

**Exploring LoRaWAN: Simulation-Based Performance
Analysis and Machine Learning-Driven Spreading Factor
Optimization**



By

Usama Mustafa

(Registration No: 00000432011)

Department of Information Security

Military College of Signals

National University of Sciences & Technology (NUST)

Islamabad, Pakistan

(2024)

**Exploring LoRaWAN: Simulation-Based Performance
Analysis and Machine Learning-Driven Spreading Factor
Optimization**



By

Usama Mustafa

(Registration No: 00000432011)

A thesis submitted to the National University of Sciences and Technology, Islamabad,

in partial fulfillment of the requirements for the degree of

Master of Science in
Information Security

Supervisor: Dr. Imran Rashid

Co Supervisor: Dr. Imran Mahkdoom

Military College of Signals

National University of Sciences & Technology (NUST)

Islamabad, Pakistan

(2024)

THEISIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS Thesis written by Maj Usama Mustafa, Registration No. 00000432011, of Military College of Signals has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations/MS Policy, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS degree. It is further certified that necessary amendments as pointed out by GEC members and local evaluators of the scholar have also been incorporated in the said thesis.

Signature: _____

Name of Supervisor: Prof Dr Imran Rashid

Date: _____

Signature (HOD): _____

Date: _____

Signature (Dean/Principal) _____

Date: _____

NATIONAL UNIVERSITY OF SCIENCES & TECHNOLOGY
MASTER THESIS WORK

We hereby recommend that the dissertation prepared under our supervision by **Maj Usama Mustafa, MSIS-21 Course** Regn No **00000432011** Titled: **"Exploring LoRaWAN: Simulation-Based Performance Analysis and Machine Learning-Driven Spreading Factor Optimization"** be accepted in partial fulfillment of the requirements for the award of **MS Information Security** degree.

Examination Committee Members

1. Name: **Maj Sarmad Idrees**

Signature: _____

2. Name: **Maj Ammar Hassan**

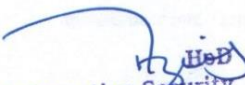
Signature: _____

Co-Supervisor's Name: **Col Imran Makhdoom, PhD.**

Signature: _____

Supervisor's Name: **Prof Dr Imran Rashid, PhD.**

Signature: _____


Information Security
Military College of Sigs
Head of Department

Date: _____

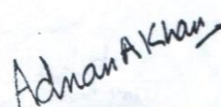
12/9/24

Date

COUNTERSIGNED

Date: _____

12/9/24


Adnan A Khan

Dean

CERTIFICATE OF APPROVAL

This is to certify that the research work presented in this thesis, entitled "Exploring LoRaWAN: Simulation-Based Performance Analysis and Machine Learning-Driven Spreading Factor Optimization." was conducted by Maj Usama Mustafa under the supervision of Dr. Imran Rashid. No part of this thesis has been submitted anywhere else for any other degree. This thesis is submitted to the Military College of Signals, National University of Science & Technology Information Security Department in partial fulfillment of the requirements for the degree of Master of Science in Field of Information Security Department of information security National University of Sciences and Technology, Islamabad.

Student Name: Usama Mustafa


Signature: 

Examination Committee:

a) External Examiner 1: Name Maj Sarmad Idrees (MCS)

Signature: 

b) External Examiner 2: Name Maj Ammar Hassan(MCS)

Signature: 

Name of Co-Supervisor: Col Imran Makhdoom

Signature: 

Name of Supervisor: Brig Dr. Imran Rashid

Signature: 

Name of Dean/HOD: Dr. Muhammad Faisal Amjad

Signature: 
HoD
Information Security
Military College of Sigs

AUTHOR'S DECLARATION

I Usama Mustafa hereby state that my MS thesis titled "Exploring LoRaWAN: Simulation-Based Performance Analysis and Machine Learning-Driven Spreading Factor Optimization." is my own work and has not been submitted previously by me for taking any degree from National University of Sciences and Technology, Islamabad or anywhere else in the country/ world. At any time if my statement is found to be incorrect even after I graduate, the university has the right to withdraw my MS degree.

Student Signature:  _____

Name: Usama Mustafa _____

Date: _____

PLAGIARISM UNDERTAKING

I solemnly declare that research work presented in the thesis titled "Exploring LoRaWAN: Simulation-Based Performance Analysis and Machine Learning-Driven Spreading Factor Optimization." is solely my research work with no significant contribution from any other person. Small contribution/ help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero-tolerance policy of the HEC and National University of Sciences and Technology (NUST), Islamabad towards plagiarism. Therefore, I as an author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS degree, the University reserves the rights to withdraw/ revoke my MS degree and that HEC and NUST, Islamabad has the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized thesis.

Student Signature:  _____

Name: Usama Mustafa _____

Date: _____

DEDICATION

This Thesis is dedicated to my Family and friends for their unconditional love and support.

ACKNOWLEDGEMENTS

In the pursuit of knowledge and the completion of this academic endeavor, I am profoundly grateful to Allah, the Most Gracious and Most Merciful, for bestowing upon me the strength, wisdom, and guidance throughout this journey. It is through His divine grace that I have been able to navigate the challenges and reach this significant milestone.

I extend my deepest gratitude to my supervisor, Dr Imran Rashid, whose unwavering support, invaluable guidance, and scholarly insights have been instrumental in shaping the trajectory of this research. His expertise, commitment, and collaborative spirit significantly contributed to the successful realization of the practical aspects of this work. His mentorship has been a source of inspiration, and I am truly appreciative of the time and expertise he generously shared with me.

I would also like to express my sincere appreciation to my elder sister Sana Mustafa for her dedicated assistance and support in the implementation phase of this project.

TABLE OF CONTENTS

LIST OF TABLES	IV
LIST OF FIGURES	V
LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS	VII
ABSTRACT	VIII
CHAPTER 1: INTRODUCTION	1
1.1. Background	1
1.2. Contribution	3
1.3. Organization of the Thesis	4
CHAPTER 2: LORA-LORAWAN-LORAWAN ALLIANCE	5
2.1. LoRa	5
2.1.1 LoRa Benefits at a glance	5
2.1.2 LoRa Modulation	5
2.1.3 CSS Inculcation in LoRa	5
2.1.4 Duration of a transmitted symbol	7
2.2 LoRaWAN - Architecture	8
2.2.1 LoRaWAN Network Framework	9
2.2.2 End Devices - LoRaWAN	9
2.2.3 Gateways – LoRaWAN	9
2.2.4 Network Server – LoRaWAN	10
2.2.5 Application Server – LoRaWAN	11
2.2.6 IoT Framework	11
2.2.7 LoRaWAN Technology Layers	12
2.3 LoRaWAN – Security Considerations	12
2.3.1 Encryption Methodology -LoRaWAN	13
2.3.2 Network Server Authentication – LoRaWAN	13
2.3.3 Merging Authentication and Encryption	14
2.4 LoRaWAN – Classes of End Devices	14
2.4.1 Class A	15
2.4.2 Class B	15
2.4.3 Class C	16
2.5 Enabling LoRaWAN End-Devices: ABP and OTAA Activation	17
2.5.1 Activation by Personalization	17
2.5.2 OTAA- Over the air activation	18
2.5.3 Join Procedure – OTAA	18
2.5.4 Advantages & Disadvantages– OTAA & ABP	20
2.6 LoRa - LoRaWAN Frames	23
2.6.1 LoRaWAN Frame Classifications	23
2.6.2 LoRaWAN Protocol layers	23
2.7 LoRaWAN Power Consumption	26

2.8	LoRaWAN MAC Commands	26
2.9	Adaptive Data Rate for LoRaWAN	27
2.10	LoRaWAN Alliance	28
CHAPTER 3: EVALUATING LORAWAN SIMULATORS: A COMPREHENSIVE REVIEW AND COMPARATIVE ANALYSIS		30
3.1	Purpose of Simulations in LoRaWAN Networks	30
3.2	Importance of Simulators in LoRaWAN Research	30
3.2.1	Testing and Validation	30
3.2.2	Cost Effectiveness	31
3.2.3	Scalability	31
3.3	LoRaWAN based simulators comparison	31
3.4	LoRaWAN Simulators survey	32
3.4.1	Hardware Specifications	33
3.4.2	LoRa SF – Simulator	33
3.4.3	LoRaSim	39
3.4.4	LoRaSim-2	47
3.4.5	Framework for LoRa	48
3.4.6	Ns-3 Module	49
3.4.7	LWN Simulator	51
3.5	Conclusion	53
CHAPTER 4: LORA SPREADING FACTOR – MACHINE LEARNING DRIVEN OPTIMIZATION		54
4.1	Packet Collision in LoRa Networks	54
4.1.1	Impact of Spreading Factor on Packet Collision/ Interference-induced collision	54
4.2	Techniques Employed for Packet collisions avoidance	55
4.2.1	Lowest Spreading Factor Technique for Packet collisions avoidance	55
4.2.2	Adaptive Data rate for Packet collisions avoidance	55
4.2.3	Managing Spreading factor for Packet Collisions avoidance	55
4.3	ML-Based SF Allocation for Packet Collision Avoidance	56
4.3.1	Using Traditional ML algorithms for enhancing SF allocation	56
4.3.2	Approaching SF Assignment as a classification problem	56
4.3.3	Applying DL algorithms to Enhance SF allocation	57
4.4	SmartLoRaML: Optimizing LoRa SF Allocation with ML Algorithms	58
4.5	SmartLoRaML – Methodology	58
4.5.1	Dataset generation	58
4.5.2	SmartLoRaML Technique	60
4.5.3	ML Algorithms used SmartLoRaML Technique	61
4.5.4	Simulation Environment	64
4.5.5	Evaluation of SmartLoRaML Technique	65
4.5.6	Simulation Results	67
4.6	Conclusion and Future works	102
	References	10

LIST OF TABLES

	Page No
Table 1.1: Symbol Transmission Duration of 125 khz BW.....	8
Table 1.2: Maximum Frame Payload Size of respective data rate.....	26
Table 3.1: Comparison of LoRaWAN Simulators.....	32
Table 3.2: Hardware Specifications for Simulations.....	33
Table 3.3: Features of LoRaSim.....	48
Table 3.4: FLoRa Features.....	49
Table 3.5: NS-3 module for LoRaWAN features.....	51
Table 4.1: Simulator Parameter Set.....	60
Table 4.2: Sample Dataset.....	60
Table 4.3: Hardware Specifications for Simulation.....	65
Table 4.4: Performance Evaluation of SmartLoRaML.....	69
Table 4.5: Performance Evaluation – PDR of SmartLoRaML.....	97
Table 4.6: Performance Evaluation Comparison – PDR of SmartLoRaML.....	99

LIST OF FIGURES

	Page No.
Figure 1.0-1 Bandwidth and Range comparison of various Wireless Networks[1]	2
Figure1.0-2: Characteristics of LPWAN Technologies[1]	3
Figure 2.0-1: Chirp Modulation.....	6
Figure2.0-2: LoRa Transmission Spectrum[1]	6
Figure 2.0-3: SF2 LoRa Modulation symbol transmission (theoretical)[1]	7
Figure 2.0-4: Symbol Transmission duration[1].....	7
Figure 2.0-5: LoRa Frame Time on Air.....	8
Figure 2.0-6: Complete LoRaWAN Architecture.....	9
Figure 2.0-7: LoRaWAN Gateway	10
Figure 2.0-8: End device and Network Server Authentication [2]	10
Figure 2.0-9: Application Server and the End Device Encryption [2]	11
Figure 2.0-10: IoT Framework and Application Server Connection [2]	11
Figure 2.0-11: LoRaWAN Technology Layers [2].....	12
Figure 2.0-12: LoRaWAN End to End Security [2]	13
Figure2.0-13: Encryption Methodology[2]	13
Figure 2.0-14: Network Server Authentication[2]	14
Figure 2.0-15: Encryption then Authentication [2].....	14
Figure 2.0-16: End device (Class A) receiving slots[2].....	15
Figure 2.0-17: End device (Class B) receiving slots[2].....	16
Figure 2.0-18: End device (Class C) receiving slots [2].....	16
Figure 2.0-19 Classes Power Consumption and Downlink Capabilities[2]	17
Figure 2.0-20: Parameters Storage in ABP [3]	17
Figure 2.0-21: Parameters stored in OTAA before the join request.	18
Figure 2.0-22: Parameters stored in OTAA after the Configuration [3].....	18
Figure 2.0-23: OTAA join procedure – Simplified [3].....	19
Figure 2.0-24: Join-Request and Join-Accept Procedures in elaboration [3]	20
Figure 2.0-25: Keys secured memory storage [3].....	21
Figure 2.0-26: Reply attack in uplink messages[3]	22
Figure 2.0-27: Reply attack avoidance through frame counter[3].....	22
Figure 2.0-28: LoRaWAN Protocol Stack[3]	24
Figure 2.0-29: Layer wise frame distribution – LoRaWAN[3]	24
Figure 2.0-30: LoRaWAN application layer[4].....	25
Figure 2.0-31: LoRaWAN MAC Layer [4]	25
Figure 2.0-32: Frame of physical layer of LoRaWAN [4]	26
Figure 2.0-33: ADR Procedure[4]	27
Figure 2.0-34: MAC commands for ADR[4]	28
Figure 3.0-1: PDR of SFs – Single Gateway	37
Figure 3.0-2: PDR of various Radii - Single Gateway	37
Figure 3.0-3: PDR of various Packet rates- Single Gateway.....	38
Figure 3.0-4: PDR of various radii- Multiple Gateway	39

Figure 3.0-5: PDR of Multiple Gateways	39
Figure 3.0-6: LoRaSim User Interface [17].....	40
Figure 3.0-7: Simulator Simulation Intervals [17].....	41
Figure 3.0-8: Throughput vs Time at DR0 and radius 2km.....	42
Figure 3.0-9: Success Probability vs time at DR0 -2km.....	42
Figure 3.0-10: Throughput vs Time at DR1 and radius 2km.....	43
Figure 3.0-11: Success Probability vs time at DR1 - 2km.....	43
Figure 3.0-12: Throughput vs Time at DR2 and radius - 2km.....	44
Figure 3.0-13: Success Probability vs time at DR2 -2km	44
Figure 3.0-14: Throughput vs Time at DR3 and radius - 3km.....	45
Figure 3.0-15: Success Probability vs time at DR2 - 3km	45
Figure 3.0-16: Throughput vs Time at DR3 and radius - 4km.....	46
Figure 3.0-17: Success Probability vs time at DR2 - 4km	47
Figure 3.0-18: User Interface of LWN Simulator	51
Figure 4-0-1: Symbol Transmission duration.....	54
Figure 4-0-2: Bootstrapping Method [39].....	62
Figure 4-0-3: Bootstrapping samples[39]	62
Figure 4-0-4: Feature set Selection by Decision Trees and Random Forest[40].....	63
Figure 4-0-5: SmartLoRaML Classifiers Accuracy at 3000m.....	91
Figure 4-0-6: SmartLoRaML Classifiers Accuracy at 5000m.....	91
Figure 4-0-7: SmartLoRaML Classifiers Accuracy at 7000m.....	92
Figure 4-0-8: SmartLoRaML Classifiers Accuracy at 10000m.....	92
Figure 4-0-9: Precision and Recall Bar Chart – SmartLoRaML Classifier RF	93
Figure 4-0-10: Precision and Recall Bar Chart – SmartLoRaML Classifier XGB.....	94
Figure 4-0-11: F1 score line chart – SmartLoRaML Classifiers	95
Figure 4-0-12: MCC score Scatter plot – SmartLoRaML Classifier RF	96
Figure 4-0-13: MCC score Scatter plot – SmartLoRaML Classifier XGB	96
Figure 4-0-14: Box Plot of SmartLoRaML Classifiers.....	98
Figure 4-0-15: Transmit Energy Consumption of SmartLoRa Classifiers	99
Figure 4-0-16: Performance Evaluation of SmartLoRa Classifiers with SVM and DT and Stratified SVM and DT Classifiers at radius 3000m	100
Figure 4-0-17: Performance Evaluation of SmartLoRa Classifiers with SVM and DT. 101	
Figure 4-0-18: Performance Evaluation of SmartLoRa Classifiers with SVM and DT and Stratified SVM and DT Classifiers at radius 7000m	101
Figure 4-19: Performance Evaluation of SmartLoRa Classifiers with SVM and DT and Stratified SVM and DT Classifiers at radius 10000m	102

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

LPWAN	Low Power Wide Area Network
LoRaWAN	Long Range Wide Area Network
PDR	Packet Delivery Ratio
MCC	Matthews Correlation Coefficient
TEC	Transmit Energy Consumption
GUI	Graphical User Interface
SF	Spreading Factor
NFC	Near Field Communication
NB-IOT	Narrowband Internet of Things
LTE	Long Term Evolution
LTE-M	Long Term Evolution for Machines
IoT	Internet of Things

ABSTRACT

This thesis provides an in-depth analysis of LoRaWAN technology, encompassing its latest trends, practical applications, and security aspects. A novel technique, SmartLoRaML, introduced leveraging machine learning to optimize spreading factor (SF) allocation in LoRaWAN networks. This approach dynamically adjusts SF based on network conditions, leading to improved collision detection. The technique is implemented using a dataset generated from a LoRaWAN simulator, which is then modified to incorporate SmartLoRaML. Performance is assessed across various scenarios using metrics such as accuracy, packet delivery ratio (PDR), energy consumption, recall, precision, F1 score, and Matthews Correlation Coefficient (MCC). Evaluations show that performance varies based on node density and communication radius. Additionally, SmartLoRaML has enhanced transmit energy consumption compared to the random SF allocation method used by the simulator. This comprehensive analysis provides valuable insights for optimizing LoRa resource allocation and enhancing IoT network anomaly detection. Furthermore, this thesis explores a selection of commonly used open-source LoRa/LoRaWAN simulation tools, providing a comparative summary based on programming language, target domain, operating system support, and the presence of a graphical user interface (GUI).

Keywords: LoRaWAN, Spreading factor, Packet delivery ratio, Collision detection

CHAPTER 1: INTRODUCTION

Frequently touted as the modern industrial revolution, the IoT often pledges intelligence across various domains like buildings, cities, and healthcare, labeling them as "smart." Nevertheless, achieving this smartness isn't straightforward, given the complexities and diversity of existing protocols. This thesis aims to offer insights into one of the primary protocols within the IoT landscape: LoRaWAN, aiding in its comprehension.

1.1. Background

In the realm of IoT protocols like Bluetooth, Zigbee, Wi-Fi, and cellular networks, classifications often prioritize bandwidth and range. However, overlooking power consumption in this assessment can be misleading. For instance, while high-range, high-bandwidth protocols seem superior, low-power options like NFC defy this logic, operating without energy storage. On the other end, LPWANs such as LoRaWAN, NB-IoT, and LTE-M offer long-range, low-bandwidth capabilities with minimal power demands. In this context, our focus lies on LoRaWAN, a standard emphasizing extended reach and low energy usage for IoT applications.

Traditional wireless methods like cellular networks (e.g., 2G, 3G, LTE) and short-range tech (e.g., NFC, Bluetooth, WiFi, Zigbee) can't offer both low power and long range simultaneously [1]. While cellular networks cover long distances with high data rates, they're complex and power-hungry, optimized for voice and data with low latency—unnecessary for most IoT applications [2]. Short-range tech consumes less power but lacks range, limited to a few hundred meters at best. LPWAN tech bridges this gap, prioritizing low power and long-range communication by sacrificing data rates and latency. Characteristics include long ranges (a few to tens of kilometers), low power (up to ten years of battery life), modest data rates (tens of kbps), affordability, and higher latency (seconds to minutes), as depicted in Figure 1.1.

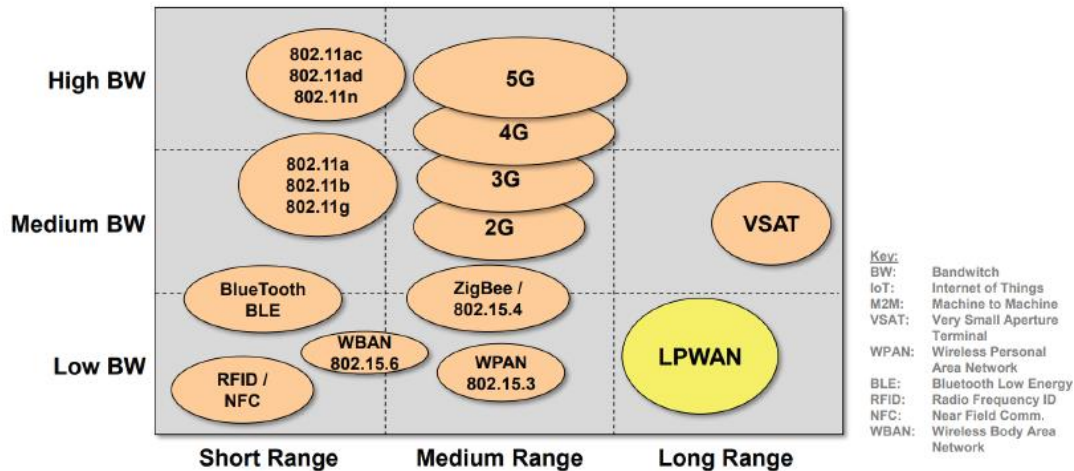


Figure 1.0-1 Bandwidth and Range comparison of various Wireless Networks[1]

Various emerging LPWAN technologies, like LoRa, Sigfox, NB-IoT, and LTE-M, are in competition. LoRa and Sigfox utilize license-free ISM frequency bands, while NB-IoT and LTE-M use licensed bands, incurring additional costs. LoRa and Sigfox excel in ultra-low power consumption and interference resilience, whereas NB-IoT and LTE-M are lauded for higher data rates. LoRa's open standard MAC protocol, LoRaWAN, and Sigfox rely on pure ALOHA medium access. LoRaWAN networks can be private, akin to WiFi, while Sigfox and NB-IoT are solely accessible via operator contracts. Sigfox limits daily messages to 140 uplink packets and 4 downlink packets, with constrained payload sizes. Conversely, LoRa and NB-IoT support larger payload sizes, with varying maximum data rates of 50 kbps for LoRa, 200 kbps for NB-IoT, and 100 bps for Sigfox [3].

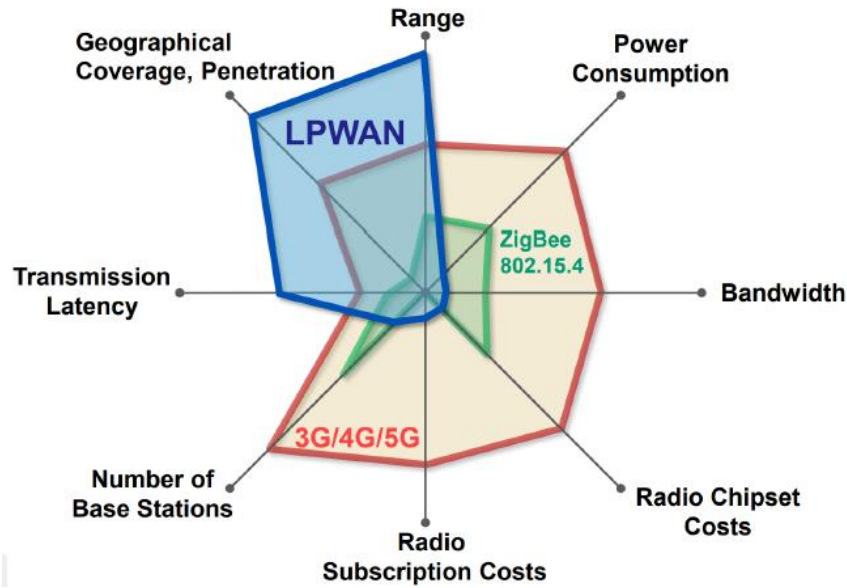


Figure1.0-2: Characteristics of LPWAN Technologies[1]

LoRa's flexibility lies in its ability to vary data rates by spreading symbols in a fixed channel width. With 6 spreading factor (SF) options, LoRa manages tradeoffs between transmission air time and receive sensitivity. While identical SF transmissions risk collision, different SF transmissions in the same channel remain orthogonal, emphasizing the importance of thoughtful spreading factor allocation for optimal network performance [4].

1.2. Contribution

This thesis introduces SmartLoRaML, a novel technique leveraging machine learning (Random Forest and Gradient Boosting) is introduced to optimize spreading factor (SF) allocation in LoRaWAN networks. This approach dynamically adjusts SF based on network conditions, leading to improved collision detection and prevention. Evaluations demonstrate that SmartLoRaML's performance varies with factors like node density and communication radius. This approach aims to significantly improve LoRaWAN performance by efficiently detecting and preventing collisions. We evaluate these classifiers across various metrics including accuracy, packet delivery ratio (PDR), energy consumption, recall, precision, F1 score, and Matthews Correlation Coefficient (MCC). The evaluations are conducted under diverse network topologies and node configurations. This comprehensive analysis provides valuable insights for optimizing LoRa

resource allocation. Besides, it outlines the challenge of spreading factor allocation in both single and multiple gateway networks, highlighting its impact on collisions and overall network. In furtherance, this thesis explores commonly used open-source LoRa/LoRaWAN simulation tools (detailed in references). Among these, LoRaWAN-SIM, NS-3, OMNeT++ (FLoRa), LoRaSim, and LoRaSF are chosen for further analysis based on their free availability for research, active development, comprehensive documentation, suitability for Internet of Things (IoT) and Wireless Sensor Network (WSN) applications, and their increasing popularity in LoRa/LoRaWAN research. This thesis summarizes these simulators by programming language, target domain (generic or LoRaWAN specific), operating system support, and presence of a graphical user interface (GUI).

1.3. Organization of the Thesis

- **Chapter 1:** This chapter includes the basic introduction, establishes the research motivation and research contribution.
- **Chapter 2:** Provide an in-depth analysis of LoRaWAN technology, encompassing its latest trends, practical applications, and security aspects.
- **Chapter 3:** This chapter reviews open-source LoRa/LoRaWAN simulators and selects some for further analysis based on their features and research suitability.
- **Chapter 4:** This chapter introduces SmartLoRaML, a machine learning approach for optimizing spreading factor allocation in LoRaWAN networks, analyzes its performance under various conditions, and evaluates the technique performance through multiple metrics.

CHAPTER 2: LORA-LORAWAN-LORAWAN ALLIANCE

2.1. LoRa

LoRa technology, devised by Semtech Corporation, is a radio frequency modulation system designed for low-power, wide area networks (LPWANs). Its name, LoRa, signifies the remarkable capability for extensive data transmission distances. It facilitates long-range communication, spanning up to three miles (five kilometers) in urban environments and exceeding 10 miles (15 kilometers) in rural settings (with a clear line of sight). An essential attribute of LoRa-based systems is their exceptionally low power consumption, enabling the creation of battery-operated devices with a potential lifespan of up to a decade. When structured in a star-shaped network layout utilizing the open LoRaWAN protocol, this technology is ideal for applications necessitating widespread, long-distance, or deep indoor communication among numerous devices with minimal power demands, collecting small data quantities.

2.1.1 *LoRa Benefits at a glance*

A single LoRa gateway covers over 10 miles in rural areas and up to three miles in urban environments for message transmission. The energy needed for small data packets means devices can last years on minimal power. A LoRaWAN network supports millions of messages, depending on the number of gateways installed. Each eight-channel gateway handles up to 1.5M messages in 24 hours, supporting around 60,000 devices sending hourly messages. Adding more gateways expands capacity as needed. With few gateways needed in a star network, costs for both equipment and operations stay low. Using affordable LoRa RF modules in end nodes with the open LoRaWAN standard can offer significant returns on investment.

2.1.2 *LoRa Modulation*

LoRa utilizes a proprietary spread-spectrum modulation method based on Chirp Spread Spectrum (CSS) technology. It achieves a balance between sensitivity and data rate by operating within fixed-bandwidth channels of 125 KHz or 500 KHz for uplink and 500 KHz for downlink. Employing orthogonal spreading factors (a technique to enable simultaneous communication between multiple devices), LoRa adapts power levels and data rates of individual end nodes, preserving their battery life. For instance, nodes near gateways transmit at low spreading factors, conserving link budget, while those farther away use higher spreading factors for enhanced reception sensitivity despite lower data rates.

2.1.3 *CSS Inculcation in LoRa*

LoRa modulation emits a symbol, depicted in figure 2.1, known as "Chirp," derived from radar technology (Chirp: Compressed High Intensity Radar Pulse). The start frequency equals the channel frequency minus half the bandwidth, while the final frequency equals the channel frequency plus half the bandwidth. Figure 2.2 illustrates the frequency domain representation of LoRa modulation.

