PERFORMANCE ENHANCEMENT OF LORA NETWORKS FOR SMART CITY APPLICATIONS

by

Shobhit Aggarwal

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Electrical Engineering

Charlotte

2023

Approved by:

_____
Dr. Asis Nasipuri

_____
Dr. Jim Conrad

_____
Dr. Ahmed Arafa

_____
Dr. Yasin Raja

ABSTRACT

SHOBHIT AGGARWAL. Performance Enhancement of LoRa Networks for Smart City Applications. (Under the direction of DR. ASIS NASIPURI)

The Internet of Things (IoT) provides unlimited possibilities for smart city applications. However, energy efficiency and scalability continue to be key considerations for the development of low-cost wireless networks for meeting the needs of the emerging world of IoT. Recent developments in low-power wide area networks (LPWAN) promise to meet these requirements by achieving long communication ranges at low data rates without increasing the energy cost. Consequently, LPWAN technologies are rapidly gaining prominence in the development of IoT networks in comparison to legacy WLANs. LPWANs address the challenges of legacy wireless technologies that use multi-hop mesh networking for increasing connectivity and coverage. Long Range (LoRa) technology is receiving increasing attention in recent years for addressing the challenges of providing wireless connections to a large number of end devices in the field of IoT. LoRa has become the most prominent LPWAN standard due to its long transmission range, low power consumption, and large network capacity. Despite these benefits, LoRa networks may not be able to achieve their full potential unless additional improvements are achieved in the network scalability domain. Specifically, the probability of success under heavy network traffic loads or a large number of end devices needs to be improved.

In this dissertation, we explore the causes of performance degradation of LoRa networks and propose several approaches to enhance their performance. Next, we present a novel MAC layer protocol to employ AI tools to make IoT applications smarter. The effectiveness of all the proposed approaches is validated using mathematical analysis as well as via simulations thereby creating the basis for further research in this area.

ACKNOWLEDGEMENTS

I would like to share my heartfelt gratitude to my advisor, Dr. Asis Nasipuri for his continuous support all through my research. His constant motivation, enthusiasm, and immense knowledge of the subject aided in significantly developing my logical and reasoning faculties which are critical for research. If it was not for his continuous support, this work would not have been a reality.

I would like to again offer my heartfelt appreciation to Dr. Asis Nasipuri, the Wireless Sensor Networks lab's principal investigator, for providing the critical infrastructure that enabled me to conduct my research. I am grateful to all of my lab colleagues and co-authors who have always supported and assisted me.

I would also like to extend my sincere appreciation to my committee members Dr. Jim Conrad, Dr. Ahmed Arafa, and Dr. Yasin Raja. Their feedback, constructive criticism, and insightful suggestions have been instrumental in improving the quality of my research.

I would at this point like to express my gratitude to the ECE department of UNC Charlotte, for providing me with access to resources, facilities, and a supportive academic environment.

I am also grateful to Mr. Joshua Cox and Mr. Peter O Connor from Oxit LLC for giving me the opportunity to gather experience and real-world data for my research.

Finally, I would like to thank my family and friends for cheering me up and standing by me through all times good and bad. Their support and patience have been invaluable

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADR Adaptive Data Rate

AI Artificial Intelligence

BW BandWidth

CR Coding Rate

CSS Chirp Spread Spectrum

DL Downlink

ECG Electrocardiography

ED End Device

FL Federated Learning

GW Gateway

HP High Priority

IoT Internet of Things

LoRa Long Range

LP Low Priority

LPWAN Low Power Wide Area Networks

LSTM Long Short-Term Memory

MAC Medium Access Control

ML Machine Learning

NS Network Server

PAN Personal Area Networks

PCA Principal Component Analysis

PDR Packet Delivery Ratio

PER Packet Error Rate

PLR Packet Loss Rate

PSP Packet Success Probability

QoS Quality of Service

RSSI Received Signal Strength Indicator

SF Spreading Factors

SINR Signal to Interference Noise Ratio

SNR Signal to Noise Ratio

ToA Time on Air

UL Uplink

WLAN Wireless Local Area Network

CHAPTER 1: INTRODUCTION

The Internet of Things (IoT) has had a significant impact on the modern world by connecting a vast array of devices to the Internet, thereby introducing ambient intelligence to the physical realm by embedding sensors, processors, and network connectivity into everyday objects. These devices are capable of gathering data from their environment, processing it, and making decisions based on the obtained insights. When connected to the Internet, organizations can leverage this capability to collect and analyze data, identify inefficiencies, and optimize processes. A significant instance of IoT implementation is highlighted in [3], which presents the benefits of linking industrial machinery to the Internet, resulting in increased efficiency, reliability, and safety. Additionally, IoT devices can be utilized for health and wellness monitoring, as well as improving quality of life, as highlighted in [4]. Recent advancements in low-cost sensor and actuator technologies, in conjunction with the emergence of widespread communication systems, have led to exponential growth in the number of IoT devices.

The application of artificial intelligence (AI) techniques, such as machine learning (ML), can enhance the capabilities of IoT devices by enabling them to learn from data and adjust to changing conditions without explicit programming. This ambient intelligence can make IoT devices more proactive and intelligent, instead of being reactive to predefined commands or rules. For instance, in the domain of energy management, [5] proposed a system that employs sensors and machine learning algorithms to optimize energy consumption in buildings based on user preferences and occupancy patterns. Additionally, [6] proposes a healthcare system that uses wearable devices and machine learning algorithms to monitor the health of elderly patients and detect falls. In the transportation domain, [7] evaluates multiple IoT technologies for Intel-

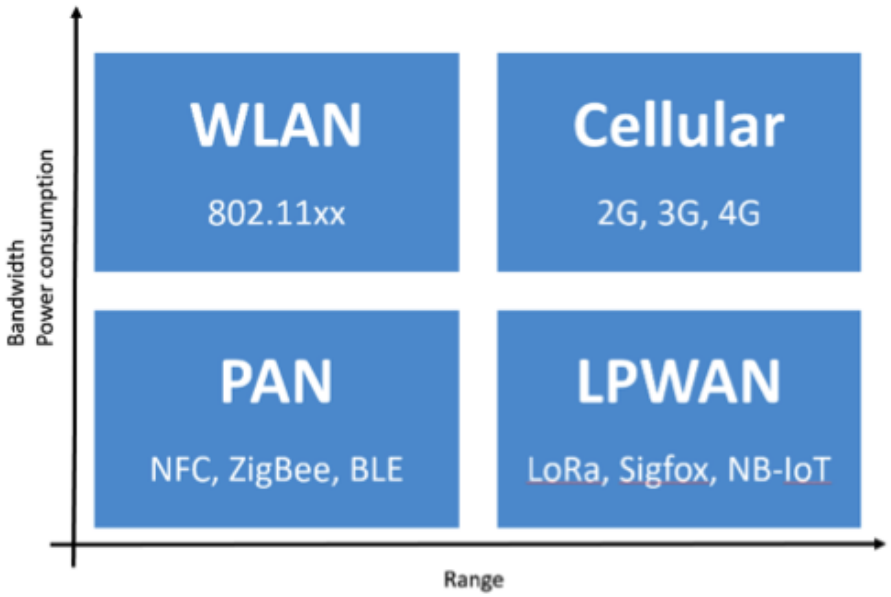


Figure 1.1: Plot of range vs bandwidth and power consumption for various wireless technologies

ligent Transportation Systems (ITS) in Smart Cities scenarios, while also discussing the relevance of machine-learning technologies in this field. Furthermore, [8] proposes a system that integrates IoT sensors and machine learning algorithms to reduce water usage and optimize crop yields.

However, to enable the widespread implementation of IoT, a variety of network infrastructure challenges must be addressed. Backhaul networks are required to connect each device directly to the Internet. Both wired and wireless technologies can be utilized as backhaul networks, but wireless technologies are much more appealing. There are various standard wireless protocols and technologies such as legacy non-cellular technologies, wireless local area networks, and cellular networks that can serve as backhauls. However, the need for low energy consumption for large-scale network deployment remains a key concern.

As depicted in Figure 1.1, non-cellular wireless technologies such as Wi-Fi, Bluetooth, Zigbee, NRF, and ANT are not ideal for connecting low-power devices distributed over large geographical areas. The transmission range of these technologies

is limited to a few hundred meters. Multihop networking principles may be employed to extend their geographical coverage, but there are challenges in terms of reliability when deployed over large areas, such as a citywide network [9]. WLANs are characterized by shorter coverage areas and higher power consumption. Cellular networks such as the Global System for Mobile Communications (GSM) and Long-Term Evolution (LTE) provide wide-area coverage but do not fulfill the energy efficiency requirements of IoT networks.

Recent advancements in radio technologies such as chirp spread spectrum (CSS) have led to the emergence of LPWAN (Low Power Wide Area Network) technologies that have the potential to address the aforementioned challenges. Generally, these technologies are energy-efficient and offer significantly higher transmission ranges (several miles) with lower data rates, making them effective in covering wide areas using a single gateway [10]. These technologies operate in the free unlicensed ISM (Industrial, Scientific, and Medical) band. LPWAN shares the same band that is shared by all "Short Range" devices. Examples of protocols that make use of this band are Z-wave, 802.11ah, 802.15.4g, and 802.15.4 [11]. Due to the prevalence of the ISM band and multiple technologies using the same band simultaneously, various regulations are imposed on these devices, which vary depending on the region [12]. Some mandate the use of LBT (Listen-Before-Talk) mechanisms and others impose duty cycles on communications, thus limiting the throughput of the devices.

LPWAN technologies can be classified into wideband or ultra-narrowband technologies [13]. To minimize the bandwidth consumed by the control plane of the network, LPWAN technologies typically use the star topology, where all nodes communicate directly with the base station. The same duty cycle restrictions also apply to the base stations, making LPWAN technologies ideal for applications that require low downlink traffic without confirmed message delivery [10]. Applications that require autonomous battery-powered nodes, long communication range, long network

lifetime, low throughput, and do not have strict latency requirements are suitable for LPWAN technologies. SIGFOX, Weightless, RPMA, and LoRa are some of the most popular LPWAN technologies that have gained significant momentum in recent years. Among all LPWAN standards, LoRa has several features and characteristics such as a subscription-free model, no limits on the number of transmissions, and ease of network deployment that make it the most prominent of all.

LoRa technology developed by Semtech [14], is a proprietary physical layer technology but offers the flexibility to utilize any of the upper layer protocols. The technology is further developed by Lora Alliance [15]. They have developed an upper layer protocol suite known as "LoRaWAN". The Medium Access Control (MAC) layer of LoRaWAN employs an ALOHA-based protocol, making the technology fairly less complex and simple to implement. However, this simplicity comes at the cost of network performance. The ALOHA-based MAC protocol makes transmissions from LoRa devices susceptible to collisions. Since the devices are free to transmit data without considering transmissions from other devices, this leads to interference under heavy traffic, resulting in performance degradation and ultimately loss of data. Technically, LoRaWAN gateways have the capability to handle hundreds of thousands of devices. However, according to studies in [16][17] the average packet delivery ratio (PDR) drops drastically in these networks when the number of connected devices exceeds a few hundred. Interference or collisions among the transmitted packets is a major reason for this decline. Although LoRa employs the power capture effect to receive transmissions with the highest signal power in case of collisions, it is not capable of maximizing network performance to its full potential.

The objective of this research is to investigate the existing challenges of using LoRaWAN networks in large-scale IoT networks and to propose approaches to improve on those challenges. One of our objectives is to explore the limitations and potential design considerations for LoRa technology that would enable the development of IoT

applications employing AI principles. With these objectives, we focus on the aspects of network scalability and supporting the communication requirements for employing AI principles in LoRa networks. We propose multiple strategies for enhancing the performance of dense LoRa/LoRaWAN networks. The key contributions of this research are outlined as follows:

1. We present a comprehensive analysis of LoRa networks to evaluate their transmission range, power consumption, and scalability. The analysis presents insights regarding the limitations of the legacy system with respect to network capacity, probability of packet success at different distances from the gateway, power consumption, and scalability [18].

2. We propose a solution to improve the scalability of LoRaWAN networks by implementing a dynamic spreading factor allocation strategy. This approach utilizes the orthogonality of spreading factors in LoRa networks to effectively manage collision domains and increase network capacity. Our analysis demonstrates that by optimizing the distribution of devices across various spreading factors, the overall network performance can be significantly improved [19].

3. We propose a novel spreading factor allocation approach that is based on the Quality of Service (QoS) needs of the Internet of Things (IoT) applications. Many IoT applications have specific QoS requirements that must be met by the underlying network for proper functioning. The legacy LoRaWAN protocol serves all devices indiscriminately, without prioritizing any specific group. Our proposed QoS-based spreading factor allocation method not only accommodates the QoS requirements of IoT applications on LoRaWAN networks but also enhances the overall network performance [20].

4. The integration of AI principles and IoT technology has the potential to create highly intelligent systems capable of processing large amounts of data to provide

insights. However, the communication requirements for such systems can be complex and challenging to understand. To gain a better understanding of the communication requirements for ML-based applications using distributed IoT devices, we conducted an evaluation of a representative application [21].

5. We propose FL-LoRaMAC, a novel framework for incorporating AI principles in IoT applications utilizing LoRa-based platforms for smart city deployments. The application of AI in smart cities can enhance the intelligence of these systems through the implementation of learning principles in IoT applications. However, traditional machine learning methods may not be suitable for the unique constraints and characteristics of IoT and LoRa ecosystems. The FL-LoRaMAC framework addresses these challenges by enabling decentralized learning in LoRa-based IoT platforms. The effectiveness of the proposed framework is demonstrated through its application in a healthcare scenario, but it can be applied to any IoT application.

The structure of this dissertation is as follows:

Chapter 2 provides an overview of the LPWAN technology and delves deeper into LoRa technology. It discusses the architecture of LoRaWAN networks, the physical layer, and the legacy MAC layer.

Chapter 3 reviews previous related works done in the field upon which we base our research. We look at several existing works done to evaluate the performance of LoRaWAN networks and approaches for improving the performance of LoRa networks. It also provides hints captivating the reader to question and argue the current approaches.

Chapter 4 presents our first contribution to the performance evaluation of LoRaWAN networks. We build on our understanding of LPWANs, especially LoRa networks to explore the scalability issue.

Chapter 5 proposes the second contribution of finding the optimum distribution of devices for various spreading factors and configuring devices on various spreading factors based on the optimum distribution to maximize network performance. We validate our proposed distribution using a network simulator.

Chapter 6 describes proposed approaches for assigning spreading factors to end nodes to satisfy the QoS requirements for IoT applications.

Chapter 7 presents an evaluation of a representative application that illustrates the advantages and challenges of incorporating AI in the field of IoT. Specifically, the study highlights the benefits of AI in IoT applications, as well as the requirement for large amounts of data for training these models.

Chapter 8 describes the novel challenges of IoT and the LoRa ecosystem for typical machine learning principles. It also proposes a novel framework to facilitate and deploy AI tools in LoRa networks. For validation and demonstration of performance, the proposed framework is applied to an ECG anomaly detection application scenario.

In Chapter 9 we summarize the contributions of this dissertation.

CHAPTER 2: BACKGROUND

2.1 Low Power Wide Area Networks (LPWANs)

Over the last two decades, the emergence of Low Power Wide Area Network (LPWAN) platforms has revolutionized the way low-power wireless sensor devices are connected over extensive geographic areas. Unlike newer wireless communication technologies that operate in higher frequency bands, LPWAN technologies primarily operate in the sub-GHz frequency band.

By utilizing the sub-GHz frequency band, LPWAN technologies achieve longer communication ranges than other wireless communication technologies [10]. Operating in the sub-GHz band allows LPWAN technologies to leverage better penetration through obstacles and attenuation, making them ideal for applications in harsh environments. These technologies can be broadly divided into two categories based on the type of modulation they use:

- Ultra Narrow Band (UNB): Technologies using 25 kHz channel bandwidth.
- Wide-band: Technologies using channels with bandwidth in the order of 125 kHz or 250 kHz. These also feature multiple-user access within a single channel by employing some form of spread spectrum multiple access techniques.

Due to its long communication range and low power consumption, LPWAN standards have been gaining momentum in the past years. A few such LPWAN standards are discussed below. Table 2.1 provides a high-level comparison of various popular LPWAN technologies.

SIGFOX is the first LPWAN technology proposed in the IoT market. It is an ultra-narrowband technology that operates in the ISM band using 868MHz in Europe

Table 2.1: Comparison of Key LPWAN Technologies

Technology	SIGFOX	LoRa	Weightless-P	RPMA
Topology	Star	Usually Star; Mesh Possible	Star	Star
Max. Data Rate per terminal	100bps	50kbps	100kbps	8kbps
Frequency band	Sub-GHz ISM	Sub-GHz ISM	Sub-GHz ISM	2.4GHz band
MAC Layer	ALOHA based	ALOHA based	–	RPMA
Range (Miles)	Urban:2-6 Rural:12-30	Urban:1-3 Rural:7-18	Urban:1	Urban:2
Modulation Technique	Ultra Narrow Band	Spread Spectrum	Narrow Band	Spread Spectrum
Proprietary Aspects	Physical and MAC Layers	Physical Layer	Open Standard	Full Stack
Nodes Per Gateway	<1,000,000	<1,000,000	<500,000	–
Encryption	Not Built-in	AES	AES-128 bit	–

and 902MHz in the US. This technology makes use of star topology and the base station is connected to servers using an IP-based network. The end devices employ Binary Phase Shift Keying (BPSK) modulation to connect to the base stations. As SIGFOX uses ultra narrowband transmission, very low noise levels are experienced and bandwidth is utilized efficiently, as a result, the receiver becomes highly sensitive and consumes ultra-low power.

Initially, SIGFOX supported only uplink communication but evolved into a bidirectional technology later. The number and size of messages over the uplink are limited to 140 12-byte messages per day and for the downlink, only 4 8-byte transmissions per day to conform to the regional regulations on the use of license-free spectrum [22]. That means the base station can not acknowledge each uplink transmission.

Weightless [23] is an open standard managed by the Weightless-Special Interest Group. Three standards have been proposed by the group namely Weightless-N, Weightless-W, and Weightless-P. However, we will focus on the most recently defined

standard which is Weightless-P.

Weightless-P is a non-proprietary physical layer technology. It uses GMSK and QPSK for modulating the signal. These modulating schemes are very well known and are used in various commercial products, hence the end devices do not require a proprietary chipset. Weightless-P, supports narrowband channels of 12.5 kHz, with Frequency Division and Time Division Multiple Access modes. It also incorporates adaptive data rates from 0.2 kbit/s to 100 kbit/s, time-synchronized aggregators, and low-cost highly energy-efficient modulations. Over-the-air upgrades of the firmware are enabled by the full support of bidirectional communication.

On-Ramp Wireless [24] came up with Random Phase Multiple Access (RPMA) which is a spread-spectrum technology operating on the 2.4GHz ISM band instead of the sub-1GHz band and leverages more relaxed regulations on the spectrum across different regions. Due to a robust physical layer design, it can still operate over long-range wireless links and under the most challenging RF environments.

A base station in RPMA's is capable of receiving transmissions on all the spreading factors. The devices employ the adaptive data rate (ADR) technique, where the devices can select optimum spreading factors based on the downlink signal strength. RPMA uses a form of Viterbi algorithm that allows guaranteed message arrival at the base station even with the Packet Error Rate (PER) of as high as 50% and security is employed using encryption.

LoRa uses chirp spread spectrum modulation to achieve long-range communication of several kilometers in rural and suburban areas. It operates with very low power consumption, allowing devices to operate for several years on a single battery. LoRa has a high capacity for connecting a large number of devices, up to millions of devices per gateway, making it ideal for large-scale IoT applications. It utilizes the LoRaWAN protocol, which is an open standard developed by the LoRa Alliance [15]. This open standard allows various manufacturers to develop LoRa-based de-

vices, potentially providing more options and lower costs for end-users. LoRa also offers the advantage of having no restrictions on the number of messages that can be sent per day. This makes it suitable for IoT applications that require frequent data transmissions. Combined with its long-range capability, low power consumption, and high device capacity, LoRa has become one of the most prominent LPWAN standards available today.

2.2 The Semtech LoRa Technology

In this section, an in-depth discussion of the technical aspects of LoRa is provided, including its modulation technique, signal structure, and network architecture.

2.2.1 LoRa Physical Layer

LoRa uses a proprietary physical layer RF technology [25]. The modulation scheme employed by LoRa radio is a derivative of the Chirp Spread Spectrum (CSS) modulation scheme. The technology enables simpler and more accurate synchronization both in terms of timing and frequency. This synchronization is achieved by ensuring the phase continuity among chirp symbols in the preamble of the packet, eliminating the need for the expensive components to generate a stable local clock in the LoRa node. The technology also provides the possibility to trade the throughput for energy consumption, or communication range, while keeping the bandwidth constant.

LoRa operates in the sub-1GHz frequency band and the regional regulations [26] govern the central frequency for various regions. For instance, in the US it operates on the band centered at 915 MHz, but in Europe, it works on the band centered at 868 MHz. In addition to the central band, the regional specifications also require the radio transceivers in Europe to adopt a transmission duty cycle restriction of 1% or 0.1%, depending on the subband used. This duty cycle restriction can be ignored if the transceiver performs Listen Before Talk, which is essentially a carrier sensing mechanism to prevent interference among devices. The US doesn't impose

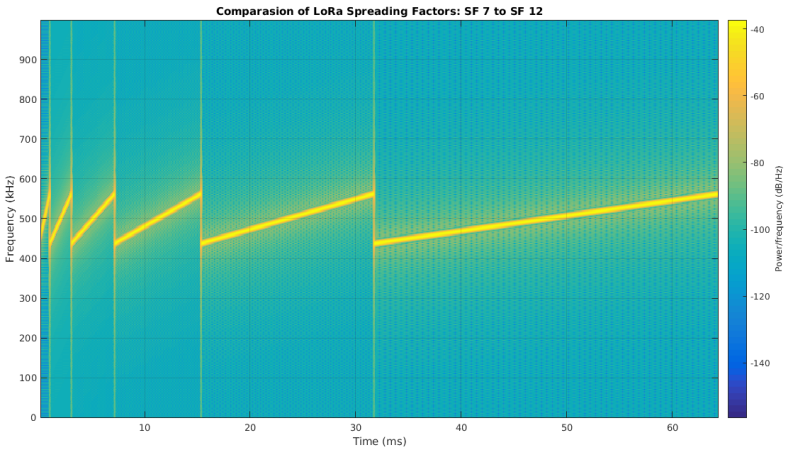


Figure 2.1: Comparison of various spreading factor (source: [1])

any duty cycle restrictions but imposes a packet dwell time restriction of 400ms. In other words, the LoRa transceivers operating in the US can not send a packet with a duration longer than 400ms.

