Scaling Up Parallel Decoding of LoRa Channels

Charles Van Hook*
ECE Department
Boston University
Boston, USA
cvanhook@bu.edu

Tasha Adler*
ECE Department
Boston University
Boston, USA
tyadler@bu.edu

David Starobinski
ECE Department and Systems Engineering Division
Boston University
Boston, MA
staro@bu.edu

***Abstract*—LoRa is a widely used IoT protocol optimized for long-range, low-power applications. LoRa networks can operate across up to 64 channels and six spreading factors. Since conventional hardware receivers are capable of monitoring only one channel and one spreading factor at a time, mapping networks and identifying devices in a varied LoRa ecosystem is difficult. To address this problem, we present `LoRa-MUX`, a real-time software-defined radio (SDR) implementation of a LoRa receiver capable of simultaneously processing multiple channels and spreading factors. Our design uses polyphase channelization to isolate each channel, squelch filtering to eliminate noise from inactive channels, and parameter-parallel demodulation to identify and decode packets. We benchmark `LoRa-MUX`'s performance in a real-world implementation, across varying distances and transmitter gains, using both RTL-SDR and USRP receivers. The USRP receiver maintains a Packet Reception Rate (PRR) above 90% while decoding up to 30 channels in parallel. Even when scaled to 50 channels, the PRR stays above 60%, 30 times better than the PRR of a single-channel receiver under the same conditions.**

***Index Terms*—LoRa, Software-Defined Radio (SDR), IoT Monitoring, LPWAN.**

I. Introduction

The physical-layer technology LoRa (Long Range) has emerged as a major Low-Power Wide-Area Network (LPWAN) solution for Internet-of-Things (IoT) applications—supporting battery-constrained devices with infrequent transmissions over long ranges. According to industry statements and academic reviews, LoRa and the associated network architecture have achieved large scale deployment, addressing use cases such as asset tracking, metering (gas/water), and industrial sensing for predictive maintenance [1].

LoRa's physical layer offers multiple configurable parameters: regionally defined frequency-plans (e.g., hundreds of narrow sub-channels in the ISM band), selectable bandwidths (125 kHz, 250 kHz, etc.), and spreading factors (SF5–SF12 or more) that are tuned for data-rate or sensitivity [8]. These parameter combinations introduce a large monitoring space when one considers all combinations of channel, bandwidth, and spreading factor (SF) settings. This variability means that passive monitoring and intrusion-detection of LoRa deployments are difficult.

Monitoring IoT networks that use channelized and parameterized wireless protocols thus poses key challenges. One simple solution is hardware duplication. One could deploy a hardware receiver for each parameter set, but this scales poorly in cost, power, and complexity. Alternately, a single receiver could hop between parameter sets, but in sparse, asynchronous transmissions (typical in LPWANs), such hopping will likely miss key packets.

In parallel, the software-defined radio (SDR) literature offers a rich foundation for high-density multi-channel reception via polyphase/FFT filter-bank channelizers. Channelization techniques for software-defined radios have been studied for many years [5], and recent implementations achieve high throughput by exploiting GPUs/CPUs [7].

In this work, we leverage these advances and apply them to LoRa monitoring: we propose a method that uses heterodyning to pack many LoRa channels into a wideband SDR capture, applies a polyphase channelizer to split the capture into many parallel sub-channels, and then runs parameter-parallel demodulators to decode multiple spreading factors and bandwidth settings simultaneously. Using LoRa as the case study allows us to explore the large parameter space (i.e., 64 possible uplink channels × two channel widths × seven spreading factors yield over 256 combinations), and to push the practical limits of how many channels can be monitored concurrently.

