LoRa++: Mixed Up-Down CSS Modulation for Enhanced Scalability and Data Rate in LPWANs

Shrutkirthi S. Godkhindi*, Deeksha P Rao*, R Venkatesha Prasad†, T V Prabhakar*,
*Department of Electronic Systems Engineering, Indian Institute of Science, Bengaluru, India
†Delft University of Technology, Delft, Netherlands
*{shrutkirthig, deeksharao, tvprabs}@iisc.ac.in; †r.r.venkateshaprasad@tudelft.nl

***Abstract*—LoRa has been widely used for various Internet of Things (IoT) applications because of its simplicity & ease of deployment. However, LoRa suffers from low data rates and higher collisions. We propose & implement a mixed index modulation scheme LoRa++ that carries two additional bits per symbol in each of the supported spreading factors. We do this by introducing UpDown and DownUp chirps to augment the existing Up and Down chirps. Modulation and demodulation are non-trivial, but we can easily incorporate them within the existing LoRa nodes and gateways. Our results show that SNR requirements for LoRa++ demodulation are within the 1-2 dB range from conventional LoRa. Moreover, LoRa++ performs equally well for inter-SF interference. Our simulation results show that LoRa++ can accommodate more nodes due to the reduced time on air. Finally, we show that our hardware implementation on USRP is simple, and the results validate the performance of LoRa++. The performance of the new scheme in terms of BER is on par with that of conventional LoRa.**

***Index Terms*—LoRa, IoT, CSS modulation, Scalability**

I. Introduction and Motivation

Long-Range (LoRa) uses the chirp spread spectrum (CSS) to achieve long-distance (hundreds of kilometer) robust transmissions at low signal to noise ratio (SNR) for IoTs [1]. LoRa deployments must adhere to regional power level and duty cycle restrictions, as governed by LoRa Alliance. In the LoRa Wide Area Network (LoRaWAN), the data rate (DR), Time on Air (ToA), and range are governed by the Spreading Factor (SF), Bandwidth (BW), and transmission power. Furthermore, SFs are orthogonal to each other; thus, simultaneous data transmission using multiple SFs is possible. SF determines the data rate, i.e., it denotes the number of bits present in a single chirp symbol, typically in the range of 7 to 12 bits. The standard BW for LoRa is the choice between 125 kHz, 250 kHz, and 500 kHz. The symbol duration (T_s) depends on the selected BW & SF.

Figure 1 shows the contour map of the range for each SF, with the center (dark purple) depicting coverage of SF7 and the farthermost (light blue) portraying coverage of SF12. It also shows the number of nodes, ToA, and DR. Here, 'N' is the maximum number of nodes with non-overlapping transmissions that a gateway for a particular SF with 1% duty cycle can accommodate. In real-world deployments, as the number of devices in a LoRaWAN increases, the probability of successful packet reception reduces due to collisions. This is because LoRaWAN uses the ALOHA-type MAC protocol. Since current LoRaWAN devices can sense the medium (a.k.a. channel activity detection (CAD)), collisions are somewhat reduced. However, if the coverage is extensive, most of the devices are hidden from each other; thus, the CSMA protocol will not be scalable.

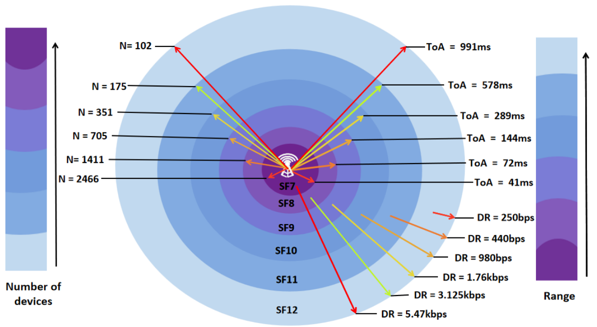


Fig. 1: **Range of different SF. The number of nodes N, communicating 8.5 kB of data over 24 hours; ToA (in ms) and DR (bps) for each SF to transmit a packet with 10 bytes is provided.**

Limitations of LoRaWAN. With the increase in the number of IoT devices, the support for scalability plays an important role in the deployment of IoT applications, and this leads to several concerns [2]–[4]: ❶ The ToA is high in LoRa since CSS is used; thus, the total number of nodes supported by a LoRa gateway is restricted, leading to lower scalability. ❷ The LoRa devices are constrained by the 1% duty-cycle restriction [5], e.g., considering 1% duty cycle for SF12, a node is allowed to transmit for only 864 s per day with a DR of 250 bps. ❸ LoRa signals are susceptible to intra-SF interference in large-scale deployments [6]. In summary, due to high ToA, higher collision probability, restrictions on DR, and low duty cycle the capacity of LoRaWAN is limited. In this paper, we introduce LoRa++, to address the DR within the framework of Conventional (Vanilla) LoRa.

LoRa++ approach. LoRa++ is a Physical (PHY) layer enhancement that still employs CSS and utilizes *different combinations of Up & Down chirps*. Vanilla LoRa conservatively uses only the Up chirp to make it easy to decode. We propose mixing the up and down chirps and generating multiple combinations to pack more bits per symbol while maintaining pseudo-orthogonality. However, the increase in bits causes rapid changes in the slope, thus requiring a slightly higher SNR to decode. In LoRa++, we add 2 bits to the symbol, striking a trade-off between SNR & DR to keep

ISBN 978-3-903176-72-0 © 2025 IFIP

bit error rate (BER) under check and increase bits per symbol. To the best of our knowledge, this is the first *implementation and Demonstration* of a mixed UpDown & DownUp chirp modulation scheme.

In the proposed LoRa++ scheme, the bits per symbol are SF+2 bits. For instance, LoRa++ employing SF7 spreading factor encodes 9 bits per symbol compared to 7 bits in vanilla LoRa. Similarly, LoRa++ with spreading factor $SF = \{8, 9, 10, 11, 12\}$ encodes $\{10, 11, 12, 13, 14\}$ bits per symbol respectively. Further, this method is extensible with a complex combination of Up and Down chirps to increase bits/symbol.

Contributions. ❶ A mixed index-based CSS modulation scheme, LoRa++ that employs mixed Up and Down chirps, is proposed, resulting in lower ToA and thus can accommodate more nodes within a network. ❷ Coherent and non-coherent decoding schemes are implemented for LoRa++ with minimal compromise on BER and SNR compared to conventional LoRa. Furthermore, the LoRa++ can be implemented on current LoRa nodes with minimal changes to the firmware. ❸ Demodulation at the gateway is implemented using open-source USRP hardware. The communication range of LoRa++ is tested for indoor and outdoor environments (with line of sight and non-line of sight conditions). ❹ An ns-3 implementation of the LoRa++ is presented where the PHY layer properties of LoRa++ modulation have also been incorporated. The simulation results provide a basis to prove the scalability of the solution. ❺ We provide thorough measurements and simulation results of the packet reception ratio, channel utilization, scalability, etc., at the LoRa gateway for varying numbers of nodes. We look at the literature and then the basics of LoRa.