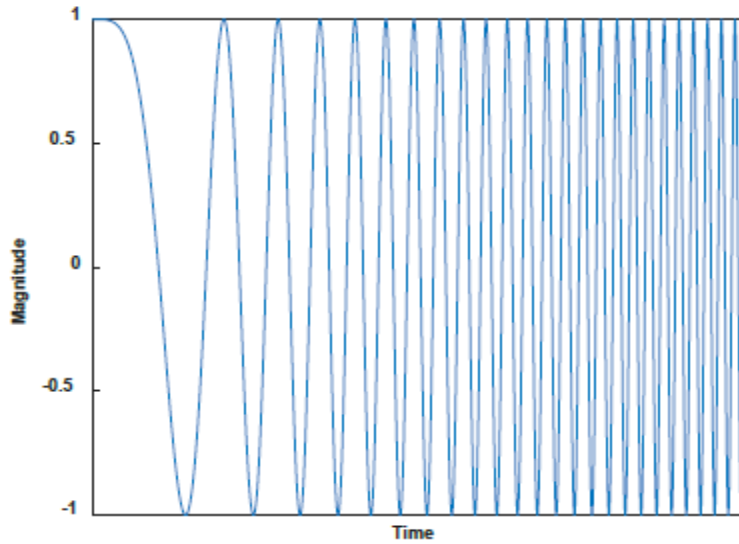


Figure 2.0-1: Chirp Modulation

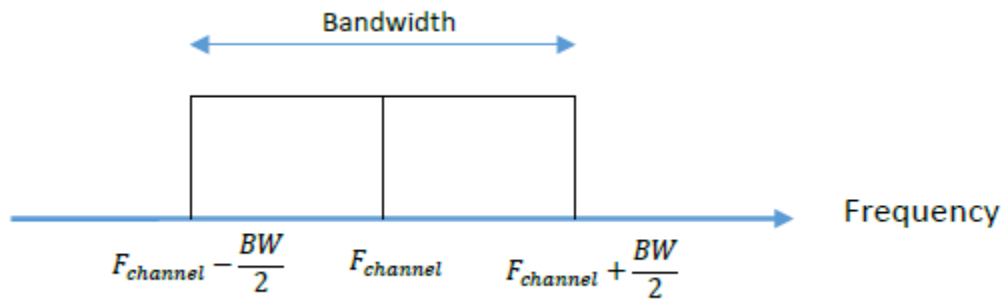


Figure2.0-2: LoRa Transmission Spectrum[1]

Within LoRa, each symbol conveys a set number of transmitted bits according to the Spreading Factor. For instance, when employing SF10, one symbol (Chirp) represents 10 bits. These bits organize into packets of SF bits during transmission, each packet linked to one of 2^{SF} potential symbol forms. While symbols differ in starting frequencies, they all encapsulate a packet of bits. Figure 2.3 illustrates a theoretical instance of SF2 modulation at 868.1 MHz using a 125 kHz bandwidth, where each symbol signifies 2 bits.

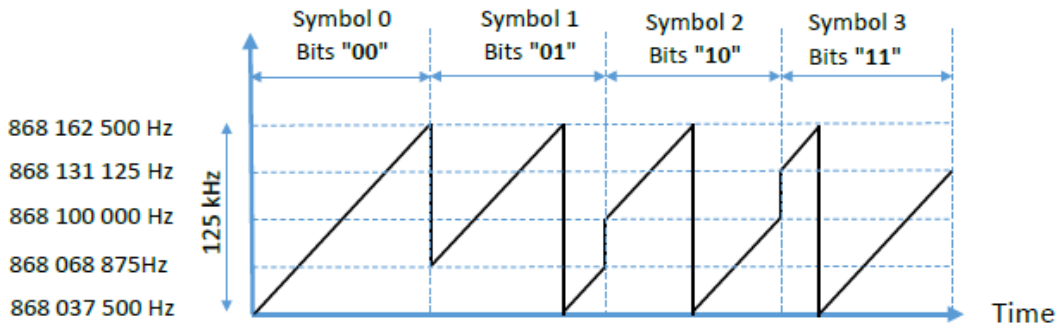


Figure 2.0-3: SF2 LoRa Modulation symbol transmission (theoretical)[1]

2.1.4 Duration of a transmitted symbol

Within LoRa, the duration of each symbol's transmission (T_{symbol}) correlates with the Spreading Factor. Increased SF values result in extended transmission times. When comparing the same bandwidth, the transmission duration of an SF8 symbol is double that of an SF7 symbol. This pattern holds true up to SF12. The transmission time for each symbol (T_{symbol}) relies on the chosen bandwidth. T_{symbol} shows an inverse relationship with the bandwidth. Considering both the Spreading Factor (SF) and the bandwidth, this dependency becomes evident.

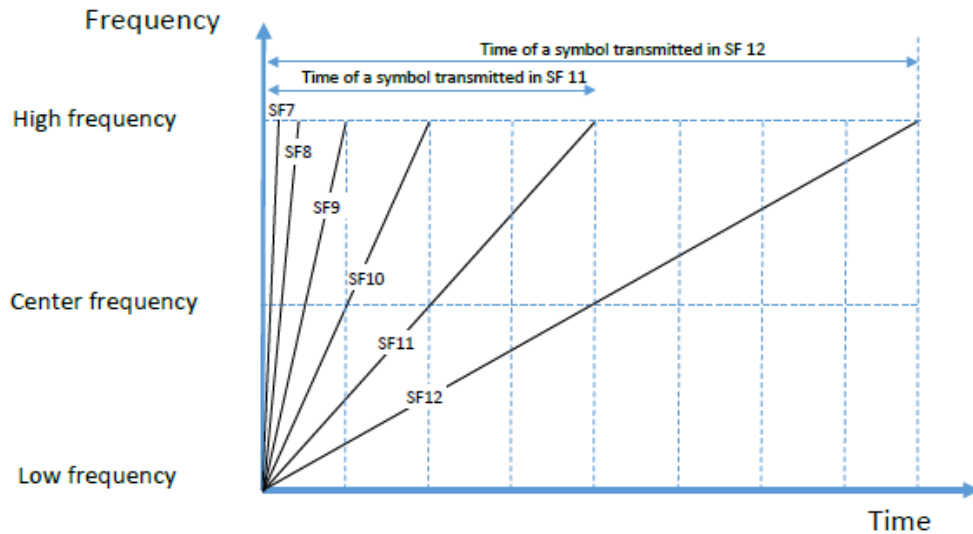


Figure 2.0-4: Symbol Transmission duration[1]

For instance, Table 2.1 illustrates how transmission time varies with SF when utilizing a 125 KHz bandwidth.

Table 1.1: Symbol Transmission Duration of 125 khz BW

Spreading Factor	Symbol Transmission Time
SF7	1.024 ms
SF8	2.048 ms
SF9	4.096 ms
SF10	8.192 ms
SF11	16.384 ms
SF12	32.768 ms

2.1.5 Time on Air – LoRa

The Time on Air represents the total duration of a LoRa frame, contingent upon the quantity of symbols within the frame, encompassing the payload, preamble, header, and CRC.

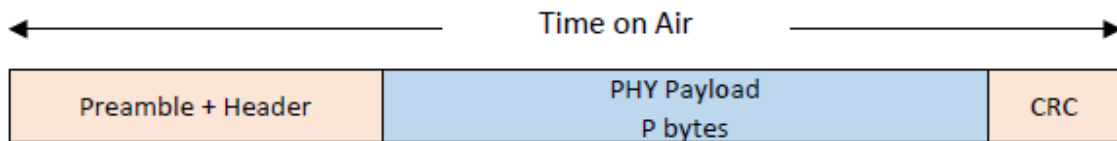


Figure 2.0-5: LoRa Frame Time on Air

Time on Air is the product of N_{symbol} and T_{symbol}

$$\text{Time on Air} = N_{\text{symbol}} \cdot T_{\text{symbol}}$$

Where N_{symbol} is the Total symbols in the LoRa Frame.

2.1.6 Data Rate – LoRa

$$\text{Data Rate} = \text{SF} \cdot [\text{BW}/2^{\text{SF}}]$$

As every symbol consists of Spreading factor the above-mentioned equation depicts the data rate.

2.2 LoRaWAN - Architecture

LoRa serves as the modulation for communication between LoRa end-devices or gateways, while the broader network infrastructure, called LoRaWAN, securely connects these devices to servers for data delivery to end users. LoRa's physical layer involves Chirp Spread Spectrum modulation and specific frame formats for data transmission between transmitters and receivers. In contrast, LoRaWAN defines the network architecture, including end-devices, gateways, and

servers, and utilizes a detailed frame format enabling secure data transmission from LoRaWAN end-devices to LoRaWAN servers.

2.2.1 LoRaWAN Network Framework

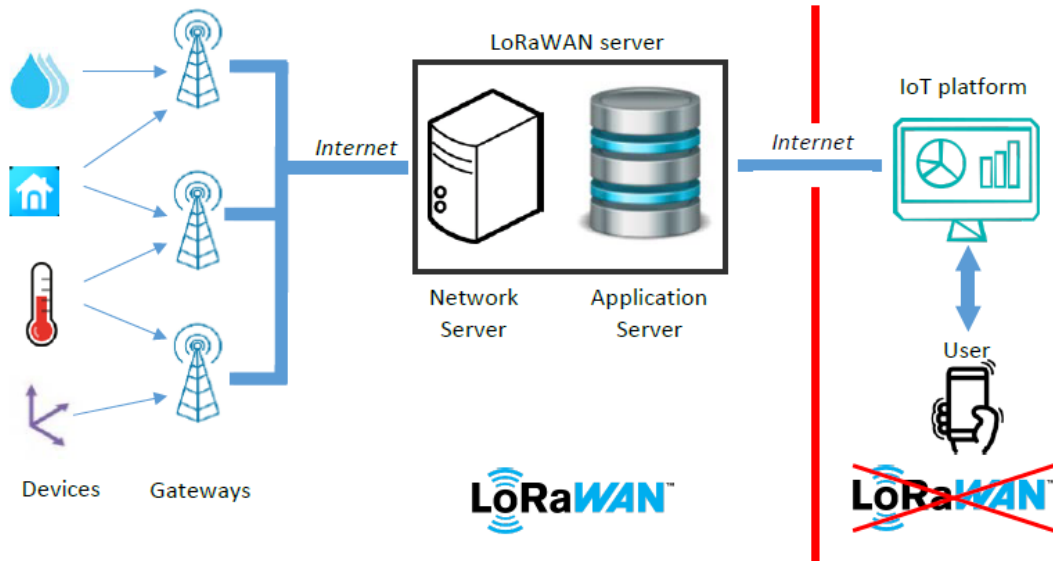


Figure 2.0-6: Complete LoRaWAN Architecture

In the depicted LoRaWAN architecture shown in Figure 2.6, the transmitting entity is the LoRaWAN end-device on the left side, while on the opposite end, the user receives the transmitted data via the network. The central components driving the LoRaWAN architecture are the LoRaWAN end-devices, gateways, Network Server, and Application Server. However, the IoT platform and user connection remain distinct and unrelated to these core components, functioning solely as conventional web services.

2.2.2 End Devices - LoRaWAN

LoRaWAN end-devices, integral to the realm of IoT, are electronic embedded systems characterized by their low power consumption, compact size, and affordability. Equipped with a LoRa radio transceiver, they establish connections with gateways. Notably, these end-devices don't communicate exclusively with a single gateway; instead, all gateways within their coverage range receive and handle the messages transmitted by the device.

2.2.3 Gateways – LoRaWAN

LoRaWAN gateways possess the ability to simultaneously listen across all channels and Spreading Factors. Upon receiving a LoRa frame, they promptly transmit its content to the preconfigured Network Server via the Internet. On one end, the gateway receives LoRa modulation through its antenna, while on the other, it establishes connectivity to the Internet through various backhaul options like 3G, 4G, 5G, Wi-Fi, Ethernet, or LTE-M. Additionally, each LoRaWAN

gateway is distinguished by a unique identifier (64-bit EUI) used for registering and activating the gateway on a Network Server.

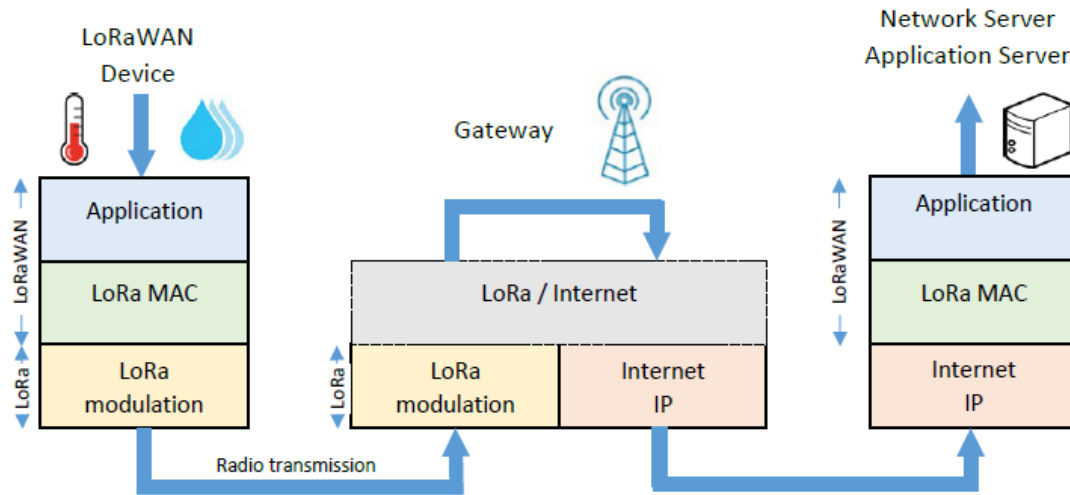


Figure 2.0-7: LoRaWAN Gateway

2.2.4 Network Server – LoRaWAN

Messages transmitted by gateways are received and processed by the Network Server, which eliminates duplicate packets—a common occurrence when multiple gateways capture the same message. The Network Server proceeds to authenticate these messages using a 128-bit AES key known as Network Session Key. It's important to note that this authentication process ensures the message's validity without encryption. Messages that fail authentication are discarded by the Network Server. However, for those that pass authentication, the Network Server forwards them to the Application Server.

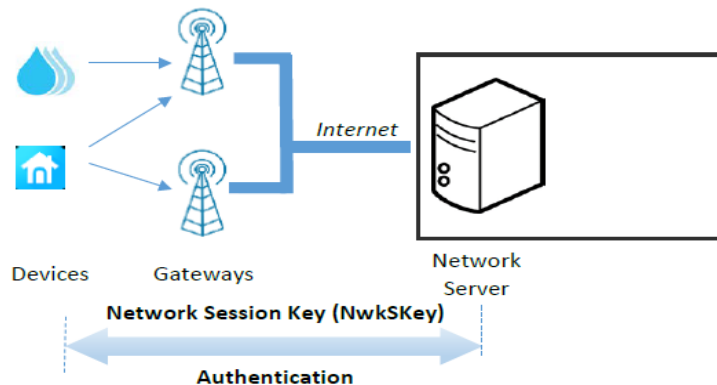


Figure 2.0-8: End device and Network Server Authentication [2]

2.2.5 Application Server – LoRaWAN

Encrypted messages from the Network Server are handled by the Application Server, with encryption and decryption facilitated by a 128-bit AES key known as AppSKey (Application Session Key).

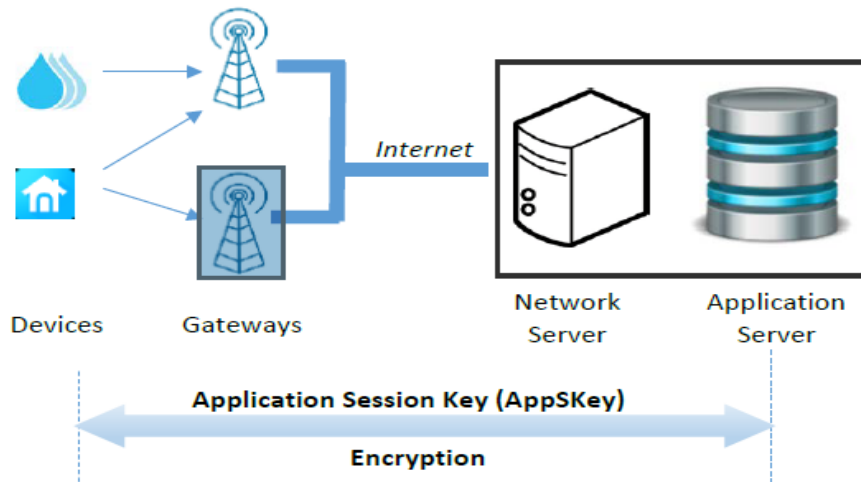


Figure 2.0-9: Application Server and the End Device Encryption [2]

2.2.6 IoT Framework

The IoT platform acts as the user application interface, encompassing three core elements: first, a connection interface with the Application Server to gather data, commonly utilizing HTTP or MQTT protocols, initially using non-secure versions during development and transitioning to HTTPS and MQTTS for secure deployment. Second, it incorporates a data storage system to retain collected information. Finally, it offers a user-friendly dashboard accessible via web browsers or mobile applications, providing users with convenient access to the data.

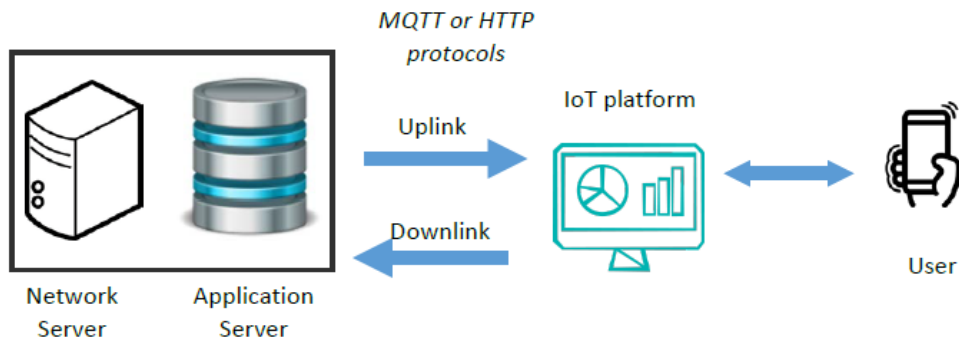


Figure 2.0-10: IoT Framework and Application Server Connection [2]

2.2.7 LoRaWAN Technology Layers

LoRa constitutes the wireless modulation in the LoRaWAN protocol architecture, establishing a long-range communication link. Meanwhile, LoRaWAN, an open networking protocol managed by the LoRa Alliance, ensures secure two-way communication, mobility, and standardized localization services. As depicted in the figure 2.11

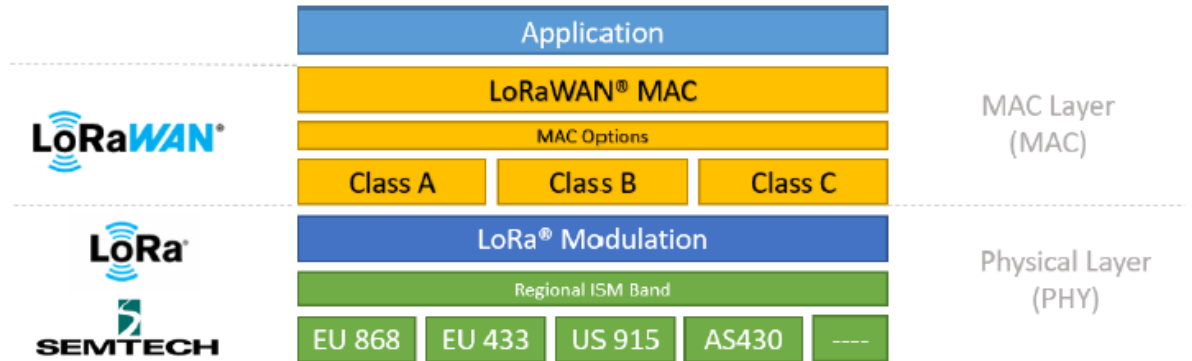


Figure 2.0-11: LoRaWAN Technology Layers [2]

2.3 LoRaWAN – Security Considerations

The LoRaWAN end-device uses a Network Session Key (NwkSKey) for authentication with the Network Server, while an Application Session Key (AppSKey) is employed for encryption between the end-device and the Application Server.

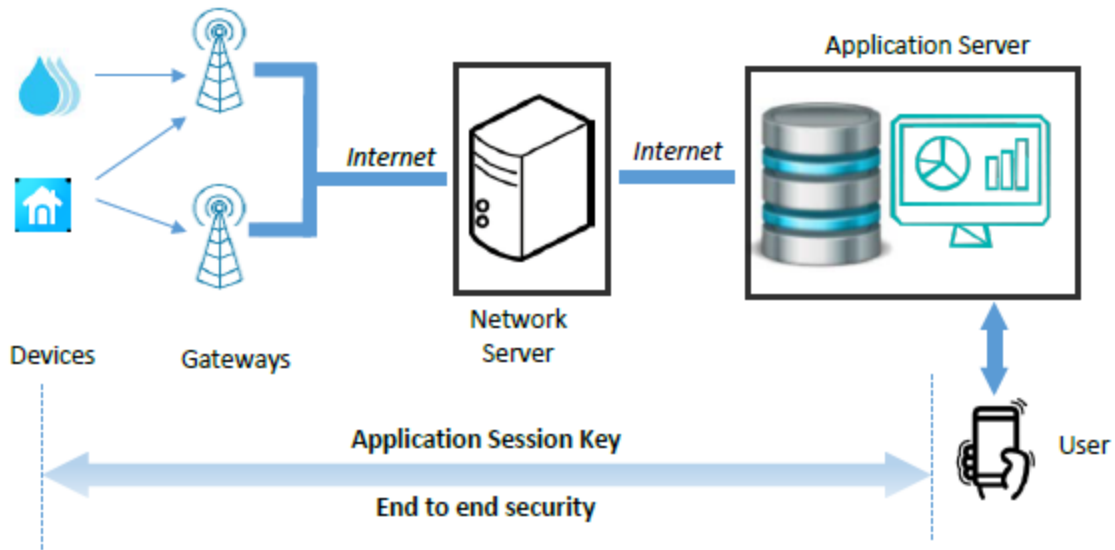


Figure 2.0-12: LoRaWAN End to End Security [2]

2.3.1 Encryption Methodology -LoRaWAN

The AppSKey, known as the Application Session Key, encrypts user data on the LoRaWAN end-device, which is later decrypted on the Application Server. This encryption is symmetric, meaning the AppSKey on the end-device aligns with the one stored on the Application Server. This process ensures that neither the gateway nor the Network Server can access the actual user data, maintaining a secure and confidential channel.

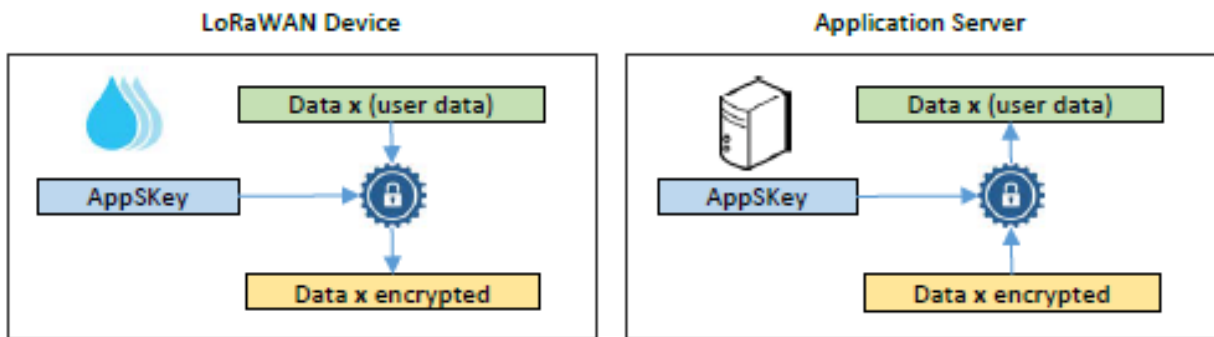


Figure2-0-13: Encryption Methodology[2]

2.3.2 Network Server Authentication – LoRaWAN

The NwkSKey, serving for authentication between the LoRaWAN end-device and the Network Server, incorporates a Message Integrity Control (MIC) field within the frame. This MIC relies on the encrypted transmitted data and the NwkSKey. Upon reception, a similar calculation

is executed. If the NwkSKey matches both on the end-device and the Network Server, the transmitted MIC should align with the one generated during reception.

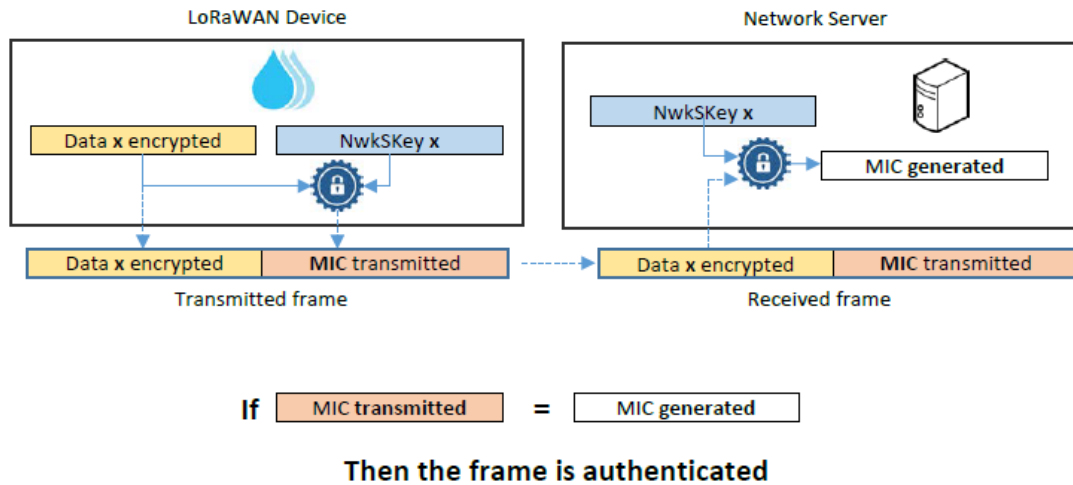


Figure 2-0-14: Network Server Authentication[2]

2.3.3 Merging Authentication and Encryption

In figure 2.15, we can now illustrate the creation of the LoRaWAN frame, integrating both authentication and encryption. On the server end, decryption occurs exclusively upon successful authentication.

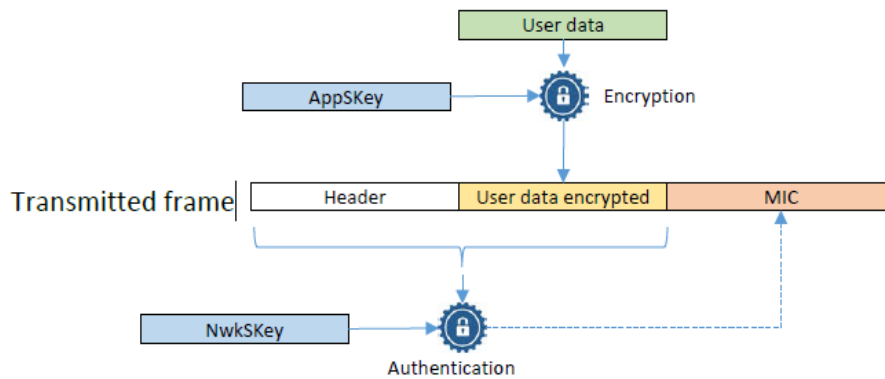


Figure 2.0-15: Encryption then Authentication [2]

2.4 LoRaWAN – Classes of End Devices

LoRaWAN end-devices are categorized into three groups (A, B, C) based on their power usage and the simplicity of sending a frame to the device's downlink capabilities.

2.4.1 Class A

All LoRaWAN devices belong to class A. Each device can transmit data to the gateway (uplink) without checking the gateway's availability. Following this transmission, two very brief reception windows occur. The server, via the gateway, can send a downlink message during either the RX1 or RX2 slot, but not both. The duration of these windows must be at least the length of the preamble (12.25 Tsymbol). Upon detecting a preamble, the receiver remains active until the transmission concludes. If the received frame during the first window was intended for the LoRaWAN device, the second window isn't activated.

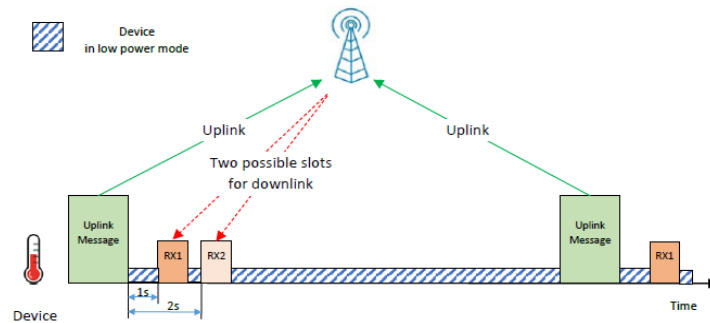


Figure 2.0-16: End device (Class A) receiving slots[2]

The initial reception window, RX1, defaults to 1 second $\pm 20 \mu\text{s}$ after the uplink transmission ends, but this timing can be adjusted based on Network Server settings. It mirrors the channel, Spreading Factor, and bandwidth chosen during the uplink. The second reception window, RX2, defaults to 2 seconds $\pm 20 \mu\text{s}$ after the uplink transmission ends, also customizable via the Network Server. RX2's channel, Spreading Factor, and bandwidth are configurable but fixed. All devices begin and join the network as class A end-devices.