Our approach is novel and non-trivial because: (1) while polyphase channelizers are known in SDR, applying them in conjunction with heterodyne packing and parameter-parallel demodulators for a real LPWAN protocol is not straightforward; (2) multiplexing many narrow LoRa channels into one wideband capture introduces issues such as cross-channel leakage, dynamic-range compression, and demodulator sensitivity degradation, which must be addressed via careful filter design, gain calibration, and scheduling; (3) we target commodity SDR hardware rather than bespoke multi-channel front-ends, thereby exploring cost/compute trade-offs and scalability in realistic deployments.

Therefore, to validate the feasibility and performance of our method, we implement a multiplexed LoRa receiver using open-source SDR LoRa blocks, open-source GNURadio blocks, and commodity hardware, which we refer to as `LoRa-MUX`. We benchmark the packet-capture rate under sparse transmission conditions, compare against sequential scanning, and profile the computational and sensitivity trade-offs across different SDR platforms.

This work makes the following contributions:

*These authors contributed equally to this work.

1) We identify the opportunity to increase observability of LoRa networks by exploiting SDR digital signal processing, and quantify the scaling and cost/compute trade-offs for parallel monitoring.
2) We propose and describe a method that combines heterodyning, polyphase channelization, and parameter-parallel demodulation to monitor many channels and parameter settings concurrently.
3) We implement our method in a LoRa case study, using open-source LoRa SDR blocks and commodity hardware, and we demonstrate that tens of channels with full spreading-factor coverage can be monitored simultaneously.
4) We empirically benchmark the system under real-world conditions, compare the system to sequential scanning baselines, and report on sensitivity, packet-capture rate, hardware cost, and processing load.
5) We discuss broader applications of our method—including passive scanning, gateway load balancing, and peer-to-peer spectrum monitoring.

The implementation of `LoRa-MUX` is made publicly available on GitHub [b14].

II. Background

LoRa is a low-bit-rate, low-power physical-layer protocol optimized for long-range communication between a hub and peripheral devices. LoRa and its network-layer, LoRaWAN, are widely utilized around the world, with more than 410 million devices deployed in agriculture, transportation, resource management, infrastructure, and manufacturing [1].

LoRa employs a modulation scheme called Chirp Spread Spectrum (CSS), derived from radar technology. Each transmitted symbol, referred to as a "chirp," is a sinusoidal signal that linearly increases (up-chirp) or decreases (down-chirp) in frequency over the symbol period. Mathematically, a chirp can be expressed as

$$s(t, f_s) = e^{j2\pi(f_0+\frac{k}{2}t)t} e^{j2\pi f_s t}, \tag{1}$$

where f_0 is the starting frequency, k is the chirp rate (rate of increasing frequency), and f_s represents the symbol frequency. The term $e^{j2\pi(f_0+\frac{k}{2}t)t}$ defines the base chirp used for modulation. At the receiver, demodulation is achieved by multiplying the received signal by the complex conjugate of the base chirp and performing a Fast Fourier Transform (FFT) that converts the chirp into a single tone in frequency space.

The **spreading factor (SF)** is a key parameter in CSS that defines the duration of each chirp and directly impacts the data rate and sensitivity. A higher SF results in longer chirp durations, reducing the effective data rate but increasing the sensitivity and resilience to noise. In contrast, a lower SF enables higher data rates at the cost of reduced sensitivity. Practical deployments typically use SF values between 7 and 12, depending on the desired trade-offs between throughput and robustness. LoRa also supports multiple channel bandwidths (commonly 125 kHz, 250 kHz, and 500 kHz), which interact with SF to determine the achievable data rate.

By combining channel, SF, and bandwidth settings, each transmission can be uniquely identified as a **parameter set**. Understanding these parameter sets is critical for receiver design: capturing multiple parameter sets in parallel enables more complete monitoring of LoRa networks and better utilization of available spectrum.

III. Related Work

Prior work on optimizing LoRa SDR decoders falls into two main categories: real-time and post-processing decoders. Real-time approaches primarily refine existing receiver designs without supporting multi-channel or multi-parameter decoding, while post-processing methods enable collision resolution and multi-parameter demodulation.