There are three parameters that specify the LoRa modulation:

- **Bandwidth (BW)**: 125 kHz or 500 kHz for US;
- **Spreading Factor (SF)**: It ranges from(7–12) and determines the length of the chirp symbol in the time domain as $T_s = 2SF.T_c$, where $T_c = 1/BW$. Figure 2.1 compares various spreading factors.

  A lower spreading factor indicates a shorter symbol duration, hence less time-on-air resulting in a higher data rate, and the higher the spreading factors; the lower will be the data rates. The spreading factors also affect the sensitivity of the receiver. A lower spreading factor corresponds to lower sensitivity, hence a lower range, and vice versa. Table 2.2 [27] tabulates various data rates and transmission ranges associated with each spreading factor at 125 kHz bandwidth.
- **Coding rate (CR)** It ranges from 0 to 4 and determines the rate of the forward error correcting code as:

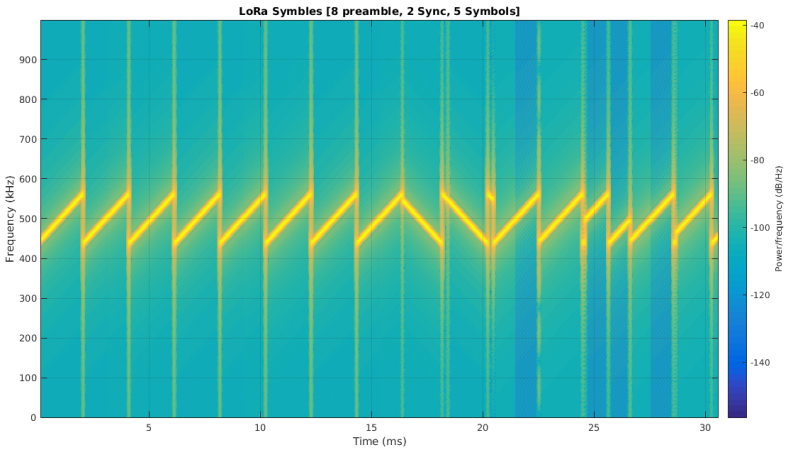


Figure 2.2: LoRa physical packet (source: [1])

$RateCode = 4/(4+R)$.

According to [28], the effective data rate of a LoRa packet is given by:

$$R_b = SF \frac{CodeRate}{\frac{2^{SF}}{BW}} [bits/s] \tag{2.1}$$

Figure 2.2 shows the physical packet structure of LoRa. LoRa physical packet starts with a preamble that is typically 8 up-chirps. It helps the receiver to lock onto the data rate. The preamble is followed by two down-chirps that demarcate the starting of the payload. The payload is followed by the CRC for error detection and correction.

All LoRa packets are considered to have a preamble so the receiver can synchronize

Table 2.2: LoRa spreading factors

Spreading factor (UL for 125 KHz)	**Physical bit rate (bits/sec)**	**Transmission Range (Depends on Terrain)**
SF7	5470	2 Kms
SF8	3125	4 Kms
SF9	1760	6 Kms
SF10	980	8 Kms
SF11	440	10 Kms
SF12	250	12 Kms

its frequency and timing according to the sender. Although the data rate ranges from 0.3-500 kbps approximately, the actual capacity of the system is much larger. The receiver exploits the orthogonal property of the spreading factors used by LoRa to detect multiple simultaneous transmissions from the different LoRa transmitters [28].

The LoRa gateways currently available allow for up to 64-channel parallel processing capabilities.

2.2.2 LoRa System Architecture

A typical LoRa network consists of three main network components:

- LoRa End-devices: They are basically sensors or actuators interfaced with the LoRa radio module.
- LoRa Gateways: Gateways act as sink nodes in the network. They act as a bridge between end devices and the LoRa Network Server.
- LoRa Network Server: The network server is the central entity in the LoRa network. It is responsible for controlling the entire network which includes but is not limited to radio resource management, admission control, security, etc.

The components of the LoRa network are typically configured in a star topology. Figure 2.3 illustrates a typical LoRa network laid out in a star-of-stars topology. The end devices are normally connected to one or many gateways via single-hop LoRa communication. The gateways as described above act as a bridge between end devices and the network server and are connected to a common Network Server via standard Internet technologies. The gateways are mainly responsible for relaying packets between end devices and the Network Server. Any gateway that successfully decodes the message sent by any of the end-device will forward the packet to the Network Server after adding link information. The Network Server can communicate with the end devices by choosing one of the gateways based on some criterion (e.g., best radio connectivity).

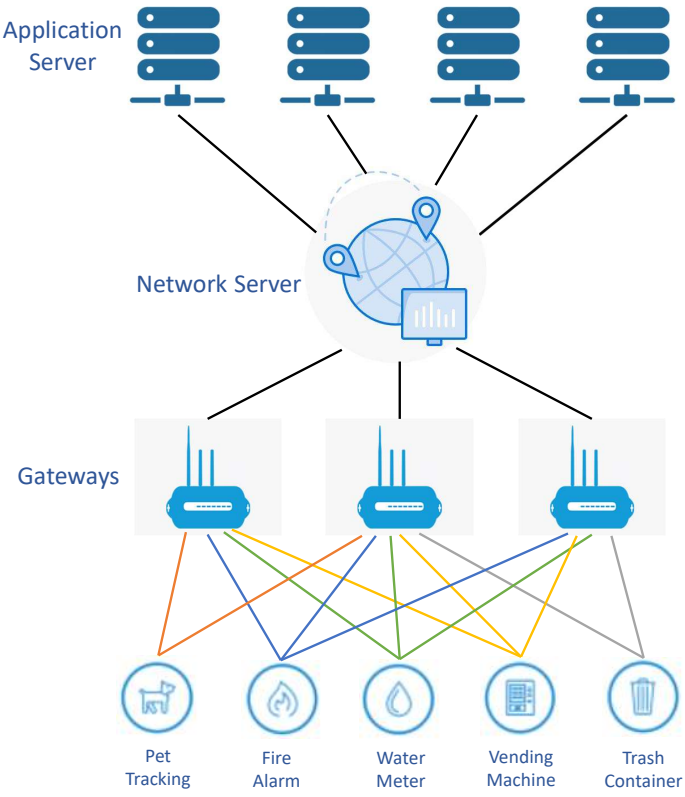


Figure 2.3: LoRaWAN architecture

2.2.3 LoRa MAC

As stated earlier, LoRa is a physical layer technology but it provides the flexibility to implement any of the upper layer protocols. The upper layer protocol suit developed by LoRa Alliance is called LoRaWAN. According to the specifications[26], the MAC layer defined under the LoRaWAN protocol suit is basically an ALOHA-based protocol.

Under LoRaWAN, the end devices can be categorized into one of the three specified classes that are Class A, Class B, and Class C as shown in figure 2.4. These classes are specified in [26].

Under the Class A mode of operation, transmissions can only be initialized by the end devices. These transmissions will occur in a totally asynchronous manner. After every uplink, the end device will open two receive windows. These receive windows

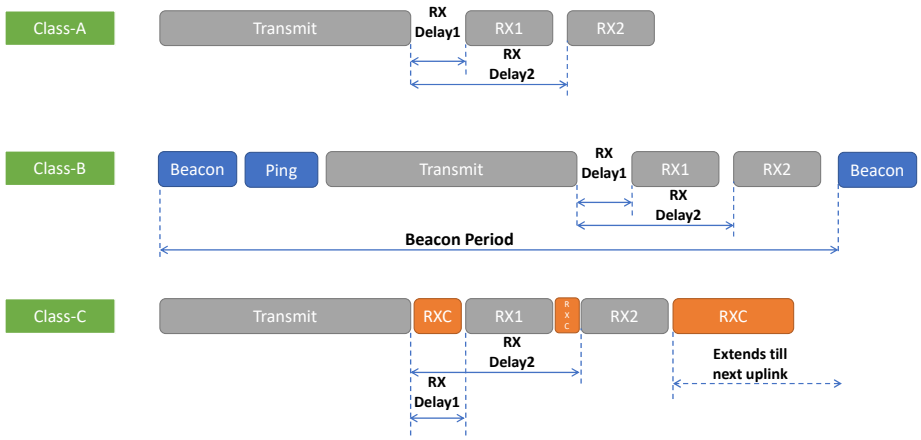


Figure 2.4: LoRaWAN classes of operation

are intended for the network server to send any command or data to the end device. The first receive window is opened according to the uplink channel and spreading factor but the second receive window is always opened on a channel and spreading factor preconfigured by the network server. Class A is the default mode of operation for LoRaWAN and is mainly intended for applications where an application server or control station collects the data produced by the end devices. In other words, Class A mode is used for devices that typically do not require downlink commands from a controller.

Class B mode of operation enables end devices to have limited capability to receive downlink communication even without having the need to send an uplink. In order to achieve this capability, the end devices under class B mode must have a tight time synchronization with the network clock. This synchronization is achieved using beacons that are broadcasted by the GPS-enabled Class B gateways. Once synchronized, the end devices can receive the downlink communication in specific time slots that are independent of the uplink traffic. Class B mode of operation is generally intended for applications that require end devices to receive downlink commands to function properly.

Finally, under Class C mode of operation, the end devices keep their transceivers

in the listen mode whenever they are not transmitting. The network server can communicate to these devices at any time without the need for any uplink traffic or time synchronization. This mode of operation is intended for end devices that do not have any energy constraints and are generally grid powered.

All communication under the LoRaWAN protocol is secured via strong AES-128 encryption. The network and application keys used for encryption and decryption are generated securely through an Over-The-Air Activation (OTAA) procedure. User-defined keys can be set using the Activation By Personalization (ABP) procedure. The procedures for activation and key generation are defined in the LoRaWAN specification [26].

2.2.4 Adaptive Data Rate (ADR) Mechanism

The MAC layer in the LoRaWAN protocol has a distinguished feature called the adaptive data rate mechanism. It allows the end devices to dynamically change their transmission data rate according to the link quality. The network server manages this mechanism by enabling end devices to change the spreading factor index. The ADR mechanism tries to find the best trade-off between energy efficiency and link robustness.

Any LoRaWAN compliant end device can enable the ADR mechanism by setting the ADR bit in its uplink communication packets. The ADR mechanism runs as two-part mechanism, running asynchronously at the end node (ADR-NODE) and at the network server (ADR-NET)

The ADR algorithm running on the end node side is a very simple and effective way of increasing its packet delivery rate to the network server by increasing the SF index of the packet. Doing so, the transmission range increases, and hence, the probability of reaching the gateway increases. The LoRaWAN specification doesn't specify any rules or mechanisms for the ADR algorithm running on the network server. This gives the network providers the flexibility to specify their own version of ADR. The ADR-

Net referred to henceforth will be based on the reference rate adaptation algorithm [29] by Semtech, implemented by the Things Network [30].

ADR-NODE: The end device that has ADR enabled will keep track of two counters: ADR_ACK_CNT and ADR_ACK_DELAY. For each uplink, the end device will increment the ADR_ACK_CNT. When the value of ADR_ACK_CNT exceeds ADR_ACK_LIMIT, which can be configured during device setup time, the end device requests an acknowledgment from the network server by setting ADRACKReq bit its subsequent uplink transmissions. The end device then waits for the acknowledgement for the next ADR_ACK_DELAY number of uplink packets. If the end device does not receive any acknowledgment once the ADR_ACK_DELAY counter expires, the device first increases its transmit power and if that is not sufficient then it increases its SF index to improve its probability of reaching the gateway [26].

ADR-NET permits the network server to control the transmit power and data rates of each node for uplink communication. It calculates the link budget by using the RSSI and SNR of recevied frames from each device. Based on the minimum SNR required for demodulation specified by Semtech, the parameters are then calculated. These parameters are then sent to the node using a downlink frame. The node configures itself according to the received parameters and uses the same for all future uplinks unless otherwise instructed.

2.3 Artificial Intelligence

Artificial Intelligence (AI) has experienced a resurgence in recent years, commonly referred to as the "third wave" of AI. This is due to the availability of faster and cheaper computation, the abundance of large datasets, and the development of new algorithms. These factors have enabled the creation of AI systems that can perform complex tasks and achieve human-like capabilities, leading to significant advancements in various fields, including healthcare, finance, transportation, and manufacturing.

AI refers to the capacity of a machine to exhibit cognitive abilities such as perception, synthesis, and inference of information, which are indicative of intelligent behavior. As defined in a 2007 paper [31], AI is the science and engineering of creating intelligent machines, particularly intelligent computer programs. It is related to the task of using computers to comprehend human intelligence, but it is not limited to methods that are biologically observable. The emergence of AI is discussed in another paper [32], which categorizes AI systems as those that mimic human behavior.

AI is a multidisciplinary field that leverages large datasets and computer science techniques to solve complex problems. It encompasses subfields such as machine learning, deep learning, and federated learning, which employ AI algorithms to create systems that can perform tasks such as classification and prediction based on input data. These techniques and algorithms are widely used in various real-world applications, including speech recognition, computer vision, healthcare, customer service, and many others.

2.3.1 Machine Learning

Machine learning is a subset of Artificial Intelligence that enables computers to learn and improve their performance on a specific task, by analyzing patterns in data without the need for explicit programming. In simple terms, it is a process of training computer algorithms to automatically improve their performance when exposed to new data. As described by [33] machine learning is when a computer program improves its performance on a task, as measured by a performance metric, through experience gained from the data provided.

It enables the creation of autonomous systems capable of performing tasks without human intervention. Through the use of learning algorithms and continuously accumulating experience, these systems are able to adapt to the complexity of problems and make decisions. While human-performed tasks such as cooking, driving, and speech recognition can also be accomplished by machine learning, the technique

is particularly useful for handling tasks that are beyond human capabilities, such as analyzing large and complex datasets in fields such as remote sensing, weather forecasting, e-commerce, and web search. With the vastness of data, it becomes increasingly difficult for humans to extract valuable insights, but machine learning algorithms are able to handle this complexity and make predictions or decisions based on the data.

In order to effectively solve a given problem, an appropriate machine learning approach must be selected. The various categories of machine learning techniques include [34]:

Classification Problem: It is a type of machine learning problem in which the goal is to predict the output class of an input based on a set of predefined classes. The number of classes and the nature of the classes are known in advance. The classification problem can be of two types: binary, in which the output can only belong to two classes, and multi-class, in which the output can belong to more than two classes.

Anomaly Detection Problem: Anomaly detection is a machine learning task that aims to identify patterns and detect variations or anomalies in these patterns. This can be used in various domains, for example credit card companies use anomaly detection algorithms to identify unusual transaction behavior and flag them as potential fraudulent activities. The goal of anomaly detection is to identify data points that do not conform to the expected pattern of the majority of data points and are considered outliers. Another example is identifying equipment failures in a manufacturing plant by monitoring sensor data from various machines and detecting any unusual patterns or deviations that may indicate an impending equipment failure.

Regression Problem: Regression algorithms are a category of supervised machine learning techniques that are used to solve problems where the output is a continuous and numeric value. These algorithms are typically used for problems that

involve making predictions of a numeric value, such as predicting the price of a house, the number of units to be sold, or the amount of rainfall.

Clustering Problem: Clustering is a type of unsupervised machine learning technique that groups similar data points together based on their characteristics, in order to discover the inherent structures in the data. The goal of clustering is to divide the data into subsets, called clusters, such that the data points in the same cluster are more similar to each other than to those in other clusters. The clusters are then labeled, and the trained algorithm can be used to assign new, unseen data points to one of the identified clusters.

Reinforcement Problem: Reinforcement learning is a machine learning technique that focuses on training agents to make decisions based on their experiences. The agent learns by interacting with an environment and receiving feedback in the form of rewards or penalties. Reinforcement learning allows for the creation of agents that can learn how to perform a task without being explicitly programmed on how to do it. It's widely used in various applications such as game-playing, robotics, and control systems.

Generic working of ML The process of Machine Learning is composed of six key components that are independent of the specific problem or algorithm being used. These components include data collection and preparation, model selection, model training, model evaluation, model fine-tuning, and model deployment and maintenance. These steps are crucial to achieving successful results with ML and are often illustrated in a figure, such as figure 2.5.

Each component has a specific task to accomplish:

- **Data collection and preparation:**: The initial stage of the machine learning process involves collecting and preparing the data in a format that can be processed by the chosen algorithm. In many cases, a large amount of data may be available for a given problem. However, the data from web sources is

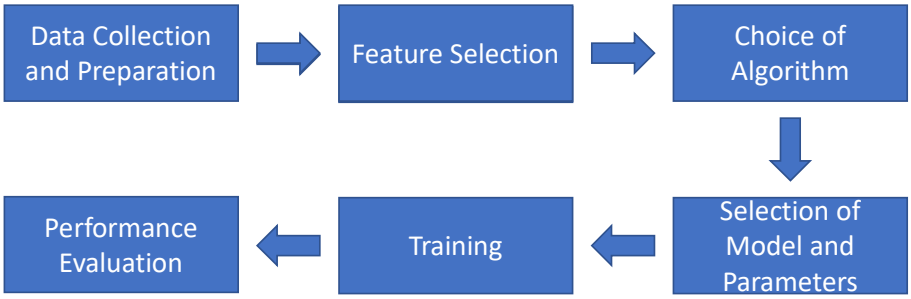


Figure 2.5: Generic ML model components

often unstructured and contains a significant amount of noise, such as irrelevant and redundant data. Therefore, the data must be cleaned and pre-processed into a structured format before it can be used for training the algorithm.

- **Feature Selection:** The data obtained from the previous step may include numerous features, some of which may not be relevant to the learning process. These features must be removed, and a subset of the most important features must be selected for use in the learning process. This process is referred to as feature selection or dimensionality reduction. The goal of this step is to improve the performance of the algorithm by reducing the computational cost and increasing the interpretability of the model.

- **Choice of Algorithm:** Not all machine learning algorithms are suitable for all types of problems. Some algorithms are better suited for certain classes of problems, as discussed above. The selection of the most appropriate machine learning algorithm for a given problem is crucial in achieving optimal results. This process is known as model selection, and it involves comparing the performance of different algorithms on a specific problem and selecting the one that performs best.

- **Selection of Models and Parameters:** Most machine learning algorithms require some manual tuning of various parameters in order to achieve optimal

performance. This process is referred to as hyperparameter tuning, and it involves selecting the most appropriate values for the parameters of the chosen algorithm. These parameters include, but are not limited to, learning rate, number of hidden layers, and number of neurons. Hyperparameter tuning can be a time-consuming process and requires some initial manual intervention.

- **Training:** After selecting the appropriate algorithm and optimal parameter values through the process of model selection and hyperparameter tuning, the model is trained using a portion of the dataset as the training data. This process is referred to as model training, and it involves using the training data to learn the underlying patterns and relationships in the data. The trained model can then be used to make predictions on new, unseen data.

- **Performance Evaluation:** Prior to the deployment of the system for real-time use, the model must be evaluated using unseen data to assess its performance. This process is referred to as model evaluation, and it involves using various performance metrics such as accuracy, precision, and recall to determine how well the model generalizes to new data. Model evaluation is crucial for identifying any potential issues with the model and for determining its readiness for deployment.

2.3.2 Data Dimensionality Reduction

In recent decades, machine learning models have been extensively utilized in a wide range of complex and data-intensive applications such as meteorology, biology, astronomy, medicine, economy, and finance. However, current ML-based systems are not efficient and extensible enough to handle massive and voluminous data. The high dimensionality of data can be a hindrance for data processing and finding the global optimum can be computationally expensive. To overcome this challenge, dimensionality reduction algorithms have been developed to reduce the number of dimensions

of the data by eliminating redundant and irrelevant information, resulting in improved accuracy of results. These algorithms operate in an unsupervised manner to uncover the underlying structure within the data. Popular dimensionality reduction techniques include Multidimensional scaling (MDS) [35], Principal component analysis (PCA) [36], Principal component regression (PCR) [37], and Linear Discriminant Analysis (LDA) [38]. These techniques can be integrated with classification and regression algorithms to enhance their performance.

PCA: Principal Component Analysis (PCA) is a widely used technique in machine learning and data analysis for reducing the dimensionality of the data. It is a linear technique that aims to project the data onto a new lower-dimensional space while preserving as much of the variation in the data as possible. The algorithm consists of the following steps:

- Data Preprocessing: The first step is to standardize the data by subtracting the mean and dividing by the standard deviation. This ensures that all features are on the same scale and have zero mean.

- Covariance Matrix Calculation: Next, the covariance matrix of the standardized data is computed. The covariance matrix is a square matrix that contains the pairwise covariance between all features.

- Eigen Decomposition: The eigenvectors and eigenvalues of the covariance matrix are calculated and the eigenvectors with the highest eigenvalues are chosen as the principal components. These eigenvectors define the directions of the new coordinate system and the eigenvalues indicate the amount of variation in the data along each direction.

- Data Projection: The original data is projected onto the new coordinate system defined by the principal components. This is done by taking the dot product of the data with the matrix of eigenvectors.

- Dimensionality Reduction: The data can be reduced to a lower-dimensional space by keeping only the first k principal components, where k is the number of dimensions desired in the reduced space.

By following these steps, the PCA algorithm enables us to reduce the dimensionality of the data while preserving most of the important information that is useful for the analysis. The result is a new set of uncorrelated features that can be used for further analysis and modeling.

2.3.3 Model Pruning

Model pruning is a technique used in machine learning to reduce the complexity of a trained model while preserving its accuracy [39] [40]. It aims to eliminate the unnecessary parameters or connections in the model that do not contribute significantly to the model's performance. This can be achieved by removing parameters with the smallest magnitude of weights or neurons/layers that have the least impact on the output.

One common approach for pruning is based on the magnitude of the weights [40]. This process starts by initializing all the weights in the model, then training the model and evaluating the performance. Next, the weights with the smallest magnitude are removed, and the model is retrained and evaluated again. This process is repeated iteratively until a desired level of compression is achieved.

It is important to note that model pruning can result in a trade-off between the performance of the model and the size of the model [40]. Pruning too aggressively can lead to a significant drop in accuracy, while pruning too little may not produce the desired reduction in model size. Therefore, it is essential to find the right balance between the performance and the size of the model through proper evaluation methods.

CHAPTER 3: RELATED WORKS

This chapter provides a comprehensive examination of the technical aspects of LoRa and LoRaWAN, which have received significant attention from the research community. The studies discussed in this chapter include evaluations of coverage performance, the effect of interference on communication, the assignment of configuration parameters, advancements in ADR, MAC designs, and IoT requirements. Additionally, the chapter also covers research on the implementation of LoRa networks in smart cities.