II. Related works

Existing works in the literature propose and implement schemes to enhance the performance of conventional LoRa under Additive White Gaussian Noise (AWGN) and fading channels. These parameters include (a) data rate, (b) scalability, and (c) capacity of the network. Many efforts have been made to study and enhance the performance of LoRa in various environments by incorporating different strategies.

Data rate enhancement. Several physical layer techniques such as interleaving [9], [10], [18], [19], phase shift keying (PSK) [8], and slope shift keying (SSK) [7] have been incorporated with conventional LoRa scheme. This allows the addition of one or more bits to each symbol. Techniques such as IQCSS [11], DCRK-CSS [12] offers a technique to double the DR by embedding phase information in both I and Q components of the chirp. Further authors in GCSS [13], TDMA-LoRa [15] symbols are superimposed and multiplexed, which provides double DR.

Strategies to improve scalability and channel capacity. CurvingLoRa [16] proposes non-linear chirps instead of linear chirps; exploiting its feature of energy scattering across the bandwidth. An extension of this work is given in [20], where the nature of the curve represents the newly added bits to the symbols. However, this demands a higher demodulation complexity. A joint up-down modulation-demodulation scheme to increase the performance of LoRa was proposed in [14], but it fails to improve the bit capacity per symbol or not increase DR & network capacity. XCopy [21] provides an algorithm to retrieve packets by combining multiple copies of low SNR packets. A backscatter technique that combines on-off keying with CSS to allow concurrent transmission from a large number of LoRaWAN nodes within a network is proposed in [22]. FTrack [17] proposes an algorithm that increases throughput by $3\times$ by employing parallel decoding of LoRa signals using edge detection of the symbol. In [23], conventional Up chirp is divided into smaller sub-chirp. The orthogonal combination of these sub-chirps are used for concurrent transmission in order to enhance throughput.

To summarize, the aforementioned works are limited to extensive simulation studies. Moreover, their hardware implementation is non-trivial and demands additional computation both at the modulator and the demodulator for implementing works like [8], [11], [19], additional issues such as phase recognition, channel estimation, demultiplexer, phase interleaver, and frequency shift have to be addressed at the receiver. Moreover, the decoding complexity increases by $O(2^p)$, where p is the number of additional bits per symbol. LoRa inherently has a low data rate, further, it has a high ToA and hence limits the number of nodes. This is one of the major causes of reduced scalability. Another approach to this is to accommodate more bits in a symbol. *Thus, there is a need for a scheme that is less complex and easily implementable even on the existing hardware. This work, LoRa++ PHY layer scheme, is motivated by the idea of packing more bits without much loss in the SNR on current LoRa chipsets.* Several retransmission, gateway adoption strategies and MAC layer techniques that are proposed can be implemented on LoRa++ without any modifications to further improve the scalability of LoRaWAN.

III. LoRa++ Scheme

LoRa++ scheme employs multiple combinations of CSS modulation such as Up chirp, Down chirp, UpDown, and DownUp chirp. These signals are selected using the index bits of each symbol. In the following sub-section, we first present a brief overview of conventional LoRa, followed by a detailed design of modulation and demodulation blocks of the proposed LoRa++.

A. *LoRa modulation and demodulation*

LoRa employs CSS modulation, in which the symbol is spread over the bandwidth using Up chirps. This provides the advantage of spreading gain. The rate at which the symbol is swept from lower to higher frequency is defined by the spreading factor (SF). The number of waveforms available is defined by the order of modulation given by $M = 2^{\mathrm{SF}}$. The symbol period T_s is defined as $T_s = M/BW$. The bit rate is defined as $R_b = log_2 M/T_s$. Each symbol is transmitted using a unique Up chirp pattern, which sweeps from frequencies $[-BW/2, +BW/2]$. The Up chirp $c[l]$ as defined in [13] is given by,

$$c[l] = \frac{1}{\sqrt{2^{\mathrm{SF}}}} e^{j\pi \frac{l^2}{2^{\mathrm{SF}}}} e^{-j\pi l} \tag{1}$$

TABLE I: **Comparison of the literature**

Schemes	**Proposed Technique**	**Enhancements**	**Order of complexity**	**Hardware implementation – PRR**
Conventional LoRa	CSS based modulation	Encodes SF bits per symbol	$O(2^{SF})$	✓ ✓
SSK-LoRa, PSK-LoRa, ICS & EICS-LoRa [7]–[10]	Employs Space shift keying, phase shift keying and interleaving	Increases datarate	Increases by $O(2^p)$; (p are additional bits)	✗ ✗
IQ-CSS [11], DCRK-CSS [12]	Employs in-phase & quadrature CSS and discrete chirp rates	Increases Spectral Efficiency	$O(2^{SF+N_e})$, N_e are the additional bits	✗ ✗
GCSS [13]	Introduces group number (GN), transmits multiplex symbol	Diversified range is achieved	Increases with increase in GN	✗ ✗
Hybrid LoRa [14]	Employs both Up and Down chirps	Improves range	$O(2^{SF})$	✗ ✗
TDM-LoRa [15]	Employs multiplexing of Up and Down chirps simultaneously	Increases spectral efficiency	Proportional to no. of multiplexed symbols	✗ ✗
CurvingLoRa [16]	Employs non-linear chirps	Increases network capacity	$O(2^{SF})$	✓ ✗
FTrack [17]	Employs time-domain information of symbol edges	Demodulates collided symbols	$O(N_s n_{fft} log_{(n_{fft})})$, n_{fft}= FFT size	✓ ✗
Proposed **LoRa++**	Employs combination of Up, Down, UpDown and DownUp chirps	Increases datarate and scalability of network	$O(4 \times 2^{SF})$	✓ ✓

where l is the index, at a given time $l = \{0, 1, \cdots, (2^{\text{SF}}-1)\}$. Further, the LoRa modulation for the k^{th} symbol consisting of SF bits is

$$x_k[l] = c[l]e^{j2\pi\frac{lk}{2^{\text{SF}}}} = \frac{1}{\sqrt{2^{\text{SF}}}}e^{j\pi\frac{l^2+2kl}{2^{\text{SF}}}}e^{-j\pi l} \tag{2}$$

The received complex-valued signal is

$$y[l] = f_h(x_k[l]) + n[l] \tag{3}$$

where $f_h(\cdot)$ is the channel function, and $n[l]$ is the AWGN with zero mean and unit variance. The demodulation process involves dechirping and Fast Fourier Transform (FFT), followed by peak detection. To dechirp, the received signal is multiplied by the SF's Down chirp symbol. The Down chirp $d[l]$ is given by,

$$d[l] = \frac{1}{\sqrt{2^{\text{SF}}}}e^{-j\pi\frac{l^2}{2^{\text{SF}}}}e^{-j\pi l} \tag{4}$$

where $l = \{0, 1, \cdots, 2^{\text{SF}} - 1\}$. The FFT of the de-chirped received signal is computed as

$$r[i] = \sum_{i=0}^{2^{\text{SF}}-1} y[l]d[l]e^{-j2\pi\frac{il}{2^{\text{SF}}}} \tag{5}$$

where $i = \{0, 1, \cdots, 2^{\text{SF}}-1\}$ is frequency index. From Eq. 5, we detect the peak, which is used to estimate the transmitted symbol. Since LoRa supports computationally efficient non-coherent demodulation the need for carrier synchronization is eliminated. However, under harsh conditions, coherent detection techniques are employed for better performance as described in [7], [24]. For example, consider four symbols, *viz,* '0', '1000', '2048', and '3000' (picked from 4096 symbols in SF12). The diagrams shown in Figure 2(a) are for conventional LoRa with SF of 12; under LoRa++ they represent symbols with two '0' bits appended at the Most Significant Bit (MSB).