Tapparel et al. introduced the original SDR LoRa receiver, `gr_lora` [2], and later optimized its processing pipeline [3]. Their work replicates the functionality of dedicated LoRa chips, but the flexibility of SDR architectures remains underutilized for parallel processing across multiple channels and parameters.

Yu et al. proposed XGate, a post-processing decoder capable of demodulating multi-parameter LoRa traffic and resolving collisions [10]. XGate detects packets by correlating preamble chirps across channels and identifies overlapping frames via frequency mismatches caused by time misalignment. While effective for offline analysis, XGate is limited to pre-recorded signals. In contrast, our system performs real-time decoding across any parameter or channel set without receiver modification.

Xia et al. developed a real-time collision-resistant LoRa demodulator that also exploits time misalignment to distinguish overlapping frames [13]. Our design instead focuses on real-time multi-channel processing. Although it does not include collision resistance, LoRa's inherent sparsity minimizes collisions, and integrating our parallel decoding architecture with Xia et al.'s techniques could yield a fully collision-resistant real-time receiver.

Claverie and Lopes Esteves presented a real-time multi-channel decoder using heterodyning to align signals with the optimal center frequency for each LoRa receiver block, still requiring one block per channel [11]. TrendMicro advanced this concept by combining multiple channel streams into a single receiver block [12]. Building on these ideas, we introduce a real-time decoder capable of processing multiple channels and parameter sets in parallel using a distinct GNURadio architecture with optimized design parameters and comprehensive system evaluation.

IV. System Architecture

Figure 1 illustrates the flowgraph of `LoRa-MUX` implemented in GNURadio. The design is intended to efficiently process multiple channels while minimizing computational load and maximizing packet detection. The system comprises three stages, each selected based on performance and practical trade-offs.

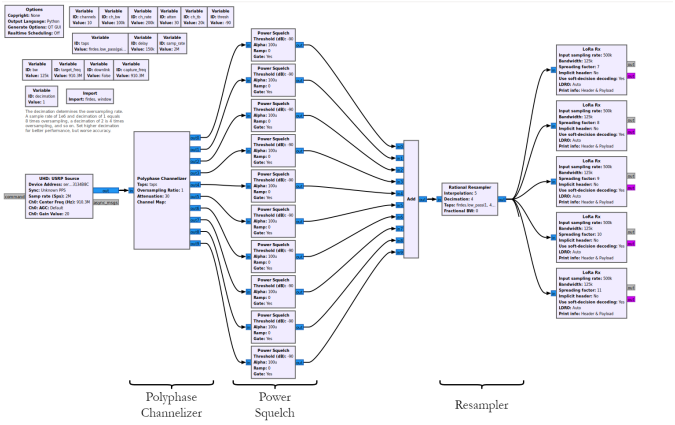

Fig. 1. Flowgraph of the GNURadio multiplexer (`Lora-MUX`) with ten active channels. Left to right - Stage 1: Polyphase Channelizer, Stage 2: Power Squelch, Stage 3: Resampler.

A. Polyphase Channelizer

The first stage is a **Polyphase Channelizer**, which splits the wideband input stream into M parallel channels of equal bandwidth. Processing multiple channels in parallel reduces the effective sample rate per channel, lowering computational requirements. Each channel is subsequently decimated to 200 kS/s.

The channelizer employs a bank of FIR filters with coefficients $h[\cdot]$, designed to minimize aliasing while maintaining sufficient transition width to prevent inter-channel interference. In GNU Radio, such filters are typically generated using the `firdes.low_pass()` command with a chosen window, such as Hamming or Blackman–Harris. The filter parameters were selected based on the MATLAB simulations described in Section IV-E, which identified configurations that balance passband flatness, stopband attenuation, and computational efficiency. In the actual implementation, a slightly shorter Hamming FIR filter was used to match LoRa's 200 kHz per-channel bandwidth while reducing computational cost.

