day, which is 8.1 pkt/sec. Although the exact size of the packets used in the evaluation was not given, the typical packet size discussed in [33] was 20 bytes, suggesting a capacity of 1.3 kbps. The capacity achieved by our receiver is therefore consistently higher than those reported earlier in all channel models.

Our receiver achieves a higher network capacity because of more advanced signal processing techniques such as SIC and IPAD. In fact, in all cases, there exists a large gap between "No-SIC" and "AS-IS," confirming the importance of SIC. The gap between "No-IPAD" and "AS-IS" is also visible in all cases, confirming the effectiveness of IPAD. As IPAD is not computationally intensive, the gain is enjoyed almost for free. The gap between "AS-IS" and "Ideal Acquisition" is small for DR9 even at the highest load, suggesting that our header acquisition component works well even with collisions. The gap is larger for DR8 primarily because the packet transmission time with DR8 is much longer, leading to more collisions. DR8, however, still achieves a higher capacity, which will be discussed further shortly.

7.3.2 Both DR8 and CR9. We also synthesized traces when an equal number of packets were randomly selected for both DR8 and DR9. Figure 25 shows the performance of "AS-IS" for DR8 and DR9, where it can be seen that there exists a significant gap between DR8 and DR9, even when the SNR of DR8 was 4 dB lower than DR9 in the simulation. This could complicate rate selection because the data rate is usually determined by the channel condition. That is, a node should choose a higher data rate if its channel is stronger; and, with correct choices of data rates, all nodes should eventually enjoy roughly the same PRR. Therefore, some future research might be needed on rate selection for LR-FHSS.

We found that DR9 suffers more loss than DR8 mainly because, by design, it receives less protection from the error correction code. We examined the results under the highest traffic load and discuss the findings in the NTN channel because the general trends are the same for all channels. Figure 26(a) shows the loss ratio of packets as a function of the fraction of symbols in data fragments under collision, where a symbol is considered under collision if there exists another symbol within 250 Hz with energy at least half of this symbol. It can be seen that many packets with DR8 can still be received correctly even when a large fraction of symbols were under collision. For example, over 84% of the packets could still be decoded correctly even when over 40% of symbols were under collision. On the other hand, DR9 is much more sensitive to collision. To our surprise, under high traffic load, packet detection is not a main factor that leads to more packet loss with DR9. Figure 26(b) shows the distributions of the number of headers detected for each packet. It can be seen that there were more DR8 packets with the 0 header detected than DR9; in other words, actually, more packets with DR8 were lost due to failures of packet detection, even though DR9 packets have one less header replica. We believe this is because headers in DR8 and DR9 use exactly the same format and code rate, and DR9's higher signal power compensates for the fewer transmitted header replicas.

8 Conclusions

LR-FHSS is an important addition to the LoRa family. Preferably, it should be evaluated in a setting as close to the real world as possible, which was difficult because there were no sufficient open