B. Design of LoRa++ scheme

In the proposed LoRa++ scheme, we have introduced an index modulation-based CSS signaling scheme. Figures 2(b, c, d) depict the employed Down chirp, mixed UpDown chirp, and DownUp chirp for SF12. A detailed description of signal generation is provided as a part of modulation in sub-section III-C. Each symbol consists of two parts, namely, index selection bits (shown in red) and Mixed CSS modulation bits. Figure 3 shows the LoRa++ symbol enhancement to 14 bits. With each chirp's order of index being $M = 2^{\text{SF}}$, the cardinality of the proposed LoRa++ scheme is $4M$. The demodulation scheme described in sub-section III-D is simple and combats phase misalignment and carrier frequency offset.

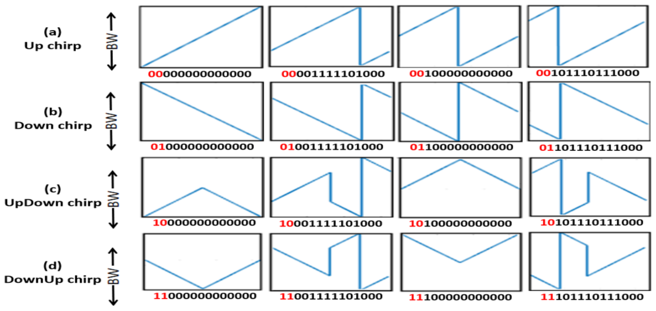


Fig. 2: **A snapshot of LoRa++'s four types of CSS modulation for SF12. The x-axis is symbol period for each chirp. First two bits of each symbol shown in red are the additional bits due to LoRa++. Among all $2^{(SF+2)}$ symbols only 4 symbols do not span the entire bandwidth.**

Frame generation. For compatibility and seamless connectivity with commercial off-the-shelf (COTS) LoRa devices, we have retained the native frame design. Similar to LoRa, LoRa++ frame supports a preamble of 8 basic Up chirps, 2 Up chirps for synchronization, and 2.25 basic Down chirps, which indicate the beginning of data, followed by data streams as payload. Figure 3 shows a spectrogram comparison of the conventional LoRa frame with LoRa++. While the preamble & sync chirps are identical, the payload in LoRa++ comprises of Up, Down, UpDown, and DownUp chirps. Also, the required number of chirps is reduced to 6 in LoRa++ as compared to every 7 chirps in conventional LoRa.

C. Modulation

Figure 4 shows the modulator design for LoRa++. The additional blocks of UpDown and DownUp are added. The

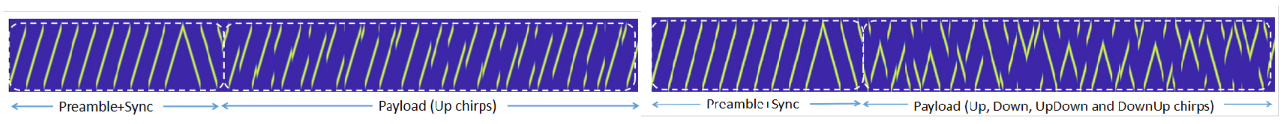


Fig. 3: **Spectrograms for Conventional LoRa and LoRa++ with SF12 spreading factor preamble, sync word and payload. Notice that preamble is kept the same as Conventional LoRa**

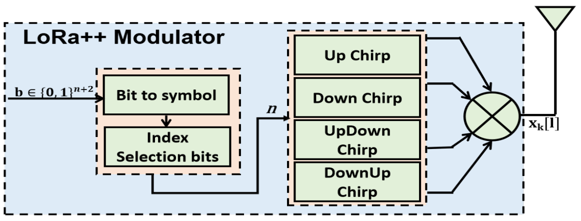


Fig. 4: **Modulator design of LoRa++ scheme. The first 2 bits of each symbol define the type of CSS modulation be employed. The remaining $n = SF$ bits are CSS modulated.**

selector block selects one out of four modulations. As explained earlier, each symbol in LoRa++ is enhanced with 2 additional bits. For example, the LoRa++ symbol with a spreading factor of 7 now represents 9 instead of 7 bits. While the first two MSB bits of the symbol are used for index selection, the remaining bits (denoted by n) are employed to generate CSS signals.

Index Selection. We call the appended 2 MSB bits index bits, and we select the type of CSS modulation scheme to be applied using these bits. Figure 2 provides the mapping between index selection bits (shown in red color) and the type of CSS modulation selected.

UpDown and DownUp modulation. The generation of LoRa's Up Chirp and Down Chirp are shown in Eq. 1 and 4. Similarly, in our scheme, we employ equation Eq. 6 and 7 to generate UpDown chirp and DownUp chirp, respectively. To generate these signals, the symbol duration (T_s) is divided into two equal parts. In UpDown chirp, we employ Up chirp for the first half-time duration ($l < T_S/2$), followed by Down chirp for the remaining time ($l \geq T_s/2$). Thus, the symbol $x_k[l]$ is now fully represented by Up chirps and Down chirps as given by,

$$x_k[l] = \begin{cases} c[l]e^{j2\pi\frac{lk}{2^{\mathrm{SF}}}} = \frac{1}{\sqrt{2^{\mathrm{SF}}}}e^{j\pi\frac{l^2+2kl}{2^{\mathrm{SF}}}}e^{-j\pi l} & l < T_s/2 \\ d[l]e^{j2\pi\frac{lk}{2^{\mathrm{SF}}}} = \frac{1}{\sqrt{2^{\mathrm{SF}}}}e^{-j\pi\frac{l^2+2kl}{2^{\mathrm{SF}}}}e^{-j\pi l} & l \geq T_s/2 \end{cases} \tag{6}$$

Similarly, Eq. 7 shows the DownUp modulated symbol ($x_k[l]$), where we employ Down chirp for the first half-time duration ($l < T_s/2$), followed by Up chirp for the remaining ($l \geq T_s/2$) duration.

$$x_k[l] = \begin{cases} d[l]e^{j2\pi\frac{lk}{2^{\mathrm{SF}}}} = \frac{1}{\sqrt{2^{\mathrm{SF}}}}e^{-j\pi\frac{l^2+2kl}{2^{\mathrm{SF}}}}e^{-j\pi l} & l < T_s/2 \\ c[l]e^{j2\pi\frac{lk}{2^{\mathrm{SF}}}} = \frac{1}{\sqrt{2^{\mathrm{SF}}}}e^{j\pi\frac{l^2+2kl}{2^{\mathrm{SF}}}}e^{-j\pi l} & l \geq T_s/2 \end{cases} \tag{7}$$