Mathematically, the k-th channel output is

$$y_k[n] = \sum_{m=0}^{L-1} h[mM + k] \cdot x[n - m] \cdot e^{-j\pi kn/M}, \qquad (2)$$

where $x[n]$ is the input signal, $y_k[n]$ is the channel output, M is the number of channels, and $L = N/M$ is the length of the prototype filter. The filter $h[n]$ was optimized using Hamming window parameters informed by extensive simulations (see Table I), balancing frequency selectivity and computational efficiency.

B. Power Squelch

The **Power Squelch** module follows the channelizer. It computes the envelope of each channel and suppresses inactive streams, reducing unnecessary processing in subsequent stages. The squelch threshold was empirically determined to be $P_{\text{thresh}} = -90$ dBm, which is low enough to detect weak packets but high enough to prevent noise-dominated channels from consuming resources. The squelched output is:

$$\tilde{y}_k[n] = \begin{cases} y_k[n], & \text{if } |y_k[n]|^2 > P_{\text{thresh}} \\ 0, & \text{otherwise.} \end{cases} \qquad (3)$$

C. Rational Resampler

After squelching, the **Rational Resampler** adjusts each channel stream to the appropriate sample rate required by the LoRa receiver modules. Since the channelizer reduces the sample rate proportionally to the number of channels, this step ensures compatibility with the demodulation algorithms, which require fixed sampling rates for reliable decoding of all spreading factors.

D. LoRa Receiver Modules

Finally, **LoRa Receiver Modules** decode the packets. One receiver is instantiated for each spreading factor (SF7–SF12). This efficient design allows simultaneous processing of all spreading factors using only six receivers, balancing real-time decoding performance with hardware and software complexity.

E. Filter Optimization and Selection

A central design decision is to select the proper polyphase filter prototype. To identify the most suitable filter for LoRa channel isolation, we conducted extensive **MATLAB simulations using synthetic LoRa signals** across the frequency spectrum. The simulations generated representative LoRa transmissions for different spreading factors and channels, allowing us to evaluate how each filter configuration would perform under realistic signal conditions.

We tested Hamming-windowed FIR filters with varying orders N and cutoff frequencies f_c, alongside standard windows such as Rectangular, Blackman, and Kaiser. The performance of each filter was assessed using two key metrics:

- **Passband Ripple [dB]**: Measures amplitude variation within the passband, where lower values indicate better signal fidelity.
- **Stopband Attenuation [dB]**: Measures suppression of out-of-band energy, critical for avoiding interference between adjacent channels.

TABLE I
FIR FILTER DESIGN GOALS FOR DIFFERENT ORDERS AND CUTOFF FREQUENCIES. THE BOLDED ROW REPRESENTS OUR FILTER TARGET.

Order (N)	Cutoff f_c [Hz]	Passband Ripple [dB]	Stopband Attenuation [dB]
256	100,000	5.98	159.09
256	125,000	5.75	166.73
256	150,000	5.81	159.60
512	100,000	5.92	137.31
512	125,000	5.47	184.76
512	150,000	5.58	153.14
1024	100,000	5.81	142.10
1024	**125,000**	**4.94**	**202.82**
1024	150,000	5.15	150.64
2048	100,000	5.58	158.76
2048	125,000	3.99	220.88
2048	150,000	4.35	157.00

Table I summarizes the results obtained from the MATLAB simulations, highlighting configurations that strike a balance between passband flatness, strong stopband attenuation, and computational efficiency. The simulations indicated that a Hamming FIR filter with $N = 1024$ and $f_c = 125$ kHz provides the most balanced performance: sufficient passband flatness (ripple of 4.94 dB) for signal fidelity and strong stopband attenuation (202.82 dB) for interference suppression. Lower-order filters failed to provide adequate stopband attenuation, while higher-order filters offered only marginal improvements at a significant computational cost. Comparisons with Rectangular, Blackman, and Kaiser windows confirmed that the selected Hamming parameters offer the best trade-off for real-time LoRa channelization.