2.4.2 Class B

Class B end-devices operate similarly to class A devices, with the addition of scheduled reception windows at specified times. To synchronize these windows, gateways transmit beacons regularly. Any end-device can opt to switch to class B if its firmware allows. The comprehensive specifications for class B end-devices have been available since the release of the LoRaWAN standard version 1.0.3.

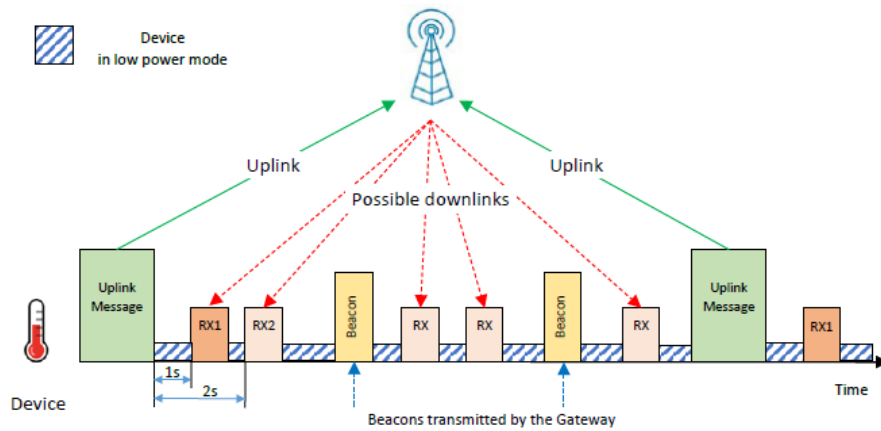


Figure 2.0-17: End device (Class B) receiving slots[2]

2.4.3 Class C

Class C devices maintain open reception windows continuously between two uplink transmissions, resulting in significantly higher power consumption compared to class A and B devices. In this mode, the LoRaWAN end-device continuously listens between uplink messages, with all RX slots configured identically to RX2, except for RX1, which retains the behavior seen in class A and B. Any end-device can opt to transition to class C if its firmware allows for it.

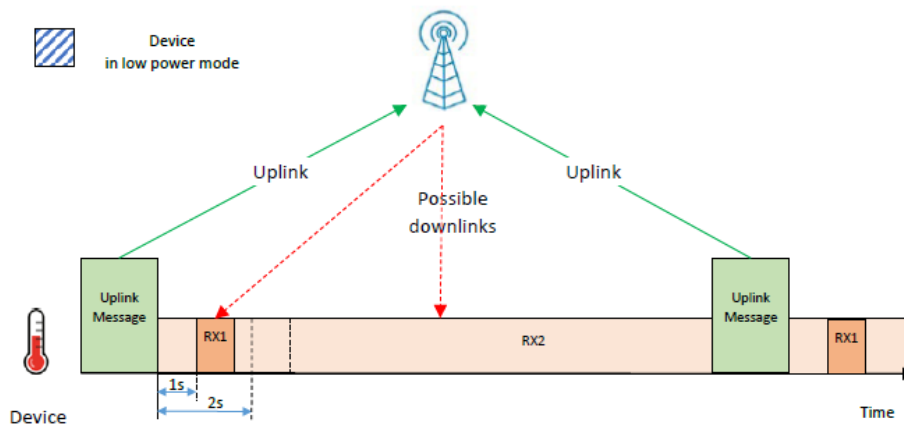


Figure 2.0-18: End device (Class C) receiving slots [2]

Figure 2.19 depicts the graphical representation of end devices classes downlink capabilities and consumption.

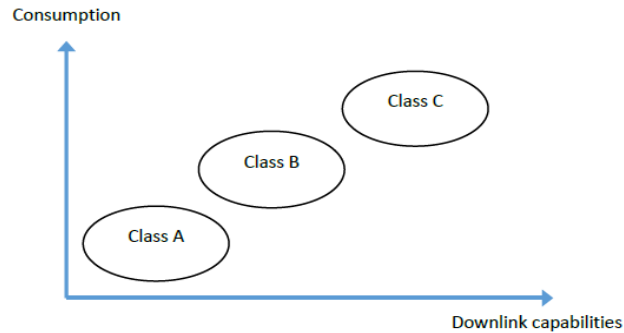


Figure 2.0-19 Classes Power Consumption and Downlink Capabilities[2]

2.5 Enabling LoRaWAN End-Devices: ABP and OTAA Activation

In LoRaWAN, communication relies on three crucial components: the DevAddr for end-device identification and two keys—the NwkSKey for authentication and the AppSKey for encryption. To provide this vital information to both the end-device and the LoRaWAN server, two distinct methods exist. The first, Activation by Personalization (ABP), involves configuring the necessary keys directly into the end-device and LoRaWAN server in advance. The second method, Over the Air Activation (OTAA), dynamically establishes these keys through mutual authentication between the end-device and the LoRaWAN server during network join.

2.5.1 Activation by Personalization

ABP serves as the simplest method, commonly used for testing prototypes and initiating LoRaWAN communication setups. It involves storing static DevAddr, NwkSKey, and AppSKey in both the end-device and the LoRaWAN server. Once configured, the end-device becomes capable of sending and receiving LoRaWAN messages.

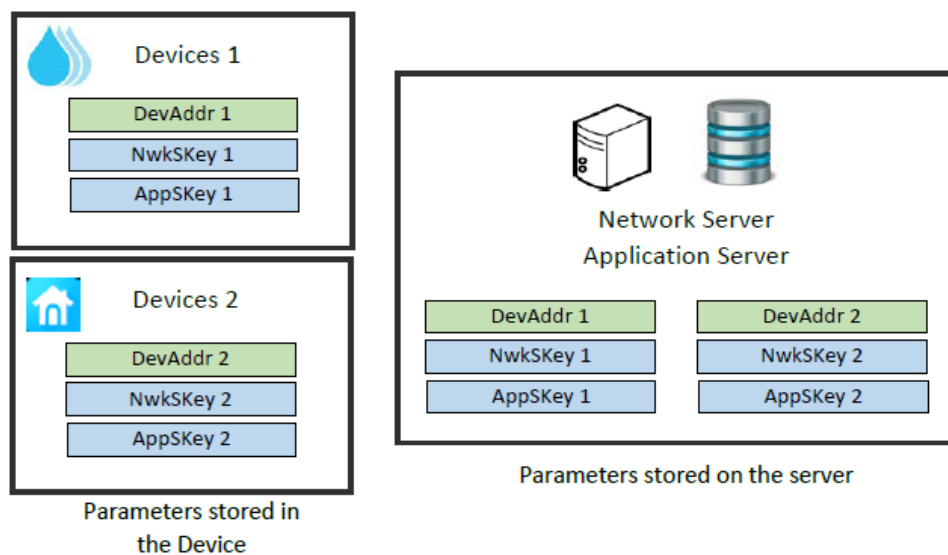


Figure 2.0-20: Parameters Storage in ABP [3]

2.5.2 OTAA- Over the air activation

OTAA, or Over the Air Activation, initiates a Join procedure for LoRaWAN end-devices to establish DevAddr, AppSKey, and NwkSKey during connection to the Network Server. To enable this procedure, the end-device requires configuration with DevEUI, AppEUI/JoinEUI, and AppKey. The Network Server must possess matching DevEUI, AppEUI/JoinEUI, and AppKey. The purpose of the Join-Request is to synchronize DevAddr, NwkSKey, and AppSKey on both ends. Following the Join-Request, these generated parameters are saved on both the end-device and the Network Server.

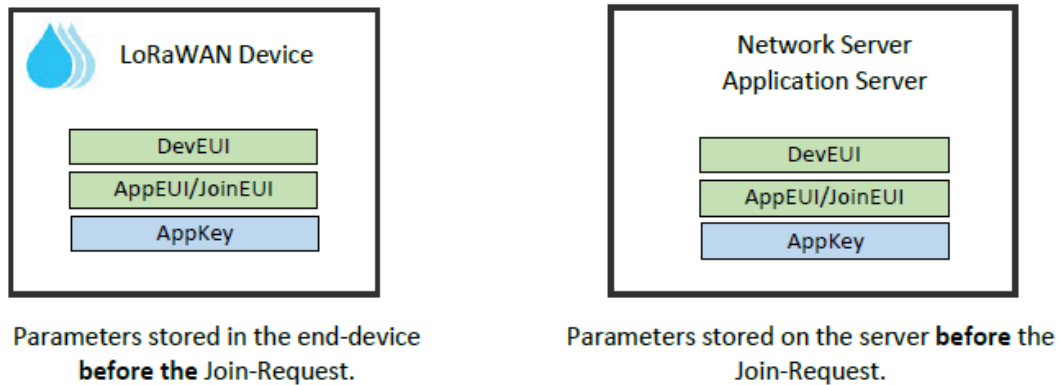


Figure 2.0-21: Parameters stored in OTAA before the join request.

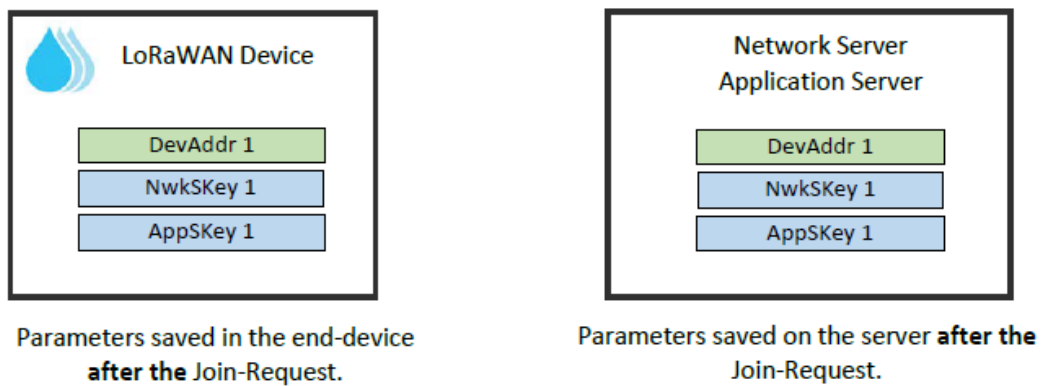


Figure 2.0-22: Parameters stored in OTAA after the Configuration [3]

2.5.3 Join Procedure – OTAA

To better understand the join procedure, one should be aware of the under mentioned terms:

- **DevEUI** serves as a unique identifier for LoRaWAN end-devices, comparable to an ethernet MAC address. Some devices come with a fixed DevEUI set during factory firmware programming, unchangeable thereafter.
- **AppKey**, an AES 128 key, is vital for authenticating Join-Requests, encrypting Join-Accepts, and generating session keys. This key is confidential and should never be shared.
- **AppEUI/JoinEUI** has varied meanings across LoRaWAN versions. In versions up to 1.0.3, it represented an application identifier (AppEUI), while in versions 1.0.4 and beyond, it has been renamed JoinEUI, functioning as a Join Server identifier.

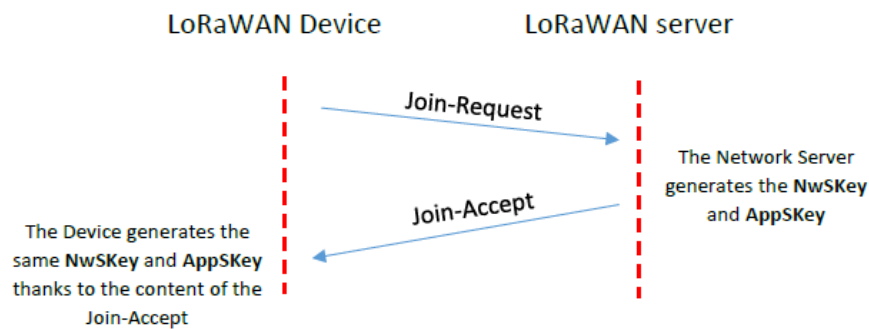


Figure 2.0-23: OTAA join procedure – Simplified [3]

The final configuration post Join procedure must include:

- **NwkSKey**: Authentication key for the Network Server.
- **AppSKey**: Encryption key for data with the Application Server.
- **DevAddr**: 32-bit identifier within the LoRaWAN network.

The Join procedure steps are as follows:

- The LoRaWAN end-device sends a Message Integrity Code (MIC) to authenticate its request, ensuring that only registered end-devices prompt a Join-Accept response from the Network Server. The 4-byte MIC is computed using the AppKey, JoinEUI, DevEUI, and DevNonce. As the Join-Request frame isn't encrypted, the AppEUI/JoinEUI and DevEUI are easily visible in the activity logs of gateways or the Network Server. The DevNonce acts as a random number to thwart replay attacks.
- Upon successful authentication of the end-device, the Network Server creates the NwkSKey and AppSKey.

- Following the Join-Request, a Join-Accept frame is scheduled either 5 seconds (RX1) or 6 seconds (RX2) later. The transfer of NwkSKey and AppSKey doesn't occur directly within this frame. Upon initiation, the end-device stores the DevAddr and NetID as its first pieces of information. It also receives essential parameters from the Network Server for proper communication:

- DownLink Settings (DLSettings)
- Downlink delay (RXDelay)
- Channel Frequency List (CFList) for end-device usage

Additionally, it receives JoinNonce, utilized for recalculating NwkSKey and AppSKey on the end-device. The Join-Accept, encrypted by the AppKey, remains understandable only to the end-device.

- Using the details from the Join-Accept, the end-device generates the NwkSKey and AppSKey. Afterwards, the AppSKey is transferred to the application server.

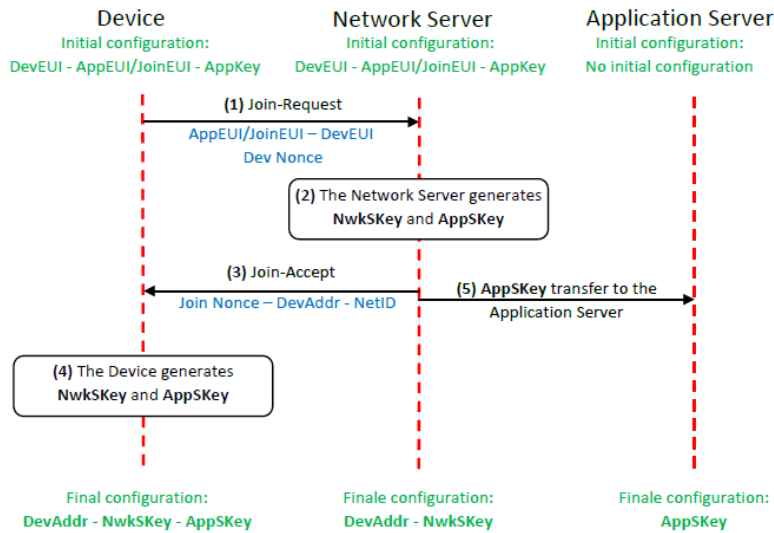


Figure 2.0-24: Join-Request and Join-Accept Procedures in elaboration [3]

2.5.4 Advantages & Disadvantages– OTAA & ABP

Having explored both activation methods, we must now clarify their suitable contexts for usage.

2.5.4.1 Security

Each method has a vulnerability linked to the key permanently stored within the LoRaWAN end-device:

- ABP holds Session keys (NwkSKey and AppSKey).

- OTAA relies on the Root Key: AppKey.

As a result, these keys must be safeguarded within highly secure memory spaces. However, in ABP, where session keys persist indefinitely, there's an increased risk of brute force attacks, particularly if the end-device replays identical sequences frequently or resets in the same manner—this was permissible until LoRaWAN 1.0.3. Furthermore, altering session keys in ABP (for network operator changes, for instance) exposes them to potential visibility, heightening the risk.

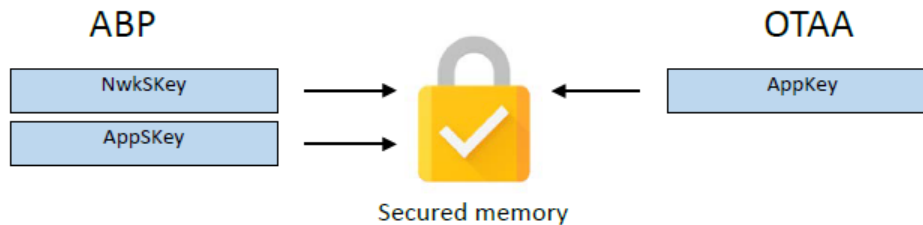


Figure 2.0-25: Keys secured memory storage [3]

2.5.4.2 Network Change

If a client opts to switch LoRaWAN server providers with ABP activation, they'll need to manually transfer all DevAddr and session keys from the current server to the new one. However, there's uncertainty regarding whether the old provider has erased all session keys, potentially allowing continued access to transmitted packet content—a significant security concern. In contrast, OTAA provides an alternative by incorporating a Join Server. While its full role will be elaborated upon later, initially, the client simply informs the Join Server about the transition to another Network Server. The root keys (AppKey) remain secure, requiring no movement. Only the session keys need regeneration through a new Join-Request process.

2.5.4.3 Protection against replay attacks in uplink messages

Despite the encryption provided by the 128-bit AES keys, a prevalent attack in wireless technology is the replay attack. In this scenario, a hacker records encrypted frames sent over the LoRa network and retransmits them at a later time. Although the hacker might not comprehend the encrypted content within the frames, the Application Server easily interprets the transported data. Consequently, simply replicating a previous frame can enable the execution of actions.

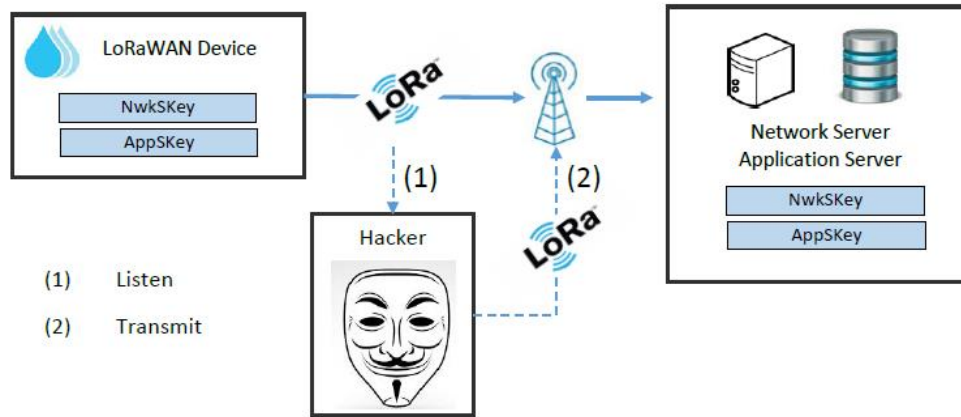


Figure 2.0-26: Reply attack in uplink messages[3]

To prevent this, the LoRaWAN frame includes a variable called the frame counter, which automatically increases each time a new frame is transmitted by the end-device. A similar counter exists on the server's side, incrementing upon receiving a valid frame.

- For validation, the server only accepts a frame if its frame counter exceeds the last valid frame counter received from that specific device.
- If a hacker replays a recorded frame, the frame counter received at the server will be lower than or equal to the server's counter. In such cases, the server silently rejects the frame.

If the hacker attempts to modify the frame counter with a random value, authentication fails because altering the frame counter invalidates the calculation of the MIC field (using the Network Session Key).



Figure 2.0-27: Reply attack avoidance through frame counter[3]

The frame counter is effective against replay attacks, but during LoRaWAN end-device development, it poses challenges. Whenever the microcontroller restarts, the device's frame counter resets to zero, while the server counter continues incrementing. Several solutions can address this issue:

- Implement the option to "Reset Frame Counter" on the end-device. Some LoRaWAN servers permit the acceptance of frames with any counter values, albeit posing security risks.

- Opt for OTAA activation instead of ABP. With each OTAA Join, both the end-device and server counters reset, resolving the synchronization.
- Store the frame counter value in non-volatile memory, allowing retrieval upon LoRaWAN end-device restarts.

2.5.4.4 Protection against replay attacks in downlink messages

Another frame counter is utilized for downlink messages. Specifically, a server-side frame counter increases every time the server dispatches a downlink message to the end-device. For successful acceptance, the end-device requires the received downlink message's frame counter to match or exceed its own frame counter value.

2.5.4.5 Protection against replay attacks in join-request messages

During the OTAA Join-Request transfer, a hacker can exploit a replay attack, where a counter known as DevNonce serves the same safeguarding purpose. In LoRaWAN specification version 1.0.4, an end-device in OTAA is mandated to store this number in non-volatile memory. Failure to do so results in rejection of the Join-Request in case of an end-device reset.

2.6 LoRa - LoRaWAN Frames

2.6.1 LoRaWAN Frame Classifications

LoRaWAN end-devices facilitate two types of frames: uplink, sent by the device, and downlink, received by the device. To ensure reliability, two frame categories exist: unconfirmed, lacking data acknowledgment from the Network Server, and confirmed, with data acknowledgment sent by the Network Server. This behavior aligns with downlink connections between the Network Server and the end-device: unconfirmed frames result in no data acknowledgment from the end-device, while confirmed frames prompt data acknowledgment sent by the end-device.

2.6.2 LoRaWAN Protocol layers

LoRa serves as a modulation technique facilitating data transfer between endpoints, forming the physical layer known as LoRa PHY. The subsequent layer, LoRa MAC, is built atop the LoRa protocol, incorporating features such as end-device authentication, data encryption, acknowledgment, and network administration within the LoRaWAN protocol. Beyond these protocol attributes, the application layer primarily encapsulates raw user data, supplemented by additional services outlined in the LoRaWAN specification.

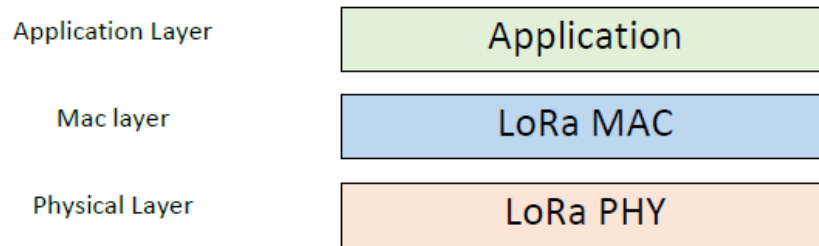


Figure 2.0-28: LoRaWAN Protocol Stack[3]

Each layer contributes specific functionalities, wherein the user data is encapsulated within each lower layer when transmitting a frame. The comprehensive breakdown of the entire LoRaWAN frame per layer is illustrated in Figure 2.29.

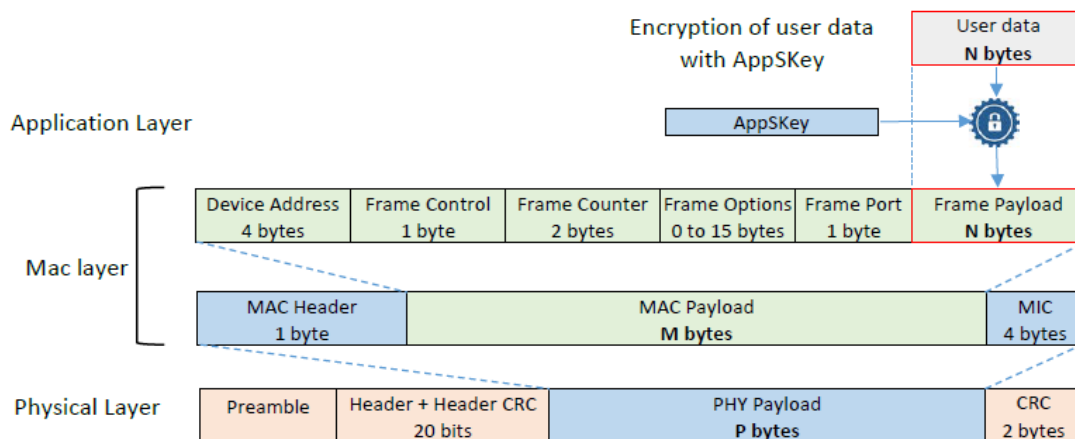


Figure 2.0-29: Layer wise frame distribution – LoRaWAN[3]

2.6.2.1 Application Layer

The application layer primarily comprises the user's data, encrypted using the AppSKey before encapsulation into the LoRaWAN frame for secure transmission. The payload's organization is flexible within the overall size limit of a LoRaWAN frame. The LoRaWAN specification introduces three services at the application layer.

- **Remote multicast Setup (port 200):** Allows frame transmission to a group of end-devices.
- **Fragmented data block transport (port 201):** Facilitates sending fragmented data to one or multiple end-devices.
- **Clock synchronization (port 202):** Enables precise time synchronization between an end-device's clock and the network GPS clock with second accuracy.

- **Firmware management (port 203):** Facilitates firmware version management requests for end-devices.

These packages collectively support FUOTA (Firmware Update Over The Air).



Figure 2.0-30: LoRaWAN application layer[4]

2.6.2.2 LoRa MAC Layer

At the core of the LoRaWAN protocol lies the LoRa MAC layer, pivotal in governing various message types such as Join-Request, Join-Accept, data up, data down, confirmed, and unconfirmed. Within this layer, key components include the DevAddr, defining the device address; the MAC Header specifying message types and options like ADR (Adaptive Data Rate), acknowledgment, and optional "Frame Option" length. Additionally, the frame option field encompasses potential MAC Commands, the Frame Port designates the application port, the frame payload holds encrypted user data, and the MIC (Message Integrity Control) authenticates the message for verification by the Network Server.

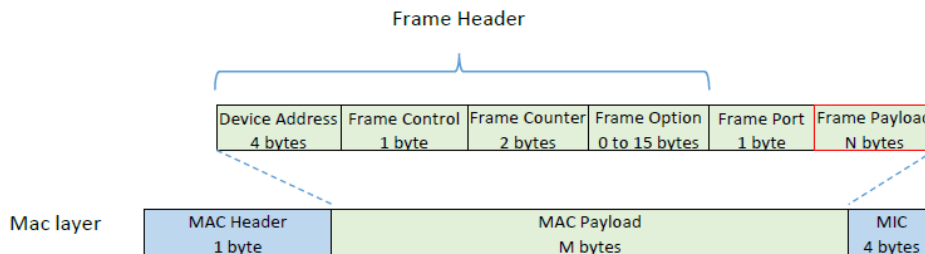


Figure 2.0-31: LoRaWAN MAC Layer [4]

Table 1.2: Maximum Frame Payload Size of respective data rate

Data Rate	Spreading Factor	Bandwidth	Max Frame Payload (Number N)
DR 0	SF12	125 kHz	51 bytes
DR 1	SF11	125 kHz	51 bytes
DR 2	SF10	125 kHz	51 bytes
DR 3	SF9	125 kHz	115 bytes

DR 4	SF8	125 kHz	242 bytes
DR 5	SF7	125 kHz	242 bytes

The table above presents the maximum bytes allowed for transmission as a MAC payload (M bytes).

2.6.2.2 LoRa Physical Layer

Sending a LoRaWAN message is straightforward as it doesn't require synchronization between the end-device and the gateway. Hence, the PHY layer is minimal, comprising a preamble, an optional header (present in default mode), and a CRC. The preamble consists of 8 symbols totaling 12.25 Tsymbol (as defined in chapter 3.1). The optional header, transmitted with a 4/8 Coding Rate, denotes data size, Coding Rate for the remaining frame, and the presence of a CRC at the frame's end. The PHY payload encapsulates all LoRa MAC Layer information, while the CRC is utilized for error detection within the LoRa frame.

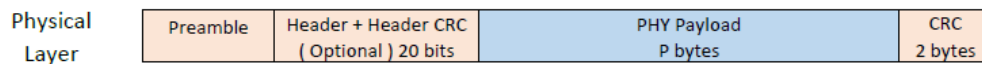


Figure 2.0-32: Frame of physical layer of LoRaWAN [4]

2.7 LoRaWAN Power Consumption

The power consumption of a LoRaWAN end-device is directly tied to two key parameters of LoRa transmission: Time on Air and transmitted power (PT). Longer messages result in prolonged radio activity, consuming more power, while higher transmission power escalates the device's power usage. Reducing transmission power affects the device's reach to the gateway, considering the gap between received power and gateway sensitivity. Lowering Time on Air is achievable by decreasing the Spreading Factor (SF), halving it for each SF reduction by one. Yet, decreasing SF notably decreases gateway sensitivity, impacting its ability to detect signals amidst noise. Balancing SF and PT adjustments is crucial, as reducing both diminishes the transmission range. Determining their values often involves an empirical approach, assessing SNR and RSSI on the gateway, or leveraging Adaptive Data Rate (ADR) for optimization.