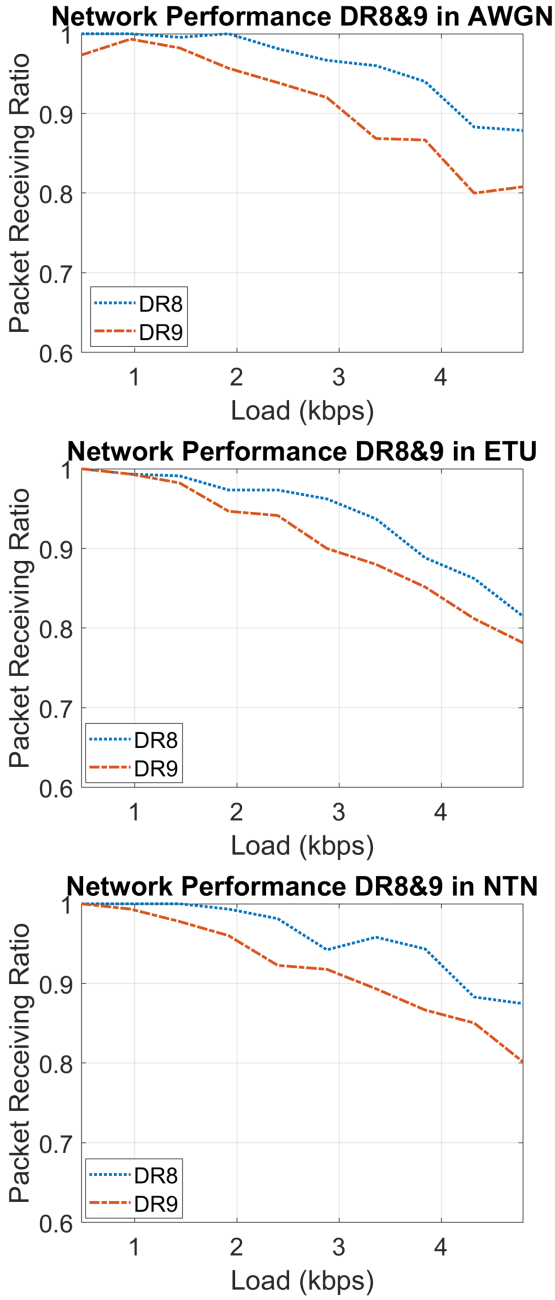


Fig. 25. Network capacity with DR8 and DR9.

documentations. We started this work facing many challenges, such as obtaining the complete understanding about the physical layer modulation and designing a receiver from scratch to convert the baseband waveform into bits. We were able to overcome these challenges and perform trace-driven simulations to reveal the performance of LR-FHSS, where real-world packet signals

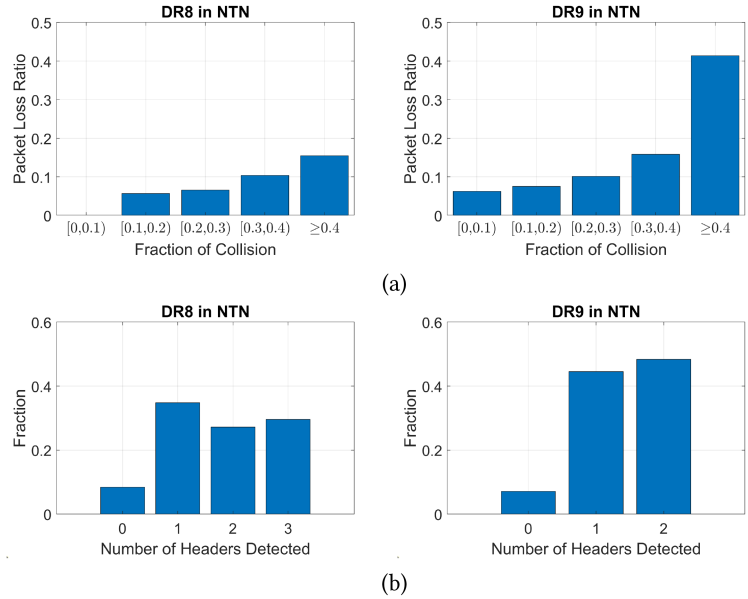


Fig. 26. Further analysis when the load is 4.8 kbps. (a) Packet loss ratio as a function of collision level. (b) Number of headers detected.

were processed and practical issues such as timing error, frequency error, error correction, and interference cancellation were taken into account. We also designed customized methods for LR-FHSS to improve its performance. We have uploaded our source code and trace files with free access.

References

[1] [n.d.]. 3GPP TS 36.101. User Equipment (UE) Radio Transmission and Reception. 3rd Generation Partnership Project; Technical Specification Group Radio Access Network. Evolved Universal Terrestrial Radio Access (E-UTRA).
[2] [n.d.]. 3GPP TS 36.104. Base Station (BS) Radio Transmission and Reception. 3rd Generation Partnership Project; Technical Specification Group Radio Access Network. Evolved Universal Terrestrial Radio Access (E-UTRA).
[3] [n.d.]. Adafruit Feather M0 with RFM95 LoRa Radio. https://www.adafruit.com/product/3178
[4] LoRaWAN 1.1 Specification. https://resources.lora-alliance.org/technical-specifications/lorawan-specification-v1-1
[5] NB-IoT. https://www.3gpp.org/news-events/3gpp-news/nb-iot-complete
[6] [n.d.]. RP002-1.0.3 LoRaWAN Regional Parameters. https://lora-alliance.org/resource_hub/rp2-1-0-3-lorawan-regional-parameters/
[7] [n.d.]. RPMA Technology. https://www.ingenu.com/technology/rpma
[8] [n.d.]. Sigfox. https://www.sigfox.com
[9] [n.d.]. SX126X driver. https://github.com/Lora-net/sx126x_driver
[10] [n.d.]. USRP B210. https://www.ettus.com/all-products/ub210-kit/
[11] 3rd Generation Partnership Project. Technical Specification Group Radio Access Network. [n.d.]. 3GPP TR 38.811. Study on new radio (NR) to support non-terrestrial networks.
[12] 3rd Generation Partnership Project. Technical Specification Group Radio Access Network. [n.d.]. 3GPP TR 38.901. Study on channel model for frequencies from 0.5 to 100 GHz.
[13] 3rd Generation Partnership Project. Technical Specification Group Radio Access Network. [n.d.]. 3GPP TR 38.821. Solutions for NR to support non-terrestrial networks (NTN).
[14] Mohammad Afhamisis and Maria Rita Palattella. 2022. SALSA: A scheduling algorithm for lora to LEO satellites. *IEEE Access* 10 (2022), 11608–11615. https://doi.org/10.1109/ACCESS.2022.3146021
[15] Guillem Boquet, Pere Tuset-Peiró, Ferran Adelantado, Thomas Watteyne, and Xavier Vilajosana. 2021. LR-FHSS: Overview and performance analysis. *IEEE Communications Magazine* 59, 3 (2021), 30–36. https://doi.org/10.1109/MCOM.001.2000627