In practice, however, the implemented filter differs slightly from this MATLAB target. To match the 200 kHz per-channel bandwidth of LoRa and reduce computational requirements in GNU Radio, we used a slightly shorter Hamming FIR filter with $N \approx 800$ taps, corresponding to polyphase subfilters of length $L \approx 40$. This modification preserved signal integrity while reducing computational cost, demonstrating that the MATLAB-derived design targets can be adapted effectively to real-time SDR implementations.

V. Experimental Design and Results

We evaluated the performance of `LoRa-MUX` against a single-threaded flowgraph, measuring **packet reception rate (PRR)** as the primary metric of receiver efficacy. Parallel processing provides multi-channel capacity, but also introduces aggregate noise from multiple channels, potentially reducing receiver sensitivity. Our experiments aim to quantify the trade-offs between parallel processing and PRR under varying receiver distances, noise levels, and packet overlaps.

Metrics Summary.

- **Packet Reception Rate (PRR):** Primary metric for receiver efficacy.
- **Channel capacity:** Maximum number of 125 kHz channels processed in parallel with PRR $\geq 50\%$.
- **Sensitivity:** Minimum USRP transmitter gain required for accurate packet reception.

A. Experimental Setup

The single-threaded flowgraph employs Tapparel's `gr-lora_sdr` receiver to sequentially monitor a predefined list of LoRa frequencies. In contrast, `LoRa-MUX` processes multiple channels simultaneously using the architecture described in Section II.

We used **Arduino M0 Feather boards with RFM95 LoRa chips** as transmitters. Transmitters were programmed to hop across channels randomly within a predefined set. Receivers included low-cost RTL-SDR and a USRP B200 mini, representing low- and high-end SDR platforms. All six LoRa spreading factors (SF7–SF12) were tested to ensure generality of results.

B. Receiver Range

We assessed how receiver type and distance affect PRR. LoRa transceivers typically provide indoor ranges of tens of meters; our goal was to evaluate SDR-based receivers in this context. The receiver was placed on the fourth floor of a multi-story building, while the transmitter was moved from the fourth floor down to the first floor, as shown in Fig. 2.

Fig. 2. The receiver was deployed on the fourth floor, denoted by the 'X' on the diagram, and the transmitter, denoted by an 'O' on the diagram, was moved from the fourth floor to the first floor.

Results shown in Fig. 3 indicate that the USRP maintained a PRR above 80% even when the transmitter was two floors below, while the RTL-SDR reached 80% PRR only on the same floor. This confirms that SDR implementations can approach the performance of more expensive hardware, but range is still limited by receiver sensitivity.

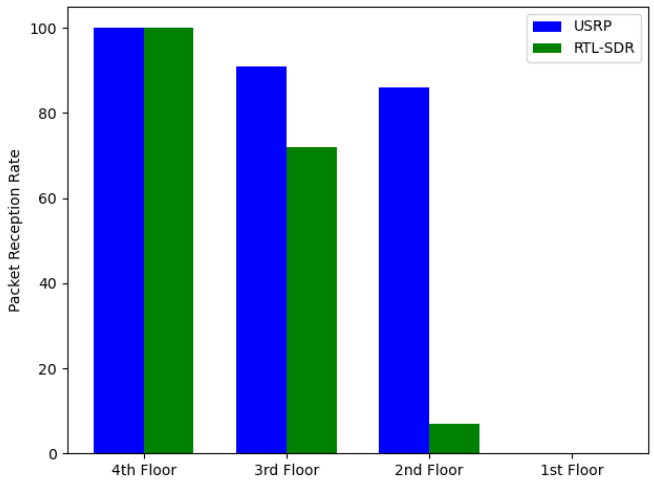

Fig. 3. Packet reception rate for two different SDR receivers (USRP in blue, RTL-SDR in green) as a function of floor separation between transmitter and receiver. The USRP maintains higher PRR at greater distances.