3.1 LoRa and LoRaWAN

This section delves into an analysis of various studies focused on evaluating the communication range of LoRa, as well as the effects of interference on the network. Additionally, it presents the different approaches taken to mitigate the detrimental effects of interference in order to optimize the performance of the network.

3.1.1 Communication Range

Like other wireless technologies, the network coverage of LoRa also depends on the transmission parameters (prominently the spreading factors and transmit power) and the environmental condition where the network is deployed. To assess the coverage for both outdoor and indoor environments, several empirical studies have been performed.

A study to measure the coverage of LoRa network range [41] was performed in the city of Oulu in Finland that has mostly a flat geographical landscape. An antenna tower that was 24 meters high above sea level was used to mount the gateway. The results showed that for end devices within the range of 2 km, the RSSI of above -100 dBm with a packet loss ratio (PLR) of 12% was observed. As the end devices are

moved to distances of 2 to 5 km, PLR of 15% surges up to 33% for distances between 5–10 km. The network starts to experience huge packet losses with a PLR of 74% beyond 10 km. These losses were lower when measured in the open sea, being 31% for a range of 5–15 km, and for a range of 15–30 km a PLR of 38% was observed. During this study, no ADR or acknowledgment mechanisms were employed. The study shows that LoRa can achieve long coverage ranges. The long coverage range observed in this study resulted from mostly flat terrain. In the case of areas with hilly terrains, LoRa is expected to have comparatively lower coverage but it will still be in terms of a few km.

A coverage test was conducted in an urban setting with buildings of 5–6 floors, as stated in [42]. A LoRa gateway was installed on the rooftop of a two-story building, achieving coverage of approximately 2 km. However, only SF12 could be utilized beyond the 1.5 km range for coverage. The authors of [42] used this coverage range to estimate a rough coverage plan for a city of 100 sq. km, requiring only 30 gateways for complete coverage. In [43], a LoRa gateway was deployed on the second floor of a house in a suburban environment. Using SF12, coverage could extend up to 2800 m with a 20% Packet Loss Rate (PLR). SF7 had a coverage of 650 m with 18% PLR, while SF9 could provide coverage up to 2300 m with a 12% PLR.

Furthermore, studies have investigated the indoor coverage capabilities of LoRa in addition to outdoor environments [44, 45, 46]. Results have shown that LoRa provides sufficient indoor coverage with RSSI values higher than -70 dBm, allowing for communication using all SFs [46]. LoRa has also been demonstrated to offer communication in large industrial areas, with an area of 34,000 square meters covered using only SF7 [47]. The coverage area can be extended by utilizing higher SFs. For industrial applications involving mobile or rotating objects, an evaluation was conducted to account for the impact of Doppler shift and angular velocity on LoRa performance [48]. Results showed that when the end node moves at a speed of 100 km/h, only one-

third of packets are received correctly. Additionally, under angular velocities higher than 78 rad/s, LoRa communication using SF12 and 125 MHz channel bandwidth becomes unreliable, resulting in up to 50% packet losses.

3.1.2 Impact of Interference on LoRa

The self-interference of LoRa is a crucial factor affecting network scalability, as it can occur between transmissions utilizing the same spreading factor (SF) as well as different SFs.

LoRa communication can experience interference from other co-located LoRa networks, which can impact communication even within the same network. Several research papers, such as [49, 50, 51, 52, 53, 54], have studied LoRa self-interference. In [52], a simulated network topology with two gateways belonging to different networks located at opposite corners of a 1000 m x 1000 m square was studied. Each network had 60 nodes distributed within the area. Results showed that end nodes at a distance of 1000 m from the gateway experienced a Packet Loss Rate (PLR) of up to 50% due to interference, despite SFs being orthogonal in theory. This is because orthogonality imperfections can lead to interference between SFs.

In [51], the effect of imperfect orthogonality among LoRa's spreading factors (SFs) on self-interference was investigated. The results showed that inter-SF interference can significantly reduce network performance, especially for higher SFs (SF10–SF12), even under low traffic loads. Log-normal channel fading was found to have a negative impact on data extraction rate (DER) in single gateway deployments, with up to a 15% reduction. However, in multi-gateway deployments, the impact of fading was negligible due to the diversity of gateways. The study in [53] also found similar results to those in [51] regarding LoRa self-interference.

The paper [51] delves into the impact of inter-SF interference on the LoRa network's performance, revealing that it can degrade the network's performance even when network traffic is low. The study also concludes that inter-SF interference affects higher

SFs (SF10-SF12) more severely. Moreover, the study shows that log-normal channel fading has an adverse effect on Date Extraction Rate (DER) in single gateway scenarios, but this effect is negligible in multi-gateway deployment due to gateway diversity. Another paper [53] also shows similar results related to LoRa self-interference.

In [49, 50], studies were conducted on the collision behavior of two different LoRa packets at the link level. The collision behavior is affected by the carrier frequency, spreading factor, received power, and time difference between the interfering transmissions. The authors in [49] demonstrated that LoRa experiences the capture effect, which means that the stronger signal can suppress the lower one and be correctly decoded by the receiver. Based on this finding, the authors proposed two solutions to decrease LoRa self-interference: using multi-gateway deployments and directional antennas for end nodes. The first solution is beneficial for removing the negative effects of fading channels [51]. The use of directional antennas for end nodes results in a stronger received signal and lower interference from other interfering networks. The study showed that using directional antennas with 8 dBi gain improves the Date Extraction Rate (DER) by 0.06 compared to cases when omnidirectional antennas are used [49].

3.1.3 LoRaWAN Improvements

Several studies have addressed the issue of network scalability in LoRaWAN and proposed improvements to deal with it. These studies include research on improving adaptive data rate (ADR) for better scalability, as well as different synchronization and traffic scheduling approaches. Many of these studies, including [41, 55, 56, 57, 58, 59], have highlighted concerns about the impact of downlink traffic on network scalability. Simulation results show that without addressing these scalability issues, the network may reach a deadlock stage. As a result, researchers have proposed various solutions such as optimizing SFs and transmit powers and improving the ADR mechanism.

The study in [16] was one of the first to analyze the scalability of LoRaWAN networks. The authors derived the capacity of a single gateway deployment under perfect synchronization, and found that less than 2% of end devices will use SF12 according to the channel attenuation model in [41]. They also found that end devices located far away from the gateway experience lower throughput due to the near-far problem. To address this issue, an algorithm for assigning SFs and power levels to end nodes was proposed in [60]. The algorithm optimizes the fraction of end nodes using a certain SF to minimize the probability of inter-SF collisions, and then assigns SFs and power levels based on the path loss of each end node. It was demonstrated that using this algorithm can decrease Packet Error Rate (PER) by 50% for nodes at the cell edge in a scenario with one end node per 1000 square meters transmitting every 10 minutes. Several other studies have also reported on scalability concerns in LoRaWAN networks and proposed improvements to the Adaptive Data Rate (ADR) mechanism and synchronization and traffic scheduling approaches to improve network scalability and reliability [41, 55, 56, 57, 58, 59].

The paper [61] explores the impact of SF distribution on packet success probability (PSP) among end nodes in a LoRa network. The authors formulate an optimization problem for assigning SFs to end nodes based on two conditions: (i) ensuring that the received power from the node at the gateway exceeds the receiver sensitivity threshold for the assigned SF, and (ii) ensuring that the signal-to-interference ratio (SIR) for that node exceeds the correctly decoded SIR threshold for that SF. They determine the regions of SF distribution in the network that maximize the average PSP by considering the distance covered by each SF. The authors demonstrate that increasing the lower SF region can improve PSP in networks with numerous end devices. Their proposed solution outperforms equal-interval and equal-area based SF distribution schemes in terms of average network PSP.

In [62], the authors propose two new algorithms for SF distribution in a cell. The

first algorithm, ExpLoRa-SF, equalizes the number of end nodes using different SFs by assigning the lowest SF to the first group of end nodes that are closer to the gateway based on their received signal strength indicator (RSSI) values and so on until the last group of end nodes. The second algorithm, ExpLoRa-AT, aims to equalize transmission times between different end nodes by distributing different SFs. This approach optimizes the usage of different SFs over time, resulting in better network performance. However, the first approach achieves the same results when all end nodes have the same traffic patterns and packet lengths.

In addition to SF and transmit power assignment algorithms, other improvements to the Adaptive Data Rate (ADR) mechanism are proposed in [63, 64]. The authors of [64] evaluate the performance of the ADR mechanism in terms of its average converge time, which depends on the network size and channel conditions. Surprisingly, highly variable channels can decrease the ADR convergence time for large networks by introducing randomness in RSSI values and increasing the impact of the capture effect. The authors optimize different parameters of the ADR mechanism and demonstrate that only reducing the ADR_ACK_DELAY parameter improves the ADR convergence time by shortening the duration of individual steps to increase transmit power and SF.

According to [59], the main factors that result in packet loss in LoRaWANs are interference, gateway receiver saturation, and network outage. When the LoRaWAN is highly congested, receiver congestion becomes the main factor for packet loss rather than network outage. In such cases, using ADR algorithms or retransmission mechanisms can further worsen the performance instead of improving it.

In [63], the authors proposed an enhancement to the existing ADR mechanism deployed at the network server to enhance network performance. Instead of utilizing the maximum SNR value from a window of past uplink packets, the authors suggested using the average SNR of that window. This modification helps to mitigate the

effect of fluctuating channel conditions. The proposed ADR mechanism was shown to achieve a packet delivery ratio that is at least 30% better than the basic ADR mechanism in scenarios with moderate variable channels.

Several studies, including [55, 58, 59], have shown that downlink traffic can have a negative impact on the overall performance of LoRaWAN networks, particularly in the case of a large number of end nodes. In [58], it was found that increasing the ratio of high priority end nodes, which require confirmed uplink traffic, can decrease the total network throughput and worsen the throughput for low priority end devices. This same relationship was observed when increasing the number of retransmissions per packet. In [55], it was shown that the number of missed receive windows increases with an increase in uplink traffic that requires downlink confirmation, due to gateway duty cycle.

3.1.4 Scheduling Based Approaches

The ADR mechanism and the use of different spreading factors and transmit power levels can enhance network scalability. However, the primary limitation of network scalability is the use of the Aloha-based MAC protocol. Furthermore, the half-duplex operation of gateways and duty cycle restrictions have further reduced network scalability. As a result, alternative solutions based on low-power time synchronization and low-overhead traffic scheduling mechanisms are being investigated to coordinate transmissions.

In [65], a lightweight scheduling mechanism is proposed for specifying the allowed spreading factors (SFs) and transmit powers for certain communication channels. The time is divided into frames and each frame is further divided into subframes, where the number of subframes is equal to the number of channels used. At the beginning of each subframe, a beacon that contains information about allowed SFs and received signal strength indicator (RSSI) values for each channel is transmitted. In the second part of the subframe, end nodes can transmit their uplink data using the Aloha protocol.

The main advantage of this approach is that it reduces the impact of the capture effect and inter-SF collisions by allowing the end nodes to select the minimum SF and transmit power that reduces negative interference with other devices. The only constraint is that uplink traffic should not interfere with beaconing traffic. However, the reliability improvements compared to LoRaWAN are not significant, with only 4-5% fewer packet losses observed for a multi-gateway deployment with 3500 end nodes.

In tele-measurement applications, accurate time synchronization between end devices and the back-end is crucial. In [66], a low-power time synchronization mechanism is proposed for Class A end devices. The mechanism synchronizes end devices a posteriori, after sending the event payload to the network. End nodes receive an ACK packet from the back-end after the main transmission, and then send a second packet indicating the time when the first packet was sent and when the ACK was received according to the unsynchronized end node time. The back-end can then calculate the uncertainty of the time-stamp and determine the actual time of the event. While this approach requires low power and an additional short packet, it cannot handle heavy loads or be used for traffic scheduling at specific times. In addition, if any of the packets are lost, the system will malfunction, and the actual time is only known at the back-end, not at the end node.

In [67], a low-overhead synchronization and scheduling mechanism is introduced that divides time into slots that are maintained by a scheduling entity in the network server. End nodes request synchronization and scheduling of time slots, which can occur in-band or through a separate channel reserved for synchronization traffic. The synchronization and scheduling entity responds with the current time slot, offset in the current time slot, and future time slots for end nodes to transmit. The information about scheduled time slots is compressed using probabilistic data structure to fit in one packet payload. Based on the response, end nodes can determine the current

time and future time slots to transmit. Synchronization can happen infrequently and considers end node clock drift. This mechanism increases packet delivery ratio by 30% for single SF deployments, practically utilizing full network capacity. Multi-SF deployments require SNR-based scheduling or using slotted-Aloha for LoRaWAN, as proposed in [68]. The proposed ACK-based time synchronization algorithm using slotted-Aloha decreases collision probability three times compared to pure Aloha.

3.1.5 New MAC Designs for LoRa

There are several MAC designs that were used by researchers on top of LoRa to overcome the limitations of Aloha. For example, in [69], it is suggested that implementing CSMA-based MAC protocols on top of LoRa can improve its performance. By incorporating CSMA, the duty cycle can be avoided and the packet delivery ratio (PDR) can be increased significantly. For example, authors found that by using CSMA, the PDR increased by seven times compared to traditional Aloha-based MAC in a scenario with 10,000 nodes. The PDR for CSMA-based MAC was over 75%, while the PDR for Aloha-based LoRaWAN MAC was only 10%. Moreover, implementing CSMA can decrease the energy consumption per node, as much energy is wasted in collisions in Aloha-based LoRaWAN MAC.

In [70], a time-power multiplexing approach was proposed to enhance the scalability of downlink traffic in LoRa-based MAC protocols. To achieve this, the uplink and downlink traffic were decoupled, with both receive windows only using a high power (27 dBm) and high duty cycle (10%) channel (869.525 MHz) for downlink traffic, while the other channels were reserved only for uplink traffic. Additionally, the downlink transmission was multiplexed in time and power, utilizing different spreading factors and transmit powers to send multiple packets simultaneously. This method resulted in decreased retransmissions of confirmed uplink packets and increased uplink throughput, including for non-confirmed traffic, by eliminating the possibility of collisions between uplink and downlink traffic.

In LoRa networks with mobile nodes, the hidden terminal problem can occur when end nodes covered by different gateways interfere with each other's traffic. To address this, a MAC protocol was proposed in [71] where nodes are always in the listening state and send a Request-to-Sent (RTS) message when they want to transmit. Multiple gateways reply with Clear-to-Sent (CTS) messages, but due to the capture effect, only one CTS succeeds. Both RTS and CTS messages include the reservation time for the channel. Before actual data transmission, the end node sends a Control CTS (CCTS) message to inform which gateway it will transmit to, allowing other gateways to serve other nodes. This MAC protocol improves the data delivery ratio by reducing the impact of the hidden terminal effect.

3.2 IoT Requirements

This section briefly discusses some of the existing works done on the QoS requirements for IoT applications.

The authors of [72] and [73] present the QoS requirements by IoT applications and network deployments that follow those requirements. In [72], the authors present a generic QoS architecture for IoT applications. The paper categorizes all the IoT applications into three categories namely: inquiry task, control task, and monitoring task. They further state that the task of inquiry (eg. status of smart logistics) focuses on timeliness and reliability. The control task (eg. remotely controlling some actuator) relies on timeliness and reliability of the information, and the monitoring task may require real-time or non-real time data. Different applications have different QoS requirements.

[74] proposes an analytical model for IoT applications. The authors classified the network traffic into two levels of priorities i.e. low priority (normal traffic) and high priority(emergency traffic). They considered the low priority traffic can only be served in absence of high priority traffic. The service to low priority traffic is immediately preempted upon arrival of high priority traffic. They assume that the low priority

traffic can be entirely sacrificed if required in order to serve to delay sensitive high priority traffic.

3.3 Artificial Intelligence in IoT

There are numerous IoT applications that have the potential to benefit from AI tools. This section discusses some of the existing works done in the field of IoT in collaboration with AI tools. Since most of the IoT applications are enabled by embedded devices, we will also discuss the existing works on AI in resource-constrained environments.

Human health is one of the fields that has benefited much from the advent of IoT and AI tools. In [75], the authors proposed a highly reliable method for measuring blood pressure without the need of putting cuffs. The proposed model uses a combination of information from the ECG signal and photoplethysmography (PPG) to measure blood pressure. The study first establishes a relationship between blood pressure and the ECG. This enables the user with ECG sensor to potentially measure blood pressure not requiring any additional device.

The authors of [76] proposed a novel solution to predict heart diseases using the cascaded deep learning model in the fog computing environment. Their solution makes use of data from multiple sensors that provide data about daily activity that is fed to an ensemble classifier. The classifier is hosted in the fog environment and computations are performed in a decentralized manner. The proposed approach achieved a prediction accuracy of 95%.

A transfer learning methodology to decrease the computational resources needed to train a deep neural network for a Reinforcement Learning (RL) problem is presented in [77]. The deep neural network is trained on a varied set of meta-environments to acquire a broad domain knowledge, which can then be transferred to test environments, and only the last few fully connected layers are trained. The performance of the algorithm is evaluated in terms of Mean Safe Flight, and it was observed that

the network's performance is comparable to that of training the network end-to-end, while significantly decreasing the latency and energy consumption by 1.8 and 3.7 times, respectively.

CHAPTER 4: PERFORMANCE EVALUATION AND LIMITATIONS OF LoRaWAN NETWORKS

4.1 Introduction

In this chapter, we present our analysis of the performance of LoRa networks. This work presents a systematic study to gain insights into the limitations of the LoRa networks. We investigate the transmission range, power consumption, and scalability of LoRa networks. We further study the impacts of various configuration parameters on the performance of the LoRa network. We have utilized "mdot" [78] by "Multitech" along with "Multitech conduit" [79] to carry out experiments in this part of the dissertation. In addition, we use NS-3 simulations for obtaining important scalability studies that are difficult to obtain experimentally. Our findings show that LoRa networks can achieve transmission ranges well above 3 miles and a AA Li-ion battery can power an end device for more than 7 years. The performance evaluation also provided insights regarding the limitations of LoRaWAN networks.

4.2 Need of LPWANs

Legacy wireless technologies like Zigbee have limited transmission ranges that cannot be used to serve a wide area such as an entire city. These devices are usually configured into a multihop mesh network to cover a larger area, as shown in figure

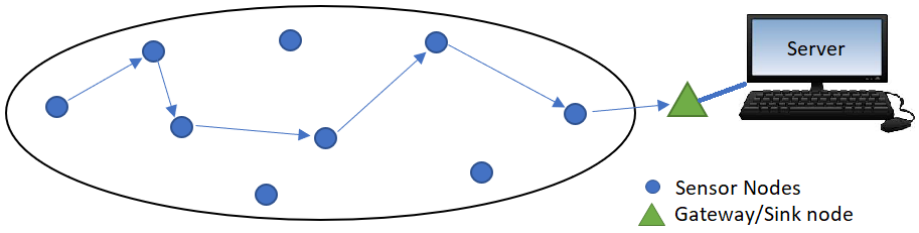


Figure 4.1: Typical multi-hop scenario

4.1. Some of the potential challenges for such networks to meet the coverage and scalability requirements of IoT networks are as follows:

- Delay: The packets in multihop networks arrive from the source node to the sink node via multiple hops and each node introduces processing and channel access delays [80] that become significant with the increasing number of hops.

- Need for robust routing: Multihop communication requires a dynamic routing protocol that must be implemented in all the nodes. This has been an area of extensive research, but in general, performance degrades drastically with the number of hops [81, 82]. Scaling up to thousands of nodes would require a hierarchical network infrastructure, which increases overhead and complexity.

- Multiple points of failure: Multihop networks need relay nodes for packet delivery. If a few of the relay nodes die out due to any reason then a part of the network will not be able to communicate to the sink.

- More network setup time: Any message from the source node to the sink node needs to be routed via multiple nodes and hence initial setup time will be required to make the routing decisions [80].

- Need for highly robust MAC protocol: The MAC protocol must resolve channel access contention from multiple neighboring nodes, which is challenging.

With an increasing number of devices connected to the Internet, there is a renewed thrust on developing low-cost and low-data-rate wireless technologies. SIGFOX built the first modern LPWAN. This came at a time when radio technology was becoming less expensive, and the tools for integrating applications were becoming easier for people to use. All these things have driven the emergence of LPWAN technologies.

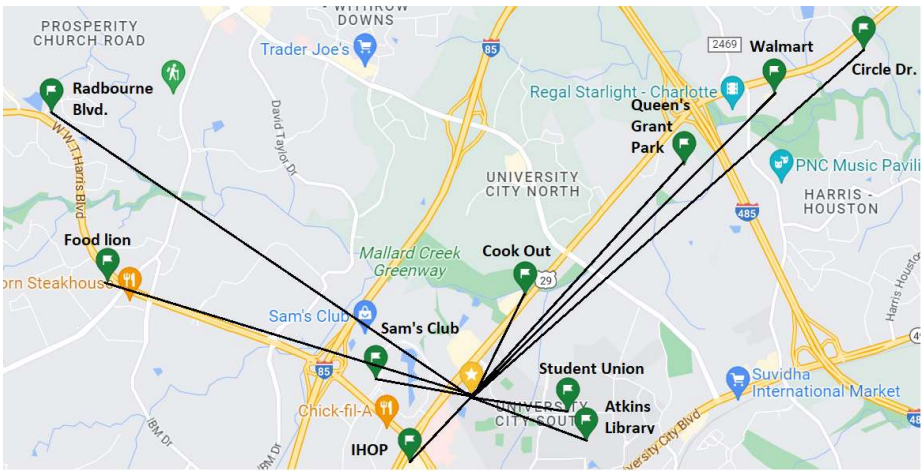


Figure 4.2: Predetermined positions for end device to take measurements

4.3 Performance Study

Long range and low power consumption are the two distinctive features of LPWANs that make them attractive for IoT deployment. In this section, we present the performance results of LoRa to evaluate its transmission range under various conditions, power consumption, and scalability.

4.3.1 Transmission Range

The sensor unit used for measuring the transmission range is composed of SX1272 LoRa module [83] that is the transceiver module and Multitech Conduit [79] is used as a gateway.