Key features. The parameters defining a conventional LoRa symbol are (a) starting frequency of the symbol (f_s at $t = 0$), (b) end frequency (f_e at $t = T_s$) and (c) the time of frequency jump ($t = t_f$) over the bandwidth from f_H ($f_c + BW/2$) to f_L ($f_c - BW/2$). Together, these three parameters will distinguish symbols from each other. For the conventional Up chirp, frequency jumps from f_H ($f_c + BW/2$) to f_L ($f_c - BW/2$), and for Down chirp frequency jumps from f_L to f_H once in symbol time T_s. In LoRa++, for UpDown and DownUp chirp, **the frequency jump occurs twice**; the additional jump occurs at $T_s/2$, where the signal reverses its slope. However, as shown in Figure 2, following the conventional LoRa, all zero CSS modulation bits do not exhibit frequency jump. A similar pattern is observed for the center codeword of UpDown and DownUp signals. Here, the parameters employed for decoding these symbols are limited to starting frequency f_s and ending frequency f_e. The time of frequency jump t_f characterizing the symbol is retained across all four CSS modulations to ensure compatibility with conventional LoRa.

D. *Demodulation*

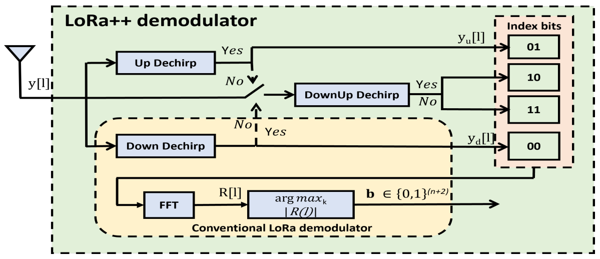


Fig. 5: **Design of the demodulator of the proposed LoRa++ scheme. At the receiver end, first, the index selection bits are estimated, and subsequently, CSS symbol detection using FFT-based energy detection is performed.**

Figure 5 depicts the block diagram of LoRa++ Demodulator. The demodulation of LoRa++ follows the conventional LoRa, which employs both coherent and non-coherent detection techniques. However, the process involves two steps: (i) detecting the index selection bits and (ii) estimating the CSS-modulated symbol. To detect the index selection bits, we first multiply the received signal with Down chirp and Up chirp individually as

$$y_d[l] = y[l]\frac{1}{\sqrt{2^{\mathrm{SF}}}}e^{-j\pi\frac{l^2}{2^{\mathrm{SF}}}}e^{-j\pi l} \tag{8}$$

$$y_u[l] = y[l]\frac{1}{\sqrt{2^{\mathrm{SF}}}}e^{j\pi\frac{l^2}{2^{\mathrm{SF}}}}e^{-j\pi l} \tag{9}$$

Eq. 8 follows the conventional demodulation, where the signal is multiplied with a Down chirp. LoRa++ introduces $y_u[l]$ as shown in Eq. 9, where the signal is multiplied with Up chirp. Multiplying two CSS signals of opposite slopes will result in the dechirping of the signal. We leverage on dechirping of CSS signals where their FFT will result in a pair of dominating peaks. The instant at which symbol jumps occur is represented by the two peaks. On the other hand, multiplying two CSS signals with the same slope will result in energy distributed over the bandwidth [25].

Figure 6(a) shows the spectrogram representation of an Up chirp symbol x =‘1000’, with a spreading factor of 12, and bandwidth of 125 kHz. The sampling frequency of 2×2^{SF} is considered for the signal generation. Figure 6(a) shows the spectrogram representation of an Up chirp symbol x =‘1000’, with a spreading factor of 12 and bandwidth of 125kHz. The sampling frequency of $2 \times 2^{\mathrm{SF}}$ is considered for the signal generation. Figure 6(b) shows the FFT of the dechirping operation where the received Up chirp symbol is multiplied by a down chirp. The x-axis represents the signal frequency, and the y-axis represents its corresponding magnitude. We observe two distributions at index x and $(2^{\mathrm{SF}} - x)$ where a symbol jump (as shown in Figure 6(a)) occurs between the two peaks. For instance, if the signal transmitted is Up chirp, the $FFT(y_d[l])$, that is, the signal multiplied with opposite slope, will resemble conventional demodulation with peaks representing energy concentration. Whereas Figure 6(c) shows the $FFT(y_u[l])$ signal multiplied by the same slope consisting of multiple small peaks, indicating that the energy is distributed over the bandwidth. A similar peak analysis is performed for Down chirp estimation, where The FFT$(y_u[l])$ will resemble conventional LoRa demodulation while FFT$(y_d[l])$ consists of energy distributed over multiple frequencies. By analyzing the magnitude of the peaks obtained from FFT$(y_u[l])$ and FFT$(y_d[l])$, the signal is categorized as either Up or Down chirp signals In the case of the UpDown and DownUp

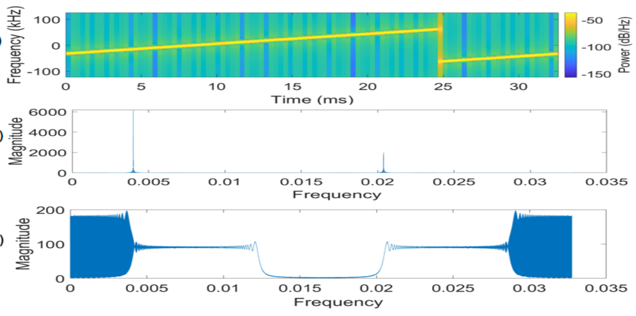


Fig. 6: **The spectrogram representation of an Up chirp, and its respective FFT when multiplied with Down chirp (opposite slope) and Up chirp (same slope).**

signals, we obtain peaks with energy in both $FFT(y_u[l])$ and $FFT(y_d[l])$. These signals are further multiplied with DownUp chirp, and their FFT is obtained. While the FFT of the UpDown signal multiplied by the DownUp chirp results in three peaks representing the symbol, the FFT of the DownUp signal multiplied by the DownUp chirp produces an energy band. Hence, by analyzing the FFT of the dechirped signals, we estimate the index selection bits.

Figure 7 describes the demodulation process. As explained earlier, the first step is to estimate the index selection bits. Subsequently, a dechirped signal is used for estimating the CSS-modulated symbol. For decoding this symbol, the highest magnitude obtained from the FFT is employed. Finally, local maximum likelihood estimation is performed to find the exact symbol transmitted. Algorithm 1 provides the steps involved in the demodulation process.

Computational Complexity of LoRa++. The demodulation of LoRa++ symbols requires the computation of a maximum of four FFT operations. Thus the maximum computational complexity of LoRa++ system is 4× that of conventional LoRa (i.e., $O(4 \times 2^{SF}) = O(2^{SF})$) However, in the case of demodulating Up Chirp & Down chirp together, the algorithm performs 2 FFTs, & thus the complexity of decoding these symbols is reduced to 2× that of conventional LoRa.