2.8 LoRaWAN MAC Commands

Within network administration, a series of MAC commands can be exchanged between the Network Server and the end-device. These commands, constituting the LoRaWAN stack, are not intended for access by the user application, ensuring they remain exclusive to the end-device and Network Server interaction. Specifically, the Application Server must not receive these commands, and the user application within the end-device should remain uninformed of their existence. Each MAC Command possesses a unique identifier known as CID (Command Identifier). LoRaWAN MAC Commands (as per LoRaWAN 1.0.4 spec – LoRa Alliance) are referenced and accessible from the aforementioned source [6].

2.9 Adaptive Data Rate for LoRaWAN

Adjusting the Spreading Factor (SF) and Power Transmitted (PT) isn't a straightforward task. Even with an optimal configuration, local conditions or weather forecasts can disrupt transmissions. To tackle this challenge, the LoRaWAN standard incorporates an automatic adjustment method known as Adaptive Data Rate (ADR). This approach delegates the computation of the best SF/PT combination to the Network Server. This computation involves several checks:

- RSSI (Received Signal Strength Indication) against receiver sensitivity, with a set margin.
- SNR (Signal-to-Noise Ratio) of the transmission above the receiver's minimum acceptable SNR, also applying a margin.
- Estimation of packet loss based on frame counter analysis from recent transmissions.

The Network Server computes results by averaging RSSI and SNR across multiple uplink frames. Notably, the specifics of the ADR algorithm aren't outlined in the LoRaWAN specification, allowing each Network Server to adopt its unique strategy. Typically, the under visualized methodology is used:

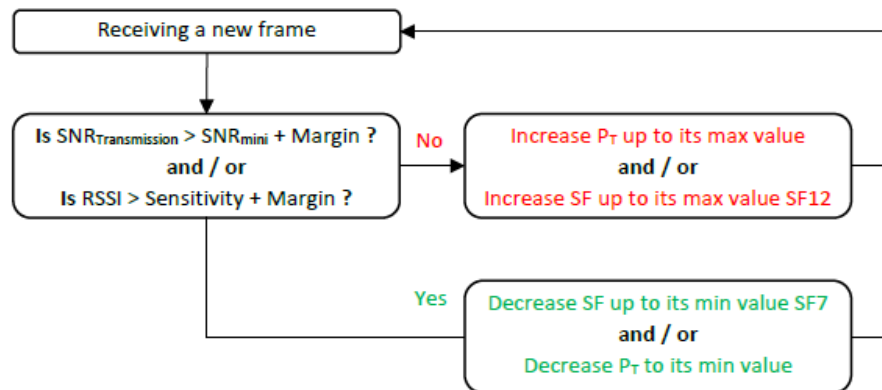


Figure 2.0-33: ADR Procedure [4]

For the ADR mode to function, a LoWAN device needs to activate the ADR Flag.

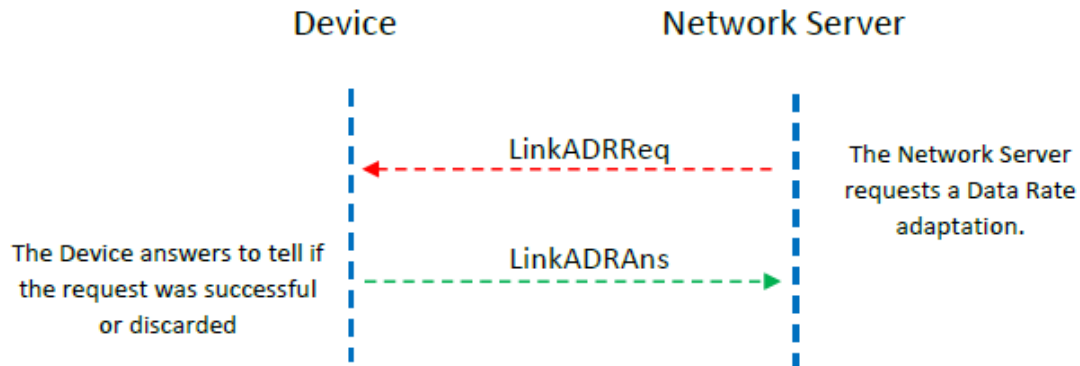


Figure 2.0-34: MAC commands for ADR[4]

For static end devices, ADR is managed by the network server using Network-managed ADR or Static ADR, based on uplink packet history.

For mobile end devices, ADR is performed by the device itself due to unpredictable channel attenuation, known as Blind ADR. In which, instead of utilizing a singular data rate, we employ three distinct data rates with varying periodicity based on each data rate [9].

2.10 LoRaWAN Alliance

The LoRa Alliance®, established in 2015 as a non-profit organization, focuses on advancing LoRaWAN technology and its ecosystem. Comprising global companies committed to its growth, the alliance welcomes applications from any organization seeking LoRaWAN development involvement.

Regarding the evolution of the LoRaWAN standard, its versions and their progressions are as follows:

- Version 1.0.0 (January 2015): Initial LoRaWAN specification.
- Version 1.0.1 (February 2016): Introduced new frequency plans for China and Australia and addressed minor clarifications and corrections.
- Version 1.0.2 (July 2016): Physical layer section separated into "LoRaWAN Regional Parameters," marking the first stable release.
- Version 1.1 (October 2017): Enhanced security and roaming, integrating new root keys, session keys, frame counters, and MAC Commands. JoinEUI replaced AppEUI and provided clarity on class B and class C devices.
- Version 1.0.3 (July 2018): Integrated the class B section of version 1.1 into version 1.0.2.

- Version 1.0.4 (October 2020): Aligned AppEUI with JoinEUI, offered numerous clarifications, and represented the final 1.0.x version.

Upon releasing a specification, the LoRa Alliance provides the LoRaWAN regional parameters document. This companion publication outlines specific parameters for regulatory regions worldwide, including channel frequencies, data rates, output power, maximum payload sizes, etc. Separating these regional parameters from the protocol specification ensures the addition or modification of regions without affecting the core protocol details, like end-device classes, message and frame formats, MAC commands, activation modes (ABP, OTAA), among others.

All versions of the LoRaWAN specification are accessible on the LoRa Alliance resource HUB [7].

CHAPTER 3: EVALUATING LORAWAN SIMULATORS: A COMPREHENSIVE REVIEW AND COMPARATIVE ANALYSIS

3.1 Purpose of Simulations in LoRaWAN Networks

Simulation in LoRaWAN networks is crucial for multiple technical reasons. It allows for performance evaluation by assessing scalability limits with varying device numbers, measuring throughput under different conditions, and testing latency critical for time-sensitive applications. In network planning and optimization, simulations predict radio coverage for optimal gateway placement, determine maximum device capacity without performance degradation, and optimize resource allocation. They facilitate protocol and algorithm testing by evaluating MAC layer protocols for collision avoidance, comparing routing algorithms for efficient data transmission, and analyzing security protocols' impact on performance. Interference and collision analysis benefits from interference modeling to assess the impact of other networks and collision detection to evaluate packet integrity and throughput. Energy consumption evaluation estimates battery life based on communication patterns and measures power consumption during transmission, reception, and idle periods. Simulations ensure protocol compliance and standardization by testing adherence to LoRaWAN standards and validating new or modified protocols. Fault tolerance and reliability are tested by simulating failure scenarios to study network robustness and measuring metrics like packet delivery ratio and uptime. Finally, traffic pattern analysis through simulations models different traffic loads to understand their impact on performance and evaluates Quality of Service (QoS) parameters to ensure reliable data transmission.

3.2 Importance of Simulators in LoRaWAN Research

LoRaWAN simulators are considered an important arsenal when it comes to research as it can be used for testing and validation, it is cost effective and scalability factor can be addressed while not compromising on the quality of the research.

3.2.1 Testing and Validation

Simulators enable comprehensive testing of network performance metrics such as throughput, latency, and packet delivery ratio under controlled and repeatable conditions, allowing researchers to validate the behavior of LoRaWAN networks under various configurations and scenarios without the unpredictability of physical deployments. They provide a platform for testing and validating new protocols and algorithms, such as evaluating MAC layer protocols for efficiency in collision avoidance and assessing new routing algorithms for their impact on data transmission efficiency. Additionally, by simulating a large number of nodes, researchers can test how the network performs as the number of connected devices increases, helping to identify potential bottlenecks and scalability issues in the network architecture.

3.2.2 Cost Effectiveness

Using simulators eliminates the need for extensive physical hardware deployments, which can be expensive and logistically challenging, particularly for large-scale network studies where setting up numerous devices and gateways would incur significant costs. Simulators allow for the efficient use of computational resources to model and test various network configurations, reducing the need for physical components such as multiple gateways and end devices, and minimizing costs associated with hardware maintenance and upgrades. Additionally, simulators accelerate the development and testing cycle by providing immediate feedback on network performance, thus reducing the time and cost associated with iterative physical testing and debugging.

3.2.3 Scalability

Simulators can model and analyze networks with thousands of nodes, which would be impractical to deploy physically, aiding in understanding the behavior of LoRaWAN networks in large-scale deployments like smart cities or widespread IoT applications. They can easily adjust network parameters to test various deployment scenarios, such as different node densities, mobility patterns, and environmental conditions, providing insights into how the network would perform under diverse real-world conditions without the need for multiple physical setups. By simulating different scales of network deployments, researchers can plan for the optimal number of gateways and nodes required to maintain performance standards, helping determine the upper limits of network capacity and ensuring efficient resource allocation for future expansions.

3.3 LoRaWAN based simulators comparison

This section provides an overview of commonly used open-source simulation tools focused on LoRa/LoRaWAN is presented in references [13–16]. The most widely used tools include LoRaSim, NS-3, OMNeT++ (FLoRa), CupCarbon, PhySimulator, SimpleIoTSimulator, and Mbed OS Simulator. Table 8 provides a comparison of IoT simulators for LoRa/LoRaWAN, detailing their programming languages, target domains (whether network generic or LoRa/LoRaWAN specific), supported operating systems, and the presence of a graphical user interface (GUI).

Table 3.1: Comparison of LoRaWAN Simulators

Ref	Simulation Type	Language	Target Domain	Operating System	GUI
[17]	LoRaSIM	Python	Specific	Linux, macOS, Windows	No
[18-21]	NS-3	C++, Python	Generic, specific	Linux, Windows	No
[22]	OMNeT++ (FLoRa)	C++	Generic, specific	Linux, macOS, Windows	Yes

[23]	CupCarbon	Java, SenScript	Zigbee, WiFi, LoRa radio	macOS	Yes
[24]	PhySimulator	MATLAB	Specific	macOS, Windows	No
[25]	LoRaFREE	Python	Specific	Linux, macOS, Windows	No
[26]	LoRaEnergySim	Python	Specific	Linux, macOS, Windows	No
[27]	LoRaWANSIM	MATLAB	Specific	Linux, macOS, Windows	No
[28]	TS-LoRa	Micropython	Specific	Linux, macOS, Windows	No
[29]	LoRaWAN-SIM	Perl	Specific	Linux, macOS, Windows	No
[30]	LoRaMACSim	Python	Specific	Linux, macOS, Windows	No
[31]	LoRa-MAB	Python	Specific	Linux, macOS, Windows	No
[32]	LoRaWANSim	Python	Specific	Linux, macOS, Windows	No
[33]	LoRaPlan	Python	Specific	Linux, Windows	Yes
[34]	AFLoRa	C++	Specific	Linux, macOS, Windows	Yes

3.4 LoRaWAN Simulators survey

For this study, we will focus on simulation tools that support the LoRa/LoRaWAN framework for conducting network simulations. Specifically, we have selected LoRaWAN-SIM, NS-3, OMNeT++ (FLoRa), LoRaSim and LoRaSF for our analysis. The rationale for this selection is as under:

- Free availability for academic and research use.
- Active development by practitioners and researchers.
- Availability of comprehensive documentation.
- Suitability for IoT and WSN applications.
- Increasing popularity among academics and researchers for LoRa/LoRaWAN simulations.

3.4.1 Hardware Specifications

The simulators will run on a HP laptop equipped with an Intel Core i5-7200U processor running at 2.50GHz, 16GB of RAM, and Windows 10 as the operating system. The components description of the device is as under:

Table 3.2: Hardware Specifications for Simulations

Components	Description
Processor	Intel(R) Core (TM) i5-7200U CPU @ 2.50GHz
Operating System	Windows 10
RAM	16 GB
System Type	64-bit operating system, x64-based processor
Graphics Card	NVIDIA GeForce MX150 (2GB GDDR5 VRAM)
Storage	512GB PCIe NVMe M.2 SSD
Display	15.6-inch Full HD (1920 x 1080) IPS display, touch
Battery Life	3-cell 52.5Whr Lithium-ion battery, upto 8 hours

3.4.2 LoRa SF – Simulator

A discrete event simulator written in Python has been developed to analyze the effects of different spreading factor strategies in LoRaWANs. The source code for this LoRaWAN spreading factor simulation tool is open source and available on GitHub [30]. The simulator supports both custom and randomly generated LoRaWAN topologies.

It can create a uniformly distributed circular network topology based on the following input parameters: radius (meters), number of nodes, and number of gateways. Global simulation parameters include simulation duration (seconds), packet size (bytes), packet generation rate (packets per second), packet generation type, and spreading factor assignment method.

Using these inputs, the simulator outputs the total number of generated packets, successfully received packets, interfered packets, packets below sensitivity, network packet delivery ratio (PDR) percentage, network throughput (bps), and total transmit energy consumption (Joules). Additionally, it provides prediction accuracy percentage and a confusion matrix for machine learning schemes.

Initially, the tool positions the gateways according to the specified input, distributing them homogeneously to maximize their separation. Subsequently, end nodes are randomly placed within the network radius.

The simulation tool focuses exclusively on LoRaWAN Class A devices, as Class A is the default operation mode and ensures the lowest power consumption. Transmissions are initiated by end nodes using a pure ALOHA method, with nodes generating new packets based on a specified packet rate parameter. Packets can be generated either at Poisson intervals or periodic intervals.

3.4.2.1 Communication Link Model

The communication link budget evaluates the wireless link quality between end nodes and gateways. It is calculated using the formula:

$$P_{dBm}^{RX} = P_{dBm}^{TX} + G_{dB}^{SYS} - L_{dB}^{SYS} - L_{db}^{PATH} \quad (1)$$

Where P_{dBm}^{RX} is the received power at the receiver, P_{dBm}^{TX} is the transmitted power of the transmitter, G_{dB}^{SYS} are the combined gains of the transmitter and the receiver, L_{dB}^{SYS} are the losses attributed to system components like lines and circuits and L_{db}^{PATH} are the losses due to propagation over the path between transmitter and receiver. orthogonality of spreading factors (SF), limiting transmissions to a single channel in the simulator. In the simulator utilized, the combined effect of system gains and losses is assumed to be +7 dB, with a maximum transmit power of 14 dBm compliant with the European ISM band regulations. The free space propagation loss is mathematically described by the formula:

$$P_{dB}^{PATH} = 40 (1 - 4 \times 10^{-3} \times h) \log_{10}(R) - 18 \log_{10}(h) + 21 l \log_{10}(f) + 80 \quad [2]$$

Here, h represents the gateway altitude, f denotes the signal frequency (assumed to be 868 MHz), and R is the radius of the network. Given the assumption of spreading factor (SF) orthogonality, only single-channel transmissions are employed. Additionally, specific parameters such as $h=15$ and $f=868$ with a 125 kHz bandwidth for receive sensitivities are considered. When the received signal power exceeds the gateway sensitivity threshold and no interfering transmissions occur, the receiver can successfully decode the signal.

3.4.2.2 Interference Model

In the simulation described in [7], to model LoRa interference between simultaneous but different spreading factor (SF) LoRa signals, a signal to interference plus noise ratio (SINR) based threshold matrix is utilized. This matrix determines whether another signal interferes with a reference signal at the receiver. The threshold matrix, denoted as $T_{i,j}$, represents the SINR margin in dB between the reference signal with SF i and the interfering signal with SF j required to correctly decode the reference signal. If multiple interfering signals are present, the reference signal must meet the SINR margin considering the total received power of all interfering signals for each SF [24]. The calculation of the SINR threshold is outlined as follows [3]:

$$SINR_{i,j} = \frac{P_{rc,0}}{\sum_{l \in I_j} P_{rc,l}} \quad [3]$$

The received power of the reference signal ($P_{rc,0}$) and the received power of the interfering signal ($P_{rc,l}$) for the j th spreading factor (SF) are considered. The packet with SF i can successfully withstand all interference if it meets the following condition for each SF j :

$$SINR_{i,j}^{db} > T_{i,j} \quad (4)$$

3.4.2.3 Software Used

The discrete event simulation tool is entirely developed in Python, specifically designed to simulate advanced spreading factor schemes with integrated machine learning libraries. Unlike existing LoRa/LoRaWAN simulation tools such as ns-3, which would require significant

modifications to incorporate smart spreading factor schemes, this tool was built from scratch to facilitate easier integration.

The tool provides a framework for simulating LoRaWAN networks, maintaining a transmission queue where each event represents a transmission. Each transmission event includes fields for time, spreading factor, source, size, duration, and status. End nodes generate events based on their traffic generation rate and add these events to the simulation queue. Initially, all events are marked as pending. The simulation tool processes these events, updating their status to transmitted, interfered, or under sensitivity. After executing all events, the tool calculates the network's packet delivery ratio (PDR), throughput, and transmit energy consumption.

A command-line parser allows users to interact with the framework without needing programming or scripting skills. Additionally, an example script is provided to demonstrate how to use the framework. The Python scikit-learn machine learning library is used for implementing smart spreading factor schemes, and the Python matplotlib library is utilized to generate figures for this thesis.

The source files of the simulation tool and their primary functions are outlined below:

- **main.py**: This is the command-line interface of the simulator. It parses simulation inputs, executes the simulation, and reports the results.
- **simulation.py**: This file contains the core simulation methods and classes. The primary function, **run**, executes the simulation steps.
- **packet.py**: This file defines classes related to LoRa packet information. It includes methods for calculating transmission duration, receive sensitivity, propagation loss, and transmission energy. It also contains the SNIR matrix and packet status enumeration types.
- **node.py**: This file defines classes related to end nodes and gateways, including methods for generating node traffic.
- **topology.py**: This file contains information about the LoRaWAN network topology, such as node and gateway locations, and includes a method for generating random topologies.
- **location.py**: This file defines the Location class, which stores the x and y coordinates of nodes or gateways.

3.4.2.4 Installation

The simulation tool was developed and tested using Python 3.x [34]. It requires specific Python modules, all of which can be installed using the "pip" package manager. To install a new Python module, use the command "pip install". The required modules for this simulation tool are:

- matplotlib
- numpy
- scipy
- scikit-learn

3.4.2.5 Command Line Interface

Below are examples of how to use the command line interface of the simulation tool:

- To show help, use the command **python3 main.py -h**.
- To set the topology radius in meters, use the command **python3 main.py -r 5000**.
- To specify the number of gateways, use the command **python3 main.py -g 3**.
- To specify the number of nodes, use the command **python3 main.py -n 300**.
- To set the spreading factor assignment method, use the command **python3 main.py -s SF_Lowest**.
- To use the smart spreading factor assignment classifier, use the command **python3 main.py -s SF_Smart -c DTC**.
- To set the simulation duration in seconds, use the command **python3 main.py -d 3600**.
- To specify the packet rate in packets per second, use the command **python3 main.py -p 0.02**.
- To specify the packet size in bytes, use the command **python3 main.py -z 80**.
- To set the proportions of different traffic generator type nodes, use the command **python3 main.py -o 0.8 0.2**.
- To set the random number generator seed, use the command **python3 main.py -e 42**.
- To specify the events log path, use the command **python3 main.py -l events.txt**.
- To set the verbose level, use the command **python3 main.py -v INFO**.

A complex example that sets multiple parameters can be executed with the command **python3 main.py -r 7000 -g 3 -n 300 -s SF_Lowest -d 3600**.

3.4.2.6 Simulation Results

The global simulation parameters are defined as follows: a packet size of 60 bytes, a simulation duration of 3600 seconds, and Poisson traffic generation. Initially, the results of single and multiple gateway LoRaWAN network simulations are presented to validate the simulator's accuracy. Following this, the simulation results for smart spreading factor schemes are discussed.

In Figure 3.1, the PDR plots for various spreading factor assignment schemes are displayed. The randomly generated network topology has a radius of 3000 meters, with one gateway and a packet generation rate of 0.01 pps. As the spreading factors increase, airtime also increases, leading to more collisions and a decreased PDR. High spreading factor schemes result in poor PDR outcomes as the number of nodes grows. Given the relatively small network topology radius, all spreading factors are capable of reaching the gateway.

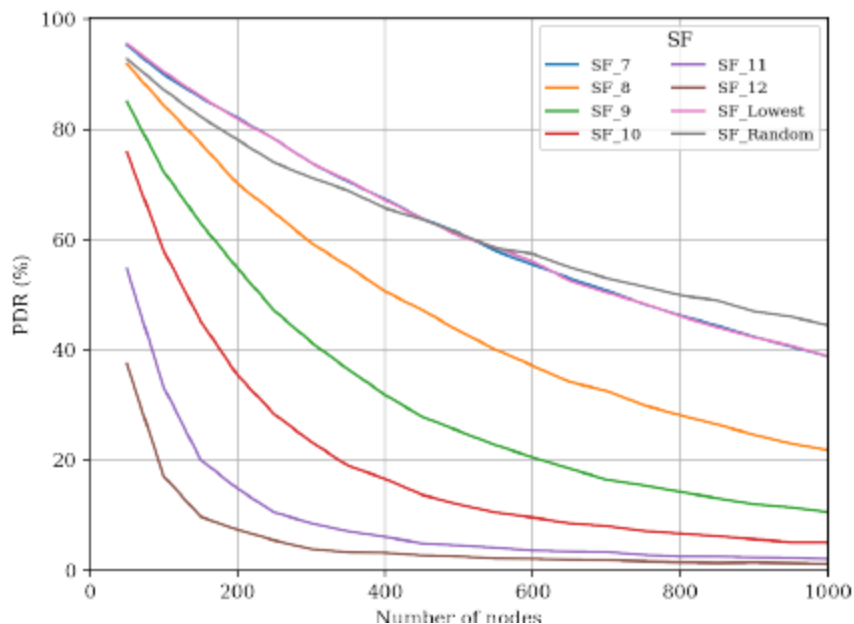


Figure 3.0-1: PDR of SFs – Single Gateway

In Figure 3.2, the PDR plots for various network radii are shown. The network configuration includes one gateway, the lowest spreading factor assignment scheme, and a packet generation rate of 0.01 pps. As the network radius increases, the number of under-sensitivity transmissions rises, resulting in a decreased PDR. Nodes farther from the gateway select higher spreading factors, which leads to longer airtime and a higher number of collisions.

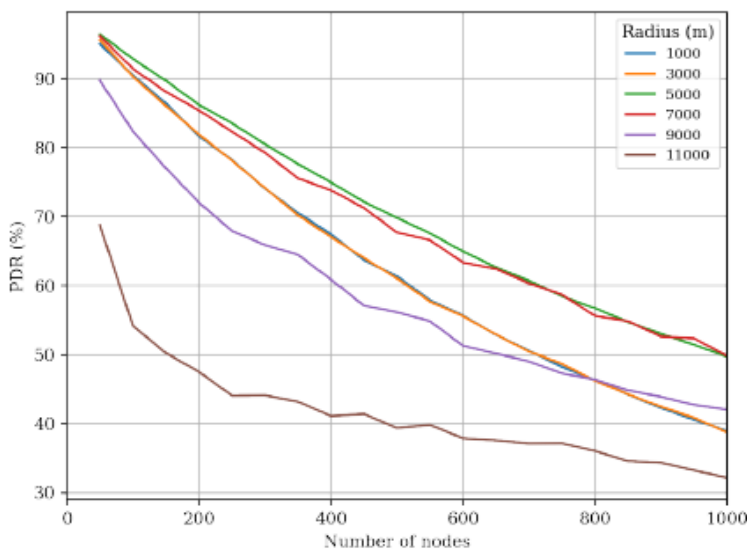


Figure 3.0-2: PDR of various Radii - Single Gateway

In Figure 3.3, the PDR plots for various packet generation rates are shown. The network configuration includes a randomly generated topology with a radius of 3000 meters, one gateway, and the lowest spreading factor assignment scheme. Increasing the packet generation rate increases airtime, which in turn raises the number of collisions and decreases the PDR of the network. Doubling the packet generation rate results in a significant impact on network performance.

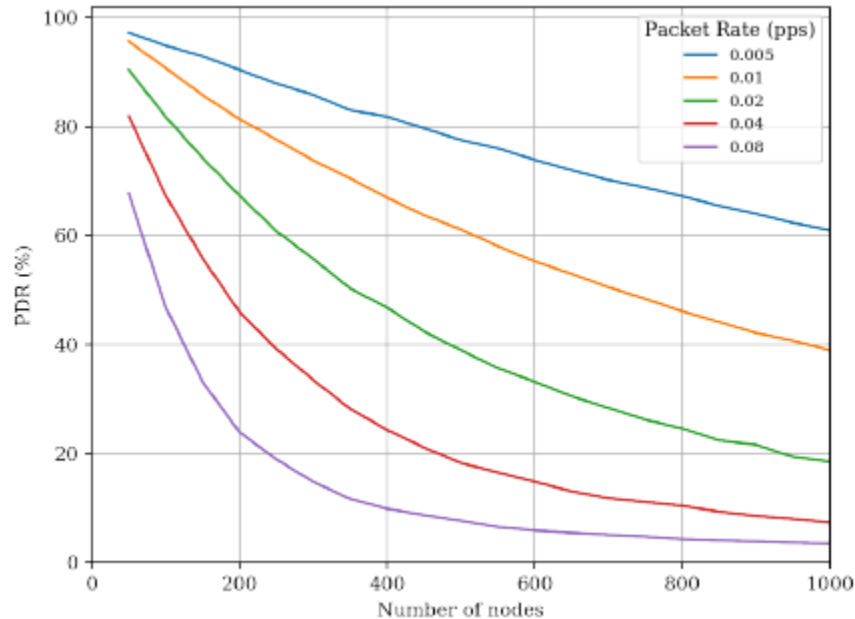


Figure 3.0-3: PDR of various Packet rates- Single Gateway

In Figure 3.4, the PDR (Packet Delivery Ratio) for various numbers of gateways is depicted. The topology radius is fixed at 3000 meters, and the lowest spreading factor assignment scheme is applied with a packet generation rate of 0.01 pps. As the number of gateways increases within this constant network radius, the distance between nodes and gateways decreases. This reduction in distance leads to fewer collisions at the receivers (gateways), thereby enhancing the network's PDR. Given the small topology radius, most end nodes consistently select the lowest spreading factor, resulting in the same airtime across all plots.

3.4.2.7 Conclusion

The simulation results validate the correctness of the simulator, demonstrating that optimizing spreading factors, managing network radius, controlling packet generation rates, and increasing the number of gateways is critical for improving Packet Delivery Ratio (PDR) and overall network performance in LoRaWANs. Higher spreading factors and packet generation rates lead to increased airtime and collisions, decreasing PDR. Larger network radii also reduce PDR due to more under-sensitivity transmissions. However, adding more gateways within a fixed network radius decreases the distance between nodes and gateways, reducing collisions and enhancing PDR. These findings highlight essential strategies for designing efficient and reliable LoRaWAN networks.

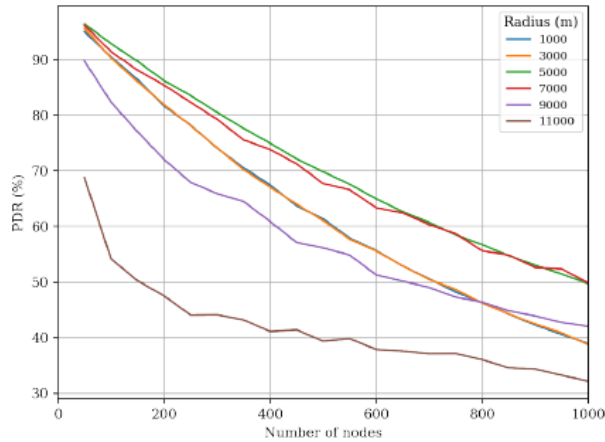


Figure 3.0-4: PDR of various radii- Multiple Gateway

Figure 3.5 presents the PDR (Packet Delivery Ratio) plots for multiple gateways is given such as when there is one gateway and subsequently increasing to 4 gateways. It is evident that one there is single gateway the PDR decreases as the number of nodes increases and this decrease becomes less when the number of gateways increases.