C. Channel Capacity

To determine the maximum number of channels that can be processed in parallel while maintaining high PRR, we measured the multiplexer's performance as the number of active 125 kHz channels increased. LoRa in the U.S. has 64 uplink channels spanning 902.3 MHz to 914.9 MHz. Our tests included hardware transmitters programmed to hop randomly across the channel list. The non-multiplexed receiver hopped sequentially, while the multiplexed receiver processed all channels simultaneously.

Fig. 4 shows that LoRa-MUX can process up to 50 channels in parallel. The PRR drops beyond 10 channels, due to aggregate noise, but stabilizes around 50%, suggesting that further optimization may enable even higher channel counts.

Note that the instantaneous bandwidth of the SDR hardware limits the channel count. As a result, the USRP B200 has an instantaneous bandwidth of 56 MHz and can theoretically receive all 64 channels in parallel, while the RTL-SDR has an instantaneous bandwidth of 3.2 MHz and can receive up to 16 channels in parallel, additionally dependent on the processing power of the receiver computer.

D. Sensitivity

To evaluate sensitivity and noise robustness, we transmitted 500 packets per round using a USRP B200 mini with a relative transmitter gain ranging from 35 dB to 100 dB. Relative USRP gain is not equivalent to absolute gain and varies by device. However, by using transmitter gain, we are able to compare the sensitivity of LoRa-MUX against the sensitivity of the single channel decoder.

As explained earlier, parallel channel processing introduces a threshold-based squelch to prevent low-power or inactive channels from feeding the demodulator. The threshold must balance packet detection with noise suppression. Fig. 5 shows

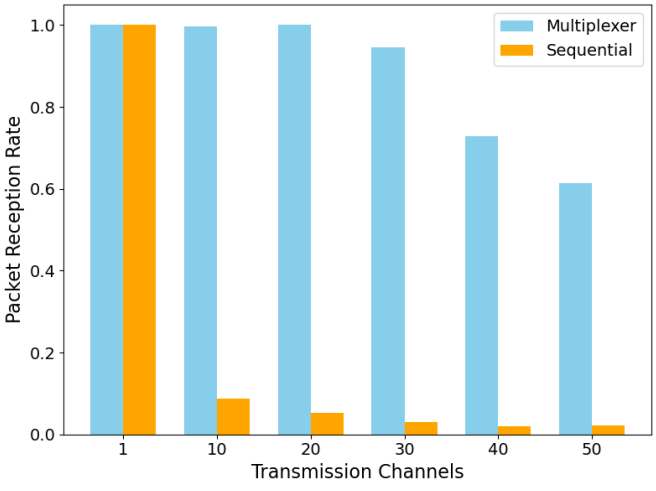

Fig. 4. Packet reception rate for LoRa-MUX (orange) and the non-multiplexed (blue) receiver as a function of the number of active channels. LoRa-MUX maintains acceptable PRR up to 50 channels, whereas the sequential receiver's PRR drops proportionally to channel count.

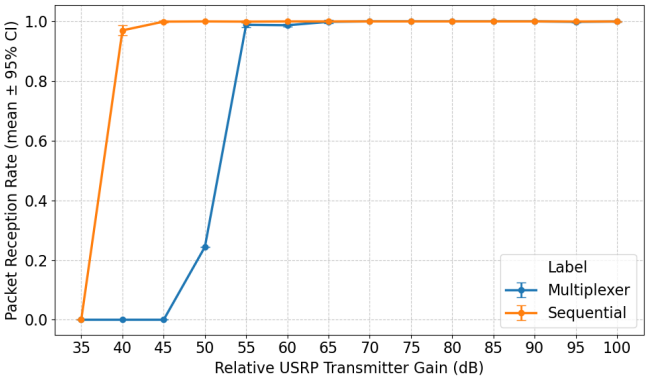

Fig. 5. Packet reception rate versus transmitter gain for LoRa-MUX and the non-multiplexed receiver with 95% confidence intervals. LoRa-MUX PRR lags slightly at low relative transmitter gains but converges at higher gains.

that the non-multiplexed receiver began detecting packets at 40 dB, while LoRa-MUX began detecting packets at 50 dB. LoRa-MUX reached comparable PRR at 55 dB, and achieved over 90% accurate decoding at higher gains. This confirms that while parallel processing introduces some sensitivity trade-offs, LoRa-MUX architecture remains highly effective under practical conditions.