Algorithm 1: LoRa++ demodulation algorithm

```
input : Received signal: $y[l]$; Variables: $c[l], d[l]$
output: Estimate of transmitted symbol: $\hat{x}_k[l]$
1 Function (Detecting index selection bits)
      De-spreading:
            $y_d[l] = y[l]d[l]$
            $y_u[l] = y[l]c[l]$
      $FFT(y_d[l])$ & $FFT(y_u[l])$
      if $FFT(y_d[l])$ || $FFT(y_u[l]) \rightarrow$ energy band then
          index selection bits $(i_b) \in \{00; 01\}$
          if $max(FFT(y_d[l])) > max(FFT(y_u[l]))$ then
              $i_b = \{00\}$
          else
              $i_b = \{01\}$
      else
          $i_b \in \{10; 11\}$
          REPEAT De-spreading steps with basic
          DownUp chirp, and take $FFT$ to determine $i_b$
      return index selection bits: $i_b$
2 Function (Detecting CSS modulated Symbol)
      Obtain dechirped signal from above steps
      Determine symbol jump time $t_f$
      Maximum Likelihood to estimate $SF_{bits}$
      $\hat{x}_k[l] = \{i_b\ SF_{bits}\}$
      return $\hat{x}_k[l]$
```

IV. Implementation of LoRa++

This section discusses the design & implementation of LoRa++ transmitter and receiver chain on the Software Defined Radio (SDR) platform. Further, the steps involved in implementing LoRa++ gateway are discussed, which can support multiple LoRa++ and conventional LoRa links.

Implementation Platform. For the experiments, we have used USRP B210 as the transmitter and RTL-SDR as the receiver. The transmit and receive chain communication blocks are processed on a Linux, Ubuntu 20.04 operating system with i7-6600U CPU and 16GB RAM. The transmitter and receiver are equipped with a 1.2 dBi gain omnidirectional antenna. We utilized Semtech’s SX1308P868GW as LoRa Gateway and Raspberry Pi 4 Model B provided the network connectivity. The LoRa end node is from Microchip technology RN2483. The operating frequency is 868 MHz, & the BW is set to 125 kHz. The transmit power is +14 dBm, & the CR is set to 4/5.

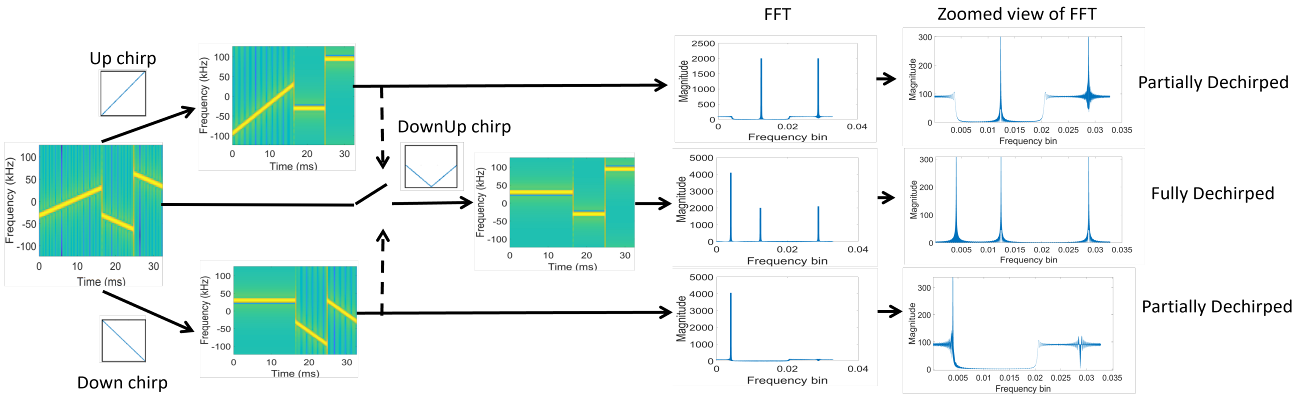


Fig. 7: **LoRa++ demodulation process: First, symbol is multiplied with both Up and Down chirp individually. Second, FFT of obtained signal is evaluated to estimate index '2' bits. Further, absolute FFT and frequency peak detection algorithm is employed to estimate remaining SF bits.**

A. *LoRa++ PHY layer GNU radio Implementation*

Several works in the literature, such as [25], [26], have presented the PHY layer implementation of the conventional LoRa scheme. In this work, we have employed the [26] implementation and enhanced the transmitter and receiver with additional chirp modulations to function as LoRa++.

Transmitter. LoRa++ packet contains a preamble, start frame delimiter (SFD), & data symbols exactly as per conventional LoRa. For LoRa++'s payload, the symbol size is increased by 2 bits compared to conventional LoRa. Once the number of symbols is determined, the data stream is modulated as described in III-C.

Receiver. One important part of receiver implementation is frame synchronization. The signal undergoes carrier frequency offset (CFO) and sampling time offset (STO), which requires correction. We have utilized the fine-grained synchronization process as described in [26].

Further, the SFDs are used to align the demodulation window. After the correction of CFO and STO and the correct alignment of the decoding window are implemented. The first step of the payload demodulation process is to recognize the type of CSS modulation, i.e., to identify the index selection bits. This involves dechirping, followed by FFT as explained in section III-D. Once the type of CSS modulation is known, the peak analysis and symbol estimation are performed as explained in algorithm 1 in section III-D. After the symbol is demodulated, the decoding process of gray demapping, de-interleaving, de-hamming, and de-whitening is performed to estimate the transmitted data.

B. *LoRa++ communication range*

The performance of LoRa++ is evaluated in both indoor and outdoor conditions for both line-of-sight and non-line-of-sight conditions. The PRR performance of LoRa++ over USRP-based hardware system is tested under *in-situ* line of sight and non-line of sight environment. Figure 8 shows the first floor of a building where the colored blocks are the rooms, partitioned by concrete, and therefore, the signals can be attenuated and blocked by the walls and centre open area.

Figure 8 shows node placement for indoor scenarios, with one gateway and four end devices. The nodes are mounted on a tripod at a 5 feet height from the floor. The

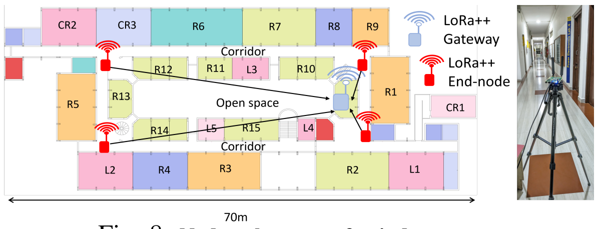


Fig. 8: **Nodes placement for indoor set-up.**

maximum distance between the transmitter and receiver using the corridor path is about 40 m. For the outdoor experiments, the placement of gateway and end devices are on the rooftop at a height of about 30 feet from the ground. The transmitting end devices are distributed around the campus within a 400 m radius with concrete buildings and thick tree canopy cover.