Table 4.1: Experimental results for range for SF=10 and transmit power of $14dBm$ towards the northeast direction from UNCC

Distance (Miles)	Total Packets Sent	Packets Sent Successfully	ACK Received	ACK lost
0.22	100	100	100	0
0.62	100	100	100	0
1.02	100	82	75	7
1.9	100	76	73	3
2.5	100	39	32	6
3	100	53	53	0

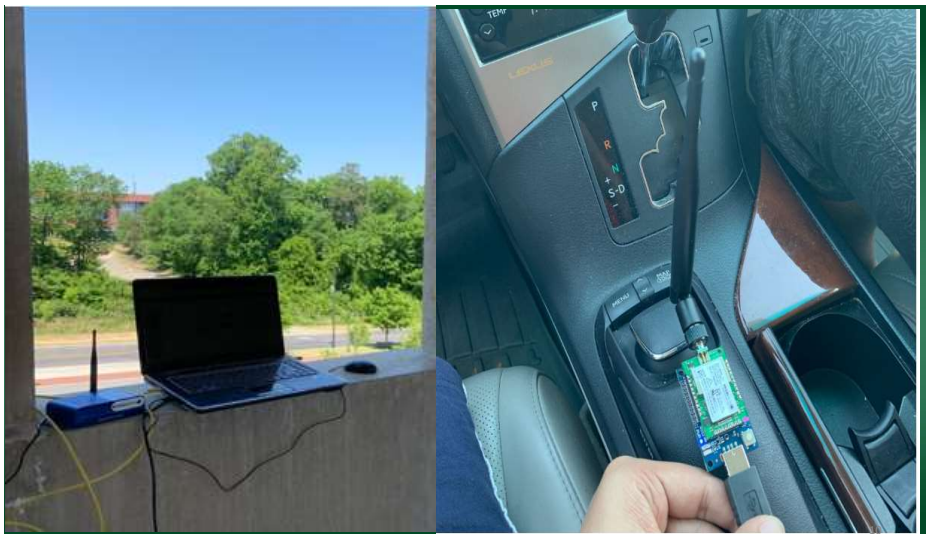


Figure 4.3: Gateway connected to the network server on the roof of CRI parking deck at UNCC and end device taken to predetermined locations

The gateway was attached to a laptop (acting as a server) that was placed on the roof of the CRI parking deck at the University of North Carolina at Charlotte (UNCC) as shown in figure 4.3. End device with different communication parameters i.e. different spreading factors, transmit powers, and coding rates was taken to predetermined distances around the UNCC campus area as shown in figure 4.2 and the corresponding communication links were evaluated. The results with the device taken to the northeast direction of the campus with an SF of 10 and transmit power of 14dBm are tabulated in Table 4.1. This region represents a semi-urban environment where the RF signal gets attenuated by buildings, trees, vehicles, etc., a LoRa node was able to achieve a maximum transmission range of 3 miles with a 53% success rate. In the northwest direction, a similar transmission range was achieved but the success rate at 3 miles distance was 70% because of less number of obstructions. However, in the southwest direction, there are several obstructions such as light-rail tracks and stations, bridges, buildings, traffic lights, etc. along this direction. The packets were able to reach a distance of 0.47 miles with a 100% success rate with all spreading factors but could not reach any further.

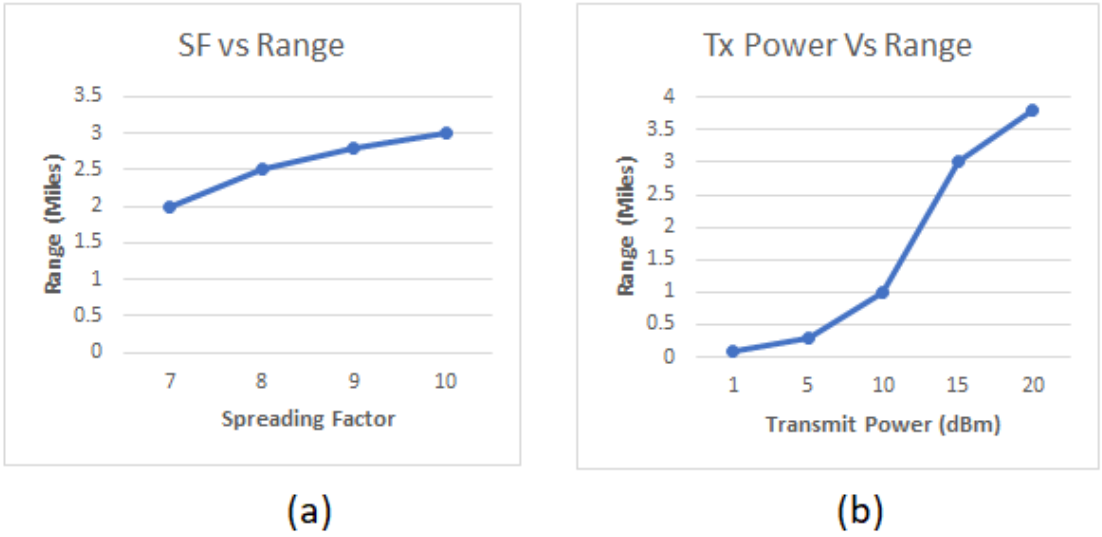


Figure 4.4: (a) Spreading Factor vs LoRa Range with Transmit power=14dBm (b) Transmit Power vs LoRa Range for SF=10 towards the northeast direction from UNCC

1) Effect of SF on Range: As the spreading factor increases, the range of the network also increases i.e. range of the network increases with the spreading factor (figure 4.4(a)).

2) Effect of Transmit power on Range: As the transmit power increases the range of the network also increases i.e. range of the network increases with the transmit power of LoRa transceiver (figure 4.4(b)).

4.3.2 Power Consumption

The current drawn by the end devices with different communication parameters was measured in the laboratory. This was done by measuring the voltage drop across a 1 ohm resistor that was connected in series with the 3.3V power supply for the end device using a data acquisition board. The power consumption was estimated from the corresponding current measurements.

1) Lifetime Expectancy of End Device: An effective and widely used mechanism for extending the battery life of wireless devices is to intelligently use sleep modes. After a specified sleep time duration, the node wakes up, performs its tasks

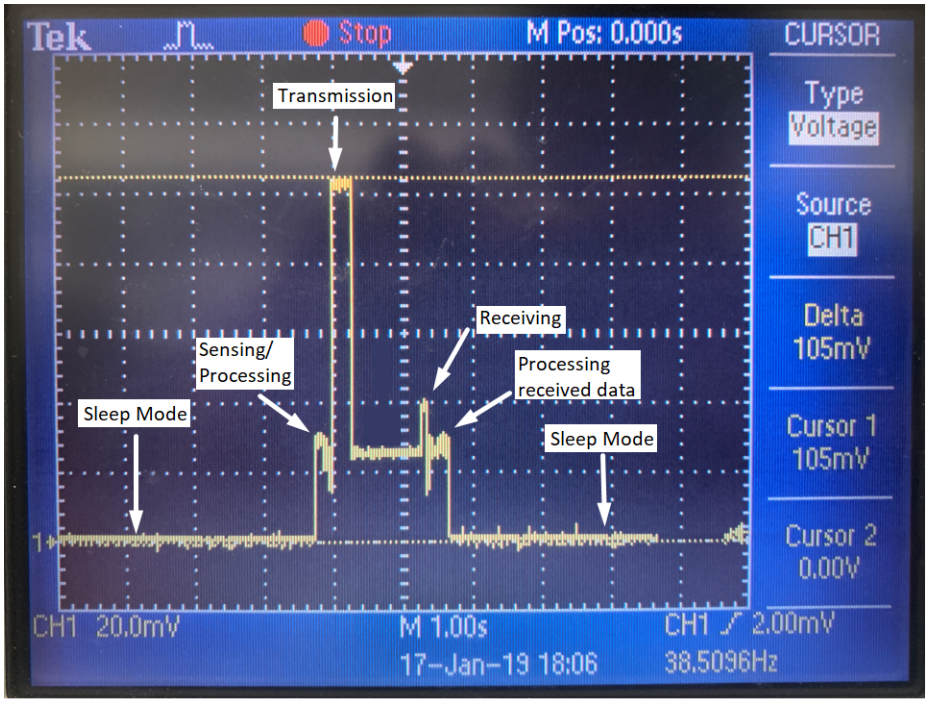


Figure 4.5: Various operations in LoRa end device seen from oscilloscope

and then goes back to sleep for the specified interval of time (figure 4.5). The average current of the node is calculated as:

$$I_{avg} = \frac{I_{tx} * T_{tx} + I_{uca} * (T_{active} - T_{tx})}{T_{active} + T_{sleep}} \tag{4.1}$$

where I_{tx} is the current drawn during transmission, I_{uca} is the active current of micro controller, T_{active} is the total active time of node, T_{sleep} is the sleep duration of node, and T_{tx} is the duration of transmission.

For calculating the lifetime of the node, the average current is divided by a battery rating of 2800mAH (considering Li-Ion battery). Table 4.2 shows the values of the life expectancy of the node for different SF and sleep times. When a sleep cycle of 10 seconds is used then the node can have a maximum lifetime of 27 days but if we increase the sleep time to 5 hours then we get a life expectancy of around 7 and a half years considering the 2800mAH battery is used (self-decay of battery not included in the calculation).

Table 4.2: Lifetime expectancy of LoRa end node for different sleep time with transmit power of $14dBm$

Battery life for sleep time intervals				
SF	10 Sec	1 Hour	5 Hour	24 Hour
7	27.65	5.95	7.47	7.87
8	25.46	5.81	7.43	7.86
9	21.83	5.54	7.34	7.84
10	19.20	5.30	7.25	7.82
	(Days)	(Years)	(Years)	(Years)

2) Effect of SF on Consumed Energy: The spreading factor does not affect the magnitude of the current drawn. By increasing the spreading factor time-on-air for a data packet increases, which in turn means that the radio transceiver module will stay ON for a longer duration and hence will increase the power consumption of the device.

3) Effect of Coding Rate on Consumed Energy: Like the spreading factor, the coding rate does not affect the magnitude of the current drawn, but as the coding rate decreases the power consumption will increase. The reason behind this is the fact that the decrease in CR means a greater number of communication bits per useful data bit and hence a longer payload. To send a longer packet, more time will be required as compared to sending a shorter message packet. Hence, the longer the message packet, the longer the transceiver module will stay ON. This accounts for the increase in power consumption.

4.3.3 Scalability

We evaluate scalability through simulations using the Network Simulator-3 (NS-3) [84]. The simulation model considers that end devices are uniformly distributed over a circular disk of radius $r = 3.7km$ [85] as shown in figure 4.6. The application running on end devices generates data every 600sec and each simulation was run for a duration of a hundred times the upstream data generation period. The transmission time of the first upstream packet for every device is picked randomly. All upstream

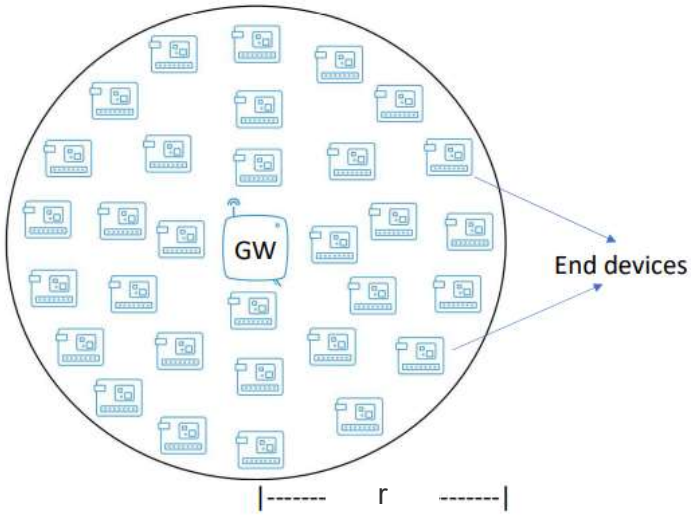


Figure 4.6: Network model to evaluate scalability

packets have an application payload of 8 bytes, which implies a PHY payload of 21 bytes. An unconfirmed upstream data packet is considered delivered if it was received successfully by a gateway node.

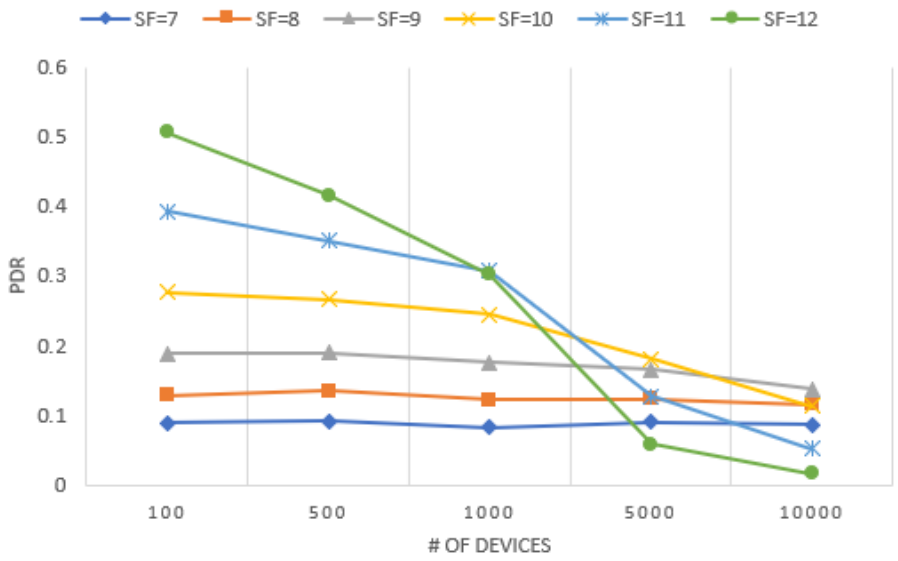


Figure 4.7: Plot showing packet delivery ratio vs number of devices with devices spread uniformly on a circle of 3 km radius

For all simulations, the average packet delivery ratio (PDR) was measured and plotted against the number of end devices for different spreading factors in figure 4.7. It is observed that with a small number of devices, the PDR is highest for the highest

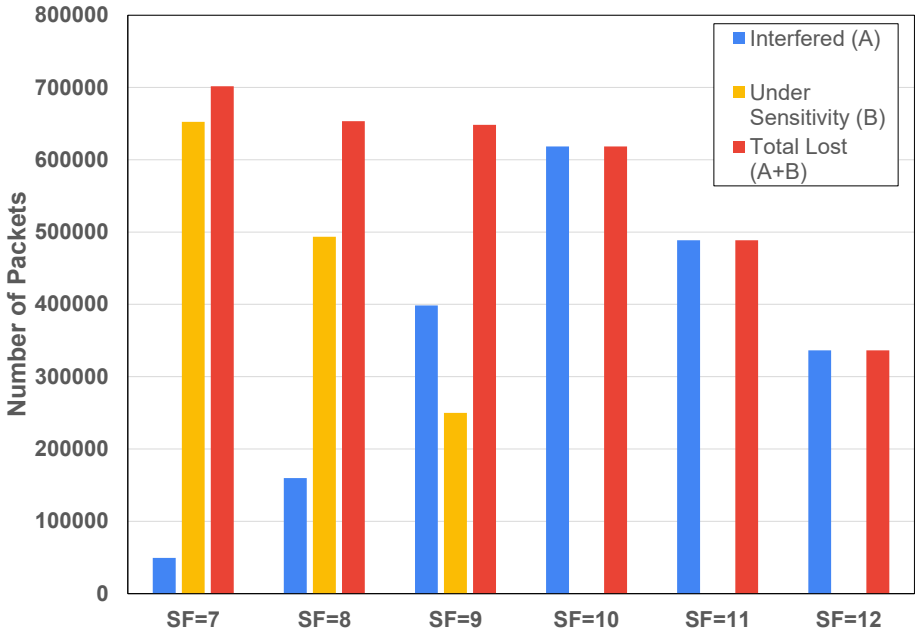


Figure 4.8: Sources of packet loss at various spreading factors with devices spread uniformly on a circle of 3 km radius

SF. The PDR decreases with decreasing values of SF when the interference is low. With the increasing number of devices in the network, the average PDR drops for all SFs but the effect is most pronounced for higher SF values. This is because a higher SF implies higher packet lengths, which increases the packet collision probability. Consequently, for heavy traffic or a higher number of devices a smaller SF works better. Overall, the performance degrades gracefully for up to 10,000 devices albeit for PDR of approximately 0.1 and low SF values, the transmission range is smaller.

The plot in figure 4.8 shows the number of packets lost and the corresponding reasons while considering 10,000 devices spread uniformly across a radius of 3 km for different spreading factors. The packets are lost due to two major reasons, collisions or interference (A), and packets received under receiver sensitivity (B). It can be observed from the plot that at lower spreading factors the major packet loss occurred due to under-sensitivity and very few packets were lost due to interference. As we move towards higher spreading factors, the major packet loss occurred due to interference, and a negligible amount of packets were lost due to under-sensitivity.

These results confirm the properties of spreading factors. The packets configured with a lower spreading factor are shorter in length as compared to packets configured with higher spreading factors. As the length of packets increases, the probability of collision also increases. Hence, more packets were lost due to interference at higher spreading factors. Transmission with a higher spreading factor also corresponds to a longer range as compared to a transmission with a lower spreading factor. Hence, more packets were lost due to under-sensitivity at lower spreading factors.

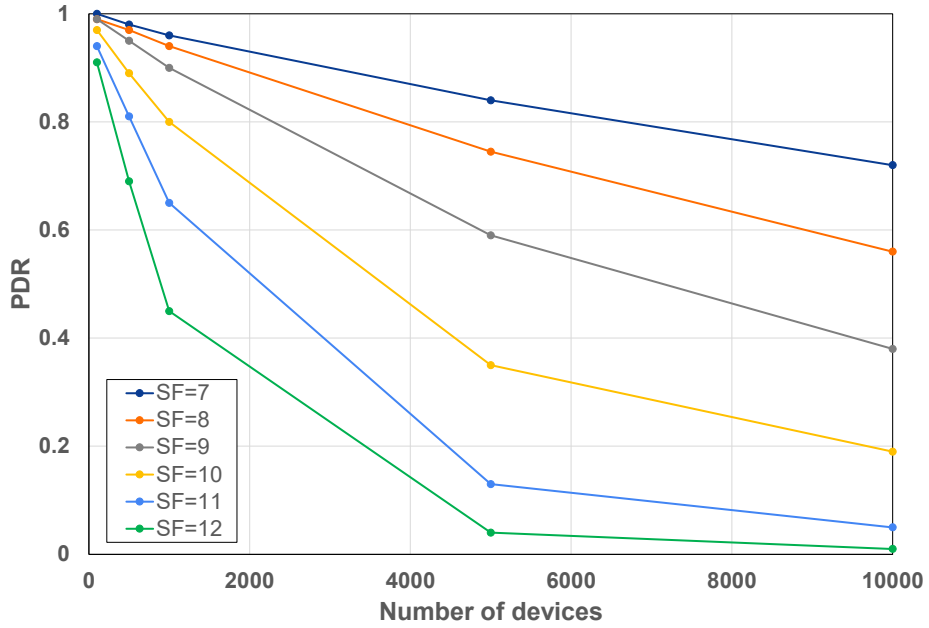


Figure 4.9: PDR for a varying number of devices spread on a circular disc of radius 1000 m with different spreading factors

In order to find the effect of spreading factors on the number of collisions, we consider a varying number of devices that are uniformly spread in a circular cell. The radius of the circular cell must be small enough that the packets sent with any spreading factor with any transmit power can reach the gateway with sufficient power. The PDR from this experiment are plotted and are shown in figure 4.9. It is evident from the plot that as the number of devices on any spreading factor increases the performance of the network degrades. It can also be observed that for a fixed number of devices, the PDR drops as the spreading factor increases.

4.4 Conclusions

Low power consumption, wide area coverage, and license-free channel operation are important characteristics of LPWAN Technologies that make them attractive for low throughput IoT applications. LPWAN technology uses single-hop long-range direct communication with the base stations that overcomes the problems of multihop delays and routing in legacy wireless technologies. LPWANs also have a high capacity, providing connectivity to tens of thousands of devices from each gateway. Key limitations found by the performance study are as follows:

- LoRa gateways can theoretically serve hundreds of thousands of devices but as the number of devices increases beyond a few hundred devices, the performance starts to degrade.
- The performance degradation happens due to two major reasons:
    1. Transmission received below the sensitivity threshold
    2. Collisions among transmissions
- As the number of devices on any one spreading factor increases the collisions among packets also increase.
- Lower spreading factors have higher data rates but smaller transmission ranges and higher spreading factors have lower data rates but longer transmission ranges.

Since the PDR, transmission range, and a high number of users, are critical for IoT networks, our research endeavors to investigate techniques for augmenting scalability beyond the capabilities offered by the legacy protocol.

CHAPTER 5: IMPROVING SCALABILITY OF LoRaWAN NETWORKS BY SPREADING FACTOR DISTRIBUTION

5.1 Introduction

In this chapter, we present an approach for improving the network capacity of LoRaWAN networks. The proposed approach is based on reducing the number of collisions in LoRa networks by allocating multiple spreading factors within the same network zones. We demonstrate that an optimum allocation of spreading factors increases the average probability of successful packet transmission to the gateway. Simulation results show that the proposed approach can substantially improve the network performance even for dense LoRa networks.

5.2 Motivation

Theoretically, a typical LoRa gateway is capable of connecting hundreds of thousands of end devices but the performance study done in chapter 4 shows that when the number of devices increases beyond a few hundred devices, the performance degrades. Hence, scalability is a major challenge for LoRa networks.

As stated in Chapter 4, collisions among the packets is the main reason for performance degradation. A packet collision takes place when two or more devices are configured on the same channel and the same spreading factor. Hence collisions can only be avoided if the devices are configured on different spreading factors and/or different transmission channels.

In LoRa networks, the transmissions on spreading factors are orthogonal to one another. If the devices in the network are configured on the same channel, the collision domain can be changed by configuring devices on different spreading factors thereby

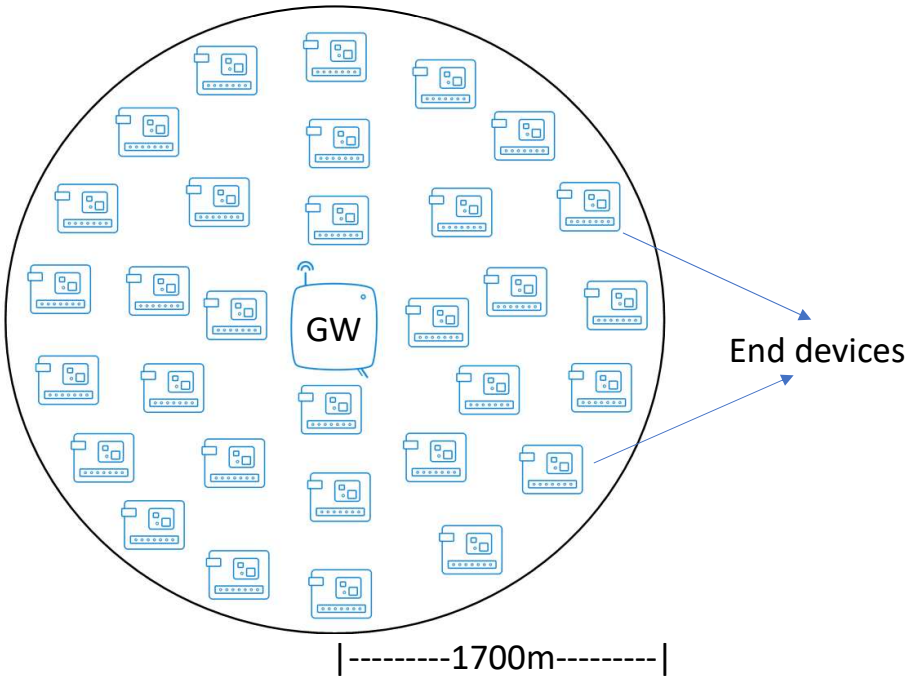


Figure 5.1: Network model

reducing the number of collisions.