[16] Joe Breen, Andrew Buffmire, Jonathon Duerig, Kevin Dutt, Eric Eide, Mike Hibler, David Johnson, Sneha Kumar Kasera, Earl Lewis, Dustin Maas, Alex Orange, Neal Patwari, Daniel Reading, Robert Ricci, David Schurig, Leigh B. Stoller, Jacobus Van der Merwe, Kirk Webb, and Gary Wong. 2020. POWDER: Platform for open wireless data-driven experimental research. In *Proceedings of the 14th International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WiNTECH '20)*. https://doi.org/10.1145/3411276.3412204

[17] Jumana Bukhari and Zhenghao Zhang. [n.d.]. LR-FHSS-receiver. https://github.com/jumanamirza/LR-FHSS-receiver

[18] Min Chen, Yiming Miao, Yixue Hao, and Kai Hwang. 2017. Narrow band Internet of Things. *IEEE Access* 5 (2017), 20557–20577. https://doi.org/10.1109/ACCESS.2017.2751586

[19] Qian Chen and Jiliang Wang. 2021. AlignTrack: Push the limit of LoRa collision decoding. In *2021 IEEE 29th International Conference on Network Protocols (ICNP '21)*. 1–11. https://doi.org/10.1109/ICNP52444.2021.9651985

[20] Roberto M. Colombo, Aamir Mahmood, Emiliano Sisinni, Paolo Ferrari, and Mikael Gidlund. 2022. Low-cost SDR-based tool for evaluating LoRa satellite communications. In *2022 IEEE International Symposium on Measurements & Networking (M&N '22)*. 1–6. https://doi.org/10.1109/MN55117.2022.9887761

[21] Rashad Eletreby, Diana Zhang, Swarun Kumar, and Osman Yagan. 2017. Empowering low-power wide area networks in urban settings. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '17)*. ACM, 309–321. https://doi.org/10.1145/3098822.3098845

[22] Juan A. Fraire, Pablo Madoery, Mehdi Ait Mesbah, Oana Iova, and Fabrice Valois. 2022. Simulating LoRa-based direct-to-satellite IoT networks with FLoRaSaT. In *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '22)*. 464–470. https://doi.org/10.1109/WoWMoM54355.2022.00072

[23] Bin Hu, Zhimeng Yin, Shuai Wang, Zhuqing Xu, and Tian He. 2020. SCLoRa: Leveraging multi-dimensionality in decoding collided LoRa transmissions. In *2020 IEEE 28th International Conference on Network Protocols (ICNP '20)*. 1–11. https://doi.org/10.1109/ICNP49622.2020.9259397

[24] Pansoo Kim, Sooyeob Jung, and Joon-Gyu Ryu. 2023. Improvements of IoT waveform for high doppler. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC '20)*. 995–996. https://doi.org/10.1109/CCNC51644.2023.10059811

[25] Tuofu Lu. 2020. LoRaWAN® Protocol Expands Network Capacity with New Long Range–Frequency Hopping Spread Spectrum Technology. https://blog.semtech.com/lorawan-protocol-expands-network-capacity-with-new-long-range-frequency-hopping-spread-spectrum-technology

[26] Alireza Maleki, Ha H. Nguyen, and Robert Barton. 2023. Outage probability analysis of LR-FHSS in satellite IoT networks. *IEEE Communications Letters* 27, 3 (2023), 946–950. https://doi.org/10.1109/LCOMM.2022.3233524

[27] Matlab. [n.d.]. Model NR NTN Channel. https://www.mathworks.com/help/satcom/ug/model-nr-ntn-channel.html

[28] Theodore J. Myers. 2008. Random phase multiple access system with meshing. US Patent 7773664B2.