Figure 9 shows a comparison of the PRR for LoRa++ and conventional LoRa. The readings are measured for varied inter-node distances from 50 m to 300 m. The payload is comprised of 32 symbols for a single packet. Since the payload supported by LoRa++ consists of two additional bits per symbol, the total payload carried is higher. It is observed

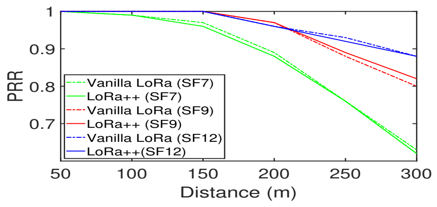


Fig. 9: **Provides the PRR for varying distances for LoRa++ and conventional LoRa with $SF = \{7, 9, 12\}$ for outdoor.**

that the performance of LoRa++ is at par with conventional LoRa for a range of 300 m. For example, in SF12 (color: blue solid and dotted respectively), for PRR of 0.9, we achieve a communication range of about 270 m for both, and for SF7, it is about 170 m. Similarly, for indoor environments, the PRR is measured for distances of $\{5, 10, 15, 20, 40\}$ meter for both line-of-sight and non-line-of-sight conditions. It is observed for line-of-sight conditions, the PRR obtained is 0.98 for all distances for both conventional LoRa and LoRa++. For non-line-of-sight conditions, the PRR of 0.97 with a spreading factor of 12, while the PRR of 0.88 is measured with a spreading factor of 7, was obtained for 70 m.

C. *Backward Compatibility*

It is important to ensure that the new scheme is backward compatible with the conventional LoRa. Therefore, it becomes crucial for the gateway to differentiate between the packet received from LoRa++ and conventional LoRa nodes. It is trivial to note that, in LoRa++, the chirp of $SF+2$ bits with index '$00'$ resembles the conventional LoRa's chirp. The following mitigation methods can be used to differentiate between LoRa++ and conventional LoRa: ❶ *Information from node ID:* It is safe to assume that the end nodes do not switch between LoRa and LoRa++, we may use the ID to quickly select the decoding mode. ❷ *Re-decode based on CRC error:* We first decode the packet as conventional LoRa. If the CRC fails, we re-decode as LoRa++ packet.

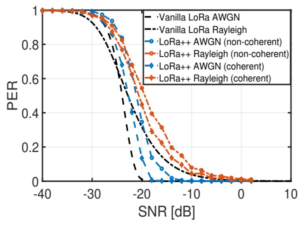
V. Evaluation

In this section, we study the packet error rate performance (PER) of LoRa++ under AWGN and Rayleigh fading channels. We further quantify the BER for inter-SF interference. To demonstrate the scalability of our scheme, we study the packet reception ratio in the presence of a large number of nodes in the LoRaWAN network. We first discuss the design for the generation of I and Q components of LoRa++. A detailed description of the metrics and parameters used for simulation is provided below.
❶***Bit Error Rate (BER)****:* Number of bits in error in a unit of time. It provides the number of bits distorted due to channel interference, synchronization, and noise. ❷***Packet reception ratio (PRR):*** We define this as the total number of successful packet receptions at the gateway to the total number of packets transmitted by end devices. In the current analysis, we do not consider retransmissions. ❸***Channel Load:*** The percentage of total channel capacity at a given time. ❹***Scalability:*** The capacity of the channel to accommodate an additional number of nodes.
Generation of LoRa++ I and Q components. For our baseline study with conventional LoRa, we utilize the MATLAB code from [27]. The code generates the BER performance results for AWGN and Rayleigh channels. To generate I and Q components for the proposed scheme, we used Equations 2, 4, 6, and 7 and accordingly modified the code. The SF varies from 7 to 12. It also incorporates our LoRa++ modulator and demodulator designs (both coherent and non-coherent) as discussed in Section III.

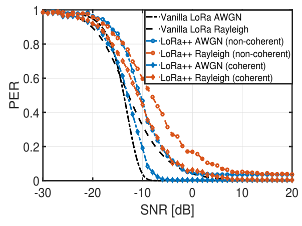
A. *Error rate*

Our packet structure retains the conventional LoRa format, with a hamming code rate of $4/5$, followed by the whitening, interleaving, and grey mapping. Further, preamble symbols are added to ensure synchronization. We perform Monte-Carlo simulation in MATLAB for 10^6 symbols and repeated the experiments 10 times.

Figure 10(a) and (b) show the results of PER of the received signal when transmitted over AWGN (blue color) and Rayleigh (brown color) fading channels. The results include both coherent and non-coherent demodulation for SF11 and SF7. The diamond mark represents a coherent



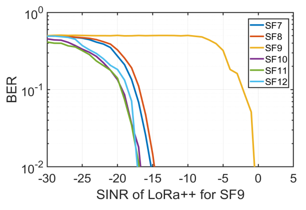
(a) For SF11



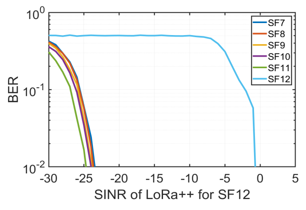
(b) For SF7

Fig. 10: **Comparison of PER of the proposed LoRa++ scheme with conventional LoRa for spreading factor of SF11 and SF7**

demodulator, while the circle mark represents a non-coherent demodulator. The black line represents the baseline analysis of conventional LoRa. To achieve a zero PER, for SF11 Figure 10(a)shows that LoRa++ coherent detection for an AWGN channel requires a 1–2 dB marginal increase in SNR over conventional LoRa. For a Rayleigh fading channel, the performance of conventional LoRa and LoRa++ are almost (about 0.5 dB difference) identical. For SF7 shown in Figure 10(b), LoRa++ performance for AWGN and Rayleigh fading channel has similar SNR requirements. Our evaluation results for other SFs not discussed in this paper confirm a similar behaviour.



(a)

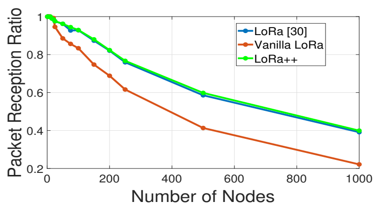


(b)

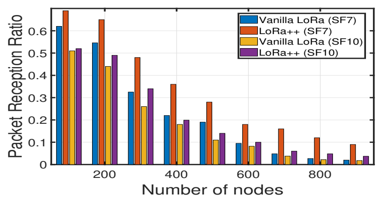
Fig. 11: **BER performance of LoRa++ for inter-SF interference caused due to other spreading factors (a) For SF9 (b) For SF12**

Simultaneous reception. While interference from other technologies with LoRa is a well-studied problem, the works in literature [28]–[30], propose a methodology to study interference at the gateway caused by simultaneous reception from multiple LoRa devices. In this standard methodology, a signal of reference SF is injected with interference from the same SF as well as other SFs. The goal here is to establish the minimum threshold Signal-to-Interference-Noise-Ratio (SINR) required for the successful decoding of the reference signal. Our LoRa++ adheres to this methodology. Figure 11 plots the SINR at the LoRa gateway when multiple signals are received simultaneously. LoRa gateway is assumed to use the SX1272 chipset, and we evaluate the SINR for LoRa++ when a pair of signals is considered. While one signal is regarded as a reference SF, we study the BER, with another interfering SF. The interfering signal is randomly generated and the time-shifted symbols overlap with reference SF as provided in [28]. For our evaluation, we have simulated for 10^6 iterations, with payload of 20 bytes and a SINR threshold of 1% BER is selected.