The main objective of the proposed solution is to reduce the number of collisions and thus improving the performance of the LoRaWAN networks by optimally distributing transmissions on spreading factors.

5.3 Methodology

The idea behind the proposed approach is that as opposed to using the default ADR mechanism that allocates higher SFs to end devices at increasing distances from the gateway to meet the required SNR, we allocate a higher SF to a fraction of the end-devices within shorter ranges as well, in order to reduce the total number of nodes configured on the smaller SF.

5.3.1 Network Model

It is assumed that a total of 'X' end devices are present in the LoRa cell that is considered to be a circular disc of radius R as shown in figure 6.1. The radius (1700 meters) is small enough that a transmission from a device configured with any

SF can reach the single-channel gateway (GW) placed at the center of the disc. At any moment, the gateway has enough resources available to demodulate any number of valid receptions. A valid reception means that the received power of the data packet should be above the sensitivity threshold of the receiver, and there should be no collisions between packets. Also, it is assumed that data traffic is generated according to Poisson distribution

5.3.2 Principle of Operation

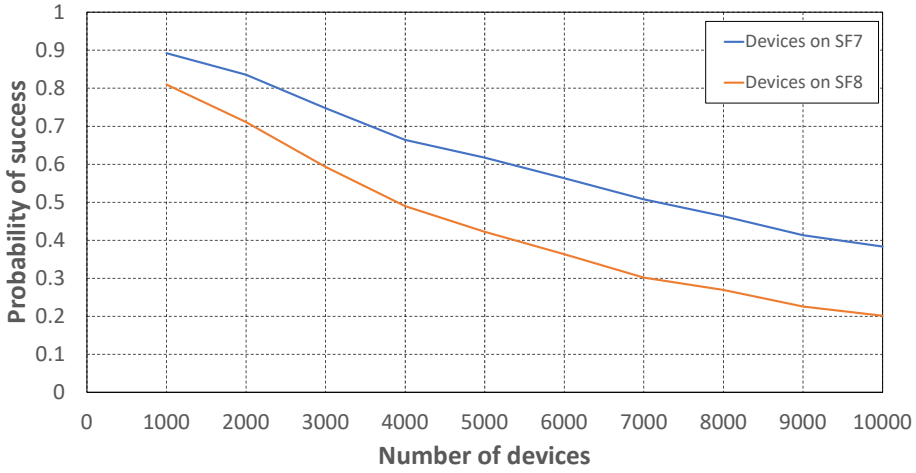


Figure 5.2: Probabilities of success with varying number of devices placed uniformly over a disc of radius 1700m

In the assumed network area, due to the proximity of the devices to the gateway, all devices can be configured to the highest data rate or lowest spreading factor SF7. However, if the number of devices configured on any specific spreading factor increases, the interference also increases. This results in higher collisions leading to lower probability of success as shown in Figure 5.2. Note that the packet success probability for the same offered traffic is lower with SF8 than in SF7, which is due to lower data rates and longer packet duration at SF8 in comparison to SF7. For instance, a success probability of 0.38 can be achieved with 10,000 devices configured on SF7, while the same success probability is only achieved with 5,500 devices configured on SF8. This is due to the fact that packets transmitted with SF8 configurations

are longer in length hence are more susceptible to collisions as compared to shorter length packets transmitted with SF7.

Another point to note is that the success probability can be improved by reducing the number of devices on the same SF. For instance, a success probability of 0.55 is achieved with 6500 devices on SF7 or with 3500 devices on SF8. The spreading factors being orthogonal to each other do not collide with one another. Hence, if 6500 devices are allocated SF7 and another 3500 devices in the same area are allocated SF8, the average probability of success will be 0.55, which is higher than that achievable with all the 10,000 devices on SF7 alone. Note that a lower spreading factor corresponds to a higher data rate and lower transmission range and vice versa. The end devices configured on a certain spreading factor may not be configured at a lower spreading factor, but they can definitely be configured to a higher spreading factor with the objective of reducing the number of collisions with some reduction in data rate. Using this principle, if the network is made aware of the number of devices configured on a certain spreading factor, it can efficiently allocate the spreading factors to the end devices, thus improving the network performance.

The distributed multiple spreading factor allocation envisions that if a certain fraction of the total number of end devices is configured to higher spreading factors, the network performance will improve many folds. Section 5.3.3 finds this optimum fraction mathematically as well as using simulations.

5.3.3 Optimum SF Allocation

For simplicity of understanding, we consider only two spreading factors (SF7 and SF8). The approach can be easily scaled for all allowed spreading factors.

Since LoRaWAN uses ALOHA-based MAC protocol, we can use the probability of success equation for pure ALOHA networks given in equation 5.1.

$$P(S) = e^{-2G} \tag{5.1}$$

$$G = \frac{N * T_{air}}{T} \tag{5.2}$$

where, 'N' is the number of devices, 'T_{air}' is the time on air for the packet, and 'T' is the total time considered.

Consider that a fraction α of the devices are configured on SF7 and $(1-\alpha)$ fraction are configured on SF8. Assume that 'i' and 'j' be the time on air for payloads of a specific size on SF7 and SF8 respectively. We can then get the probability of success from equation 5.3.

$$P(S) = e^{\frac{-2i\alpha}{T}} + e^{\frac{-2j(1-\alpha)}{T}} \tag{5.3}$$

By maximizing $P(S)$ with respect to α and for total devices N to be 10,000, time on air for 18-byte payload for SF7 and SF8 be 0.051 seconds and 0.092 seconds, we get the optimum value of α to be 0.64.

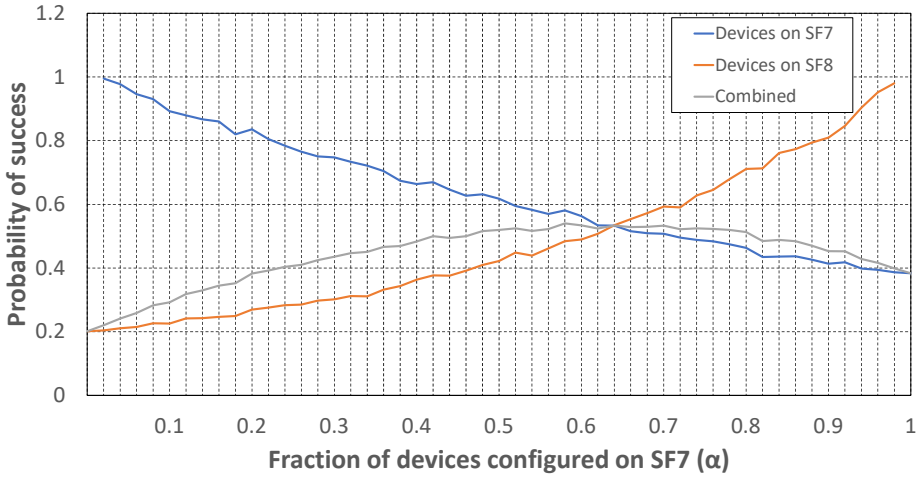


Figure 5.3: Plot of probability of success with α devices on SF7 and $(1-\alpha)$ devices on SF8 simulated independently and their combined weighted mean

We also confirm these findings with simulations. We assume a total of 10,000 end devices are placed uniformly over a disc of a radius of 1700m. The transmission from all the end devices can successfully reach the gateway placed on the center of the disc when configured with SF7. In order to find the optimum fraction of devices 'α', to be

configured on SF7 and the fraction $(1-\alpha)$ to be configured on SF8, the value of α is increased from 0 to 1 in steps of 0.02. For each value of α the probability of success is calculated for devices configured on SF7 and SF8. The calculated values are plotted in Figure 5.3.

As α increases, the number of devices configured on SF7 increases, resulting in an increase in collisions, and hence, the probability of success decreases for devices on SF7 as α increases. As α increases, $(1-\alpha)$ decreases and the number of devices configured on SF8 decreases hence decreasing the number of collisions and increasing the probability of success for SF8 devices. In summary, as α increases, the probability of success for devices on SF7 decreases, and that of devices on SF8 increases. The point of intersection of the two curves gives the optimum fraction at which the average probability of success for the whole network is maximum and that optimum fraction came out to be 0.64. In this case with 10,000 total devices, if 6400 devices are configured on SF7 and the remaining 3600 devices are configured on SF8, a success probability of 53% is achieved as compared to 38% with all devices on SF7. Although the above results were obtained considering the distribution of transmissions over two adjoining spreading factors, the performance can be further improved by considering the optimum distribution of transmissions over all spreading factors, which is presented in the next section.

5.4 Results and Discussions

The effect of the proposed approach on the performance can be studied using the simulation of the LoRaWAN network. NS-3 tool was used for the purpose of the analysis. The simulation model used to simulate the LoRa network is available at [85]. The configuration parameters used for the simulation are tabulated in Table 6.3

To simulate the real-life scenario, the devices were placed uniformly over the circular disc. Here the power capture effect also plays its role in network performance. The simulations were conducted for 5000 end devices with increasing values of α from 0

Table 5.1: Configuration parameters for simulation

Number of end devices, x	5000
Number of Gateways	1
Number of Channels	1
Spreading factors	7, 8
LoRa cell	Circular disc of radius 1700 meters
Simulation time	600 Seconds
Packet generation model	Poisson Distribution
Mean packet arrival time	$600/x$ Seconds

to 1. For each simulation probability of success was calculated and plotted in figure 5.4.

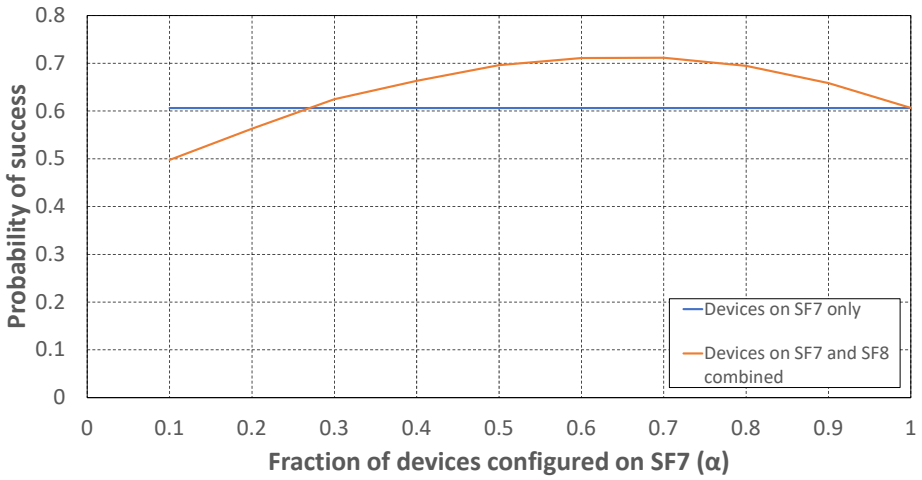


Figure 5.4: Probability of success with 5000 devices with varying α

Figure 5.4 shows the probability of success for the fraction of devices configured on SF7 and SF8 according to α. It can be observed that the maximum probability of success occurs at $\alpha = 0.64$. The optimum fraction found in Section 5.3.3 is expected to improve the network performance for any number of devices in the network. It can also be inferred that with the optimum fraction, the network can support more devices for a given probability of success. Hence, the capacity of the network will be improved.

The optimum fraction was also found considering all spreading factors using simulations. It was found by configuring 34.25% of total devices on SF7, 31.25% on SF8,

24.5% on SF9, 6% on SF10, 2% on SF11, and 2% on SF12, the maximum possible average probability of network can be achieved. The results are shown in figure 5.5.

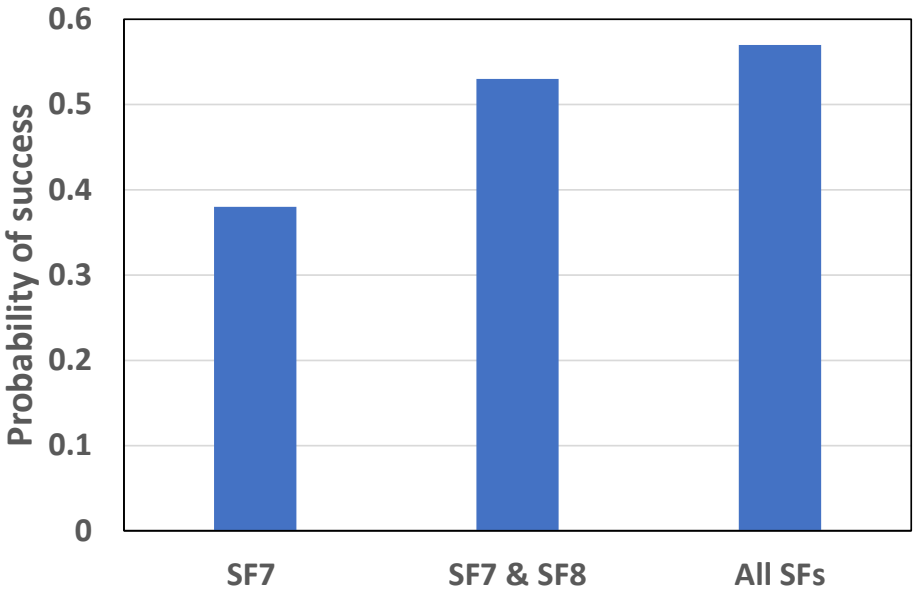


Figure 5.5: Probability of success with 10,000 devices configured on spreading factors according to optimum fraction

The approach can also be scaled to a wide area network cell that considers concentric discs of increasing radii with gateway situated at the center of the cell as shown in Figure 5.6. The entire cell can be divided into zones (Z1 to Z6). Due to the distance from the gateway, the devices in Z6 can only be configured on SF12 and not any lower spreading factor as it will not be able to reach the gateway with sufficient power and will be lost. Similarly, devices in Z5 can be configured on SF11 & SF12, Z4 on SF10, SF11 & SF12 and so on. The above sections calculated the optimum ratio for Z1 only in which the devices can be configured on any six spreading factors. In a similar way, an optimum ratio of devices can be found for all the zones. By doing so, a single gateway will be able to explore its full potential of covering a larger area and the optimum fractions will make sure of the reduced number of collisions.

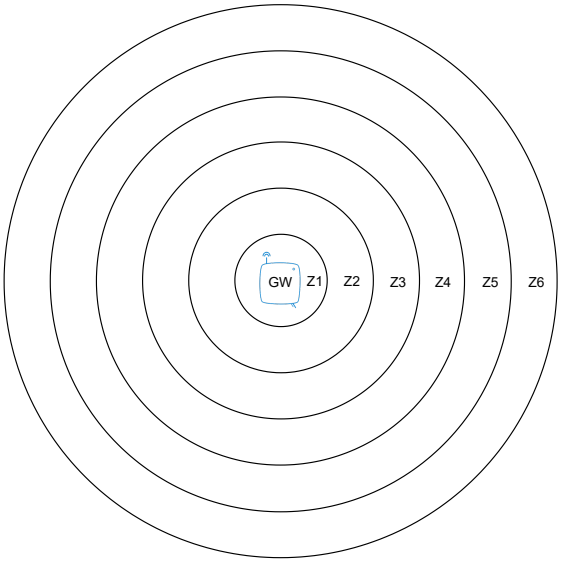


Figure 5.6: Network cell divided into various zones with increasing radii

5.5 Conclusions

We evaluated the possibility of improving the packet success probability in LoRaWAN by allocating multiple spreading factors (SF) to devices in a zone where all devices could be potentially assigned the smallest SF. The study indicates that there is an optimum fraction for the allocation of two different SFs in the same zone that maximizes the packet success probability. The average success probability using the optimum SF allocation increases significantly. It does so by configuring a fraction of the total devices to a higher spreading factor thus changing the collision domain and hence reducing the number of collisions. The optimum fraction at which the improvement is maximized was found both mathematically as well as using simulations and it came out to be $\alpha = 0.64$ for SF7 and SF8. The optimum fractions were also found considering all spreading factors. The results support this claim.

CHAPTER 6: QoS BASED SPREADING FACTOR DISTRIBUTION FOR LoRaWAN NETWORKS

6.1 Introduction

Recent developments in technology had a great impact on the exponential increase in the number of IoT applications [86]. One of our interests is with the smart city scenario, comprising various types of applications such as HVAC control, smart street lights, smart utilities, smart waste management, etc., running on IoT devices. IoT devices in such applications are typically embedded devices that are connected to the Internet using backhaul technologies such as Wi-Fi, Low Power Wide Area Networks (LPWAN), Zigbee, Cellular technologies, etc. Most embedded devices run a single application, hence one category of devices will be running a single application and other categories of devices will be running an entirely different set of applications. For example, one category of devices that consists of smoke sensors and buzzers can be running a fire alarm application and another category of devices consisting of a temperature and humidity sensor can be running an application to control the Heating, ventilation, and air conditioning (HVAC) system. Each application has different Quality of Service (QoS) requirements. These QoS requirements may be specified in terms of packet success rates, latency requirements, data rate requirements, bandwidth requirements, etc. Also, some applications have strict QoS requirements and others have relaxed requirements[72]. In the example application case of fire alarm and HVAC control discussed above, the fire alarms will have strict QoS requirements as compared to the HVAC control application. The underlining network must satisfy these requirements in order for the application to run efficiently on the IoT devices.

LoRaWAN does not differentiate between different types of devices and considers

all to be the same. Due to these considerations, specific QoS requirements are not guaranteed to be satisfied.

In the previous chapter, we exploited the orthogonality property of spreading factors and calculated an optimum distribution of devices under different spreading factors that reduced the total number of collisions in the network. This optimum distribution improves the network performance significantly but does not guarantee to satisfy the QoS requirements. In this work, we propose a protocol that will guarantee to satisfy the QoS requirements for the applications. We demonstrate analytically as well as using simulations that the proposed approach improves the performance of the network while satisfying the QoS requirements.

6.2 Motivation

The performance of the LoRaWAN network degrades as the number of devices increases and as such the network can not guarantee to satisfy the QoS requirements for IoT applications. The ADR mechanism is also not able to improve the network performance. According to the ADR mechanism, all devices start with the highest spreading factor and at higher traffic loads, this will add to the number of collisions. The algorithm will either take too long to converge or it will fail to converge due to the high number of collisions. Different SF assignment schemes that try to improve the performance of the networks are discussed in chapter 3 but they do not address the posed QoS requirements. One of the articles, [87] discusses the research gap in LoRaWAN technology regarding the QoS requirements. Our proposed approach tries to fill this gap and improve the performance of LoRa networks while satisfying the QoS requirements posed by IoT applications. The approach exploits the orthogonality property of spreading factors and performs the assignment based on the QoS requirements.

6.3 QoS Based Spreading Factor Allocation

This section describes the proposed mechanism to assign spreading factors according to the QoS requirement. The main idea is to assign spreading factors to devices based on the application QoS requirements as opposed to using the default ADR mechanism that allocates higher SFs to end devices at increasing distances from the gateway to meet the required RSSI and SNR thresholds[29].

6.3.1 Defining QoS Requirements

For ease of understanding, we define two levels of QoS requirements considering two different types of IoT applications running on the application server:

1. ***High priority (HP) applications:*** Applications that require a guaranteed QoS such as those that are characterized by the maximum latency or minimum delivery rate. Examples of such applications include fire and intrusion alarms.

2. ***Low priority (LP) applications:*** Applications that have relaxed latency and delivery requirements such as environmental monitoring sensors used for HVAC control, smart agriculture, etc.

Both types of applications consist of sensing devices that transmit their status to the server via gateway through the LoRa link. We assume that the QoS requirements can be achieved by *a minimum probability of success for packet delivery* for the HP devices. This assumption is analogous to the assumptions made in [74], which preempts the low-priority traffic to serve the high-priority traffic. Preempting the low-priority traffic to serve the high-priority traffic means sacrificing all the low-priority traffic for the sake of high-priority traffic. Instead, we consider the high-priority traffic must have a certain probability of success as the strict QoS requirement but for the low-priority traffic, there are no such strict QoS requirements.

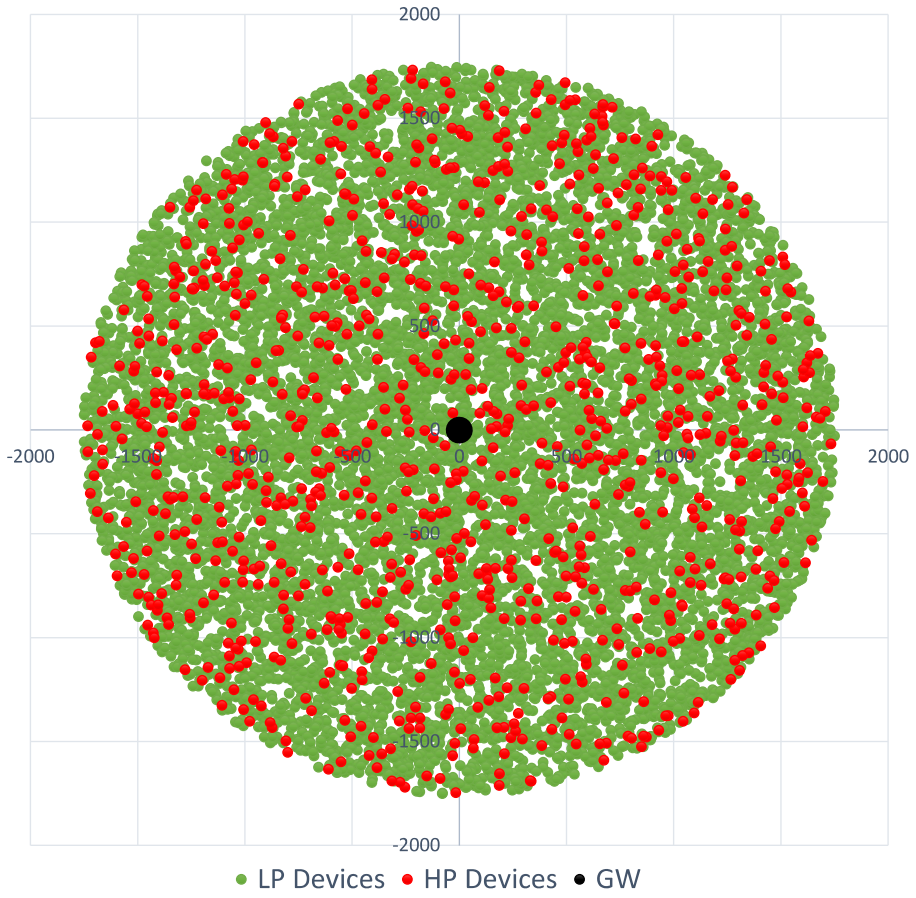


Figure 6.1: Proposed network model with single gateway serving a circular area of radius 1700 meters

6.3.2 Network Model

We assume a network comprising N devices in the network that are uniformly distributed over a circular area of radius r, as shown in Figure 6.1. A fraction β of total devices consists of high-priority devices and the remaining are low-priority devices. A gateway is placed at the center of this disc. The radius is small enough that transmission from any device configured with any spreading factor can reach the gateway. It is assumed that the gateway has enough resources available to demodulate any number of received transmissions that are above the sensitivity threshold. The traffic in the network is generated according to Poisson distribution.