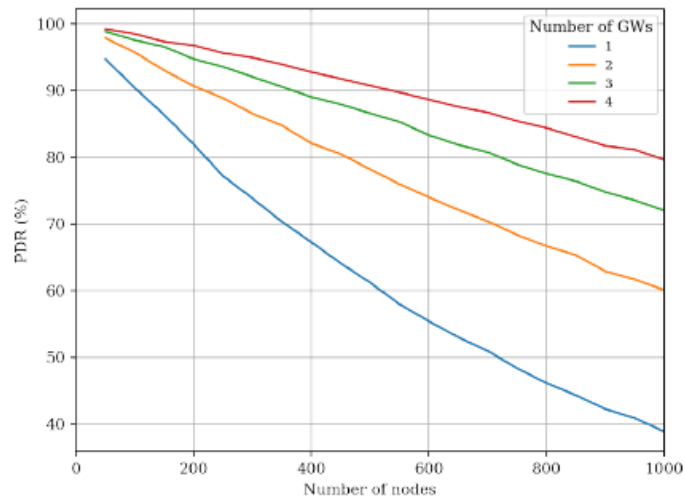


Figure 3.0-5: PDR of Multiple Gateways

3.4.3 LoRaSim

LoRaSim utilizes Markov Chains to model and simulate LoRa traffic. The simulator includes a variety of pre-built models, but users can also expand the collection with their own models. These models can be interleaved across different time intervals to study the effects of interference or changes in transceiver settings.

3.4.3.1 Markov Chains

Markov Chains are mathematical models that transition between different states within a finite or countable set of possible states. They are defined by the "memorylessness" property, meaning the probability of moving to a future state depends only on the current state, not on the sequence of states that led to it. This characteristic is formally known as the Markov property.

In a Markov Chain model for LoRaSim, each state represents a specific condition or configuration of the LoRa transceiver, such as different spreading factors, channel frequencies, or power levels. Transitions between states signify changes in the transceiver's configuration or environmental conditions, such as switching channels or adjusting spreading factors, and these transitions follow probabilistic rules. Transition probabilities are defined for each state pair, determining the likelihood of moving from one state to another, which is essential for accurately simulating realistic LoRa traffic patterns and network behavior under various conditions.

3.4.3.2 Models Employed

In the context of my thesis, the simulator utilized for analyzing LoRaWAN communication encompasses a diverse array of model configurations tailored specifically for urban environments. These configurations, encapsulated within a series of distinct models, vary according to the radius of the simulated environment and the Data Rate (DR) at which communication occurs. For instance, models such as "urban_2km_DR0" and "urban_650m_DR3" signify scenarios set within urban landscapes with differing radii and DR settings. Additionally, certain models incorporate specific interference scenarios, adding further depth to the simulation capabilities. By harnessing these varied configurations, researchers can comprehensively explore the nuances of LoRaWAN network performance in urban settings, fostering a deeper understanding of deployment considerations and optimization strategies within this context.

Figure 3.5 illustrates the User Interface of the Simulator where one can add the simulation intervals having the start time, duration in ms and the model employed.

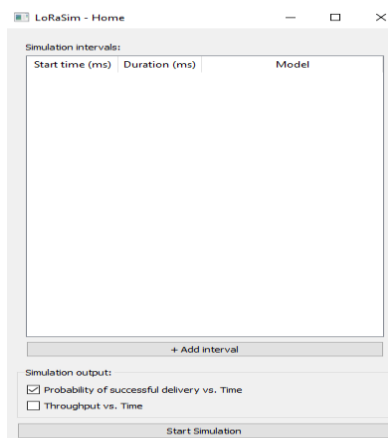


Figure 3.0-6: LoRaSim User Interface [17]

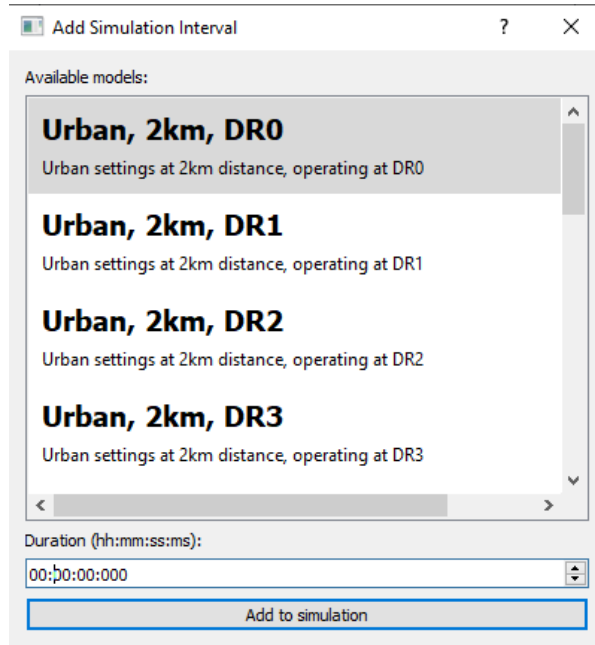


Figure 3.0-7: Simulator Simulation Intervals [17]

As depicted, in figure 3.7 at constant radius (2km and 650m) one can run various simulations from DR0 to DR6.

For Validation of the Simulator, it is tested across varying data rate from DR0 to DR4 at radius 2km.

In Figure 3.8, The x-axis of the graph is time in milliseconds (ms). The y-axis is throughput in bytes per second (byte/s). There are two lines on the graph. The blue line with square markers shows the instantaneous throughput, which is the throughput at any given moment in time. The green line with circle markers shows the average throughput, which is the average throughput over a period of time. The throughput starts at 0 bytes/s and increases to about 11 bytes/s at around 1 millisecond. It then fluctuates slightly around 11 bytes/s for the rest of the simulation. The 95% confidence interval (CI) for the throughput is shown by the shaded area around the blue line. The CI shows the range of values that the throughput is likely to fall within 95% of the time.

In conclusion, the graph shows that the average throughput of the message is about 11 bytes/s. The throughput fluctuates slightly over time, but it remains relatively constant throughout the simulation.

Similarly in Figure 3.9, is a graph about the reception probability of a project over time. The red line represents the probability of the project being successful, while the blue line represents the failure probability. The text on the y-axis reads "Reception Probability" however it likely refers to "Success Probability" based on the context of the graph. The x-axis is labeled "Time (ms)". It goes from 0 milliseconds to 1,000,000 milliseconds (1 second). The y-axis goes from 0 to 1. The success probability starts at 1 and slowly decreases over time. It reaches a value of about 0.6 at the 1 second mark. This means that the project has a 60% chance of being successful after 1 second.

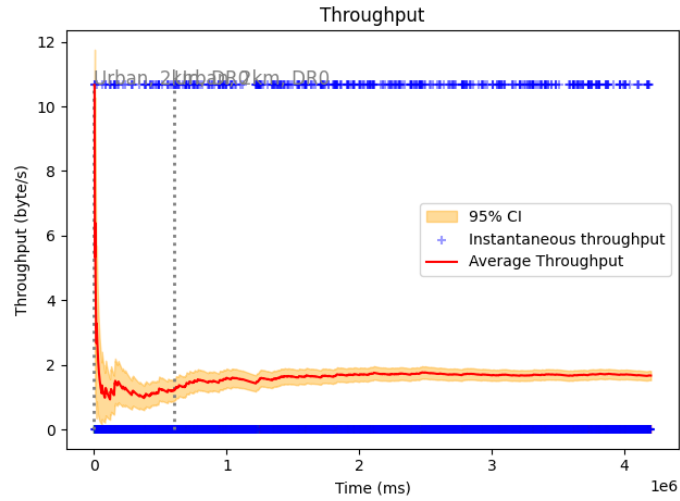


Figure 3.0-8: Throughput vs Time at DR0 and radius 2km

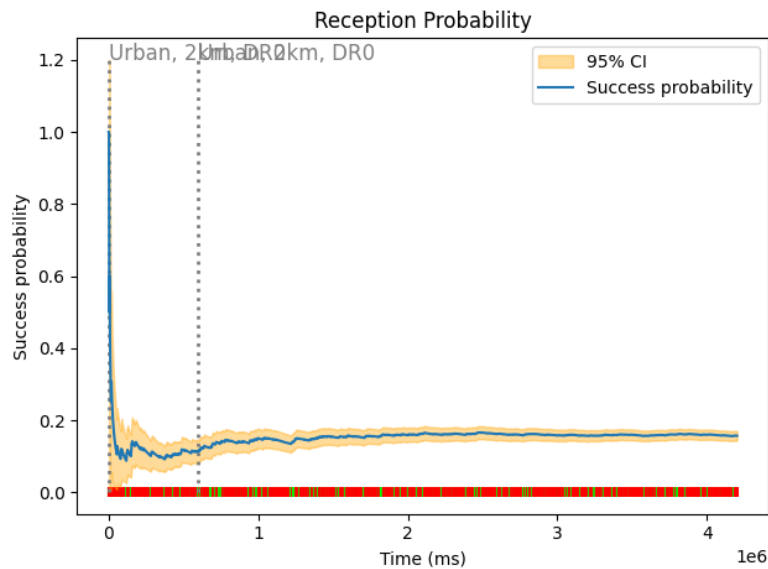


Figure 3.0-9: Success Probability vs time at DR0 -2km

In Figure 3.9. The average throughput graph indicates a stable system performance around 11 bytes/second with an initial data processing burst. Slight fluctuations might be due to variable data size or resource sharing, and the narrow confidence interval suggests these fluctuations are minor.

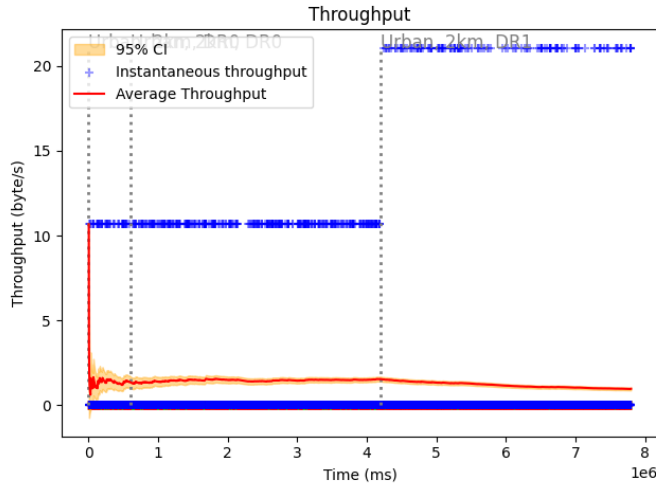


Figure 3.0-10: Throughput vs Time at DR1 and radius 2km

In figure 3.10, The graph depicts a project's success probability diminishing over time, starting at 100% and dropping to roughly 60% within the first second. While the cause for this decline is unclear without further context, it suggests potential challenges emerging as the project progresses. The limited timeframe of the graph leaves the possibility of the success probability dipping further unanswered, and the analysis is likely based on a model, so real-world outcomes may differ.

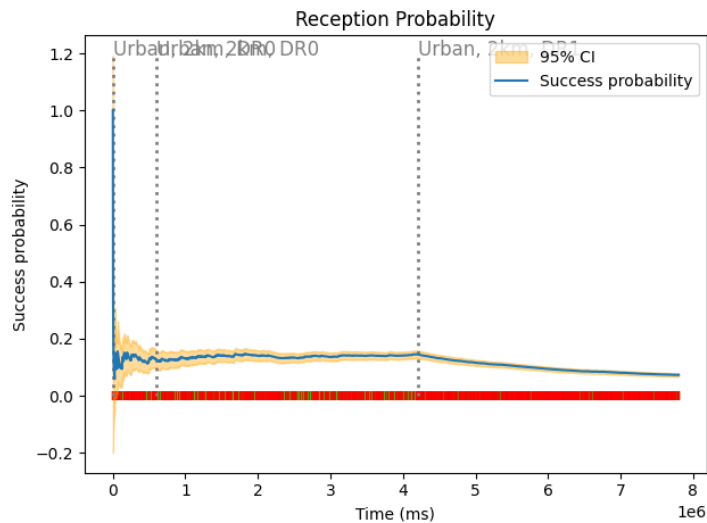


Figure 3.0-11: Success Probability vs time at DR1 - 2km

In Figure 3.12, the cumulative distribution function (CDF) of packet reception probability over a communication channel. The x-axis represents the packet reception probability, which goes from 0 to 1. The y-axis shows the cumulative distribution, expressed as a percentage, which goes from 0% to 100%. The curve on the graph shows that as the probability of receiving a packet increases, the percentage of packets successfully received also increases. For instance, at a packet reception probability of 0.2 (20%), there is a 60% chance of successfully receiving a packet. This means that

60% of the packets transmitted will be received successfully with a probability of 20% per packet. The steeper the curve, the more likely packets are to be received with a higher probability. In this graph, the curve is steeper at the beginning, indicating a higher probability of successful reception at lower individual packet reception probabilities. Overall, the graph depicts a communication channel where the probability of successful packet reception increases as the individual packet reception probability goes up. The curve suggests that a significant portion of packets might be received with a lower probability of success.

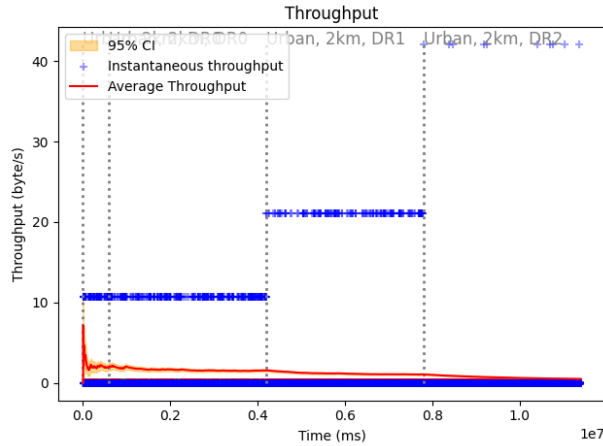


Figure 3-0-12: Throughput vs Time at DR2 and radius - 2km

Figure 3.13 depicts the success probability of a project over time. The y-axis labeled "Reception Probability" likely refers to "Success Probability" in this context. It starts at 1, signifying a 100% chance of success, and gradually decreases to about 0.6 over the course of a second (represented on the x-axis). This suggests a project with a promising initial likelihood of success that faces diminishing odds as time progresses. The reason for this decline is unclear without further information, but it highlights potential challenges that may arise during project execution.

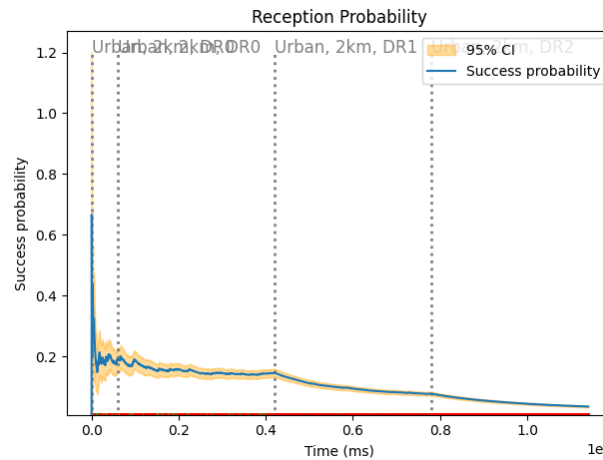


Figure 3-0-13: Success Probability vs time at DR2 -2km

Figure 3.14 depicts a system's average throughput hovering around 11 bytes/second with an initial surge in data processing. The slight fluctuations likely stem from variations in data size or resource sharing, and the narrow confidence interval signifies these fluctuations are minor. For a more thorough analysis, details regarding system specifications, data characteristics, and throughput requirements would be beneficial.

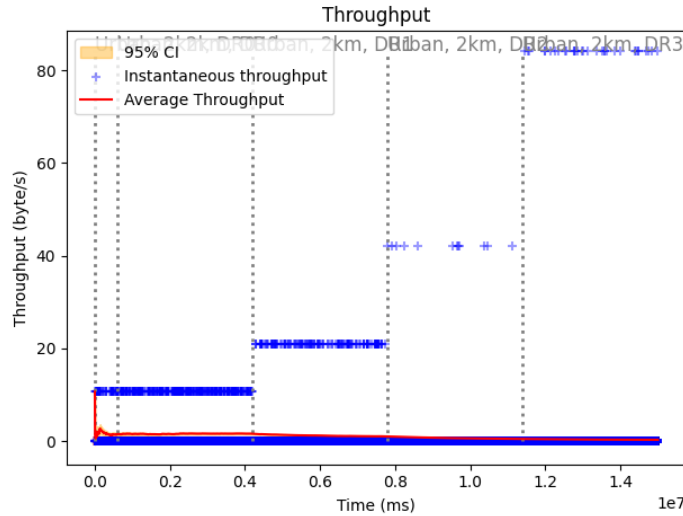


Figure 3-0-14: Throughput vs Time at DR3 and radius - 3km

Figure 3.15 depicts a cumulative distribution function (CDF) of packet reception probability over a communication channel. It shows that as the likelihood of receiving a single packet increase (x-axis), the overall percentage of successfully received packets also rises (y-axis). The steeper curve at the beginning suggests a higher chance of successful reception for packets with a lower individual probability of success. However, it also indicates a significant portion of packets might be received with a lower probability of success overall.

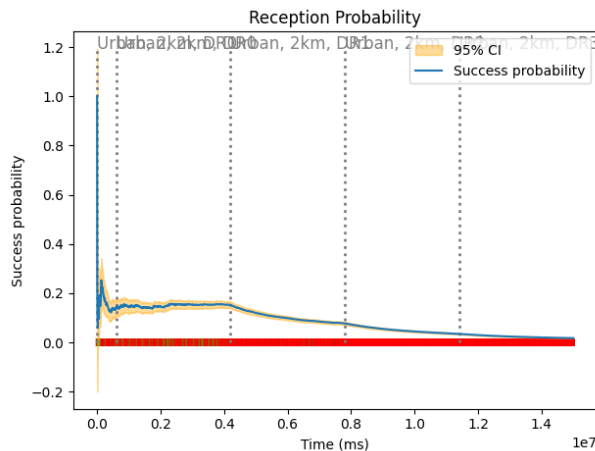


Figure 3-0-15: Success Probability vs time at DR2 - 3km

Figure 3.16 depicts there's an initial burst in throughput followed by fluctuations around the average. This suggests the system efficiently handles data but experiences minor variations in processing, likely due to fluctuations in data size or resource sharing. The narrow confidence interval around the blue line indicates these fluctuations are relatively small.

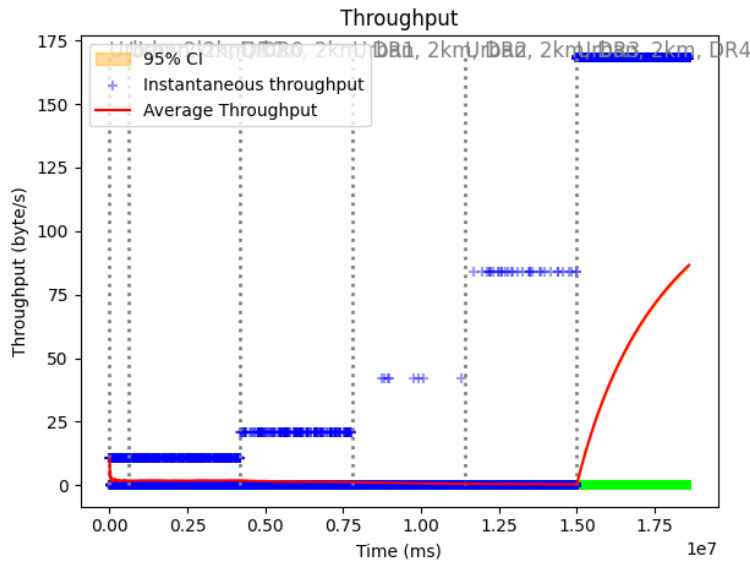


Figure 3-0-16: Throughput vs Time at DR3 and radius - 4km

Figure 3.16 illustrates y-axis, labeled "Reception Probability", likely refers to "Success Probability" in this context. The line starts at 1, signifying a 100% chance of success, and gradually decreases to about 0.6 over the course of a second (represented on the x-axis). This suggests a project with a promising initial likelihood of success that faces diminishing odds as time progresses. The reason for this decline is unclear without further information, but it highlights potential challenges that may arise during project execution. It's important to note that the graph only shows a one-second timeframe, so the success probability could continue to decrease beyond that point.

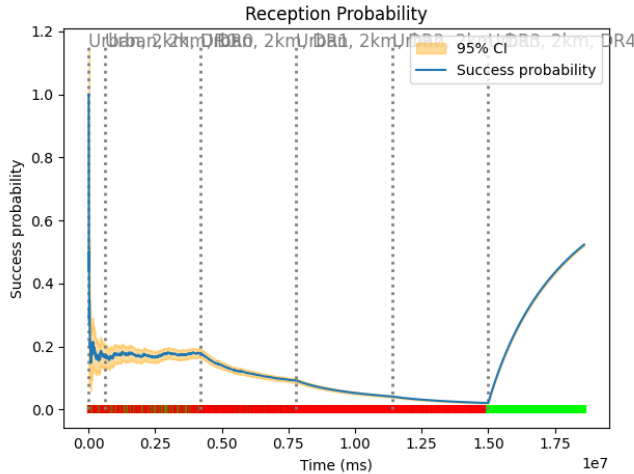


Figure 3-0-17: Success Probability vs time at DR2 - 4km

3.4.4 LoRaSim-2

LoRaSim is a program written in Python that simulates how well LoRa networks can handle increasing numbers of devices. It allows you to set up a network with any number of sensors (called end devices) and gateways (also called base stations) arranged in a grid or scattered randomly.

LoRaSim uses a standard model to calculate how signal strength weakens with distance. While it's a simple tool, it provides valuable insights into network performance. However, it has limitations. It doesn't take into account acknowledgment signals, which are messages sent back from gateways to let devices know if their data was received. This means it can't be used to study how changing signal strength settings (spreading factor) based on these signals affects the network. It can only simulate data going from devices to gateways (uplink), not the other way around (downlink). This prevents it from evaluating a key feature called Adaptive Data Rate (ADR) that optimizes network performance. Despite these limitations, LoRaSim allows you to run simulations with multiple gateways, automatically adjusting device settings based on their distance. It requires additional software packages to function and display results as text and plots, not through a graphical user interface.

Despite its limitations, LoRaSim has been a popular tool for researchers studying LoRa networks, with many modifying it to suit their specific needs. Table 3.2 illustrates the features of LoRaSim.

Table 3.3: Features of LoRaSim

Feature	LoRaSim
Underlying Simulator	Python
Programming Language	Python

Event Handling	Discrete
Licensing	Open source
GUI Availability	Plot only
Power Management	Yes
Low-Power Protocol Support	Yes
Additional Libraries	SimPy, NumPy, matplotlib
Energy Modeling	Yes
Adaptive Data Rate (ADR)	No
Example Scenarios	Yes
Acknowledgment Support	No
Imperfect Spreading Factors	No
Capture Effect Handling	Yes
Device Classes Supported	A
Multiple Gateway Support	Yes
Uplink Confirmation	Yes
Downlink Traffic Handling	No
Network Server Type	Simple
Urban Propagation Modeling	Yes
Research Popularity	High
Documentation Quality	Good
Community Assistance	Limited
Energy Consumption Tracking	Yes
Latest Release / Year	0.2.1 / 2017

3.4.5 Framework for LoRa

FLoRa is a simulation framework that leverages the OMNeT++ simulator and the INET framework to conduct comprehensive end-to-end simulations for LoRa networks. It enables the complete simulation of LoRa/LoRaWAN networks, incorporating all key components. FLoRa is implemented according to the LoRaWAN specification for Class A End Devices (EDs) using unconfirmed transmission mode. The framework supports dynamic management of configuration parameters through the Adaptive Data Rate (ADR) mechanism, which adjusts the Spreading Factor

(SF), Bandwidth (BW), and Transmission Power (TP) of EDs. Unlike other simulators, FLoRa features a user-friendly interface and graphical representations of network scenarios.

Additionally, FLoRa provides an accurate physical layer model for LoRa and allows for end-to-end simulations involving one or more gateways. Communication between the gateways and the network server(s) is facilitated via the Internet Protocol (IP), with the physical layer interactions handled by existing INET framework modules. However, FLoRa has some limitations. It does not account for interference or mobility, and its ADR algorithm does not support unconfirmed transmission mode or the assignment of SFs by the network server. To overcome these issues, researchers have developed the Advanced Framework for LoRa (AFLoRa), an enhanced version of FLoRa with significant improvements and additional LoRaWAN features. Numerous researchers have validated their work using the FLoRa framework. Table 3.3 illustrates the features of FloRa Framework.

Table 3.4: FLoRa Features

Feature	FLoRa Framework
Base Simulator	OMNeT++
Programming Language	C++
Event Handling	Discrete
License Type	Open source
GUI Support	Yes
Power Management	Yes
Low-Power Protocols	Yes
Additional Libraries	INET framework
Energy Modeling	Yes
Adaptive Data Rate (ADR)	Yes
Example Scenarios	Yes
Acknowledgment Support	Yes
Imperfect Spreading Factors	No
Capture Effect Handling	Yes
Device Classes Supported	Class A
Multiple Gateway Support	Yes
Uplink Confirmation	Yes
Downlink Traffic	Yes
Network Server Communication	Via IP
Urban Propagation Modeling	Yes
Research Popularity	Medium
Documentation Quality	Good
Community Assistance	Limited
Energy Consumption Tracking	Yes
Latest Release / Year	1.1.0 / 2022

3.4.6 Ns-3 Module

NS-3 is an open-source discrete-event network simulator primarily designed for educational and research purposes. It operates under the GNU GPLv2 license and aims to enhance the realism of network models by closely aligning their implementation with actual software or real-world systems. The core and models of NS-3 are written in C++, with an optional Python scripting API. Users can write their simulation scripts using either C++ main() or Python.

The LoRaWAN module for NS-3 extends NS-3 to simulate LoRaWAN networks. Each LoRa end device and gateway in this module includes a LoRaWAN MAC/PHY pair, and communication between an end device's PHY layer and its corresponding gateway's PHY layer occurs via the spectrum channel module. This module supports LoRaWAN Class A end devices and utilizes the capture effect for its collision model. This effect occurs when two simultaneous uplink transmissions with the same frequency and spreading factor collide, and the stronger signal captures the weaker one, allowing the gateway to receive only the frame with the highest signal strength.

Over the years, researchers have developed various versions of NS-3 modules for simulating LoRaWAN networks. A comprehensive survey of four different implementations of LoRaWAN modules in NS-3, labeled Module I through IV, is presented by the authors in a study, which compares these modules to identify the most suitable scenarios for each. These modules are freely available for download on GitHub. Many LoRaWAN specifications not found in the FLoRa framework are implemented in the NS-3 LoRaWAN module. Additionally, implementing FLoRa is considered more complex compared to NS-3 LoRaWAN. Numerous researchers have validated, improved, or extended their work using the various implementations of NS-3-based LoRaWAN modules or their own proposed modules in the NS-3 simulator.

Table 3.5: NS-3 module for LoRaWAN features

Feature	NS-3 LoRaWAN Module
Underlying Simulator	NS-3
Supported Languages	C++ and Python
Event Management	Discrete
License Type	Open source
Graphical User Interface (GUI)	Not available
Power Efficiency Features	Yes
Low-Power Protocols	Yes
Additional Library Support	Import all libraries online
Energy Consumption Modeling	Yes
Adaptive Data Rate (ADR)	Yes
Predefined Scenarios	Yes
Acknowledgment Mechanism	Yes

Support for Imperfect SF	Yes
Handling of Capture Effect	Yes
Device Classes	Class A
Multi-Gateway Support	Yes
Uplink Confirmation Messages	Not supported
Downlink Traffic	Yes
Type of Network Server	Simple through IP
Urban Propagation Models	Yes
Popularity in Research	High
Quality of Documentation	Excellent
Community Support	Very Good
Energy Usage Tracking	Yes
Most Recent Version / Year	0.3.0 / 2021

3.4.7 LWN Simulator

LWN Simulator is a LoRaWAN node simulator featuring a web interface. It facilitates communication with a real LoRaWAN infrastructure or an ad-hoc setup, such as Chirpstack.