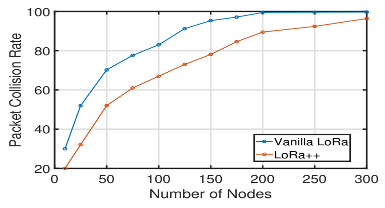
Figure 11 provides the BER for varying SINR values, for SF9 and SF12 for LoRa++. This represents the minimum SINR threshold values for different interfering SFs required to decode the reference signal. In Figure 11(a), the threshold



(a)



(b)



(c)

Fig. 12: **(a) Variation of PRR with an increasing number of nodes to a gateway. (b) PRR for SF7 and SF10 for the varying number of nodes (N) with channel load fixed to 100% for $N = 100$. (c) Packet loss with varying N for SF7 with channel load fixed to 100% for $N = 100$.**

for decoding SF9 across SFs is within a SINR band of -17 dBm. Similarly for decoding SF12 the SINR requirements is within -24 dBm band as shown in Figure 11(b). As expected the higher SFs will provide lower SINR thresholds.

B. Scalability analysis of LoRa++

In this section, we examine the performance of LoRa++ for a LoRaWAN network comprising of a large number of nodes. We implemented our scheme by modifying the open-source code of ns-3 [29], [31] and simulated the network as specified in the standard. Our simulated LoRaWAN environment on ns-3 platform uses the base code of the work from [29], [32] with 50 to 1000 devices, where each device transmits within the 1% duty cycle limits. All the nodes transmit 23 bytes of payload, with 8 bytes of preamble and 1 CRC byte. The transmit power is set to $+14$ dBm, the center frequency is selected between $\{868.1, 868.3, 868.5\}$ MHz, the coding rate and bandwidth are set to 1 (i.e. $4/5$) and 125 kHz respectively. The placements of nodes are assigned by a uniform random variable. The simulation environment is considered for the radius of 6.4 km. Channel parameters is set to log-distance propagation loss model with a path-loss distance exponent of 3.76, and a constant speed propagation delay model. The simulation time considered is 24 hours.

PRR and Scalability (maximum number of nodes N). Since LoRa++ supports SF+2 bits per symbol and if the total number of bits to be transmitted is fixed, we expect the network to support additional LoRa end-devices. A theoretical maximum value obtained from calculations under the zero collision assumption is available in the section VI. In this section, to estimate N under experimental conditions we perform ns-3 simulations for both conventional and LoRa++ schemes.

Figure 12(a) shows the PRR comparison between Vanilla LoRa, LoRa design from [30] and LoRa++. Though the ToA of LoRa++ is lowered as compared to conventional LoRa, the PRR performance of LoRa++ is at par with at LoRa [30]. While realistic and usable PRRs of 0.99 to 1 are obtained for 50 nodes in both schemes, it is interesting to note that a near 10% improvement in LoRa++ is observed when the devices in LoRaWAN are above 500. Clearly, the two additional bits carried in each SF pulls up the PRR.

Figure 12(b) provides the PRR for Vanilla LoRa and LoRa++ for spreading factors of 7 and 10. The channel load for $n = 100$ is set to be 100%. The PRR is evaluated for 100 to 800 nodes with a step-size of 100. It is observed that PRR of LoRa++ outperforms conventional LoRa by $\sim 10\%$.

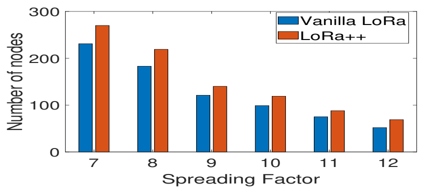


Fig. 13: **Number of devices supported by different SF for Vanilla LoRa & LoRa++ for PRR of 0.7 with channel load of 75%.**

Figure 12(c) shows the packet loss in terms of collision with varying number of nodes for channel load of 100% at $n = 100$, and SF7. The packet loss for LoRa++ is $\sim 15\%$ less than conventional LoRa, due to its reduced ToA.

The benefits of LoRa++ extend across several SFs as well. Figure 13 shows the number of end devices supported by a single gateway for varying spreading factors with a fixed PRR of 0.7 and with channel load of 75%.

This value of PRR is selected, as shown in [33] the data can be successfully recovered even with a low PRR of 0.7. The red color indicates that LoRa++ supports a maximum of 25% more end-devices and a minimum of 18% more end-devices compared to conventional LoRa. This is because the ToA increases with SFs, and thus the number of devices also progressively reduces.

VI. Discussions

Advantages of LoRa++. The LoRa++ modulation technique is simple and easy to implement. For instance, the selector block's function (Figure 4) has to choose one out of four signal, instead of one out of two. Thus, the adaptation of this modulation in commercial chipsets is expected to be easy with minimal code addition. Some of the advantages are: (i) The LoRa++ scheme is designed to preserve the strengths of conventional LoRa chirps. All the three f_s, f_e and t_f parameters of a symbol are retained. This assists in the correct detection of SF bits in the symbol without additional decoding complexity. (ii) The packet design of LoRa++ is kept the same as LoRa; aided with the CRC bits to increase the robustness of the transmitted signal. (iii) LoRa++ gateways can be made available by software upgrades to existing commercial LoRa gateways.

Demodulator SNR for LoRa++. In the design of LoRa++, the user can opt for a maximum of $SF + 2$ bits per symbol. This scheme is simple, and trivial in implementation. The hardware has the flexibility to support conventional LoRa, LoRa++ with $SF + 1$ bits per symbol and LoRa++ with $SF + 2$ bits per symbol. In LoRa, the standard defines discrete SFs and their associated SNRs for successful decoding (say 1% PER) for a given receiver sensitivity. The user sets an

SF to support the application's data requirement. LoRa++ introduces a smaller step size because of the possibility of supporting conventional LoRa, LoRa++ with $SF+1$ bits, and $SF+2$ bits per symbol.

TABLE II: **SNR requirements in dB for different SFs for conventional LoRa and LoRa++ for $SF+2$ bits/symbol.**

	SF7	SF8	SF9	SF10
Conventional LoRa	-9	-11	-14	-17
LoRa++	-7	-10	-12	-15

Table 2 proves that for the available SNR can be efficiently utilized to support higher data rate. For example, if the available SNR is -10 dB, in conventional LoRa the maximum DR achieved is 3.125 kbps with SF8. Whereas with LoRa++ one can achieve DR of 3.515 kbps with $SF+1$ bits and 3.906 kbps with $SF+2$ bits using LoRa++'s SF8. A similar observation is seen when the available SNR is -12 & -15 dB.