Table 6.1: Time on air for various SFs for a payload of 8 bytes with 8 preamble symbols.

Spreading Factors	**Time on Air (sec)**
SF7	0.0361
SF8	0.06195
SF9	0.1239
SF10	0.24781

6.3.3 Principle of Operation

Due to the proximity of devices to the gateway in the assumed LoRa cell, the fastest data rate or the lowest spreading factor i.e., SF7 can be assigned to all the end devices. However, as observed in studies such as [18],[19],[88], the probability of collisions may be reduced by distributing the devices over multiple spreading factors. The transmissions on different spreading factors being orthogonal to each other do not collide with one another and by configuring devices on different spreading factors changes the collision domain of the devices.

Assuming a Poisson arrival model with an average arrival rate of λ, the packet success probability P_s can be calculated using the analysis applied to the ALOHA protocol

$$P_s = P(k = 0) = e^{-2G} \tag{6.1}$$

where $P(k)$ represents the Poisson probability of k transmissions per packet duration, and G is the average number of transmission attempts per frame time. In our case, G can be calculated as

$$G = \frac{D * ToA}{Total\ time\ period} \tag{6.2}$$

where D is the total number of devices attempting to transmit during the total time period, and ToA is the time-on-air, i.e., the duration for which a packet stays over the transmission medium.

Table 6.2: Maximum number of devices that can be configured on various SFs transmitting payload of 8 bytes with more than 90% probability of success.

Spreading Factors	**Max. number of devices**
SF7	870
SF8	500
SF9	250
SF10	120

Using the values of ToA from Table 6.1 and equations (6.1) & (6.2), and setting the value of probability according to the QoS requirements of high-priority traffic, the value D can be calculated. The value D, in this case, will represent the maximum number of devices that can be assigned to a specific spreading factor satisfying the QoS requirements. The values were calculated for a 90% probability of success and are tabulated in Table 6.2.

From Table 6.2 it can be inferred that the maximum number of devices that can be configured on a spreading factor decreases as we move from a lower spreading factor to higher one. This corresponds to the fact that as we move from the lower spreading factor to a higher spreading factor the packet length increases and that increases the probability of collision.

6.4 Proposed Approaches

In this section, we propose the rules that will be followed to allocate the spreading factors. Before delving into the proposed algorithm we define some of the variables that will be used for the remainder of the chapter:

N: Total number of devices in network

β: Fraction of high priority devices in network

p: Number of high-priority devices in network

q: Number of low-priority devices in network

x: Probability of success threshold for high-priority devices

a: Number of devices supported by the SF7 satisfying the QoS requirements

b: Number of devices supported by the SF8 satisfying the QoS requirements

c: Number of devices supported by the SF9 satisfying the QoS requirements

d: Number of devices supported by the SF10 satisfying the QoS requirements

6.4.1 Approach-1: SFA-1 Approach

In SFA-1, HP devices are assigned SFs starting with the highest SF first, and then going down to lower SFs as they get filled up to the limit where QoS can be guaranteed. SF allocations also take into account optimum distributions as discussed in Chapter 5. To clarify, we describe SFA-1 by performing SF assignments based on the following rules:

1. For a given N and β, if 36% of $p < d$ then p devices will be distributed according to 64% and 36% of p devices among SF9 and SF10 respectively and the q devices will be distributed among SF7 and SF8 in 64% and 36% respectively.

2. For a given N and β, if 36% of $p > d$ then d number of devices out of p devices will be configured on SF10 and remaining p-d devices will be configured on SF9, given that $p - d < c$. The q devices will be distributed among SF7 and SF8 in 64% and 36% respectively.

3. If $p > c + d$ but $p < b + c + d$ then all the low priority devices will be configured on SF7 and high priority devices will be configured on SF8, SF9 and SF10 according to optimum distribution.

4. If $p > b + c + d$, decrease the x and find the values of b, c, and d for this new x.

5. Configure all p devices on SF8, SF9, and SF10 in optimum fraction. The thresholds b, c, and d must be maintained at all times.

6.4.2 Approach-2: SFA-2 approach

Contrary to SFA-1, the SFA-2 starts by assigning the lower spreading factors to HP devices and then going up to higher SFs as they get filled up to the limit where QoS can be guaranteed. Similar to SFA-1, allocations also take into account optimum distributions. The rules to allocate the spreading factors to high-priority devices and low-priority devices under the SFA-2 approach are as follows:

1. For a given N and β, if 64% of $p < a$ then p devices will be distributed according to 64% and 36% of p devices among SF7 and SF8 respectively and the q devices will be distributed among SF9 and SF10 in 64% and 36% respectively.

2. For a given N and β, if 64% of $p > a$ then a number of devices out of p devices will be configured on SF7 and remaining p-a devices will be configured on SF8, given that $p - a < b$. The q devices will be distributed among SF9 and SF10 in 64% and 36% respectively.

3. If $p > a + b$ but $p < a + b + c$ then all the low priority devices will be configured on SF10 and high priority devices will be configured on SF7, SF8 and SF9 according to optimum distribution.

4. If $p > a + b + c$, decrease the x and find the values of a, b, and c for this new x.

5. Configure all p devices on SF7, SF8, and SF9 in optimum fraction. The thresholds a, b, and c must be maintained at all times.

6.5 Performance Evaluation

The performance of the proposed allocation schemes have been evaluated mathematically as well as via network simulations. We assume the QoS requirement for the high priority devices to be defined by a minimum packet success rate of 90% or above. For comparison, we evaluate the packet success probabilities of the high priority and low priority devices as evaluated with the proposed SFA-1 and SFA-2 schemes with that of the legacy LoRaWAN network as well as that using the optimum distribution of spreading factors as presented in [19].

6.5.1 Mathematical Evaluation

The packet success probabilities of the proposed approaches can be evaluated using Poisson's distribution equation. In order to find the performance of proposed approaches, end devices need to be configured on different spreading factors according to the rules stated in section 6.4. Once the configuration is complete for any number of total devices, the probability of success for each spreading factor can be found using equations 6.1 and 6.2. A weighted average of the probability of success for different spreading factors on which high priority devices are configured gives the probability of success for high priority devices. The same process is followed to get the probability of success for low priority devices. This process was done for a different number of total devices in the network.

6.5.2 Simulation model

The Network Simulator-3 (NS-3) [84] simulator was used for the analysis. Details of the simulation model are presented in [85]. To simulate the real-life scenario, the model considers uniformly spread end devices on a circular disc of radius 1700 meters. Selection of HP devices as well as assignment of spreading factors are done randomly, using uniform distributions. The network traffic is generated according to Poisson distribution. The simulation is based on the configuration parameters tabulated in

Table 6.3: Configuration parameters for simulation

Number of end devices, N	1000-10,000 in increments of 500
Number of Gateways	1
Number of Channels	1
Spreading factors	7, 8, 9, 10
LoRa cell	Circular disc of radius 1700 meters
Simulation time	600 Seconds
Application Payload	8 bytes
Packet generation model	Poisson Distribution
Transmit Power	Default power 14dBm
Mean packet arrival time	$600/N$ Seconds

Table 6.3.

The simulations were conducted for number of end devices varying from 1000–10,000 devices increasing in steps of 500 devices. We measured the average probability of success for high priority and low priority devices individually from each simulation.

6.5.3 Results and discussions

This section presents the performance results obtained mathematically as well as from simulations. The legacy LoRaWAN configures the end devices on spreading factors based on the RSSI and SNR, whereas the optimum distribution approach distributes spreading factors that maximize the average packet success rate. Both of these schemes consider all transmission to have the same level of priority.

Figure 6.2 shows the results from the mathematical evaluation of the packet success rates for the proposed SFA approaches as well as those obtained for the legacy network and the optimum distribution approach. It can be observed from the plot that the legacy approach achieves success rates of about 95% for fewer number of total devices (500 devices). Even for 1000 devices in the network the legacy approach does not seem to satisfy the QoS requirement. The optimum distribution approach achieves better performance than the legacy approach. It achieves a success rate of 97% for 500 devices in the network but this approach also fails to satisfy the QoS requirements once the number of devices exceeds 1500 devices. SFA-1 and SFA-2 both are able to

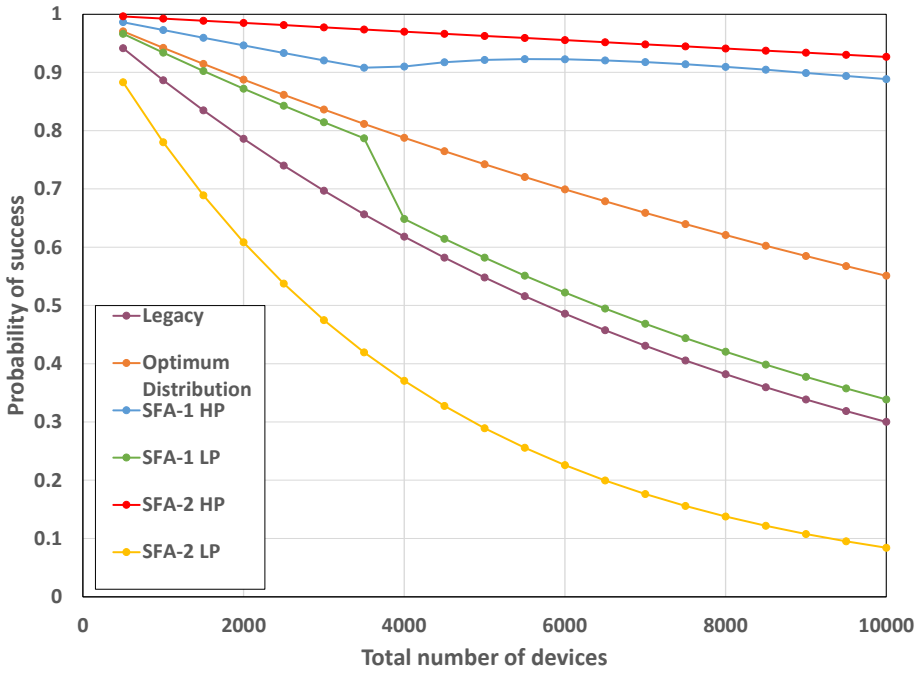


Figure 6.2: Probability of success for various approaches with increasing number of total devices in the network as obtained from the mathematical model with $\beta = 0.1$

satisfy the QoS requirements for the given range of the number of devices. In addition to satisfying the QoS requirements, SFA-1 was also able to achieve better performance for low priority devices. It can also be observed that for SFA-1, the probability of success for low priority devices decreases gradually with the increasing number of devices, except for 4000 total number of devices, where a sudden degradation in performance is observed. This sudden degradation occurred due to fact that SF10 and SF9 combined could not serve the increasing high priority devices and hence, according to the rules, they need to be configured on SF8 in addition to SF10 and SF9, limiting low priority devices to SF7 only. This resulted in the observed sudden degradation in probability of success.

Both the legacy and the optimum distribution approaches do not differentiate devices on priority levels and hence all the devices will have the same probability of success irrespective of their priority level. The optimum distribution performs better than the legacy because it distributes devices on all spreading factors according to an

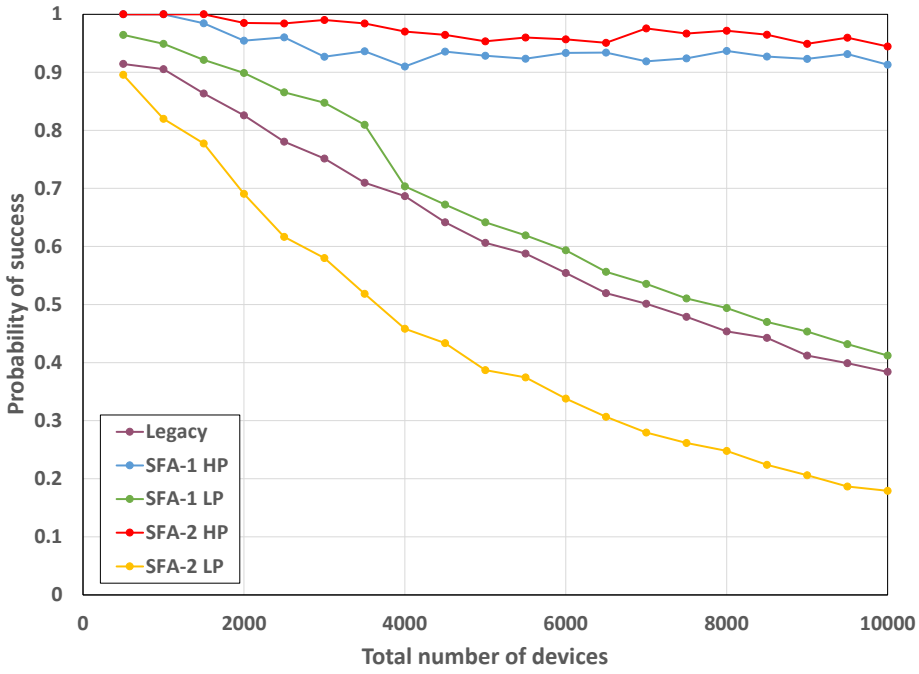


Figure 6.3: Probability of success as obtained from simulation of proposed and legacy approaches with $\beta = 0.1$.

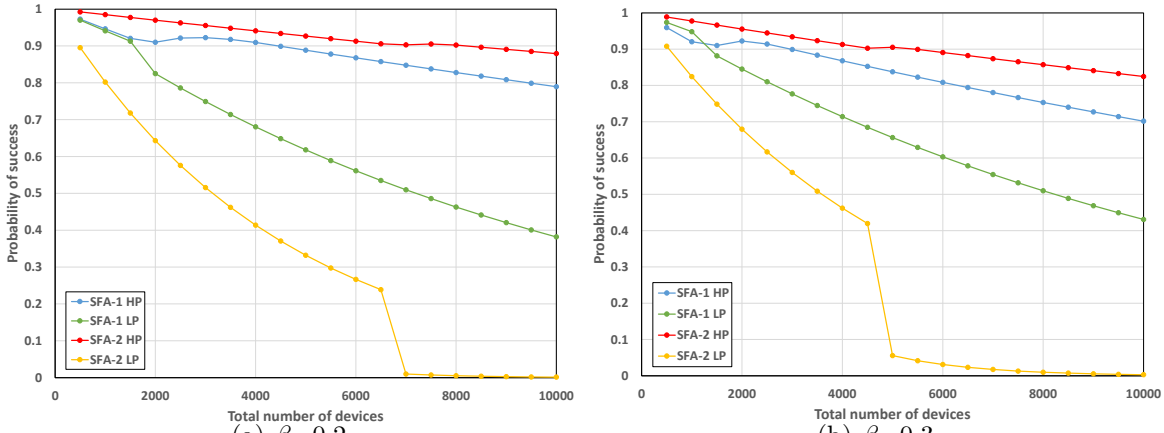
optimum fraction that improves the performance as compared to the legacy.

The proposed allocation schemes consider low priority and high priority devices differently. As SFA-2 starts configuring the high priority devices from the lowest spreading factor and significantly higher number of low priority devices on high spreading factors, the high priority devices enjoy a very high probability of success while the performance of low priority devices substantially degrades.

The higher spreading factors can serve a lesser number of devices as compared to lower spreading factors for the same performance levels, hence by configuring high priority devices in the network, which are fewer in number as compared to low priority devices, to the higher spreading factors and low priority devices on lower spreading factors, the SFA-1 approach tries to optimize the performance of both the high and low priority devices.

As the spreading factors are not perfectly orthogonal, some collisions can occur due to imperfection in orthogonality among spreading factors [89]. In order to take these

collisions into account both of the proposed approaches were simulated. The results of the simulation, as plotted in figure 6.3, indicate slightly better performance for all schemes as the simulator considers the power capture effect that is not included in the mathematical model.



(a) β=0.2 (b) β=0.3

Figure 6.4: Probability of success with both proposed allocation schemes with (a) β=0.2, and (b) β=0.3 with the varying total number of devices

Figure 6.4 shows the performance comparison of SFA-1 and SFA-2 for higher values of β. For both of the allocation schemes, it can be observed that whenever the number of high priority devices increases more than the respective thresholds, $b+c+d$ in the case of SFA-1 and $a+b+c$ in the case of SFA-2, neither of the allocation schemes is able to satisfy the QoS requirements, and hence x was reduced. It is also intuitive that for a fixed N, p is greater when $\beta = 0.3$ as compared to when $\beta = 0.2$, resulting in a lesser number of q devices in the former case. Hence, the performance of low priority devices should improve and the same can be observed for SFA-1 but in the case of SFA-2, as N exceeds 5000 devices, the performance starts to degrade even further. This degradation stems from the fact that in SFA-2, lower spreading factors that have higher thresholds, are allocated to p devices that are lesser in number as compared to q devices. SFA-2 provides better QoS for high priority devices when compared with SFA-1 by almost sacrificing the performance of low priority devices.

However, SFA-1 tries to optimize the performance of low priority devices while trying to maintain the requirements of high priority devices. Hence, the SFA-1 approach is better than SFA-2 in terms of success rates for low priority devices while maintaining QoS for high priority devices.

This work has considered devices within 1700 meters radius but full coverage potential of LoRa gateway can also be considered. In that case, the devices that are farther away from the gateway can only be configured to higher spreading factors. Hence, the high priority devices that are at the maximum coverage distance can only be configured to SF10, but according to SFA-2, high priority devices will be configured on lower spreading factors so high priority devices that are farther from the gateway will not be served and hence the QoS requirements may not be satisfied. However, SFA-1 configures high priority devices on higher spreading factors and low priority devices on lower spreading factors, hence SFA-1 will be able to scale to the full coverage potential of the gateway.

Hence, the SFA-1 approach achieves better performance than other approaches.

6.6 Conclusions

In this chapter, we presented design considerations for a LoRaWAN network to meet specific QoS requirements for IoT applications. Specifically, we proposed spreading factors assignment schemes that try to meet a minimum packet success rate for end devices that require high-priority service while still providing an adequate packet success rate for other devices, i.e., those for which the packet success rate is not highly critical. Such differentiated QoS requirements are typical of IoT applications in smart city environments.

Legacy LoRaWAN networks typically assign spreading factors depending on the signal-to-noise ratios of the end devices. A higher spreading factor enables communication at a lower SINR at the cost of lower data rates. Here, we propose spreading factor assignment schemes that utilize their orthogonality to limit the number of con-

tending end devices using the same spreading factor. Two different spreading factor allocation schemes are introduced that assign all high-priority devices to spreading factors such that the packet success probability is within acceptable limits. The approaches were evaluated mathematically as well as using a network simulator. It was observed that by assigning the spreading factors according to the proposed SFA-1 approach the network was able to satisfy the QoS requirements of the IoT applications that can be deployed in smart cities.

This work considers transmissions on a single channel. In a multi-channel deployment, the end devices perform frequency hopping for every transmission. They can select transmission channels randomly.

CHAPTER 7: SHORT TERM TRAFFIC CONGESTION PREDICTION WITH DEEP LEARNING FOR LoRa NETWORKS

7.1 Introduction

This chapter explores the application of machine learning in an IoT network for traffic monitoring to enhance its services using intelligent transportation systems (ITS). The objective is to deepen our understanding of the constraints and demands associated with the communication cost of implementing machine learning in IoT applications.

The application of IoT technology in ITS enables the development of smart roads where IoT sensors are embedded across the roadways to deliver real-time traffic data for learning and monitoring traffic patterns. This data can be fed to machine learning and deep learning algorithms in the cloud to extract vital information. Such a system can help in improving the efficiency of the ITS services in terms of modeling, planning, and management.

With the increase in the number of vehicles and highways, traffic congestion problems are an increasing concern in most urban areas. In 2020, there were an estimated 286.9 million registered vehicles in the US, which also has the largest road network in the world, comprising 6.59 million kilometers [90]. Further increase in traffic leads to critical issues threatening the safety of the drivers and harming the environment. High traffic congestion leads to delays, drivers' mental stress, incidents, and pollution [91]. Real-time traffic congestion prediction information can be exploited by transportation services to make better decisions and improve traffic rules and infrastructures. Predicting traffic patterns involves collecting and periodically recording vehicles' speed, position, and traffic flow [92]. With LoRa devices deployed, smart

traffic monitoring can provide signal lights adjustment, advance planning to reach a destination, trip scheduling, and a safer and better driving experience [93].

Most of the existing traffic congestion prediction solutions are video-based, where data collection is often unstable and computationally expensive in such varying conditions. To capture the dynamic changes in the traffic flow in real-time, advanced sensors with specific capabilities are required for data acquisition. Moreover, advanced learning models can be explored for real-time traffic congestion prediction with labeled and unlabeled data for high accuracy. In this chapter, we consider a LoRa network using low-power IoT devices, which is motivated by its low-cost and wide-area coverage capabilities. However, LoRa devices have limited bandwidth for data transmission, which is restricted to only 125 kHz and 500 kHz in the US [15]. Such bandwidth limitation implies that the system can capture only basic information about the traffic state for prediction.

This work aims at developing and training machine learning models using site-specific data that can successfully predict traffic congestion with low bandwidth data available from the IoT sensors, and evaluate the use of machine learning for smart city IoT applications. To achieve this objective, we used a dataset with temporal data collected from a wide range of traffic sensors instead of real-time data from LoRa sensors. We trained and tested centralized machine learning models using this dataset in order to evaluate their accuracy. We approached intelligent traffic congestion prediction as a short-term task to predict the short-term future of 15 minutes to a few hours. We modeled the traffic parameters to learn about the traffic flow patterns. The long short-term memory (LSTM) algorithm is used for model training and testing. We present numerical results that indicate that the LSTM-based traffic congestion prediction model using low-bandwidth data provides better prediction performance in comparison to other models.