These results validate the design choices in Section IV and demonstrate that LoRa-MUX achieves a favorable balance between multi-channel capacity and receiver sensitivity, while remaining compatible with low-cost SDR hardware.

VI. Applications

The multiplexing approach can also be applied to monitor devices using non-LoRa protocols, provided the monitoring device has sufficient bandwidth to cover multiple channels. LoRa represents an ideal case for this tactic, as the multiplexer does not handle collisions intelligently, and LoRa transmissions are typically sparse. Thus, transmission sparsity and hardware bandwidth are the primary limiting factors.

A. Scanning

IoT-Scan is a multi-protocol scanning tool capable of hopping between channels and protocols to assess IoT activity [9]. Scanning LoRa channels is challenging due to transmission sparsity and the large number of possible channel–parameter combinations. By integrating LoRa-MUX's multi-channel, multi-SF capabilities into IoT-Scan or similar tools, full-spectrum LoRa scanning becomes faster and more efficient.

B. Peer-to-Peer LoRa Mesh Networks

Meshtastic is an off-grid, encrypted mesh communication network that uses LoRa for long-range, public-spectrum communication [6]. The network employs six bandwidth–spreading factor pairs, called 'presets,' which act as public channels. Standard Meshtastic receivers can only interact with one preset at a time. Unlike LPWAN protocols such as LoRaWAN, Meshtastic traffic is chaotic and unpredictable, making it impossible to anticipate activity on any single preset.

Using our multiplexing receiver architecture, it is possible to receive traffic from all presets in parallel, enabling more comprehensive network coverage.

C. Load Balancing

In regions with high densities of LoRa end nodes, gateways can become overloaded. Collisions lead to packet loss, and since LoRaWAN does not provide acknowledgment mechanisms, end nodes cannot recover lost messages. While end nodes are optimized for long battery life and cannot rely on power-hungry SDRs, gateways have fewer constraints. By programming end nodes to broadcast on different channels and employing a gateway with parallel reception and collision-resistant postprocessing, network throughput and reliability can be significantly improved [10].

VII. Conclusion

This work presents, to the authors' knowledge, the first systematic approach to designing and evaluating a multi-channel LoRa receiver architecture that balances real-time processing, channel capacity, and sensitivity using low-cost software-defined radios. Through careful architectural design, including a polyphase channelizer, power squelch, rational resampling, and a minimal number of LoRa receiver modules, we achieve a system capable of processing up to fifty channels in parallel, while maintaining high packet reception rates (PRR) across a range of spreading factors and channel configurations.

Extensive simulations in MATLAB using synthetic LoRa data guided the selection of optimal filter parameters for the polyphase channelizer. By systematically varying filter order and cutoff frequency, we identified a configuration that provides a near-ideal tradeoff between passband ripple and stopband attenuation, thereby maximizing signal integrity while minimizing computational load. Comparative experiments with other window types (Rectangular, Blackman, Kaiser) and varying filter parameters validated the superiority of the chosen Hamming-based design. This data-driven design approach ensures that the multiplexer maintains reliable PRR even under high channel occupancy and moderate noise conditions.

Experimental validation with both RTL-SDR and USRP receivers confirms the effectiveness of our architecture. The multiplexed receiver consistently outperforms a single-threaded sequential flowgraph in channel throughput, while retaining over 90% PRR across 30 active channels. Sensitivity experiments further show that the multiplexer achieves more than 90% packet decoding accuracy at practical transmitter gains, demonstrating robustness to environmental variations and overlapping transmissions. These results illustrate that it is possible to combine computational efficiency, broad spectral coverage, and practical sensitivity in a low-cost multi-channel LoRa monitoring system.