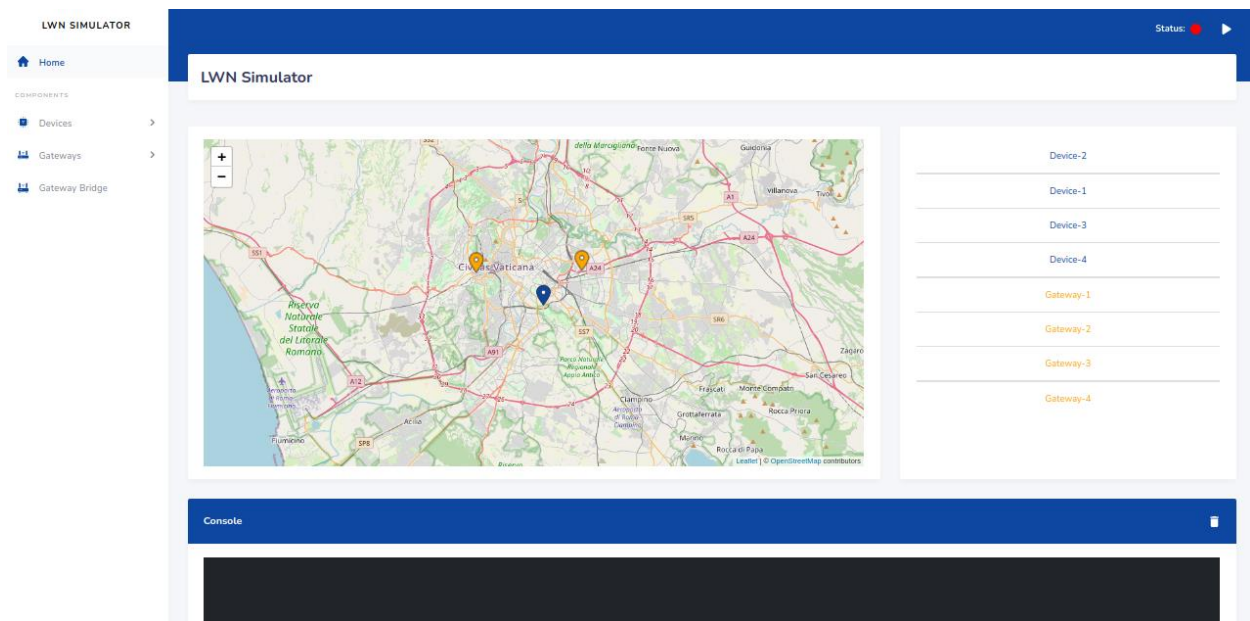


Figure 3-0-18: User Interface of LWN Simulator

The project comprises three primary components: devices, forwarders, and gateways.

The device:

- Adheres to LoRaWAN v1.0.3 specifications.
- Supports all LoRaWAN Regional Parameters v1.0.3.
- Implements Class A, C, and partially even Class B.
- Incorporates the Adaptive Data Rate (ADR) Algorithm.
- Periodically sends frames containing configurable payloads.
- Supports MAC commands and implements the FPending procedure.
- Enables real-time interaction.

The forwarder:

- Receives frames from devices and encapsulates them within an RXPk object.
- Forwards the encapsulated frames to gateways.

The gateway:

- Includes two types: a virtual gateway communicating with a real gateway bridge (if available), and a physical gateway to which UDP datagrams are forwarded.

If a real infrastructure is unavailable, one can opt to download ChirpStack, an open-source LoRaWAN® Network Server, or similar software for demonstration purposes. Alternatively, if a real infrastructure is accessible, ensure that the gateways and LoRaWAN servers are reachable from the simulator.

3.4.7.1 Installation Method

From Binary File: You can acquire the pre-compiled binary file by downloading from the releases section.

- Prerequisites: Ensure Go is installed, with a version greater than or equal to 1.16. Clone the repository using the following command.
`git clone https://github.com/UniCT-ARSLab/LWN-Simulator.git`
- Navigate to the main directory.
`cd LWNSimulator`
- Install all dependencies required to build the simulator.
`make install-dep`
- Initiate the build process for the simulator.
`make build`

Starting the Simulator: There are two options to start the simulator:

- From source (without building the source): `make run`

- From the built binary: make run-release

Configuration File: The simulator relies on a configuration file (config.json) to specify various configurations:

```
{ "address": "0.0.0.0", "port": 8000, "configDirname": "lwnsimulator" }
```

- address: Specifies the IP mask from which the web UI is accessible.
- port: Defines the web server port.
- configDirname: Indicates the directory name where all status files will be saved and created.

3.5 Conclusion

This survey offers a new taxonomy of LoRa/LoRaWAN simulators to help researchers identify the most suitable one for their research. While general-purpose simulators have versatility to their advantage, ChirpStack-enabled simulators broaden the horizon and let the users interact with a real-life LoRaWAN deployed Network Server. The onset of machine learning and blockchain-integrated simulators is evidence of progressive advances in network optimization and data protection in wireless sensor networks. Future surveys should include more established and source simulators in the catalogue; this way, the segmentation of the users must be made according to categories resulting from the research interests.

CHAPTER 4: LORA SPREADING FACTOR – MACHINE LEARNING DRIVEN OPTIMIZATION

4.1 Packet Collision in LoRa Networks

In LoRa networks, packet collisions happen when multiple nodes try to send data at the same time and frequency, employing identical spreading factors. This simultaneous transmission leads to interference as the signals intersect at the receiving gateway. Consequently, this interference disrupts the reception process, resulting in packet loss and undermining network stability. Simultaneous transmissions with different spreading factors in LoRa are somewhat orthogonal, allowing a gateway to receive multiple transmissions concurrently. However, simultaneous transmissions with identical spreading factors may collide, hindering reception. Hence, strategic spreading factor assignment is pivotal for network optimization.

4.1.1 Impact of Spreading Factor on Packet Collision/ Interference-induced collision

Lower SFs result in shorter symbol durations, allowing for faster transmission of data. With shorter symbol durations, more symbols can be transmitted within a given time period, leading to higher data rates. Lower SFs also leads to narrower frequency spreads, meaning the signal is concentrated within a smaller portion of the frequency spectrum. This concentration increases the spectral density of the signal, making it more susceptible to narrowband interference, such as adjacent channel interference or narrowband noise, increasing the likelihood of packet collisions. Lower SFs typically result in shorter transmission ranges due to increased susceptibility to interference. However, the effective range is not solely determined by SF; other factors such as transmit power, receiver sensitivity, and environmental conditions also play significant roles.

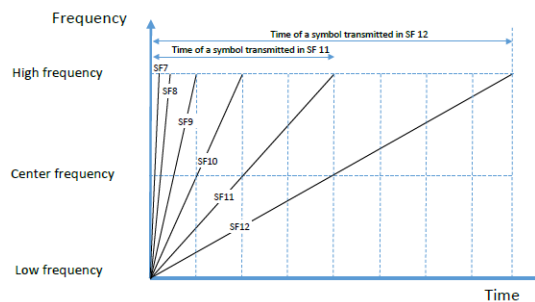


Figure 4-0-1: Symbol Transmission duration

Higher SFs spread the signal over wider frequency ranges, reducing the spectral density of the transmitted signal. These wider spread decreases the spectral density, making the signal less susceptible to narrowband interference, such as adjacent channel interference or narrowband noise, decreasing the likelihood of packet collisions. Higher SFs provide longer transmission ranges by spreading the signal energy over a wider area in the frequency domain. With reduced susceptibility

to interference, signals transmitted using higher SFs can maintain their integrity over longer distances, enabling communication over extended ranges at the cost of increased power consumption and lower data rate, while also extending the time on air (TOA), thereby raising the risk of packet collisions due to prolonged transmission duration.

In furtherance, collisions can still occur if transmissions are not properly coordinated or if nodes transmit simultaneously without collision detection mechanisms. Therefore, choosing the right SF is crucial for balancing data rate, range, and collision likelihood in LoRa networks to ensure dependable communication.

4.2 Techniques Employed for Packet collisions avoidance

Packet collision avoidance in LoRaWAN networks is achieved through techniques such as utilizing the lowest spreading factor, adaptive data rate mechanisms, and strategic spreading factor management among nodes. These strategies optimize transmission efficiency and minimize interference in dynamic network environments.

4.2.1 Lowest Spreading Factor Technique for Packet collisions avoidance

In a LoRaWAN network, nodes initially lack knowledge of their distance from a gateway. However, they can estimate this distance by monitoring the received signal power of downlink transmissions. If the received signal power is excessive, nodes can adjust their next transmission spreading factor downwards to reduce power consumption [10].

4.2.2 Adaptive Data rate for Packet collisions avoidance

In LoRaWAN networks, adaptive data rate (ADR) mechanisms can dynamically adjust spreading factors based on network conditions. ADR algorithms aim to balance data rate, transmission range, and collision probability by dynamically optimizing spreading factor assignments for individual nodes. As described in Chapter 2.9, ADR adjusts the data rate of LoRaWAN nodes based on the available link budget [8].

- If the link budget is high, it increases the data rate by reducing the Spreading Factor (SF).
- If the link budget is low, it decreases the data rate by increasing the SF.

This dynamic adjustment helps mitigate collision events by adapting to changing network environments and traffic patterns.

4.2.3 Managing Spreading factor for Packet Collisions avoidance

In this technique, some nearby nodes need to use higher spreading factors to avoid collisions, even if they could manage with lower ones. This helps prevent clashes by taking advantage of how spreading factors differ. However, it's crucial to carefully distribute spreading factors among nodes. When a node's spreading factor is increased, it takes longer to transmit, increasing the chance of

colliding with other high spreading factor transmissions. In setups with multiple gateways, this can lead to more collisions with nodes in the coverage of other gateways. Nodes in areas where gateways overlap need special attention [10].

Proposing a single spreading factor assignment rule for all LoRaWAN topologies is challenging due to the unique characteristics of each network. Optimizing spreading factors for nodes necessitates tailored rules, as every network configuration varies.

4.3 ML-Based SF Allocation for Packet Collision Avoidance

This section scrutinizes the efficacy of machine learning-based SF allocation techniques in LoRaWAN networks, highlighting methodological strengths and limitations. Through an objective lens, it evaluates the utilization of various classification methods, emphasizing the importance of robust evaluation metrics and considerations such as cross-validation. By dissecting recent studies, it aims to provide technical insights for advancing SF allocation strategies in IoT environments.

4.3.1 *Using Traditional ML algorithms for enhancing SF allocation*

In [7], the authors have introduced a technique for SF assignment utilizing traditional ML-based support vector machine (SVM) and decision tree (DT) methods. The evaluation of classifier performance focused solely on accuracy concerning the prediction of smart SVM and smart DTC schemes applied to Table III's data on network radii and the number of nodes, as depicted in Fig. 7. PDR for lowest and smart SF schemes ($r = 5000$ m, $GW = 3$). Notably, the evaluation overlooked critical metrics such as Recall, Precision, F1 Score, and Matthews Correlation Coefficient (MCC), which are essential in comprehensively assessing classifier efficacy in this context. Moreover, Recall, Precision, F1 Score, and MCC each offer distinct insights into the classifier's ability to accurately predict interfered and successful transmissions within the network. Neglecting these metrics hinders a thorough understanding of the classifier's performance, particularly concerning its capacity to differentiate between various transmission outcomes.

Furthermore, the absence of cross-validation in the evaluation raises concerns regarding the robustness and generalization capability of the trained classifiers. Cross-validation is pivotal in assessing model performance on unseen data and guarding against overfitting, thereby ensuring the reliability and applicability of the classifier in real-world deployments.

In this specific application scenario, where the focus lies on enhancing network PDR through smart SF allocation, the inclusion of comprehensive evaluation metrics is imperative for gaining insights into the classifier's effectiveness in predicting interference and successful transmissions accurately. Failure to consider such metrics may lead to misconceptions about the classifier's performance and its suitability for practical deployment in wireless network environments.

4.3.2 *Approaching SF Assignment as a classification problem*

In [12], the comparison of classification techniques for SF allocation in LoRa networks was conducted, focusing on k-nearest neighbor (KNN), Naive Bayes, and Support Vector Machine (SVM). The study treated the SF assignment problem as a classification task, addressing discrete

SF values ranging from 7 to 12. The experiments, conducted in the FLoRa simulator environment, indicated that SVM and KNN demonstrated superior predictive performance for optimal SF allocation.

Despite the study's utilization of several crucial evaluation metrics such as accuracy, Recall, Precision, and F1 Score, it overlooked the inclusion of the Matthews Correlation Coefficient (MCC). MCC stands as a valuable metric for evaluating classifier performance, particularly in scenarios characterized by imbalanced datasets or differential costs associated with false positives and false negatives.

The omission of MCC diminishes the comprehensiveness of the evaluation, limiting insights into the effectiveness of classification techniques for SF allocation in LoRa networks. Including MCC alongside the other metrics would have provided a more holistic assessment of classifier performance, offering deeper understanding and potentially highlighting nuances in performance across different classification techniques.

Despite this omission, the study showcased robustness in its evaluation methodology by considering multiple metrics. However, the absence of MCC restricts the extent to which the study can draw conclusions about the relative effectiveness of the classification techniques evaluated for SF allocation in LoRa networks.

4.3.3 Applying DL algorithms to Enhance SF allocation

The study detailed in [13] introduced an SF assignment methodology employing traditional ML methods like Support Vector Machine (SVM) and Decision Tree (DT), alongside deep learning techniques such as Fully Connected Neural Network (FCNN) and Convolutional Neural Network (CNN). Evaluation metrics encompassed prediction accuracy, packet delivery ratio (PDR), and total transmit energy consumption (TEC). However, the assessment overlooked critical metrics like Recall, Precision, F1 Score, and Matthews Correlation Coefficient (MCC), which are pivotal for a comprehensive evaluation of classifier performance, particularly in contexts with imbalanced datasets or mission-critical applications.

The absence of cross-validation on the dataset raises concerns regarding the robustness and generalization capability of the trained models. Cross-validation serves as a fundamental method for evaluating model performance on unseen data and mitigating risks associated with overfitting, thereby ensuring the reliability and applicability of the classifiers in diverse scenarios.

Furthermore, employing deep learning techniques for IoT applications may pose challenges in terms of resource constraints. Deep learning models typically demand substantial computational resources for training and deployment, which may not be feasible in resource-constrained IoT environments. Hence, the potential advantage of traditional ML methods lies in their ability to offer efficient solutions suitable for deployment in such environments.

4.4 SmartLoRaML: Optimizing LoRa SF Allocation with ML Algorithms

In this study, we introduce SmartLoRaML, a technique aimed at enhancing LoRa SF allocation through the utilization of random forest (RF) and gradient boosting for a more comprehensive analysis. The LoRa simulator explained in chapter 3 [11] is used to generate the dataset.

A thorough comparative evaluation of these classifiers, assessing their performance across metrics such as accuracy, packet delivery ratio (PDR), transmit energy consumption, recall, precision, F1 score, and Matthews Correlation Coefficient (MCC) is provided. This evaluation is carried out across diverse network topologies and configurations of end nodes which will be subsequently explained in detail, providing valuable insights into the effectiveness of SmartLoRaML in optimizing LoRa resource allocation.

Additionally, an enhanced evaluation of classifiers, namely Support Vector Machine (SVM) and Decision Tree (DT), is conducted within the context of [11], utilizing the same simulation environment and dataset as in the aforementioned study. This new evaluation encompasses metrics such as recall, precision, F1 score, and Matthews Correlation Coefficient (MCC) to gauge the classifiers' performance in accurately classifying instances, detecting positive samples, balancing precision and recall, and measuring the overall agreement between predicted and observed classifications. This additional evaluation through employing various metrics serves to provide a more nuanced assessment compared to the previous evaluation [11]. Furthermore, cross-validation techniques are employed to facilitate a comparative analysis between the accuracies, packet delivery ratios (PDR), and energy consumption of the prior study [11] and the updated model.

The code updates, including modifications to the simulator and simulation environment, along with the revised evaluation methodologies and results, are accessible via the GitHub link provided below.

4.5 SmartLoRaML – Methodology

The SmartLoRaML methodology includes the simulator used for dataset generation, the technique, evaluation metrics,

4.5.1 Dataset generation

In the LoRa Simulator [11], the first step involves configuring the transmission parameters, which are also outlined in Table 4.1. These parameters serve critical roles: the **Topology_Radius** parameter defines the maximum distance from the network center within which nodes are placed or communication is allowed. This parameter is crucial for determining the coverage area and spatial arrangement of the LoRa network. **Number_Of_Gateways** represents the count of gateway devices deployed in the network infrastructure, responsible for relaying messages between end-devices and network servers. It significantly influences the network's capacity, coverage, and reliability. **Number_Of_Nodes** indicates the total quantity of end-devices or nodes participating in the network, affecting the overall network load and communication dynamics. **SF Assignment**

Method refers to the approach used to assign spreading factors (SF) to nodes, which determines the modulation scheme and data rate for communication. The simulator automatically assigns a random SF assignment method if not coded otherwise, a critical aspect for optimizing network performance and spectral efficiency. **Simulation_Duration** specifies the length of time for which the simulation of the LoRa network model is executed, providing insights into network behavior over time. **Packet_Rate** denotes the rate at which data packets are generated and transmitted within the network, influencing network traffic and resource utilization. **Packet_Size** represents the size of data packets transmitted, including headers and payloads, affecting the efficiency of data transfer and network capacity. Lastly, **Traffic_Type** characterizes the nature of data transmission, such as periodic or random (e.g., Poisson), influencing network traffic patterns and performance metrics. Each of these parameters plays a critical role in defining the behavior and performance of the LoRa network under simulation.

The simulation tool is executed to generate transmission logs. These logs contain three main features: the X coordinate and Y coordinate of the transmission source, and the spreading factor used in the transmission. The X and Y coordinate values are continuous, while the spreading factor values are integers ranging from 7 to 12. The class label assigned to each transmission in the dataset indicates the transmission result, categorized as either successful, interfered, or under sensitivity, as defined by the simulation outcomes. A sample dataset is presented in Table 4.2.

Table 4.1: Simulator Parameter Set

Serial no	Parameters
1	Topology Radius
2	Number Of Gateways
3	Number Of Nodes
4	SF Assignment Method
5	Simulation Duration
6	Packet Rate
7	Packet Size
8	Traffic_Type

In this simulator, it is assumed that the node locations are known to the Network Server (NS) through triangulation techniques.

Table 4.2: Sample Dataset

X- Coordinate	Y-Coordinate	Spreading Factor	Result
-2033.1	713.0	8	successful
995.5	-2148.6	7	under sensitivity
-3738.6	-4665.3	12	interfered
-1268.9	1348.2	9	successful
-427.1	593.7	9	successful
-193.4	-4047.5	11	interfered
-2729.1	-3663.3	10	interfered
637.3	-715.3	7	successful
3843.7	1996.3	8	successful

4.5.2 SmartLoRaML Technique

The SmartLoRaML technique involves two key phases. In the initial stage, known as Round 1, random spreading factor (SF) values are initialized, and machine learning algorithms such as Random Forest and Gradient Boosting are trained using transmission log data. These algorithms are covered subsequently in the sub section. Cross-validation is employed during this training to ensure robustness and prevent overfitting, focusing on preparing for collision prediction. In Round 2, termed Collision Prediction and SF Adjustment, trained classifiers predict collisions and adjust the SF accordingly to optimize transmission outcomes, thereby enhancing the performance of the LoRa communication system. These two rounds are elaborated upon in the subsequent sections.

4.5.2.1 Round 1: Initial Training Phase

The simulator begins by employing a random Spreading Factor (SF) scheme as a SF assignment method, as discussed in Section 4.5.1. Following the completion of this initial random SF simulation phase, data is extracted from transmission logs. The dataset comprises of three feature columns (x, y, SF) and a label column indicating collision status (successful, interfered, under sensitivity). Subsequently, the extracted dataset undergoes training using both Random Forest and Gradient Boosting algorithms. The dataset is split into 80% training and 20% testing subsets. Cross-validation with a fold value of 5 (cv=5) is employed during training to ensure robustness and avoid overfitting. In this setup, each fold serves as a validation set for the others.

Incorporating cross-validation in the training process offers several advantages, including:

- **Mitigation of Overfitting:** Cross-validation helps in evaluating the model's performance on multiple subsets of data, reducing the risk of overfitting to a particular training set.
- **Better Generalization:** By testing the model on multiple validation sets, cross-validation provides a more accurate estimate of the model's performance on unseen data, leading to better generalization.
- **Robustness Assessment:** Cross-validation allows for assessing the robustness of the trained models by evaluating their performance across different data partitions.

4.5.2.2 Round 1 - Mathematical Representation

The mathematical representation for round 1 is as follows:

$$D_{train} = CombineLogs(D) \quad (4.1)$$

$$M = TrainModel (D_{train}) \quad (4.2)$$

where

D_{train} represents the training dataset created by combining transmission logs

$CombineLogs(D)$ Combine logs denotes the process of combining transmission logs into the training dataset.

M represents the trained classifier.

$TrainModel ()$ is the function used to train the model using the training dataset.

4.5.2.3 Round 2: Collision Prediction and SF Adjustment

The trained classifiers from Round 1 are then applied in the second simulation phase for collision prediction. The tool selects optimum spreading factor for transmissions considering prediction of the transmissions. The outcomes from the trained classifiers (Random Forest and Gradient Boosting) aid in proposing an SF assignment algorithm to mitigate collision events. The simulator dynamically adjusts the SF based on the output class label from the classifiers. For each transmission, the simulator checks the transmission result label for the lowest possible SF. If the predicted transmission label is interfered with, the simulator increases the SF and transmits again, predicting a new transmission outcome (successful, interfered, or under sensitivity). If a successful transmission label is predicted, the simulator continues operation. However, if all SF values are tested and no successful transmission label is predicted, the simulator resumes operation with the lowest SF.

4.5.2.4 Round 2 - Mathematical Expression

For each transmission, the classifier predicts the transmission result R for the lowest possible spreading factor SF_{min} .

$$R = Predict (M, SF_{min}) \quad (4.3)$$

If the result R indicates successful then the simulator will use the SF_{min} in 4.3 for future transmissions. If the result indicates interference, the spreading factor is increased by one.

$$SF_{new} = SF_{min} + 1 \quad (4.4)$$

If the new transmission result R_{new} is successful, the simulator continues with the selected spreading factor SF_{min} . However, if no successful transmission result is predicted, the simulator continues with the lowest possible spreading factor SF_{min} .

4.5.3 ML Algorithms used SmartLoRaML Technique

SmartLoRaML technique employs Random Forest and Gradient Boosting as the classifiers which will get trained on the dataset derived from the transmission log data. In this section, random forest and gradient boosting will get explained and how these algorithms actually work and why these algorithms are chosen for this technique.

4.5.3.1 Random Forest

An improvement to decision tree algorithm is using several decision trees at a time rather than one. This innovative technique is termed as Random Forest. The RF algorithm adds a substantial amount of enhancement to the accuracy of a model because of multiple decision tree handling capability. The multiple trees generate a unique result about the given sample and later the basis of majority of decision tree's result is provided. [25].

Random forests are created utilizing a technique called bagging. In this technique the decision trees are used as parallel estimators. If Random Forest is employed for a classification problem, then the result of algorithm is based on majority vote of each result received from each decision

tree. Otherwise, if this algorithm is being used for regression, then the forecast of a leaf node is considered as the mean value of the target values in that leaf. The mean value of results by decision tree is taken by Random Forest regression as the final output.

The accuracy of Random forests is much higher than a single decision tree and it also reduces the risk of overfitting. Furthermore, there is no bottleneck in time as the decision trees run in parallel making this algorithm time efficient.

Random forest works efficiently where the decision trees are uncorrelated. If there is high degree of similarity between the decision trees, then the random forest result will not be different from a single decision tree. Random forest uses bootstrapping and feature randomness to achieve the uncorrelated decision trees.

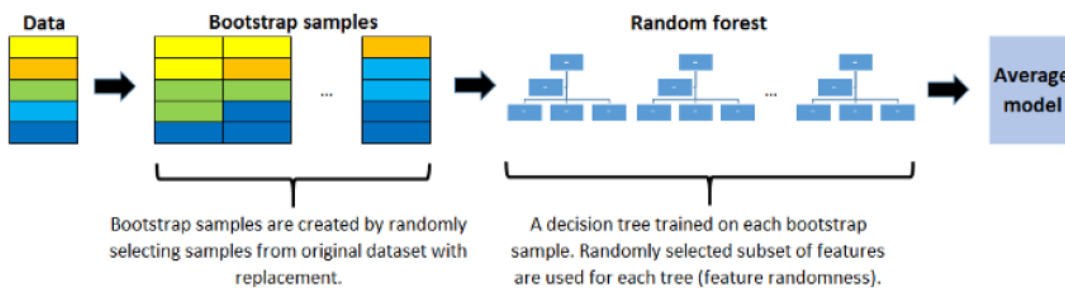


Figure 4-0-2: Bootstrapping Method [39]

Bootstrapping is a technique where samples are randomly selected samples from training data with replacement. They are called bootstrap samples.



Figure 4-0-3: Bootstrapping samples[39]

Feature randomness is attained by randomly selecting features for every decision tree. `max_features` parameter is used to control the number of features used for each tree in random forest.

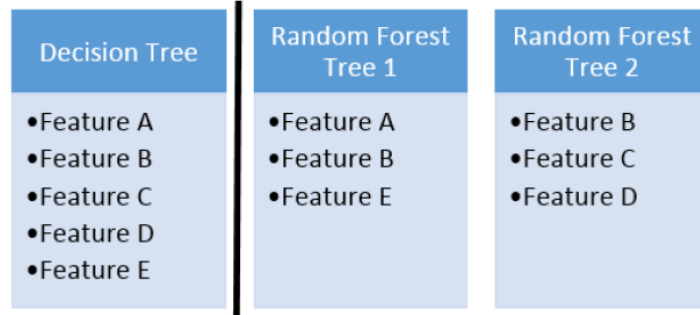


Figure 4-0-4: Feature set Selection by Decision Trees and Random Forest[40]

Random forest does not require normalization or scaling, and it is an extremely precise model for numerous different problems. However, it doesn't work well with high-dimensional data sets (i.e., classification of text) as compared to other fast linear models such as Naive Bayes.

4.5.3.2 Gradient Boosting

Gradient Boosting stands out as a formidable member of the ensemble learning family, renowned for its remarkable predictive capabilities spanning diverse tasks. This section delves into the technical intricacies of Gradient Boosting, elucidating its foundational principles, pivotal components, and operational mechanisms.

At the heart of Gradient Boosting lies the pursuit of minimizing a predefined loss function. With each iteration, the algorithm strategically introduces a new weak learner, typically decision trees, tailored to the residual errors of the current ensemble predictions. This iterative refinement gradually diminishes the overall error, culminating in a finely tuned final model.

In Gradient Boosting, several key components synergistically contribute to its effectiveness. First and foremost are the base learners, primarily decision trees, chosen for their versatility and ease of implementation. To mitigate the risk of overfitting and maintain computational efficiency, these trees are deliberately kept shallow, limiting their depth to only a few levels. Secondly, the selection of an appropriate loss function is crucial, dictating the direction of model optimization. Common choices include the mean squared error (MSE) for regression tasks and the logistic loss for binary classification endeavors. Leveraging gradient descent optimization, the algorithm systematically navigates towards minimizing the loss function by iteratively adjusting model parameters based on computed gradients. Moreover, to regulate the impact of each base learner on the ensemble, Gradient Boosting incorporates a shrinkage parameter, often referred to as the learning rate. This parameter governs the pace of learning, facilitating a gradual refinement process that enhances generalization and overall model performance. Together, these components form the foundation of Gradient Boosting, enabling it to excel in predictive modeling across diverse domains.

The training process of Gradient Boosting follows a systematic series of steps aimed at iteratively enhancing the predictive capabilities of the model. It begins with the initialization phase, where a basic model, often a single base learner, is established to lay the groundwork for subsequent iterations. Following initialization, the algorithm proceeds to calculate the negative gradient of the loss function based on the ensemble's predictions. This gradient computation guides the introduction of a new base learner, specifically designed to address the residual errors of the current

ensemble. As the new base learner is fitted to the ensemble, its predictions are integrated into the ensemble's overall predictions, with the integration weighted by the learning rate to control its influence. This process iterates through steps 2 to 4, with each iteration refining the ensemble's predictive accuracy. Iterations continue until either a predefined iteration limit is reached or until the improvement in the loss function falls below a specified threshold. Through this iterative process, Gradient Boosting progressively improves its predictive performance, resulting in a highly accurate final model.

Gradient Boosting offers several advantages, including exceptional predictive performance that often surpasses alternative machine learning algorithms. Its systematic, sequential training approach and the use of shallow base learners contribute to its resilience against overfitting. Moreover, Gradient Boosting demonstrates versatility across various data types and modeling tasks, including regression, classification, and ranking. However, it is sensitive to hyperparameters like the learning rate and tree depth, requiring careful calibration for optimal performance. Additionally, it may incur computational overhead, especially with large datasets or deep trees, potentially extending training durations. Furthermore, Gradient Boosting is susceptible to overfitting in cases of suboptimal hyperparameter tuning or when dealing with noisy or outlier-laden training data.

4.5.4 Simulation Environment

The Lora Simulator [11] used for the implementation of the SmartLoRaML technique, specifications and installation method has already been covered in chapter 3. In this section, the hardware used for running the simulator and its specifications will be covered. In furtherance, the different scenarios in which the SmartLoRaML technique will be implemented is also covered.