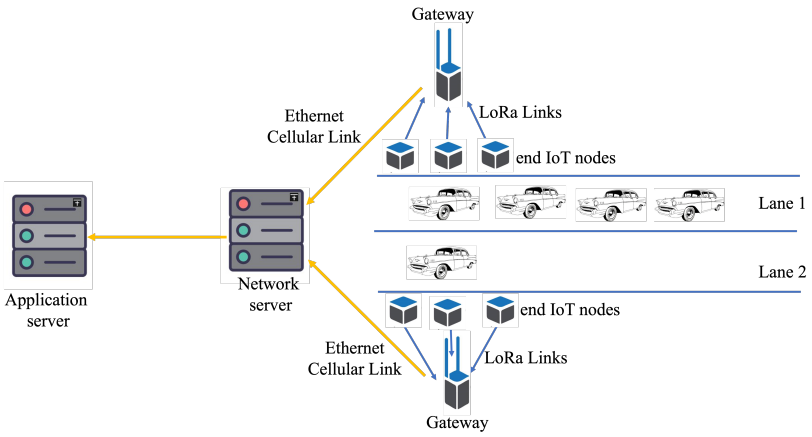


Figure 7.1: Block diagram of the proposed traffic congestion prediction system.

7.2 Methodology

The recurrent neural network (RNN model) is suitable for time series processing and sequence classification and can learn from large datasets of sequential data flow with long intervals. However, it cannot perform well when sufficient data are unavailable with input and output gates. LSTM is an improved version of RNN that is able to deal with missing data as it has an additional gate called forget gate. This gate allows discarding former values and updating new memories if it determines they cannot be passed to the output gate. LSTM allows exploiting datasets with missing data or unavailable data at timestamps, which makes it a suitable algorithm for traffic congestion prediction [94].

Fig. 7.1 represents a conceptual block diagram of how the traffic data is proposed to be collected to predict congestion. The IoT end nodes collect data and send them to the gateways over LoRa links. The gateways then forward the collected data to the network server through TCP/IP links. An application server receives the collected data for processing and model training. As illustrated in Fig. 7.2, the proposed system model of the traffic congestion prediction can be represented in four layers: data layer, processing layer, serialization layer, and deserialization layer. The data layer as presented in Fig. 7.1 includes two main units: collection and transmission.

It is assumed that at the collection unit, traffic data are collected using LoRa sensors deployed across the road segments to capture real-time traffic data. These sensors are dedicated sensors deployed permanently for specific IoT applications. At the data transmission unit, each sensor node periodically sends the collected data to one of the gateways, which act as relays between the sensor nodes and the network server. The processing layer involves two main units: data pre-processing and model development. Data pre-processing aims at cleaning and preparing the collected data to feed the machine learning model for training. At the model development unit, the model is built and trained by applying one or more machine learning algorithms to the training dataset.

The trained model is then tested with the testing dataset to verify the fitted model on data that were not involved in its training. The model accuracy is then evaluated by comparing the true values with the predicted values. If the accuracy is not satisfied, the trained model must be updated by considering more input features or more training data. Model tuning can be used to improve the model by boosting the tuning parameters, namely the number of iterations, complexity, and learning rate. After the trained model is validated, it is serialized and saved in a particular format for deployment at the serialization layer. At the deserialization layer, the saved model can be then loaded and applied to new data for future prediction in the production environment. Intelligent traffic management systems can use the saved model to save resources instead of re-training the model when new data are available.

7.2.1 Data source

Traffic datasets are classified into two main categories: probe and stationary datasets. Probe datasets provide GPS data from moving vehicles covering the entire road network. Stationary datasets provide data from sensors installed at known and specific locations. Stationary sensors include inductive loop detectors, imaging sensors, and audio-based sensors. Inductive loop detectors are point sensors buried in the road

pavement to identify the presence of a vehicle using an induced current. Imaging sensors are video camera systems installed across the road to capture images for traffic conditions recognition. Audio-based sensors are also fixed sensors used to record vehicle sounds. Stationary sensors are low-cost solutions followed by audio-based sensors. The imaging-based solution depends on the light conditions for reliable image analysis and consumes high bandwidth when sending large images to the traffic monitoring system. GPS-based solution is not appropriate for localization for IoT applications within a LoRa network.

We consider a system where traffic data are gathered from low-power sensors installed over the roads. A number of inductive loop detectors are distributed at specific locations to continuously measure the data traffic and forward them to the central server at low bandwidth. These sensors can measure traffic flow, vehicle speed, and road occupancy. The collected data are then explored for further analysis with centralized machine learning. Since such a system of sensors was not available at this time, for this work we used a public dataset comprising similar traffic information for training and testing the models. Our models can be then used for traffic congestion predictions with real-time data when a real-life system of sensors is available. A number of datasets from different sources were investigated and extensively evaluated to find the one that meets our objectives.

7.2.2 Pre-processing

As the efficiency of traffic congestion prediction depends mainly on the quality of the data and their characteristics, data pre-processing is one of the fundamental steps before training. It aims at dealing with missing data by applying one of the available solutions to retrieve the data by excluding these data or replacing them with others. Some parameters are excluded from the dataset as they do not have any impact on traffic prediction. Before building the model, the dataset is split into 70% for training and 30% for testing.

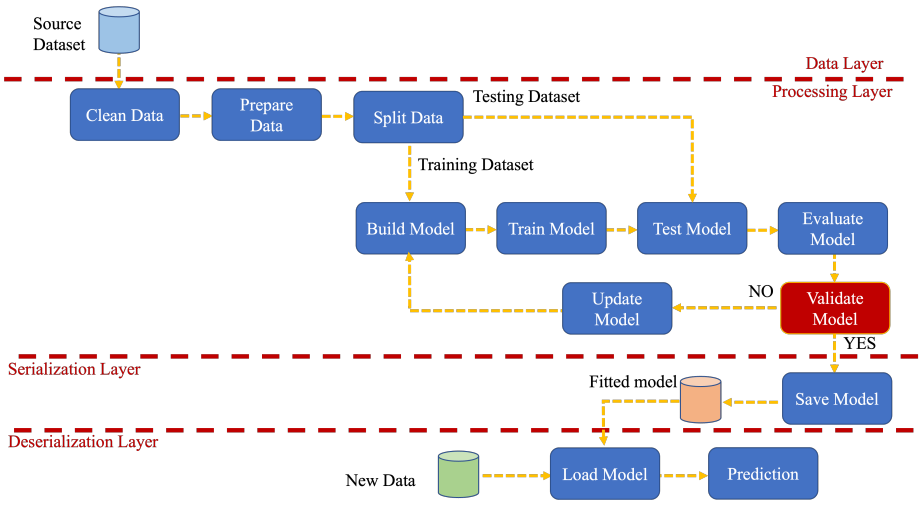


Figure 7.2: System model.

7.2.3 Modeling

Traffic congestion prediction can be represented as a classification problem that aims at classifying the traffic flow into different classes: heavy, moderate, and low. It can also be represented as congested traffic and no congested traffic. Congested traffic class corresponds to traffic flow higher than the maximum load capacity, low average speed, high average occupancy, and high average delay with respect to the threshold speed. No congested traffic class corresponds to smooth traffic flow with minimum delays. We explored both cases and adopted the second one for this work. Based on the training dataset, the model learns how to predict the traffic congestion output from the extracted features including traveling time (timestamp) and traffic flow during each 5 minutes interval at different locations.

As a time series model, the LSTM model is used to predict congestion by handling periodic traffic data at peak hours, weekdays, weekends, nights, and days. With periodic features, it is expected that the traffic flow is heavy at the commuting time. To extract the traffic flow pattern during a specific window of the day, previous window data is considered for temporal dependencies of the traffic data. Time series clustering is not considered as grouping data points based on similarity is not needed.

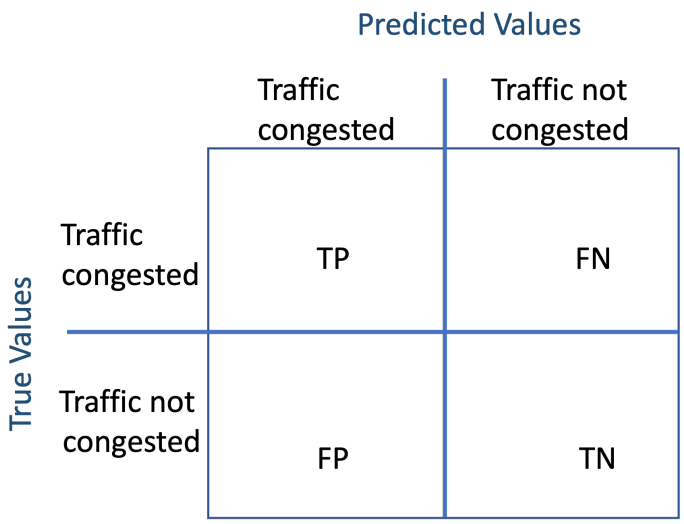


Figure 7.3: Confusion matrix for traffic congested and traffic not congested.

Long-term dependency is considered to minimize the impact of factors impacting the congestion prediction.

7.2.4 Evaluation

To evaluate the LSTM model and measure its performance, a number of metrics are considered, namely accuracy, loss function, confusion matrix, recall, precision, mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE), success rate, and computing time. Accuracy refers to the number of times the classification is correctly performed by the total number of points. It is expressed using the confusion matrix (see Fig. 7.3), which is a 2-by-2 matrix representing the correct prediction in classifying the traffic into two classes: congested and not congested. The confusion matrix represents four metrics corresponding to the outcomes of the machine learning classifier, namely true positive (TP), true negative (TN), false positive (FP), and false negative (FN).

True positive refers to the number of times the model correctly predicted the output as traffic congested. True negative refers to the number of times the model correctly predicted the output as traffic not congested. False positive refers to the number of times the model incorrectly predicted the output as traffic congested. False negative

refers to the number of times the model incorrectly predicted the output as traffic not congested. Accuracy is then expressed as

$$A_{cc} = \frac{TP + TN}{TP + TP + FP + FN} \tag{7.1}$$

Accuracy allows evaluation of the effectiveness of the trained model in predicting traffic congestion, but it depends on the datasets and may be misleading the results. The recall metric can be then computed as the number of times the algorithm predicted congestion by the total number of points and it is expressed as

$$Rec = \frac{TP}{TP + FN} \tag{7.2}$$

Recall evaluates the sensitivity of the model. High accuracy may indicate that the algorithm always predicts no congestion and never predicts congestion, which corresponds to low recall. Precision is also needed as the algorithm can have recall equal to 1, which indicates the algorithm did not predict any false negative values. It evaluates the accuracy of the positive predictions and it is expressed as

$$Prec = \frac{TP}{TP + FP} \tag{7.3}$$

On the other hand, the error in predicting traffic congestion can be measured with a number of metrics including MSE, RMSE, and MAE. MSE is given as

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{y_i})^2 \tag{7.4}$$

where y and $\tilde{y}$ denote the true values and the predicted value of the target, respectively, n is the total number of data points. RMSE metric allows identifying if the model is making mistakes in approximating data and it is given as

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}} \tag{7.5}$$

MAE is expressed as

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \widehat{y}_i| \tag{7.6}$$

RMSE and MAE allow identifying any error variation by comparing their corresponding values. As RMSE is equal to or higher than MAE, equal values mean errors are of equal magnitude and a larger difference means larger error variations. In addition, RMSE and MAE evaluate the sensitivity to outliers in data where MAE is robust while RMSE is more sensitive to outliers. Success rate refers to the number of correct trials to the total number of trials. Computing time evaluates how fast an algorithm performs.

7.3 Results and Discussion

We implemented the short-term traffic congestion prediction model using real traffic data from the Caltrans Performance Measurement System (PeMS) [95]. This dataset is used to train and validate the model to ensure its efficiency for future predictions. A performance evaluation is conducted by comparing the LSTM model with two other machine learning models: artificial neural network (ANN) and K-nearest neighbors (KNN). The prediction models are evaluated and extensively tested based on a number of metrics, namely accuracy, loss function, confusion matrix, MSE, RMSE, MAE, precision, recall, computing time, and success rate.

As one of the most used public datasets, the PeMS dataset contains over than ten years of traffic data collected across the highways of California. The data collection was performed using 18,711 stations equipped with 45,697 sensors across 41,236.0 mi of directional distance managed by 7001 controllers. In this chapter, we selected the traffic data reported every 5 minutes on a given day on one of the highways for

one month of 2021. At each time stamp of 5 minutes, a number of parameters are collected, including timestamp, station identifier, district number, route number, the direction of travel, lane type, station length, the total number of received samples for all lanes, number of observed points, average occupancy, average speed, and average delay over the segment length. Timestamp indicates an interval of time and date (MM/DD/YYYY HH24:MI:SS), station length indicates the segment length in miles, and traffic flow indicates the number of vehicles passing a given point per unit of time.

The performance and the simulation setting of the LSTM model depend on a combination of parameters, including the number of hidden layers, number of hidden neurons, batch size, number of epochs, learning rate, activation function, and optimization algorithm. We performed several experiments to find out the optimal parameters for maximizing the model's performance using data with 5 minutes intervals. We used LSTM with two layers for model training with 64 hidden neurons per layer. We used the Adam algorithm for optimization as it is effective for noisy data. The number of communication rounds corresponds to the number of epochs. Table 7.1 summarizes the LSTM structure used for high-performance deep learning with the TensorFlow framework (TFF).

Table 7.1: LSTM structure

Number of layers	2
Step per epoch	20
Number of epoch	200
Batch size	64
Activation function	Sigmoid function
LSTM hidden function	tanh function
Optimization algorithm	Adam
Learning rate	0.001
Loss function	Binary cross-entropy loss

The algorithm for the LSTM-based traffic congestion prediction system is given in Algorithm 1.

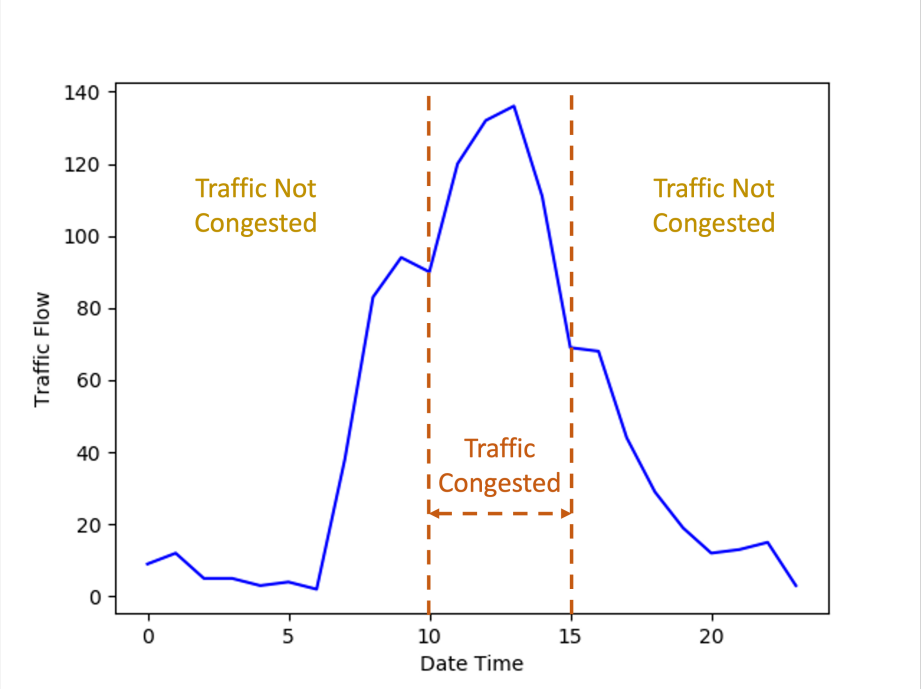


Figure 7.4: Traffic Flow for 24 Hours, sensor Id=3084071.

We opt for a graphical residual analysis to check the different models' fit and compare their performances. Examples of results are presented in Fig. 7.4 through Fig. 7.6. Fig. 7.4 represents the traffic flow captured by an observed sensor with id:308471 for 24 hours on a weekday with no unusual event happening. It can be observed that the traffic flow is low in the date time window of 00:00-06:00 and the date time window of 20:00-00:00. The traffic flow is very high from 10:00 to 15:00, which corresponds to peak hours. The behavior of the traffic flow is almost similar for all weekdays of the month over the year.

After the traffic is predicted with the LSTM model, the prediction results can then be used to evaluate how congested the traffic is over time in a given road segment by considering the congestion level. Congestion level represents the state of the traffic from normal to extremely congested. Not congested prediction corresponds to a normal level while congested prediction corresponds to three levels heavy, moderate, and low. Based on the LSTM model with 200 epochs, Fig. 7.5 compares the pre-

Algorithm 1: Traffic congestion prediction

Data: Data set

training dataset, testing dataset, learning rate, Predicted value of the next time stamp **for** *training dataset* **do**
- - Define model as sequential
- - Add LSTM layer with 64 units as hidden layer, sigmoid as activation function, input shape
- - Add Dense layer with 1 unit as output layer
- - Compile with Adam optimizer and MSE
- - Reshape input for each sample [timestamp, samples] to [timestamp, samples, features]
- **for** *each timestamp in the training dataset* **do**
  - - Fit the model with epochs 200

for *testing dataset* **do**
- - Predict with the model using new data
- - Evaluate the model

dicted values with the true values of the traffic congestion levels for 24 hours of a weekday with normal traffic conditions. As can be seen, congestion levels indicate that: 0.0 corresponds to normal traffic, 1.0 and 2.0 correspond to low congestion, 3.0 corresponds to moderate congestion, and 4.0 corresponds to heavy congestion. Low congestion occurs during the mornings while heavy congestion occurs during the pick hours. The predicted values are almost correct compared to the actual values of the congestion and slightly worse in missing some values.

Fig. 7.6 represents the loss function against the number of communication rounds for the LSTM model. It is observed that the loss function decreases with the increase of the number of epochs to converge to a minimum of 0.14 for more than 100 epochs.

In Table 7.2, we present a quantitative comparison between LSTM, ANN, and KNN-based models. The LSTM model reaches 90% of accuracy and outperforms the other models with 77% and 80%, respectively. It has a low MSE of 2.4 followed by ANN with an MSE of 5.01. The KNN model reaches the highest MSE with a value of 11.02. MAE of the LSTM model is reduced by 33% and 45%, respectively. Large MAE is due to the heavy traffic flow in the used dataset. The LSTM model performs

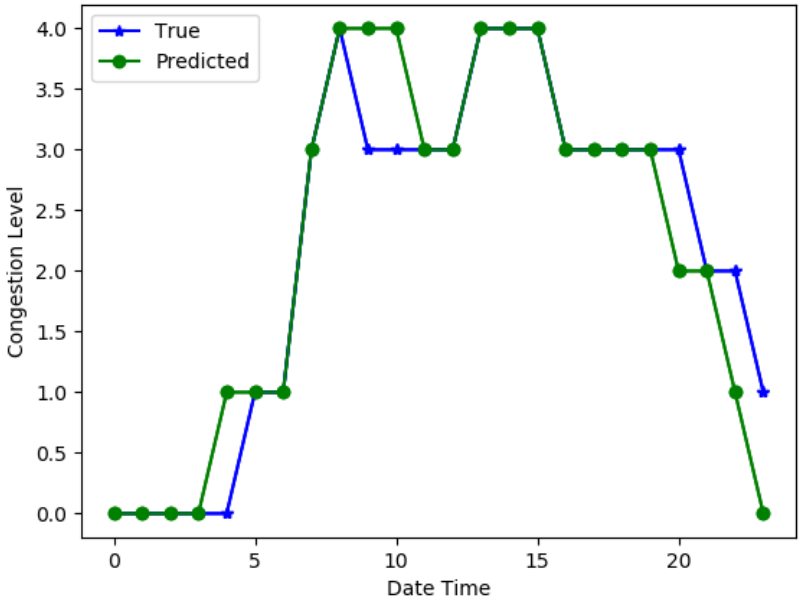


Figure 7.5: Traffic Congestion for 24 Hours.

with a lower RMSE of 28 while ANN and KNN models perform similarly with a high RMSE. For precision, the LSTM model learns with high precision and achieves 92% of precise prediction followed by 71% for the ANN model and 33% for the KNN model. The performance of the three models in terms of recall metric is almost similar to the precision metric with 97% for the LSTM model, 47% for the ANN model, and 25% for the KNN model. In terms of computation time, traffic congestion prediction takes only 0.9 minutes to perform with the LSTM model while it takes longer with ANN and KNN. The LSTM model is faster than the other models, which makes it a good candidate for time-sensitive predictions. Thus, LSTM-based traffic congestion prediction can improve the prediction performance compared to other models.

For the success rate, the LSTM model achieves 75% in successfully predicting congestion with 100 as the total trials. Therefore, the traffic congestion prediction-based LSTM model can achieve high accuracy, high precision, and high recall. It can perform faster in predicting traffic congestion with a high success rate and minimal

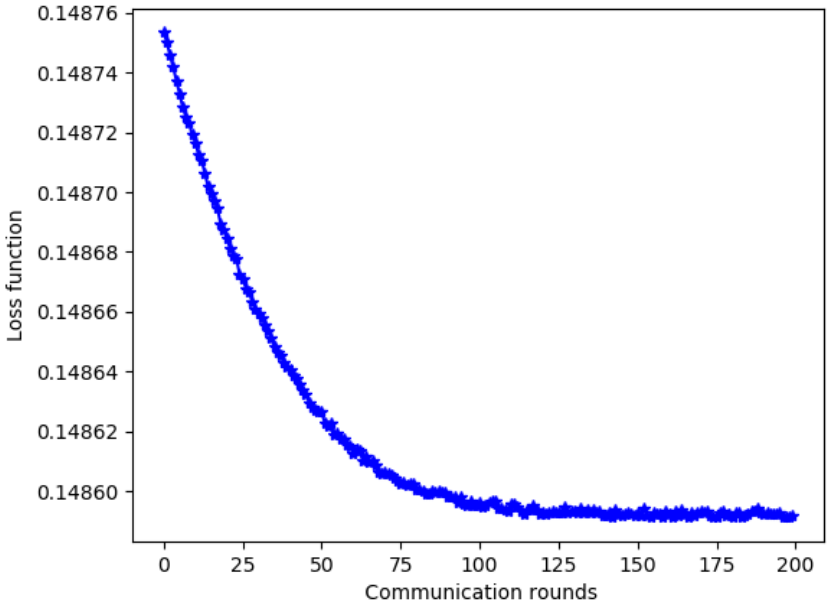


Figure 7.6: Loss function with LSTM model.