[29] Raghav Rathi and Zhenghao Zhang. [n.d.]. TnB Implementation. https://github.com/raghavrathi10/TnB

[30] Raghav Rathi and Zhenghao Zhang. 2022. TnB: Resolving collisions in lora based on the peak matching cost and block error correction. In *The 18th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '22)*.

[31] Ettus Research. [n.d.]. Universal Software Radio Peripheral (USRP). https://www.ettus.com/products/

[32] Semtech. [n.d.]. SX1261DVK1BAS: Sub-GHz Development Tools SX1261 868MHZ DVK 2-Layers EU.

[33] Semtech. 2022. Application Note: LR-FHSS System Performance. https://www.mouser.vn/pdfDocs/AN1200-64_LR-FHSS_system_performance_V1_2.pdf

[34] Muhammad Osama Shahid, Millan Philipose, Krishna Chintalapudi, Suman Banerjee, and Bhuvana Krishnaswamy. 2021. Concurrent interference cancellation: Decoding multi-packet collisions in LoRa. In *ACM SIGCOMM 2021 Conference*, Fernando A. Kuipers and Matthew C. Caesar (Eds.). ACM, 503–515. https://doi.org/10.1145/3452296.3472931

[35] Shuai Tong, Jiliang Wang, and Yunhao Liu. 2020. Combating packet collisions using non-stationary signal scaling in LPWANs. In *The 18th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '20)*. ACM, 234–246. https://doi.org/10.1145/3386901.3388913

[36] Shuai Tong, Zhenqiang Xu, and Jiliang Wang. 2020. CoLoRa: Enabling multi-packet reception in LoRa. In *39th IEEE Conference on Computer Communications (INFOCOM '20)*. IEEE, 2303–2311. https://doi.org/10.1109/INFOCOM41043.2020.9155509

[37] Muhammad Asad Ullah, Konstantin Mikhaylov, and Hirley Alves. 2022. Analysis and simulation of LoRaWAN LR-FHSS for direct-to-satellite scenario. *IEEE Wireless Communications Letters* 11, 3 (2022), 548–552. https://doi.org/10.1109/LWC.2021.3135984

[38] Muhammad Asad Ullah, Anastasia Yastrebova, Konstantin Mikhaylov, Marko Höyhtyä, and Hirley Alves. 2022. Situational awareness for autonomous ships in the arctic: mMTC direct-to-satellite connectivity. *IEEE Communications Magazine* 60, 6 (2022), 32–38. https://doi.org/10.1109/MCOM.001.2100810

[39] Xiong Wang, Linghe Kong, Liang He, and Guihai Chen. 2019. mLoRa: A multi-packet reception protocol in LoRa networks. In *2019 IEEE 27th International Conference on Network Protocols (ICNP '19)*. 1–11. https://doi.org/10.1109/ICNP.2019.8888038

[40] Y.-P. Eric Wang, Xingqin Lin, Ansuman Adhikary, Asbjörn Grövlen, Yutao Sui, Yufei W. Blankenship, Johan Bergman, and Hazhir Shokri-Razaghi. 2017. A primer on 3GPP narrowband internet of things. *IEEE Communications Magazine* 55, 3 (2017), 117–123. https://doi.org/10.1109/MCOM.2017.1600510CM

[41] P Series; Radio Wave Propagation. [n.d.]. ITU-R Recommendation P.681-11 (08/2019). Propagation data required for the design systems in the land mobile-satellite service.

[42] Xianjin Xia, Ningning Hou, and Yuanqing Zheng. 2021. PCube: Scaling LoRa concurrent transmissions with reception diversities. In *ACM Mobicom*.

[43] Xianjin Xia, Yuanqing Zheng, and Tao Gu. 2020. FTrack: Parallel decoding for LoRa transmissions. *IEEE/ACM Transactions on Networking* 28, 6 (2020), 2573–2586. https://doi.org/10.1109/TNET.2020.3018020

[44] Zhenqiang Xu, Pengjin Xie, and Jiliang Wang. 2021. Pyramid: Real-time LoRa collision decoding with peak tracking. In *IEEE Conference on Computer Communications (IEEE INFOCOM '21)*. 1–9. https://doi.org/10.1109/INFOCOM42981.2021.9488695

[45] Guido Álvarez, Juan A. Fraire, Khaled Abdelfadeel Hassan, Sandra Céspedes, and Dirk Pesch. 2022. Uplink transmission policies for LoRa-based direct-to-satellite IoT. *IEEE Access* 10 (2022), 72687–72701. https://doi.org/10.1109/ACCESS.2022.3189647

Received 21 December 2023; revised 5 June 2024; accepted 30 August 2024