Future work will focus on further improving sensitivity and noise resilience. Possible directions include adaptive thresholding for the squelch module, dynamic resampling strategies, and incorporating real-time interference estimation to further enhance packet recovery. Extending the architecture to support additional spreading factors, higher instantaneous bandwidths, or multi-antenna diversity could allow monitoring of the full LoRa spectrum simultaneously. Finally, the design methodology, combining MATLAB-based simulations, filter optimization, and hardware-in-the-loop experiments, provides a framework that can be generalized to other wideband IoT or SDR systems requiring efficient, real-time multi-channel processing.

Acknowledgments

This research was supported in part by the U.S. NSF under grant AST-2229104.

References

[1] A. Yegin. (2025). What is LoRaWAN? Retrieved from https://blog.lora-alliance.org/what-is-lorawan

[2] J. Tapparel, O. Afisiadis, P. Mayoraz, A. Balatsoukas-Stimming and A. Burg, "An Open-Source LoRa Physical Layer Prototype on GNU Radio," 2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Atlanta, GA, USA, 2020, pp. 1-5.

[3] J. Tapparel and A. Burg, "Design and Implementation of LoRa Physical Layer in GNU Radio". Proceedings of the GNU Radio Conference, Knoxville, TN, USA, 2024.

[4] P. Robyns, P. Quax, W. Lamotte, W. Thenaers. (2017). gr-lora: An efficient LoRa decoder for GNU Radio. Zenodo. 10.5281/zenodo.853201

[5] L. Pucker. (2003). Channelization techniques for software defined radio. SDR Technical Conference and Product Exposition.

[6] Meshtastic. (n.d.). Meshtastic: An open source, off-grid, decentralized, mesh network built to run on affordable, low-power devices. Retrieved from https://meshtastic.org

[b14] Nislab. 2026. LoRa-MUX. https://github.com/nislab/LoRa-MUX

[7] S.C. Kim and S.S. Bhattacharyya. (2014). Implementation of a high-throughput low-latency polyphase channelizer on GPUs. EURASIP J. Adv. Signal Process. https://doi.org/10.1186/1687-6180-2014-141

[8] Semtech. (2025). What is LoRa? Retrieved from https://www.semtech.com/lora

[9] S. Gvozdenovic, J. K. Becker, J. Mikulskis and D. Starobinski, "IoT-Scan: Network Reconnaissance for Internet of Things," in IEEE Internet of Things Journal, vol. 11, no. 8, pp. 13091-13107, 15 April 15, 2024, doi: 10.1109/JIOT.2023.3327293.

[10] S. Yu, X. Xia, N. Hou, Y. Zheng, and T. Gu. 2024. Revolutionizing LoRa Gateway with XGate: Scalable Concurrent Transmission across Massive Logical Channels. In Proceedings of the 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '24). Association for Computing Machinery, New York, NY, USA, 482–496. https://doi.org/10.1145/3636534.3649375

[11] T. Claverie and J. Lopes Esteves (2021). A LoRaWAN Security Assessment Test Bench. Proceedings Of The GNU Radio Conference, 2(1). Retrieved from https://pubs.gnuradio.org/index.php/grcon/article/view/88.

[12] Trend Micro Research. (2020). Compromised Comms: Assessing the Security of LoRaWAN Radio. Retrieved from https://www.trendmicro.com/vinfo/us/threat-encyclopedia.

[13] X. Xia, Y. Zheng and T. Gu, "FTrack: Parallel Decoding for LoRa Transmissions," in IEEE/ACM Transactions on Networking, vol. 28, no. 6, pp. 2573-2586, Dec. 2020, doi: 10.1109/TNET.2020.3018020