4.5.4.1 Hardware specifications

The simulator runs on an HP laptop equipped with an Intel Core i5-7200U processor running at 2.50GHz, 16GB of RAM, and Windows 10 as the operating system. The components description of the device is as under.

Table 4.3: Hardware Specifications for Simulation

Components	Description
Processor	Intel(R) Core (TM) i5-7200U CPU @ 2.50GHz
Operating System	Windows 10
RAM	16 GB
System Type	64-bit operating system, x64-based processor
Graphics Card	NVIDIA GeForce MX150 (2GB GDDR5 VRAM)
Storage	512GB PCIe NVMe M.2 SSD
Display	15.6-inch Full HD (1920 x 1080) IPS display, touch
Battery Life	3-cell 52.5Whr Lithium-ion battery, upto 8 hours

4.5.4.2 Different Scenarios for Implementing SmartLoRaML Technique

The simulator can create a network topology resembling a circular shape, with parameters such as radius, number of nodes, and number of gateways. This topology is uniformly distributed across the simulated area. The simulation tool is capable of accommodating both custom-designed LoRaWAN topologies and randomly generated ones.

Users can define global simulation parameters, including the duration of the simulation, packet size, packet generation rate, packet generation type, and spreading factor assignment method.

The simulation tool visualizes the placement of LoRaWAN network gateways for different quantities of gateways. Initially, the tool distributes gateways evenly across the network, aiming to maximize their separation. Subsequently, it randomly positions end nodes within the network's radius.

The simulation tool exclusively focuses on LoRaWAN Class A devices due to their default operational mode and minimal power consumption. In this mode, end nodes always initiate transmissions in a pure ALOHA manner. Nodes generate packets based on the specified packet rate parameter, with options for generating packets according to either Poisson interval or periodic interval.

In conclusion, the tool offers a comprehensive framework for simulating LoRaWAN networks, managing a transmission queue where each event represents a transmission. Each transmission includes fields such as time, spreading factor, source, size, duration, and status. End nodes generate events based on their traffic generation rate parameter, adding them to the simulation event queue with an initial pending state. The tool iterates through and executes events, updating their status as transmitted, interfered, or under sensitivity. Upon completion of all events, the tool computes network packet delivery ratio (PDR), throughput, and transmit energy consumption.

4.5.4.3 Implementation of the SmartLoRaML technique

After the installation of LoRa Simulator [11] as mentioned in chapter 3, the following global simulation parameters are used: the packet size is set to 60 Bytes, the simulation duration is set to 3600 seconds, and the traffic generation type is set as Poisson. The packet rate is set at 0.01 per second, and the packet size is 60 bytes, including both header and payload.

4.5.5 *Evaluation of SmartLoRaML Technique*

In this section, the performance of the SmartLoRaML technique will be evaluated using a carefully selected set of metrics. These metrics have been specifically chosen to comprehensively assess the efficacy of SmartLoRaML in enhancing LoRa SF allocation and suitability for real-life deployment. This evaluation aims to provide a robust and holistic understanding of the technique's effectiveness in real-world scenarios.

The following evaluation metrics have been selected, each serving a distinct purpose in assessing various aspects of SmartLoRaML's performance:

4.5.5.1 Stratified Cross Validation:

Stratified cross-validation is a variation of the traditional cross-validation technique, designed to address the potential issue of imbalanced class distributions in the target variable. In stratified cross-validation, the dataset is divided into folds while ensuring that each fold maintains the same proportion of samples from each class as the entire dataset. This ensures that the distribution of classes remains consistent across all folds, mitigating the risk of biased model evaluation. By preserving class proportions, stratified cross-validation provides a more reliable estimate of model performance, particularly in scenarios where certain classes are underrepresented or overrepresented. This technique is particularly beneficial for classification tasks, where maintaining class balance is crucial for accurate model assessment. Stratified cross-validation is widely adopted in machine learning practice to enhance the robustness and generalization ability of predictive models, especially in the presence of imbalanced datasets.

4.5.5.2 Accuracy:

This metric will measure the overall correctness of SmartLoRaML's predictions, providing insight into the accuracy of its transmission outcome predictions. Accuracy serves as a fundamental metric in evaluating the overall correctness of the SmartLoRaML technique in predicting transmission outcomes. It provides insight into the percentage of correctly classified instances, including successful transmissions and mitigated collisions. Accuracy is crucial for assessing the overall effectiveness of the SmartLoRaML technique in predicting transmission outcomes accurately, especially in terms of successfully transmitted packets and mitigated collisions. Accuracy measures the overall correctness of the classifier's predictions. It is essential for understanding how well the classifier performs in terms of correctly classifying instances.

4.5.5.3 Packet Delivery Ratio (PDR):

PDR will be utilized to evaluate the reliability of packet delivery within the LoRa network. It will help gauge the effectiveness of SmartLoRaML in ensuring successful packet transmissions.

4.5.5.4 Energy Consumption:

Energy consumption will be assessed to understand the efficiency of SmartLoRaML in optimizing transmission outcomes. Lower energy consumption would indicate more efficient operation, crucial for IoT device longevity.

4.5.5.5 F1 Score:

The F1 score will provide a balanced assessment of SmartLoRaML's performance by considering both precision and recall. It will offer insights into the technique's effectiveness in optimizing LoRa SF allocation.

4.5.5.6 Matthews Correlation Coefficient (MCC):

MCC will be employed to assess the overall performance of SmartLoRaML, particularly in scenarios with imbalanced datasets. It will offer a balanced measure of the technique's predictive capability.

4.5.6 *Simulation Results*

In this section, the accuracy, packet delivery ratio (PDR), transmit energy consumption, recall, precision, F1 score, and Matthews Correlation Coefficient (MCC) of SmartLoRaML technique is provided. In furtherance, an improved evaluation of classifiers, specifically Support Vector Machine (SVM) and Decision Tree (DT), within the framework of [11] is conducted by utilizing the same simulation environment and dataset as [11], employing stratified cross-validation (CV) mirroring the methodology of SmartLoRaML. Subsequently, we compare these results with the simulation outcomes of [11] conducted without the advantage of stratified CV.

4.5.6.1 SmartLoRaML Performance Evaluation

In the simulator, the nodes were scaled from 100 to 1000, and concurrently, the network radius also scaled from 3000 meters to 10000 meters. Randomly generated network topology radius is set to 3000 meters, number of gateways is set to 1 and packet generation rate is set to 0.01 pps, whereas when the radius is increased to 5000 m and above, the number of gateways increases to 3. The table presents the accuracy results of two classifiers, Random Forest and Gradient Boosting, under these varied conditions. The accuracy, expressed in percentage, demonstrates the performance of these classifiers across different combinations of node numbers and network radii. This analysis provides valuable insights into the effectiveness of the classifiers in diverse scenarios, shedding light on their suitability for varying network sizes and spatial extents.

Table II presents the prediction accuracy of various algorithms across different network coverage radii: 3000 meters, 5000 meters, and 7000 meters from the gateway. For each radius, we explored different numbers of deployed LoRa nodes: 100, 500, and 1000 nodes.

Table 4.4: Performance Evaluation of SmartLoRaML

Number of nodes	Radius (m)	SmartLoRaML Classifiers	Accuracy	Precision	Recall	F1-Score	MCC
100	3000	Random Forest	87.79	0.9443	0.89	0.93	0.12
		Gradient Boosting	96.64	0.9680	0.86	0.94	0.11
500	3000	Random Forest	76.88	0.823	0.81	0.86	0.25
		Gradient Boosting	85.93	0.84	0.98	0.93	0.12
1000	3000	Random Forest	72.65	0.754	0.77	0.81	0.359
		Gradient Boosting	78.41	0.77	0.94	0.87	0.371
100	5000	Random Forest	88.74	0.951	0.88	0.937	0.112
		Gradient Boosting	97.06	0.982	0.886	0.985	0.14
500	5000	Random Forest	76.34	0.742	0.754	0.727	0.239
		Gradient Boosting	85.62	0.77	0.75	0.741	0.14
1000	5000	Random Forest	71.79	0.742	0.799	0.787	0.345
		Gradient Boosting	77.85	0.777	0.810	0.878	0.360
100	7000	Random Forest	86.96	0.936	0.865	0.897	0.390
		Gradient Boosting	93.45	0.940	0.865	0.911	0.390
500	7000	Random Forest	75.41	0.799	0.755	0.773	0.317
		Gradient Boosting	82.95	0.799	0.755	0.760	0.399
1000	7000	Random Forest	72.44	0.752	0.729	0.737	0.422
		Gradient Boosting	75.80	0.753	0.729	0.729	0.429
100	10000	Random Forest	87.89	0.939	0.869	0.900	0.658
		Gradient Boosting	88.94	0.891	0.869	0.865	0.495

4.5.6.2 SmartLoRaML Performance Evaluation –Accuracy

As mentioned in the above-mentioned table, the accuracy of SmartLoraML classifiers is visualized as under for various radius 3000m ,5000m, 7000m and 10000m.

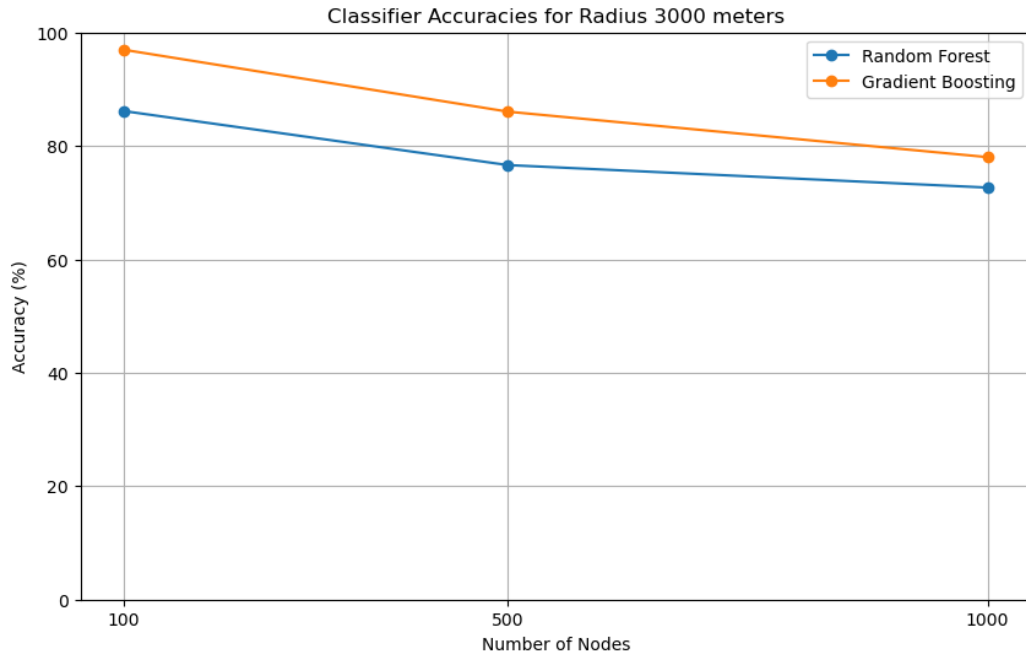


Figure 4-0-5: SmartLoRaML Classifiers Accuracy at 3000m

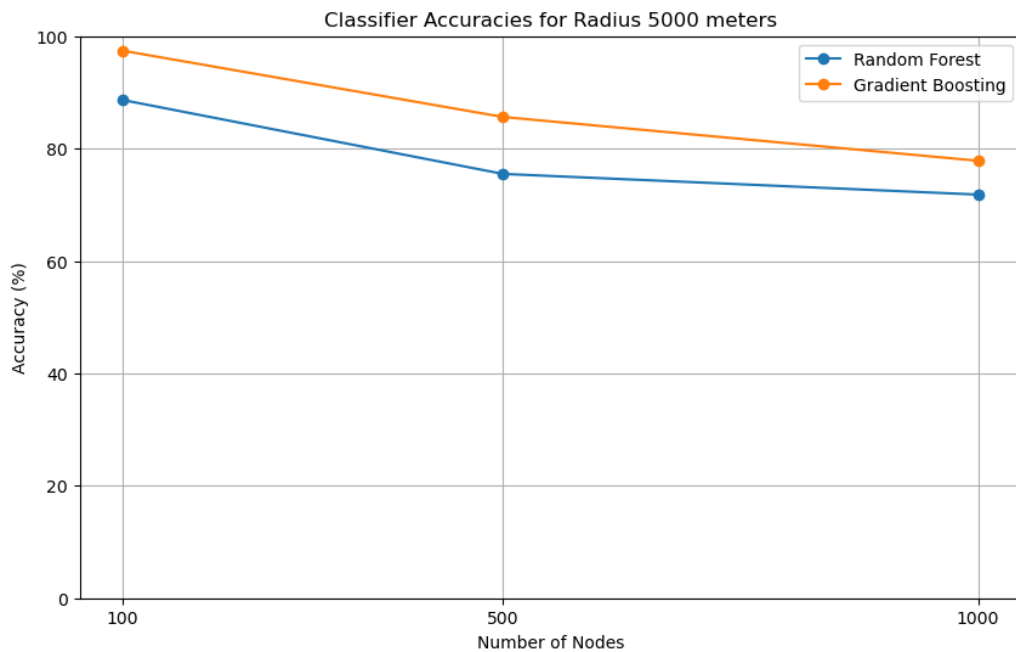


Figure 4-0-6: SmartLoRaML Classifiers Accuracy at 5000m

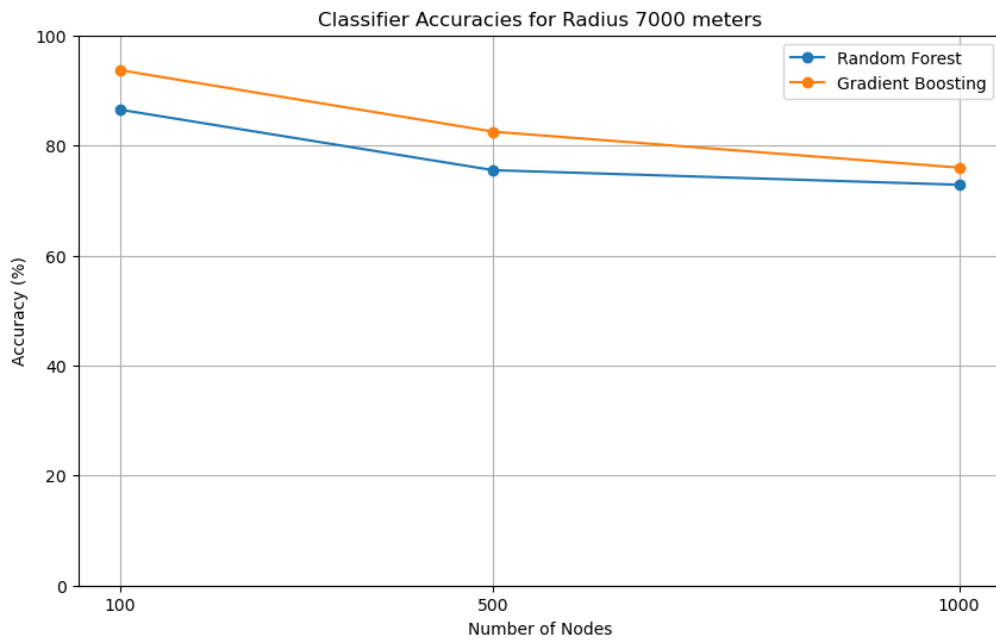


Figure 4-0-7: SmartLoRaML Classifiers Accuracy at 7000m

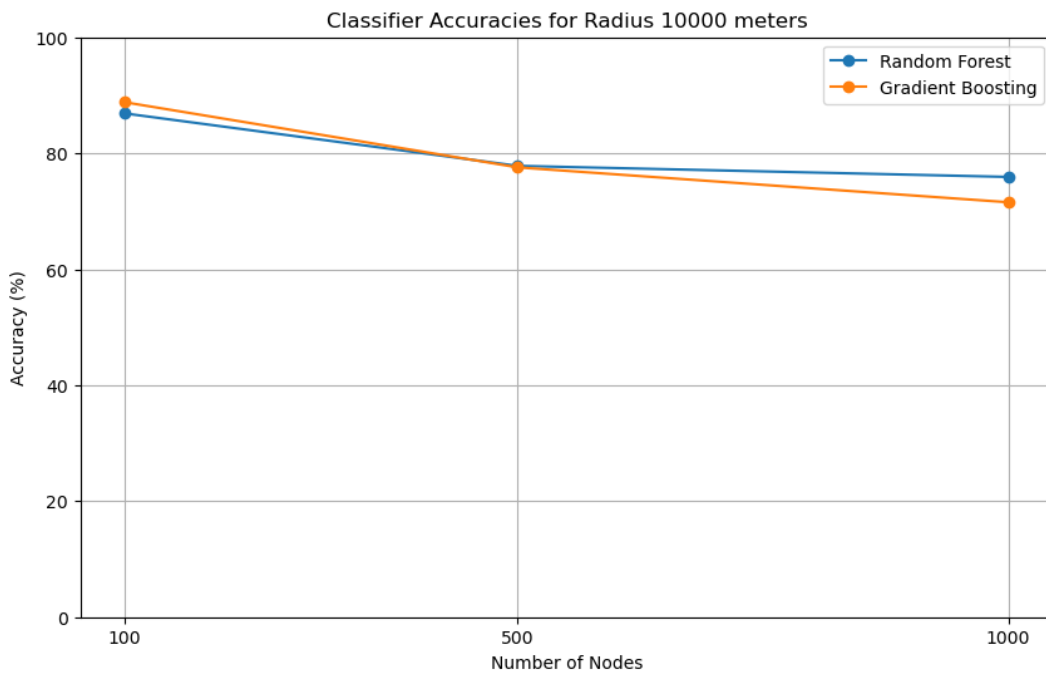


Figure 4-0-8: SmartLoRaML Classifiers Accuracy at 10000m

4.5.6.3 SmartLoRaML Performance Evaluation – Recall & Precision

The analysis of precision and recall across different network configurations, including varying radii and node densities, provides valuable insights into SmartLoRaML classifier performance. Gradient Boosting demonstrates higher precision values, indicating strong accuracy in positive predictions. However, this sometimes corresponds to lower recall, implying potential misses in identifying positive cases (leading to false negatives). In contrast, Random Forest exhibits a more balanced performance in terms of precision and recall across diverse network settings.

A bar chart has been created illustrating the precision and recall performance of both classifiers separately. This visualization showcases how each classifier performs in terms of precision and recall across different network configurations, such as varying radii and node densities. The bar chart provides a clear comparison of the strengths and weaknesses of Random Forest and Gradient Boosting in LoRaWAN network simulations, highlighting their respective precision-recall trade-offs and overall performance characteristics. This graphical representation enhances the understanding of classifier performance and aids in the evaluation and selection of suitable classifiers for real-world LoRaWAN applications.

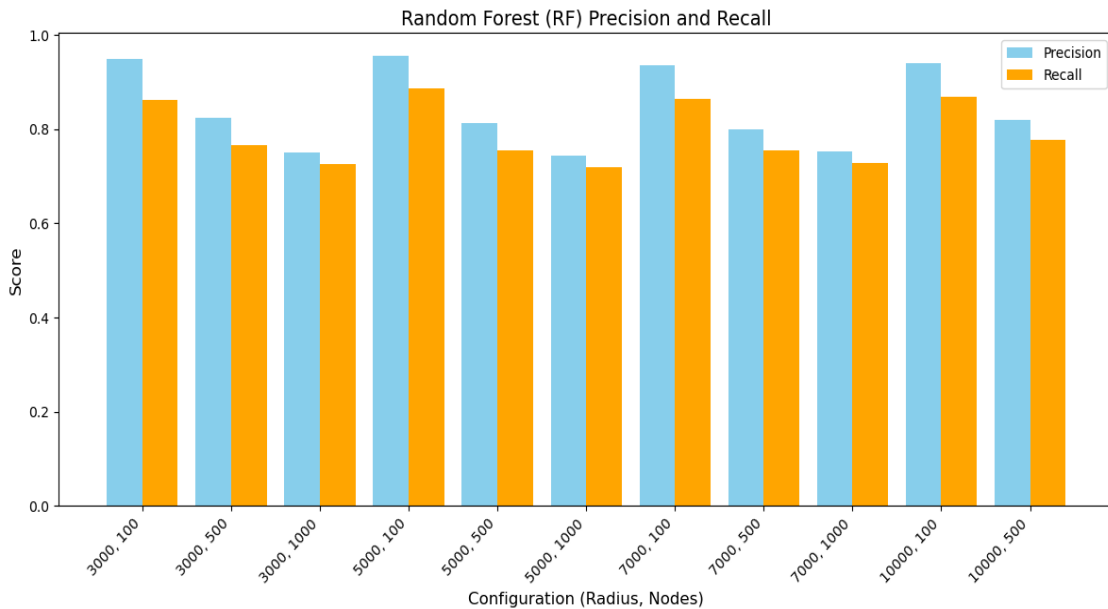


Figure 4-0-9: Precision and Recall Bar Chart – SmartLoRaML Classifier RF

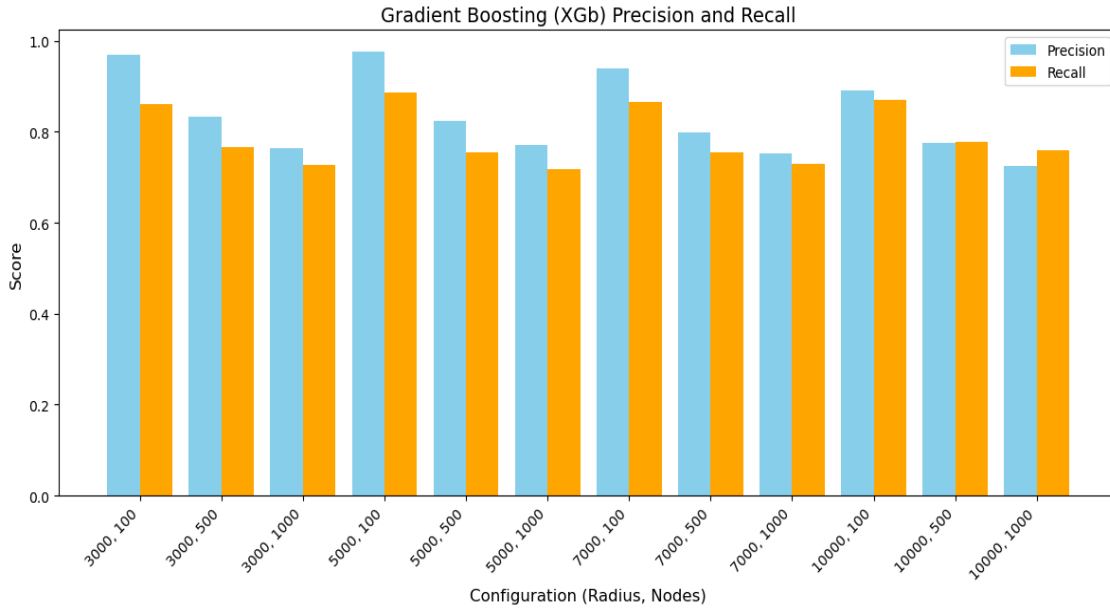


Figure 4-0-10: Precision and Recall Bar Chart – SmartLoRaML Classifier XGB

4.5.6.4 SmartLoRaML Performance Evaluation – F1 Score

The F1 score values calculated for both Random Forest and Gradient Boosting classifiers provide additional insights into their performance within the context of LoRaWAN network simulations across varying radii and node densities. Based on the analysis of F1 scores across different radius and node configurations, Gradient Boosting consistently demonstrates superior performance compared to Random Forest. The trend of decreasing performance with increasing radius is evident for both classifiers, although more pronounced for Gradient Boosting. Specifically, when considering smaller node counts, such as 100, Gradient Boosting consistently outshines Random Forest across various radii settings. Interestingly, Random Forest's performance shows a slight improvement with an increase in the number of nodes, particularly noticeable in configurations with smaller radii. These observations highlight the nuanced impact of radius and node parameters on the classification performance of these algorithms, emphasizing the advantages of Gradient Boosting in scenarios with varying spatial scales and node capacities.

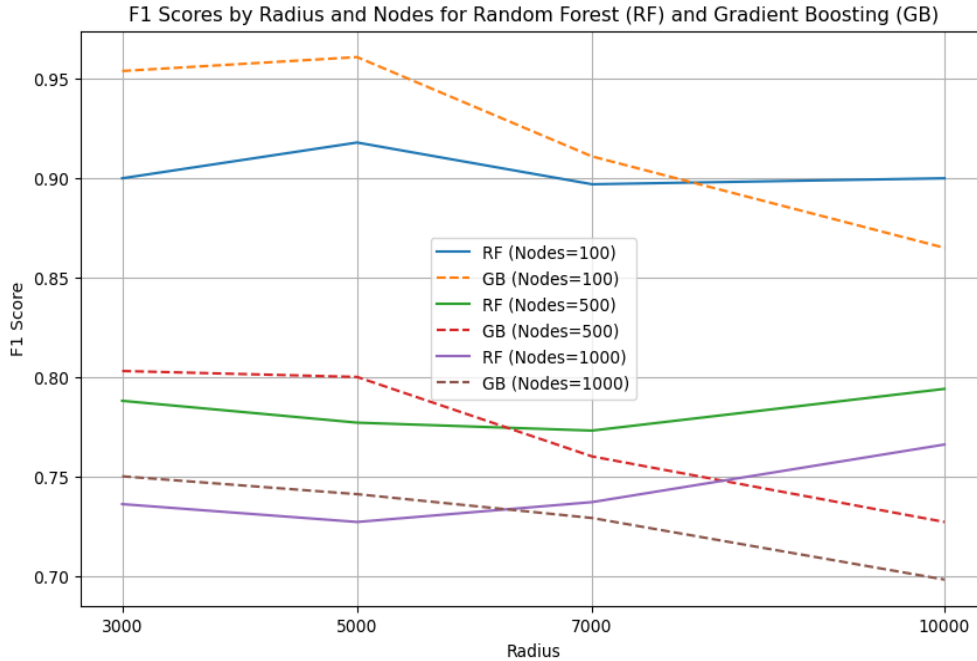


Figure 4-0-11: F1 score line chart – SmartLoRaML Classifiers

4.5.6.5 SmartLoRaML Performance Evaluation – MCC Score

The Matthews Correlation Coefficient (MCC) values calculated for both Random Forest and Gradient Boosting classifiers provide additional insights into their performance within the context of LoRaWAN network simulations across varying radii and node densities. The MCC measures the quality of binary classifications, considering both true positive and true negative predictions while accounting for imbalanced datasets.

Analyzing the MCC values reveals that both classifiers exhibit varying degrees of effectiveness based on network parameters. Generally, higher MCC values indicate better classification performance. Notably, Random Forest tends to achieve higher MCC values compared to Gradient Boosting across most simulation configurations, especially at larger radii and node densities.

The "SmartLoRaML" technique, which leverages both Random Forest and Gradient Boosting classifiers, likely combines the strengths of these algorithms to enhance overall classification accuracy and reliability in LoRaWAN applications. By utilizing ensemble learning techniques, such as combining predictions from multiple classifiers, SmartLoRaML aims to optimize performance and mitigate individual classifier limitations.

Furthermore, the comparison of classifiers using a scatter plot can visually highlight the differences in their performance across various simulation scenarios. A scatter plot plotting MCC values for Random Forest versus Gradient Boosting can provide a clearer understanding of which classifier consistently outperforms the other under different network conditions.