Maximum number of nodes for an SF. We calculated the maximum number of nodes accommodated by LoRa++ network with non-overlapping communication, for a particular SF at 1% duty cycle. We have considered the BW of 125 kHz, for a total payload of 8.5 kB over 24 hours, with each packet carrying 10 b of payload. The maximum number of nodes accommodated by a single gateway for SFs $\{7, 8, 9, 10, 11, 12\}$ are $\{3167, 1764, 862, 421, 207, 119\}$ respectively. It is observed that with LoRa++ scheme the gateway can support 25% (maximum) to 18% (minimum) increased number of nodes, with at par PRR performance.

VII. Conclusion

In this work, we have proposed and implemented LoRa++, a LoRa technology mechanism that increases the total amount of data to be transmitted for a specified ToA. Our 2-bit enhancement introduces two new chirps, viz. UpDown and DownUp chirps augment existing chirps, and thus each SF supports 4× more symbols than the conventional LoRa. Furthermore, this increase in data rate is achieved with a marginal increase in SNR requirement. The PRR of LoRa++ is at par with the conventional LoRa. As a direct impact of savings in ToA, our LoRa++ supports a maximum of 25% (18% minimum) more number of devices in the LoRaWAN.

Acknowledgment

We express our gratitude to SMDP-MeitY : SP/MITO-20-0009 for their encouragement and support. Further, we acknowledge the support of the project no. 10115715, ENACT by EU HADEA.

References

[1] C. Sarkar *et al.*, "VSF: An energy-efficient sensing framework using virtual sensors," *IEEE Sensors Journal*, vol. 16, no. 12, 2016.

[2] C. Daniele *et al.*, "Lora technology demystified: From link behavior to cell-level performance," *IEEE Transactions on Wireless Communications*, vol. 19, 2020.

[3] O. Georgiou and U. Raza, "Low power wide area network analysis: Can lora scale?" *IEEE Wireless Communications Letters*, vol. 6, 2017.

[4] M. C. Bor *et al.*, "Do lora low-power wide-area networks scale?" in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2016.

[5] "Things network," 2023, https://www.thethingsnetwork.org/docs/lorawan/regional-parameters/ [Accessed: October 26, 2023].

[6] D. Bankov *et al.*, "Mathematical model of lorawan channel access with capture effect," in *28th annual international symposium on personal, indoor, and mobile radio communications*, 2017.

[7] M. Hanif and H. H. Nguyen, "Slope-shift keying lora-based modulation," *IEEE internet of things journal*, vol. 8, no. 1, 2020.

[8] R. Bomfin, M. Chafii, and G. Fettweis, "A novel modulation for iot: Psk-lora," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, 2019, pp. 1–5.

[9] T. Elshabrawy and J. Robert, "Interleaved chirp spreading lora-based modulation," *IEEE Internet of Things Journal*, vol. 6, no. 2, 2019.

[10] Y. Shi, W. Xu, and L. Wang, "An enhanced interleaved chirp spreading lora modulation scheme for high data transmission," in *2022 Wireless Telecommunications Symposium (WTS)*. IEEE, 2022, pp. 1–6.

[11] I. B. F. de Almeida, M. Chafii, A. Nimr, and G. Fettweis, "In-phase and quadrature chirp spread spectrum for iot communications," in *GLOBECOM*, 2020, pp. 1–6.

[12] I. Bizon Franco de Almeida, M. Chafii, A. Nimr, and G. Fettweis, "Alternative chirp spread spectrum techniques for lpwans," *IEEE Transactions on Green Communications and Networking*, vol. 5, 2021.

[13] Q. Yu, H. Wang, Z. Lu, and S. An, "Group-based css modulation: A novel enhancement to lora physical layer," *IEEE Wireless Communications Letters*, vol. 11, no. 3, pp. 660–664, 2022.

[14] M. Noor-A-Rahim *et al.*, "Hybrid chirp signal design for improved long-range (lora) communications," *Signals*, vol. 3, no. 1, 2022.

[15] S. An, H. Wang, Y. Sun, Z. Lu, and Q. Yu, "Time domain multiplexed lora modulation waveform design for iot communication," *IEEE Communications Letters*, vol. 26, no. 4, pp. 838–842, 2022.

[16] C. Li, X. Guo, L. Shangguan, Z. Cao, and K. Jamieson, "Curvinglora to boost lora network capacity via concurrent transmission," *arXiv preprint arXiv:2201.05179*, 2022.

[17] X. Xia *et al.*, "Ftrack: Parallel decoding for lora transmissions," *IEEE/ACM Transactions on Networking*, vol. 28, no. 6, 2020.

[18] P. Edward *et al.*, "On the coexistence of lora- and interleaved chirp spreading lora-based modulations," in *International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2019.

[19] P. Edward, M. El-Aasser, M. Ashour, and T. Elshabrawy, "Interleaved chirp spreading lora as a parallel network to enhance lora capacity," *IEEE Internet of Things Journal*, vol. 8, no. 5, 2021.

[20] G. Lee *et al.*, "Bic-lora: Bits in chirp shapes to boost throughput in lora," in *2024 23rd ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, 2024.

[21] X. Xia *et al.*, "Xcopy: Boosting weak links for reliable lora communication," in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, 2023, pp. 1–15.

[22] M. Hessar, A. Najafi, and S. Gollakota, "NetScatter: Enabling Large-Scale backscatter networks," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, 2019.

[23] P. Xie, Y. Li, Z. Xu, Q. Chen, Y. Liu, and J. Wang, "Push the limit of lpwans with concurrent transmissions," in *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, 2023, pp. 1–10.

[24] O. Afisiadis, S. Li, J. Tapparel, A. Burg, and A. Balatsoukas-Stimming, "On the advantage of coherent lora detection in the presence of interference," *IEEE Internet of Things Journal*, vol. 8, no. 14, 2021.

[25] Z. Xu, S. Tong, P. Xie, and J. Wang, "From demodulation to decoding: Toward complete lora phy understanding and implementation," *ACM Transactions on Sensor Networks*, vol. 18, no. 4, pp. 1–27, 2023.

[26] J. Tapparel *et al.*, "An open-source lora physical layer prototype on gnu radio," in *21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2020.

[27] B. Al Homssi *et al.*, "IoT network design using open-source lora coverage emulator," *IEEE access*, vol. 9, pp. 53 636–53 646, 2021.

[28] D. Croce *et al.*, "Impact of lora imperfect orthogonality: Analysis of link-level performance," *IEEE Communications Letters*, vol. 22, no. 4, 2018.

[29] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of lora networks in a smart city scenario," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.

[30] C. Goursaud and J.-M. Gorce, "Dedicated networks for iot: Phy/mac state of the art and challenges," *EAI endorsed transactions on Internet of Things*, 2015.

[31] V. den Abeele Floris *et al.*, "Scalability analysis of large-scale lorawan networks in ns-3," *IEEE Internet of Things Journal*, vol. 4, no. 6, 2017.

[32] N. Kouvelas *et al.*, "Employing p-CSMA on a lora network simulator," 2018. [Online]. Available: https://arxiv.org/abs/1805.12263

[33] P. Marcelis J *et al.*, "DaRe: Data recovery through application layer coding for lorawan," in *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, 2017.