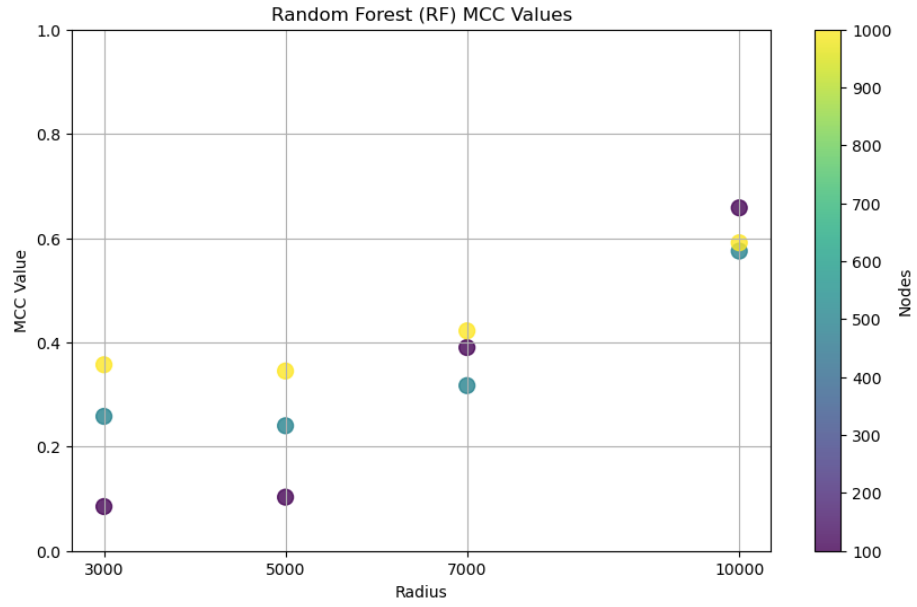


Figure 4-0-12: MCC score Scatter plot – SmartLoRaML Classifier RF

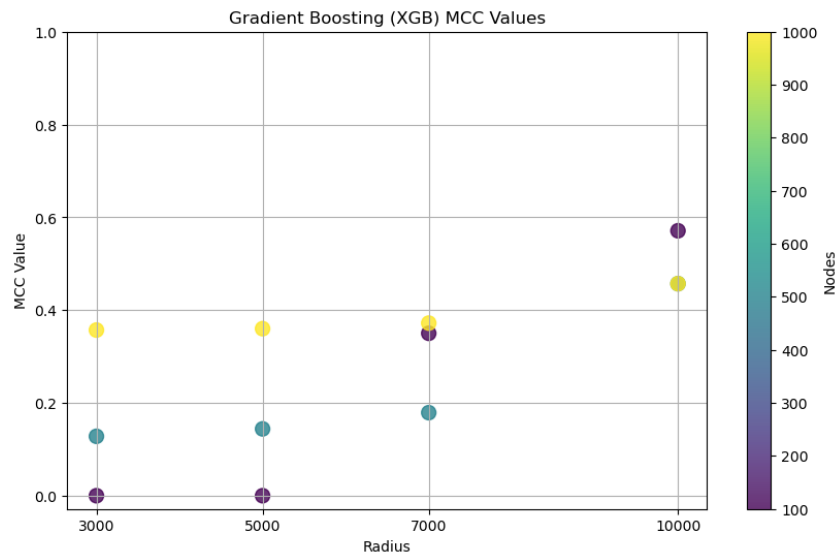


Figure 4-0-13: MCC score Scatter plot – SmartLoRaML Classifier XGB

4.5.6.6 SmartLoRaML Performance Evaluation – PDR

Table 4.5: Performance Evaluation – PDR of SmartLoRaML

Number of nodes	Radius (m)	SmartLoRaML Classifiers	Packet Delivery Ratio
100	3000	Random Forest	97.8
		Gradient Boosting	97.3
500	3000	Random Forest	87.9
		Gradient Boosting	86.4
1000	3000	Random Forest	76.0
		Gradient Boosting	71.9
100	5000	Random Forest	97.6
		Gradient Boosting	97.1
500	5000	Random Forest	88.7
		Gradient Boosting	85.2
1000	5000	Random Forest	78.7
		Gradient Boosting	71.2
100	7000	Random Forest	98
		Gradient Boosting	97.4
500	7000	Random Forest	89.5
		Gradient Boosting	87.3
1000	7000	Random Forest	80.5
		Gradient Boosting	75.9
100	10000	Random Forest	97.9
		Gradient Boosting	98
500	10000	Random Forest	90.9
		Gradient Boosting	90.9
1000	10000	Random Forest	81.6
		Gradient Boosting	81.3

The evaluation of Packet Delivery Ratio (PDR) across different node and radius configurations provides valuable insights into the performance attributes of Random Forest and Gradient Boosting classifiers within the SmartLoRaML technique. Random Forest consistently demonstrates higher PDR values compared to Gradient Boosting across various settings, as indicated by higher median values and upper quartiles in the box plot representation in the figure 4.12. Despite generally lower PDR values, Gradient Boosting exhibits relatively consistent performance across different configurations, showing less variability compared to Random Forest. Both classifiers show a notable decrease in PDR as the coverage radius increases, suggesting challenges associated with maintaining reliable packet delivery over larger areas. Additionally, increasing the number of nodes tends to result in reduced PDR for both classifiers, potentially due to increased network congestion or interference. Overall, these findings underscore the influence of node count

and coverage radius on classifier performance in wireless communication scenarios, emphasizing the need for further investigation into optimizing PDR under varying network conditions.

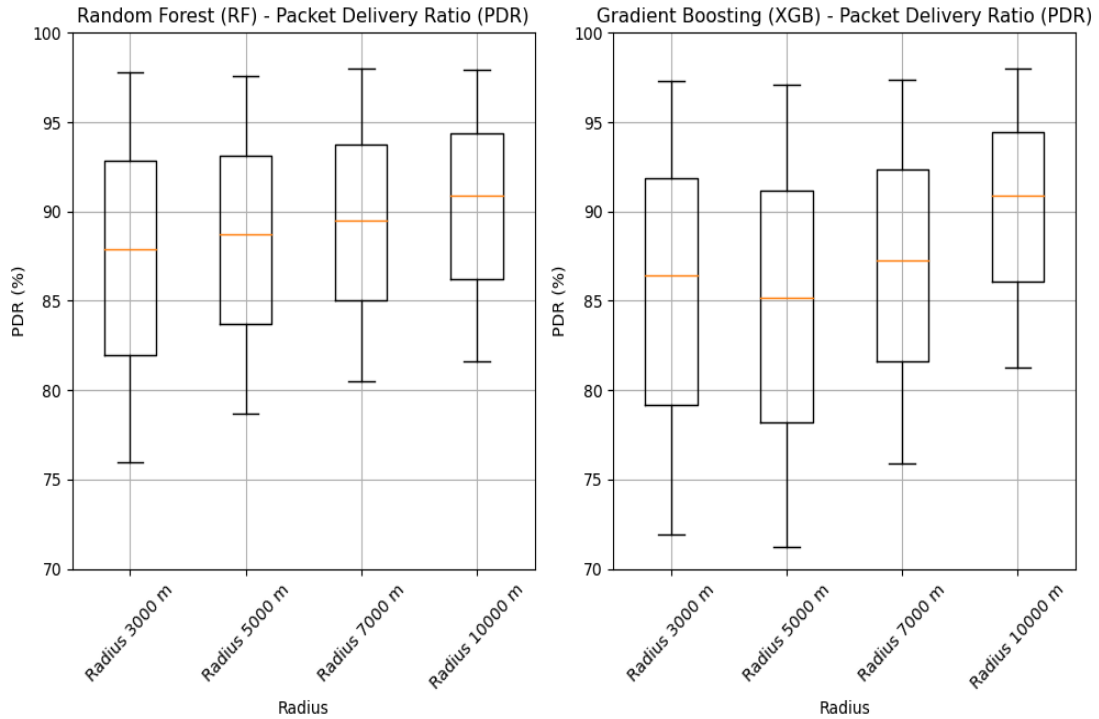


Figure 4-0-14: Box Plot of SmartLoRaML Classifiers

4.5.6.7 SmartLoRaML Performance Evaluation – Transmit Energy Consumption Plots

In this section, transmit energy consumption of varying nodes is given for three different area radii of 3000, 5000 and 7000 meters away from the gateway. In furtherance, a comparison of Random SF which is the inbuilt SF allocation technique of LoRa Simulator with SmartLoRa Classifiers is given. It is evident from the figure 4.13 that SmartLoRa Classifiers portray far better results than Random SF technique and within the SmartLoRa Classifiers, Gradient Boosting gives better results than Random Forest.

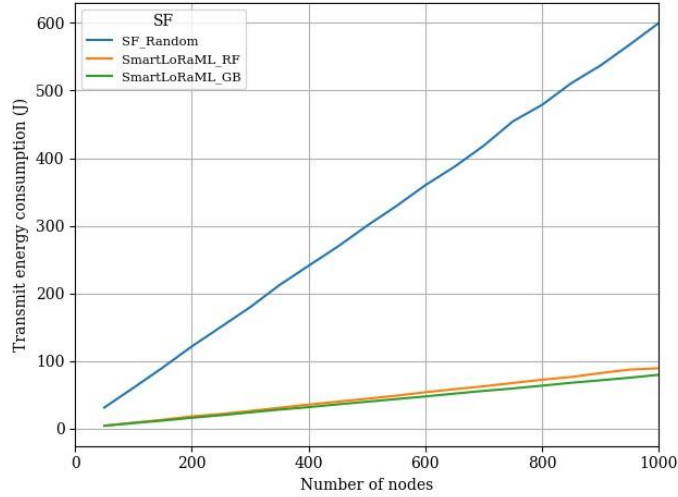


Figure 4-0-15: Transmit Energy Consumption of SmartLoRa Classifiers

4.5.6.8 Enhanced Classifier Evaluation Using Stratified Cross-Validation: A Comparison with Previous Studies

In this section, we enhance the assessment of classifiers, like Support Vector Machine (SVM) and Decision Tree (DT), within the framework of [11]. We do this by using the same simulation setup and data as [11], but with a method called stratified cross-validation (CV), mirroring SmartLoRaML methodology. Afterward, we compare these findings with the results from [11] that didn't use stratified CV.

Table 4.6: Performance Evaluation Comparison – PDR of SmartLoRaML

Number of nodes	Radius(m)	Previous DTC - Accuracy	DTC with Stratified Cross Validation Accuracy	Previous SVM Accuracy	SVM with Stratified Cross Validation Accuracy
100	3000	86.0	87.42	82.4	83.6
500		67.3	69.29	70.4	70.34
1000		70.4	69.53	71.7	71.22
100	5000	84.5	88.72	79.5	84.58
500		67.3	68.72	69.00	70.59
1000		69.5	68.43	71.1	70.15
100	7000	84.5	85.79	79.5	80.89
500		67.7	67.37	70.6	69.58
1000		69.2	69.26	71.2	71.05
100	10000	83.8	86.90	79.2	83.07
500		70.7	72.28	74.4	74.10
1000		74.3	73.35	70.7	75.40

The analysis of decision tree classifier (DTC) and support vector machine (SVM) accuracies highlights the significant impact of employing stratified cross-validation (CV) on improving model performance. In previous studies, without stratified CV, both DTC and SVM accuracies exhibited fluctuations, particularly notable when transitioning from 100 to 500 nodes where accuracy temporarily dropped before increasing again. However, with the application of stratified CV, the results demonstrate more consistent and enhanced performance. Stratified CV ensures that each fold of the data preserves the proportion of classes, which can mitigate issues related to imbalanced datasets and improve generalization. Consequently, the utilization of stratified CV leads to more stable accuracy trends and consistently higher performance levels across different configurations of nodes and radius values for both DTC and SVM. This observation is reinforced by the line chart depicting these accuracies at radius values of 3000m, 5000m, 7000m, and 10000m, where the trends are more favorable and show clearer improvements when stratified CV is employed. This underscores the importance of employing robust validation strategies like stratified cross-validation to achieve more reliable and improved machine learning model outcomes. Further exploration of optimal parameter settings within the context of stratified CV could yield even more refined and dependable results in future studies.

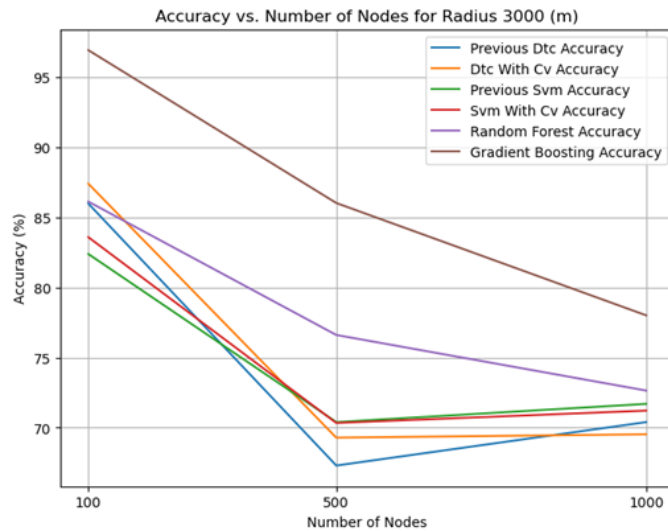


Figure 4-0-16: Performance Evaluation of SmartLoRa Classifiers with SVM and DT and Stratified SVM and DT Classifiers at radius 3000m

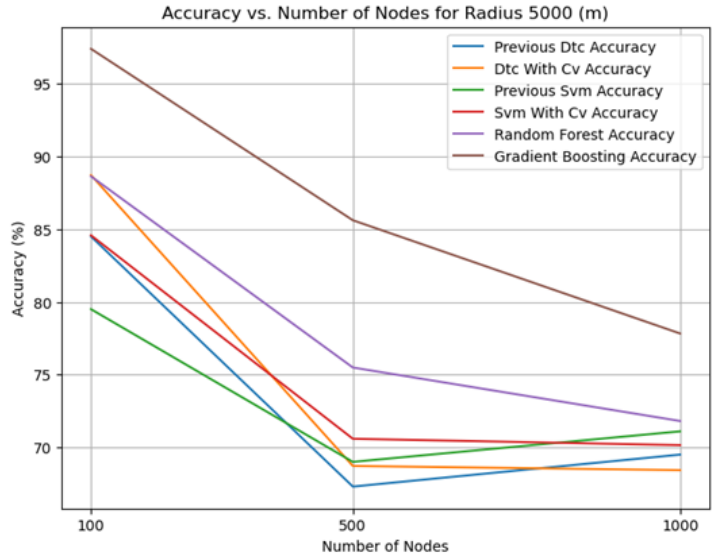


Figure 4-0-17: Performance Evaluation of SmartLoRa Classifiers with SVM and DT

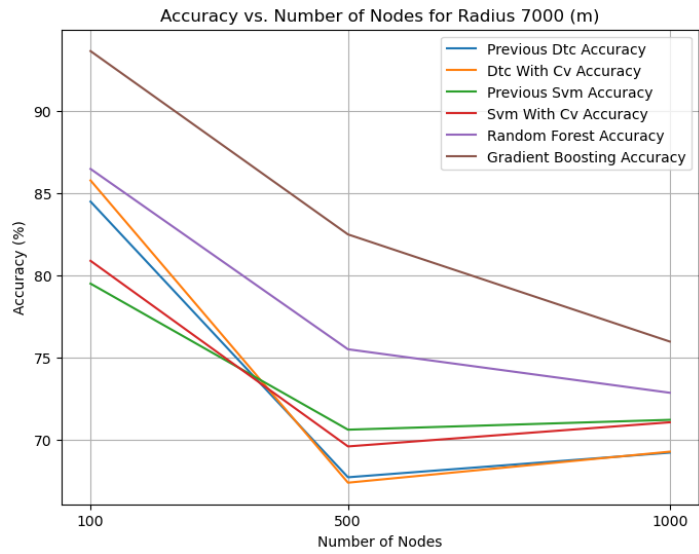


Figure 4-0-18: Performance Evaluation of SmartLoRa Classifiers with SVM and DT and Stratified SVM and DT Classifiers at radius 7000m

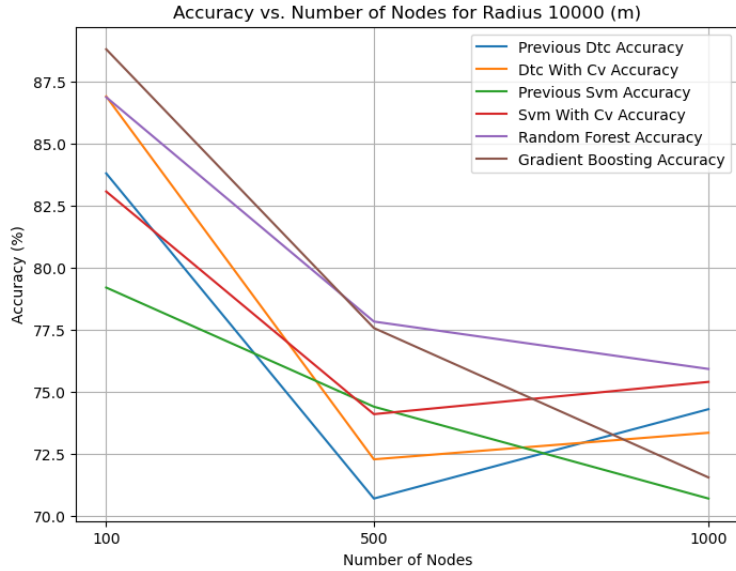


Figure 4-19: Performance Evaluation of SmartLoRa Classifiers with SVM and DT and Stratified SVM and DT Classifiers at radius 10000m

4.6 Conclusion and Future works

The SmartLoRaML technique, utilizing Random Forest and Gradient Boosting for SF allocation optimization in LoRaWAN, demonstrates promising performance across varying network conditions. The evaluation underscores the impact of node density and radius on classifier effectiveness and Packet Delivery Ratio (PDR). At smaller radii with lower node densities, both classifiers achieve high precision and recall, with Random Forest exhibiting consistent accuracy. However, performance diminishes with increased radius due to network congestion. SmartLoRaML improves transmit energy consumption compared to random SF allocation methods. Future advancements could investigate a classifier that directly predicts the optimal spreading factor (SF), thereby eliminating the intermediate step of generating transmission outcome categories. Additionally, the intelligent method could be expanded to include optimization of transmission power for further efficiency gains.

REFERENCES

- [1] Raza, U., Kulkarni, P. and Sooriyabandara, M. (2017). Low Power Wide Area Networks: An Overview, *IEEE Communications Surveys Tutorials*, 19(2),855–873.
- [2] Finnegan, J. and Brown, S. (2018). A Comparative Survey of LPWA Networking, *CoRR*, abs/1802.04222.
- [3] Sigfox (2017). Sigfox Technical Overview, Technical Report, Sigfox, Labège, France. ? *IEEE Wireless Communications Letters*, 6(2), 187–196.
- [4] Georgiou, O. and Raza, U. (2017). Low Power Wide Area Network Analysis: Can LoRa Scale? *IEEE Wireless Communications Letters*, 6(2), 162–165.
- [5] S. Devalal and A. Karthikeyan, "LoRa Technology - An Overview," *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, Coimbatore, India, 2018, pp. 284-290, doi: 10.1109/ICECA.2018.8474715.
- [6] H. U. Rahman, M. Ahmad, H. Ahmad and M. A. Habib, "LoRaWAN: State of the Art, Challenges, Protocols and Research Issues," *2020 IEEE 23rd International Multitopic Conference (INMIC)*, Bahawalpur, Pakistan, 2020, pp. 1-6, doi: 10.1109/INMIC50486.2020.9318170.
- [7] J. de Carvalho Silva, J. J. P. C. Rodrigues, A. M. Alberti, P. Solic and A. L. L. Aquino, "LoRaWAN — A low power WAN protocol for Internet of Things: A review and opportunities," *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, Split, Croatia, 2017, pp. 1-6.
- [8] A. Lavric and A. I. Petrariu, "LoRaWAN communication protocol: The new era of IoT," *2018 International Conference on Development and Application Systems (DAS)*, Suceava, Romania, 2018, pp. 74-77, doi: 10.1109/DAAS.2018.8396074.
- [9] M. Jouhari, N. Saeed, M. -S. Alouini and E. M. Amhoud, "A Survey on Scalable LoRaWAN for Massive IoT: Recent Advances, Potentials, and Challenges,"

- in *IEEE Communications Surveys & Tutorials*, vol. 25, no. 3, pp. 1841-1876, thirdquarter 2023, doi: 10.1109/COMST.2023.3274934.
- [10] A. J. Wixted, P. Kinnaird, H. Larijani, A. Tait, A. Ahmadinia and N. Strachan, "Evaluation of LoRa and LoRaWAN for wireless sensor networks," *2016 IEEE SENSORS*, Orlando, FL, USA, 2016, pp. 1-3, doi: 10.1109/ICSENS.2016.7808712.
- [11] T. Yatagan and S. Oktug, "Smart spreading factor assignment for lorawans," in 2019 IEEE Symposium on Computers and Communications (ISCC), 2019, pp. 1–7.
- [12] B. Christos, G. Apostolos, S. S. A. Katsampiris, and P. Nikolaos, "Spreading factor analysis for lora networks: A supervised learning approach," 2021, pp. 344–353.
- [13] Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., and Ayyash, M., "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," in *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347-2376, Fourthquarter 2015, doi: 10.1109/COMST.2015.2444095.
- [14] D. Davcev, K. Mitreski, S. Trajkovic, V. Nikolovski and N. Koteli, "IoT agriculture system based on LoRaWAN," *2018 14th IEEE International Workshop on Factory Communication Systems (WFCS)*, Imperia, Italy, 2018, pp. 1-4, doi: 10.1109/WFCS.2018.8402368. [15] The Things Industries. (n.d.). Large-Scale LoRaWAN Deployments.
- [16] K. -H. Phung, H. Tran, Q. Nguyen, T. T. Huong and T. -L. Nguyen, "Analysis and assessment of LoRaWAN," *2018 2nd International Conference on Recent Advances in Signal Processing, Telecommunications & Computing (SigTelCom)*, Ho Chi Minh City, Vietnam, 2018, pp. 241-246, doi: 10.1109/SIGTELCOM.2018.8325799.
- [17] Chéour, R., Jmal, M.W., Lay-Ekuakille, A., Derbel, F., Kanoun, O., and Abid, M. "Choice of efficient simulator tool for wireless sensor networks." In *Proceedings of the IEEE International Workshop on Measurements & Networking (M&N)*, Naples, Italy, 7–8 October 2013; pp. 210–213.
- [18] Das, A.P., and Thampi, S.M. "Simulation tools for underwater sensor networks: A survey." *Netw. Protoc. Algorithms* 2016, 8, 41–55.
- [19] Abuarqoub, A., Hammoudeh, M., Alfayez, F., and Aldabbas, O. "A survey on wireless sensor networks simulation tools and testbeds." *Sens. Transducers Signal Cond. Wirel. Sens. Netw. Adv. Sens. Rev.* 2016, 3, 283–302.
- [20] A.S., and Jain, A. "A survey on wireless network simulators." *Bull. Electr. Eng. Inform.* 2017, 6, 62–69.

- [21] Mouiz, A., Badri, A., Baghdad, A., Ballouk, A., and Sahel, A. "Analysis of Modeling Performance and Simulation Tools for Wireless Sensor Networks." *Int. J. Comput. Appl. Technol. Res. (JCATR)* 2017, 6, 9–12.
- [22] Pesic, D., Radivojevic, Z., and Cvetanovic, M. "A survey and evaluation of free and open source simulators suitable for teaching courses in wireless sensor networks." In *Proceedings of the 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia, 22–26 May 2017; pp. 895–900.
- [23] Vasanthi, V. "Simulators and emulators used for wireless sensor network." *J. Adv. Res. Comput. Comm. Eng.* 2017, 6, 171–175.
- [24] Sarkar, N.I., and Halim, S.A. "A review of simulation of telecommunication networks: Simulators, classification, comparison, methodologies, and recommendations," *Cyber J. Multidiscip. J. Sci. Technol. Spec. Issue J. Sel. Areas Telecommun. (JSAT)*, vol. 2, pp. 10-17, 2011.
- [25] Moravek, P., Komosny, D., and Simek, M. "Specifics of WSN simulations," *ElektroRevue*, vol. 2, pp. 15-21, 2011.
- [26] Fahmy, H. M. A. "Simulators and emulators for WSNs," in *Wireless Sensor Networks*, Singapore: Springer, 2016, pp. 381-491.
- [27] Khan, M. Z., Askwith, B., Bouhafis, F., and Asim, M. "Limitations of simulation tools for large-scale wireless sensor networks," in *Proceedings of the IEEE Workshops of International Conference on Advanced Information Networking and Applications*, Biopolis, Singapore, 22-25 March 2011, pp. 820-825.
- [28] Živković, M., Nikolić, B., Protić, J., and Popović, R. "A survey and classification of wireless sensor networks simulators based on the domain of use," *Adhoc Sens. Wirel. Netw.*, vol. 20, pp. 1-30, 2014.
- [29] Owczarek, P., and Zwierzykowski, P. "Review of simulators for wireless mesh networks," *J. Telecommun. Inf. Technol.*, vol. 3, pp. 82-89, 2014.
- [30] Nayyar, A., and Singh, R. "A comprehensive review of simulation tools for wireless sensor networks (WSNs)," *J. Wirel. Netw. Commun.*, vol. 5, pp. 19-47, 2015.
- [31] Sharif, M., and Sadeghi-Niaraki, A. "Ubiquitous sensor network simulation and emulation environments: A survey," *J. Netw. Comput. Appl.*, vol. 93, pp. 150-181, 2017. [CrossRef]
- [32] Ojie, E., and Pereira, E. "Simulation tools in internet of things: A review," in *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, Liverpool, UK, 17-18 October 2017, pp. 1-7.

- [33] Pandey, D., and Kushwaha, V. "Experimental tools and techniques for wireless sensor networks," *Int. J. Recent Technol. Eng. (IJRTE)*, vol. 8, pp. 1674-1684, 2019.
- [34] Kulkarni, V., Narayana, V. L., and Sahoo, S. K. "A survey on interference avoiding methods for wireless sensor networks working in the 2.4 GHz frequency band," *J. Eng. Sci. Technol. Rev.*, vol. 13, pp. 59-81, 2020. [CrossRef]
- [35] Bouras, C., Gkamas, A., Katsampiris Salgado, S.A., and Kokkinos, V. "Comparison of LoRa Simulation Environments," in *Proceedings of the Advances on Broad-Band Wireless Computing, Communication and Applications, BWCCA 2019, Antwerp, Belgium, 7–9 November 2020, Lecture Notes in Networks and Systems*, vol. 97, Springer, Cham, Switzerland, 2020, pp. 374–385.
- [36] Khan, F.H., and Portmann, M. "Experimental Evaluation of LoRaWAN in NS-3," in *Proceedings of the 28th International Telecommunication Networks and Applications Conference (ITNAC), Sydney, NSW, Australia, 21–23 November 2018*, pp. 1–8.
- [37] Luvisotto, M., Tramarin, F., Vangelista, L., and Vitturi, S. "On the Use of LoRaWAN for Indoor Industrial IoT Applications," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–11, 2018. [CrossRef]
- [38] Ta, D. T., Khawam, K., Lahoud, S., Adjih, C., and Martin, S. "LoRa-MAB: A flexible simulator for decentralized learning resource allocation in IoT networks," in *Proceedings of the 12th IFIP Wireless and Mobile Networking Conference (WMNC), Paris, France, Sep. 2019*, pp. 55–62.
- [39] Zorbas, D., Caillouet, C., Hassan, K. A., and Pesch, D. "Optimal data collection time in LoRa networks—A time-slotted approach," *Sensors*, vol. 21, no. 4, p. 1193, 2021. [CrossRef]
- [40] Pop, A.-I., Raza, U., Kulkarni, P., and Sooriyabandara, M. "Does bidirectional traffic do more harm than good in LoRaWAN based LPWA networks?" in *Proceedings of the GLOBECOM2017—2017 IEEE Global Communications Conference, Singapore, Dec. 2017*, pp. 1–6.
- [41] Casals, L., Gomez, C., and Vidal, R. "The SF12Well in LoRaWAN: Problem and End-Device-Based Solutions," *Sensors*, vol. 21, no. 19, p. 6478, 2021. [CrossRef] [PubMed]
- [42] G., Ottoy, G., and van der Perre, L. "Cross-Layer Framework and Optimization for Efficient Use of the Energy Budget of IoT Nodes," in *Proceedings of the 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, Apr. 2019*, pp. 1–6

- [43] Zorbas, D., Kotzanikolaou, P., and Pesch, D. "TS-LoRa: Time-slotted LoRaWAN for the industrial internet of things," *Comput. Commun.*, vol. 153, pp. 1–10, 2020. [CrossRef]
- [43] Bounceur, A., Marc, O., Lounis, M., Soler, J., Clavier, L., Combeau, P., Vauzelle, R., Belmeguenai, A., and Ladsous, L. "Intensive LoRa Simulations using Multiple Interfering Spreading Factors in Ns-3," in *Proceedings of the 24th IEEE Symposium on Computers and Communications (ISCC)*, Barcelona, Spain, 29 June–3 July 2019, pp. 1–6.
- [44] Zennaro, M., and D'Alessandro, S. "On using the Lorawan simulator to compare mac layer protocols," in *Proceedings of the 2017 IEEE International Conference on Computing, Networking and Communications (ICNC)*, Silicon Valley, CA, USA, Jan. 2017, pp. 1–6.
- [45] Pêgas, M. A., Coimbra, A., Martins, J. C., Monteiro, E., and Krsul, I. V. "A Comparative Study of LoRaWAN Simulators," *Journal of Computer Networks and Communications*, vol. 2020, p. 8895146, 2020. [CrossRef]
- [46] Toledo, J., Da Costa, I., Souto, E., and Sadok, D. "Energy efficiency and network performance analysis of LoRa networks," *Wireless Communications and Mobile Computing*, vol. 2019, p. 8753461, 2019. [CrossRef]