Table 7.2: Quantitative comparison of the prediction models

Model	LSTM	ANN	KNN
Accuracy	0.90	0.77	0.80
MSE	2.14	5.01	11.02
MAE	30	89	66
RMSE	28	42	47
Precision	0.92	0.71	0.33
Recall	0.97	0.47	0.25
Computing time	0.9	2.7	5.99

error rate.

7.4 Conclusions

In this chapter, we presented a traffic congestion prediction approach that can be applied to LoRa networks for ITS in smart cities. It aims at predicting traffic congestion with low-bandwidth data that are typically available from LoRa networks. The LSTM model is applied as a fast and effective solution to efficiently make proper predictions and reduce congestion. Training and performance evaluations are performed using traffic data from an online dataset. The proposed system can be used to ef-

ficiently predict congestion leading to intelligent traffic congestion management for reducing the waiting time and providing drivers with alternative and shortest routes.

However, LoRa is a low-bandwidth technology and the model used in this chapter requires the sensors to transmit raw data measurements to the central server for processing. Hence, the same model will face challenges when applied to real-time traffic data collected through pre-installed sensors over road networks. Sending an extensive amount of raw data with LoRaWAN technology is not feasible.

In the next chapter, we discuss the communication requirements and the impact of communication impairments of the LoRa network when using the proposed machine learning approach. A number of requirements need to be considered when exchanging data among LoRa network components, including volume of data, data loss, bandwidth, data rate, and cost. In the following chapter, we present a proposed approach for addressing these challenges. Our approach reduces the bandwidth requirements by using federated learning [96]. The federated learning approach would involve performing model training at the edge devices, also called clients, with limited data exchange over the LoRa links.

The proposed work involves a decentralized approach, where a central server coordinates with multiple clients by sharing a global model and each client trains the model locally using only its own data. Clients perform the local training and send back only their training results to update the global model at the central server by combining or averaging them. With only a limited amount of data exchanged, the central server updates the global model and forwards it to all the clients for local training on their own data. These steps are repeated until the global model converges to the optimal solution. Since legacy LoRaWAN may not be able to support federated learning requirements, this approach requires developing a MAC protocol to implement federated learning on LoRa-based sensor networks.

CHAPTER 8: FL-LoRaMAC: A NOVEL FRAMEWORK FOR ENABLING ON-DEVICE LEARNING FOR LoRa BASED IoT APPLICATIONS

8.1 Introduction

As the IoT revolution continues to gather momentum, we are seeing an unprecedented explosion in the amount of data being generated. With estimates predicting over 55 billion connected devices generating a staggering 80 zettabytes of data by 2025 [97], the potential for Artificial Intelligence (AI) to unlock insights and improve the management and quality of life is enormous. However, the unique challenges presented by the IoT and LoRaWAN ecosystems make it difficult to implement traditional AI tools. In this chapter, we unveil a novel framework that allows for the implementation of a cutting-edge AI technique known as "federated learning" in LoRa-based IoT devices, bringing the power of AI to the forefront of the IoT revolution.

In this chapter, we delve into the challenges of implementing traditional machine learning methods in IoT ecosystems, specifically within the context of LoRaWAN networks. Due to the vast amounts of data generated by IoT devices, traditional AI techniques often struggle to be implemented in those ecosystems. To overcome these challenges, the chapter proposes the use of Federated Learning as an alternative to traditional machine learning methods. The communication requirements and limitations of Federated Learning are also examined, leading to the development of a novel framework for its implementation in a representative healthcare application. Additionally, the chapter explores potential communication optimization techniques to enhance the efficiency of the system.

8.2 Machine Learning in IoT Applications

Machine learning is a tool that has the ability to demystify hidden patterns in the data. It can provide hidden insights from the data generated by IoT devices for improved decision-making and rapid automated responses. Machine learning tools can be of immense help to project future trends or detect anomalies in IoT applications.

In recent years, IoT and Machine learning are used in conjunction in multiple domains such as agriculture, manufacturing, healthcare, smart cities, etc. [98][99]. In healthcare scenarios, the increasingly high costs have been encouraging the masses to use remote health monitoring systems. Machine learning is applied in a wide range of use cases such as predicting and treating diseases [100], organizing medical records [101], providing medical diagnostics, etc. [102]. A large number of organizations such as Microsoft [103], Tempus [104], Beta Bionics [105], Insitro [106], etc. are using machine learning tools in the field of healthcare. IoT devices generate data from the physical world using sensors and transmit this data to the Internet. Machine learning algorithms use this data to create insights. With the advancements in sensor technology, the sensors used for getting data such as the electrocardiogram (ECG), blood sugar, blood pressure, heart rate, SPO2, etc. have become cheaper and smaller to the extent that they can be embedded into wearable devices such as watches, pendants, vests, etc.

In this chapter, we use ECG signal monitoring as an example application of applying AI for detecting anomalous conditions using sensor information. ECG is a commonly used clinical tool that involves attaching up to twelve sensors or electrodes to a patient's chest and limbs to measure the electrical activity of the heart. These sensors are equipped with wires that transmit electrical signals to a computer or a monitor which displays the heart's electrical activity [107] in the form of amplitude vs time data points as shown in figure 8.1 [2]. In recent times, smartwatches have been developed that can accurately measure ECG data without requiring the wearer

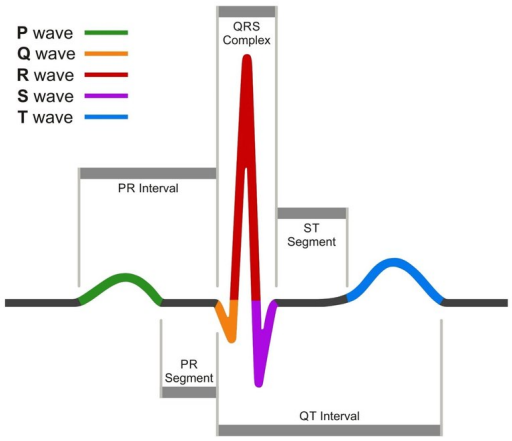


Figure 8.1: ECG signal generated in one heartbeat (source: [2]).

to visit a clinic or hospital, enabling remote monitoring of cardiac health.

Smartwatches utilize photoplethysmography (PPG) technology for heart rate monitoring by using light to detect blood flow. PPG technology uses sensors that emit light onto the skin and measure the amount of light that is reflected back [108]. This data is then used to determine the heart rate and rhythm. By continuously monitoring the heart rate and rhythm, smartwatches can predict or detect various heart conditions such as irregular heart rhythms, heart attacks, and the functioning of pacemakers, among others.

The easy availability of ECG data from FDA-approved sensors integrated into smartwatches by manufacturers such as Apple, Google Pixel, Fitbit, and Samsung has paved the way for machine learning applications. In this chapter, we will leverage this ECG data for such applications. By training machine learning algorithms with ECG datasets, they can detect or predict heart conditions. However, the model must be able to differentiate between healthy (normal) and unhealthy (abnormal) ECG signals to detect health conditions effectively. This binary classification of ECG data is known as ECG anomaly detection and serves as a representative application for this chapter.

There are now a variety of publicly available ECG datasets that are labeled and

accessible for use. Rather than needing to collect a large amount of data ourselves, we can load these datasets onto devices and use them to simulate real-world scenarios. This saves the time and cost associated with data collection, while still allowing us to train and test machine learning models on representative data. Using these datasets can also help ensure that the models we develop are more robust and accurate since they are based on a diverse range of real-world data. Additionally, the availability of labeled ECG datasets can facilitate more widespread research and development in this area, as it makes it easier for others to replicate and build upon our work.

Overall, leveraging publicly available ECG datasets can help accelerate progress in ECG anomaly detection and other machine learning applications in healthcare, ultimately leading to better patient outcomes and more efficient medical care.

Challenges faced by machine learning in IoT ecosystem: In a typical machine learning scenario, the data from all the sources is sent to the central server where the model resides. The model is trained on the collected data from various sources. Once the model is trained and is ready to be tested, the test data is sent to the central server where decisions or predictions are made. It can be inferred that in both cases whether the testing phase or the training phase, the data must be collected at the central server.

The IoT ecosystem poses some novel challenges for the implementation of typical machine learning techniques with respect to sending data to the central server. Some of the major challenges are:

- **Intermitted Internet connection:** The Internet connectivity to the central server is not always available to all IoT devices, hence restricting communication.
- **High data volume:** A sensor node may capture data once every few minutes but the gateway is connected to multiple such sensor nodes. All the data is

collected by the gateway and sent to the central server. This can be difficult and costly to achieve in a real-time environment.

- **Data privacy:** As discussed earlier, in the case where the data is residing on the cloud and the training is happening in a batch process or the model is already deployed, the data is sent in real-time to the cloud, and the trained model is trying to make the predictions. In both cases, we are providing data that can be private to an individual. So, we are providing individuals' private data to a third-party service provider that hosts the model and the service. Regulatory organizations like California Consumer Privacy Act (CCPA) in the US, impose a lot of restrictions in terms of the privacy of sensitive customer data. We have to comply with a lot of regulations to share the data with a third party even to provide benefits to the customers themselves.

In IoT applications that involve private user data, the need to send data to a central server for machine learning purposes can be a challenge. This challenge can be addressed by using federated learning, which is an artificial intelligence technique that allows for machine learning to be performed without sending data to a central server. In the following section, we will discuss the principles of federated learning and how it can be used as an alternative to traditional machine learning in IoT applications.

8.3 Federated Learning: A viable machine learning alternative

Federated Learning (FL) is a distributed machine learning approach that allows training models across decentralized devices without transferring data to a central server. In FL, each device locally trains a model on its own data, and only the updated model parameters are transmitted to the central server for aggregation. This process is repeated iteratively until the model converges to an acceptable accuracy. The overall architecture of FL is shown in figure 8.2. FL is especially useful in IoT

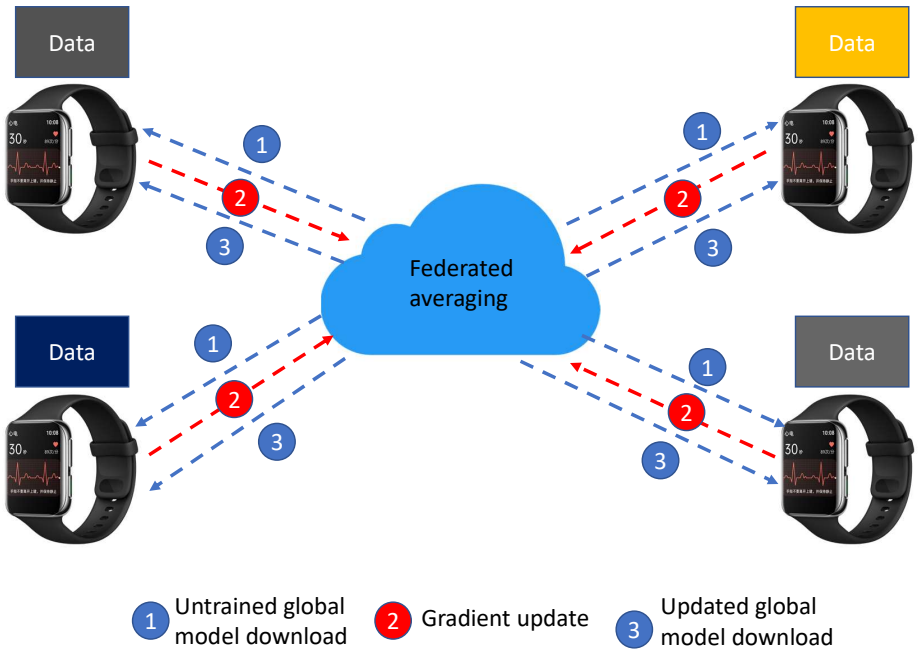


Figure 8.2: Working model of federated learning.

applications that involve private user data since it does not require the data to be sent to a central server, addressing privacy and security concerns.

Typically, federated learning is employed to address privacy concerns in machine learning. In this context, we aim to explore the possibility of leveraging federated learning to decrease the amount of data that needs to be transmitted while maintaining comparable performance. The training in federated learning occurs in a decentralized manner, with local model parameters being transmitted from devices to a central server for aggregation and then sent back to the devices. To support this process, the network infrastructure should enable independent two-way communication between the sensor nodes and the central server, provide uplink capabilities for local updates, and downlink capabilities for the updated global model.

It is important to note that while machine learning relies on transmitting and processing all sensor data centrally to obtain the trained model, federated learning aggregates locally trained models. In this chapter, we compare and analyze the performance of both technologies. Performance was evaluated using three parameters

that were experimentally computed by implementing both technologies on an ECG anomaly detection application.

1. Model Performance: Comparison of model performance metrics.
2. Volume of data traffic: The number of bytes that needs to be communicated between the end device and the server for training.
3. Computational time: The time it takes to train a model, excluding the communication time.

To evaluate their performance, both federated learning and machine learning were implemented on an ECG anomaly detection application using the ECG-5000 dataset. A dense neural network-based autoencoder was trained on the dataset and was used to make the predictions. The details for both the dataset as well as the autoencoder are discussed in section 8.6.1.

It must be noted that we use ECG as an application for evaluation purposes only. The approach of federated learning to preserve privacy while conserving communication bandwidth may be applied to a number of different IoT applications that involve large data sets and artificial intelligence.

In the case of machine learning this model stays on the central server. All sensors or end devices transmit their data to the central server. Once a sufficient amount of data is collected at the server, the model is trained on the collected data. However, prior to training some pre-processing is performed on the data. Typically, the data is provided to the model in batches. The number of data samples in each batch depends upon a parameter known as "batch size." The entire dataset is fed to the model in batches for one training iteration. This single training iteration is termed an "epoch." Hence, the model is trained on the dataset for an appreciable number of epochs. In each epoch, the data samples are randomized and fed to the model in batches. Training sets the model's weights based on the dataset that was used for training.

Therefore, the dataset must contain an appreciable amount of all permutations and combinations of possible outcomes. Once the training is complete the model is then fed a different set of data samples and its performance is evaluated based on the predictions it makes. All the computations and predictions are made by the central server.

Contrary to the centralized machine learning approach, federated learning assumes a distributed approach. In the case of federated learning, instead of the end devices transmitting all their data to the central server, the central server sends the global model (initially untrained) to the participating end devices. Once the untrained model is received by the end device, it trains the received model based on its locally collected data. This training is done for a certain number of epochs similar to the training in the case of machine learning discussed above, but its performed on the edge and based only on the data collected by the edge itself. Once the local training is complete, the model weights or parameters are sent to the central server. These local model weights are numerical vectors that do not provide any information about the type or kind of data that was used for training. At no point, does the local data leave the end device. Once, the central server receives the model parameters from all participating end devices, it performs a federated averaging on all the received model parameters and outputs an improved global model. This entire cycle of sending the global model, local training, sending local updates, and getting an updated global model by federated averaging is termed as "communication round." Federated learning is performed for a certain number of communication rounds till the desired performance is achieved. Once the training is complete the global model is sent to all end devices. It is then fed with local test samples and its performance is evaluated based on the predictions it makes. In federated learning under no circumstance does the local data leave the edge device, hence the data privacy is preserved at all times.

Challenges faced by Federated Learning in LoRaWAN ecosystem: The

federated learning mechanism involves four major steps.

1. Federation Construction: Under federated construction, the global model is downloaded from the server to the end nodes. For this step to be successful, it is necessary that the end nodes are listening when the server broadcasts the global model.

2. Decentralized Training: This step does not involve any communication because under this step the model is trained locally on the end device using the data collected locally.

3. Model Accumulation: This step involves the transmission of gradient updates from the end devices to the central server.

4. Model Aggregation: Once the server receives all the gradient updates from the end devices, this step computes the average of the model parameters and outputs an improved global model.

To summarize, the major communication design considerations for federated learning are:

- Reliable transmission of the global model from server to end devices
- Reliable transmission of gradient updates from end devices to the server.

The three classes of operation defined under LoRaWAN pose some challenges to downlink communication design considerations. Class-A mode of operation prioritizes uplink communications and only opens two short receive windows for downlink after the uplink transmission. If the device does not receive any data during these receive windows, the device goes to sleep. Class-B mode of operation opens additional time synchronized receive windows but requires additional resources like GPS, and real-time clocks(RTC) for the synchronization. This adds to the complexity of the system.

Class-C is the most power-hungry mode of operation and hence is not advised for battery-powered devices.

LoRa is a physical layer technology and allows any upper protocols to be implemented. The following section presents the proposed framework. The framework includes a MAC protocol that enables bidirectional communication between sensor nodes and the server, without requiring additional resources or complexity.

8.4 Proposed Framework: FL-LoRaMAC

In this section, we will describe the proposed framework FL-LoRaMAC that satisfies the communication design considerations for the implementation of various steps of federated learning. This includes global model downloads, local gradient updates, and downloading the updated global model. Additionally, we will describe how these gradients are processed on the end devices as well as on the centralized server and some mechanisms to optimize communication bandwidth.

8.4.1 Network Joining

Under FL-LoRaMAC, all devices taking part in the training need to join the LoRa network first. The joining procedure is similar to that of the legacy LoRaWAN protocol as mentioned in [26] with some additional steps and information in join messages.

Whenever a device is powered ON, it sends a join request 'J_Req' to the network server using via LoRa transceiver. Once the J_Req is sent the device waits for the join response 'J_Res'. If the end device receives the J_Res, it configures itself according to the J_Res.

On the network server side, a join timer is started periodically. If the J_Req from the end device is received before this timer is expired, the network server prepares a J_Res. The network server has information regarding the AI model architecture, let's term this information as 'MODEL_INFO'. It also contains information regarding the

downlink channels and spreading factor on which the global model parameters will be broadcasted (DL_INFO), the information about the data fragments that will be sent (FRAG_INFO), the information about how frequently the device should perform the listening and for how much duration (LISEN_INFO). The network server will also send MODEL_INFO, DL_INFO, FRAG_INFO, and LISTEN_INFO along with the legacy LoRaWAN J_Res. Once the timer is expired, no J_Res for the end devices will be sent. This process is also illustrated in figure 8.5.

8.4.2 Proposed MAC layer

In the federation construction phase, a patented model (untrained) is present at the central server. This model needs to be transmitted to the end devices for training. In order for end devices to receive this model, they must have their transceivers in listening mode while the server is transmitting. However, the end devices are not synchronized to the network clock.

Hence to achieve this, we propose an elongated preamble approach. Under this approach, the end devices join the network according to the network joining procedure discussed in section 8.4.1. Once the joining procedure is complete, the end device will configure itself according to the information received in the J_Res. It will also generate the local model according to the MODEL_INFO. Once the end device is configured, it will periodically open receive windows for receiving the global model parameters according to the DL_INFO, LISTEN_INFO, and FRAG_INFO as shown in figure 8.3. If the end device does not hear any LoRa preamble, it will close the receive window and the transceiver will go to sleep till the next period to conserve power. If the end device finds a preamble, it will continue to listen to the packet.

The gateway will periodically transmit the global model according to the channel and spreading factor in DL_INFO. The transmission will consist of an elongated preamble according to the FRAG_INFO. The length of the preamble will be slightly longer than the periodicity of the end device at which it is opening the receive windows

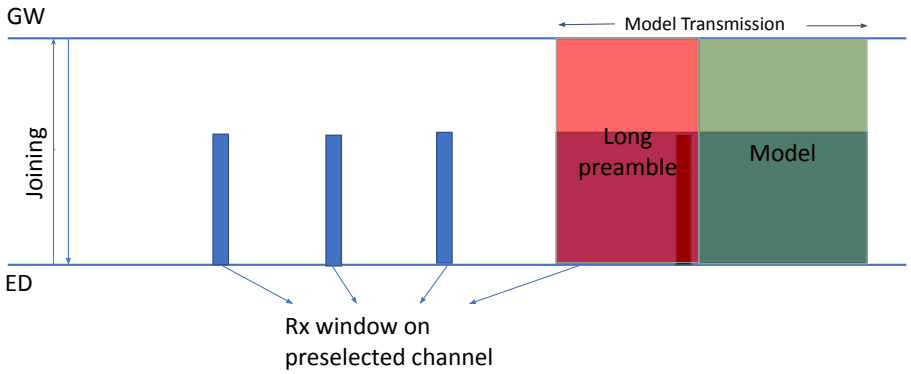


Figure 8.3: Approach for global model transmission from gateway

as shown in figure 8.3. This elongated preamble enables the end devices to receive the transmission without being synchronized with the gateway. Once all the fragments of the global model are received by the end device, the fragments will be serialized according to the frame numbers. If the end device encounters missing frames while serializing, the retransmission of those frames will be requested otherwise, it will send an acknowledgment to the gateway indicating the device received all the global model parameters. After this, the end device will stop opening the receive windows.

Until the model is not received, the end device will keep collecting the data locally and once the model is received, it will be trained on the data collected by the end device.

The end device trains the model that it received from the network server. Once the local training cycle is complete for a certain number of epochs, the local weights must be sent to the server for averaging. To do so, the model parameters need to be packed into various data fragments. Once the fragments are created, the end device transmits these fragments one by one using the uplink transmissions. After fragment transmission is complete, the end device will follow the elongated preamble approach to receive the updated global model.

8.4.3 Decentralized Training & Model Aggregation

Section 8.4.2 provides the details about the communication of the model parameters. This section explains how those model parameters are generated and processed when they arrive at the end device or at the network server. The entire process of FL-LoRaMAC taking place at the end device and the network server is illustrated using flowcharts in figure 8.4 and figure 8.5.

First, we will look at the decentralized training that happens at the end device as shown in figure 8.4. The end devices participating in the federated learning process will receive the MODEL_INFO. This contains detailed information about the model. For instance, in the case of a neural network, it contains information about the number of neural layers and the number of neurons in each layer that are present in the model. The end device compiles a local model based on this information. As soon as the end device receives all the global model parameters, they are saved into its local model parameters. The global model is not received as a single frame but as a number of data fragments. These fragments are saved in a buffer, serialized, and checked for any missed packets. If there are missing fragments then re-transmission of missing fragments is requested.

Otherwise, if the device has enough amount of data to train the model, the model training starts on this local data. Once the training for a certain number of epochs has elapsed, the updated local model parameters are sent for averaging and local memory used for saving local model parameters is cleared. As discussed earlier, these model parameters are sent in fragments. The data frames for the uplink transmission are constructed by flattening the parameter matrix and packing the pre-determined number of parameters in the frame. Also, each frame includes a frame number for identification and serializing purposes.

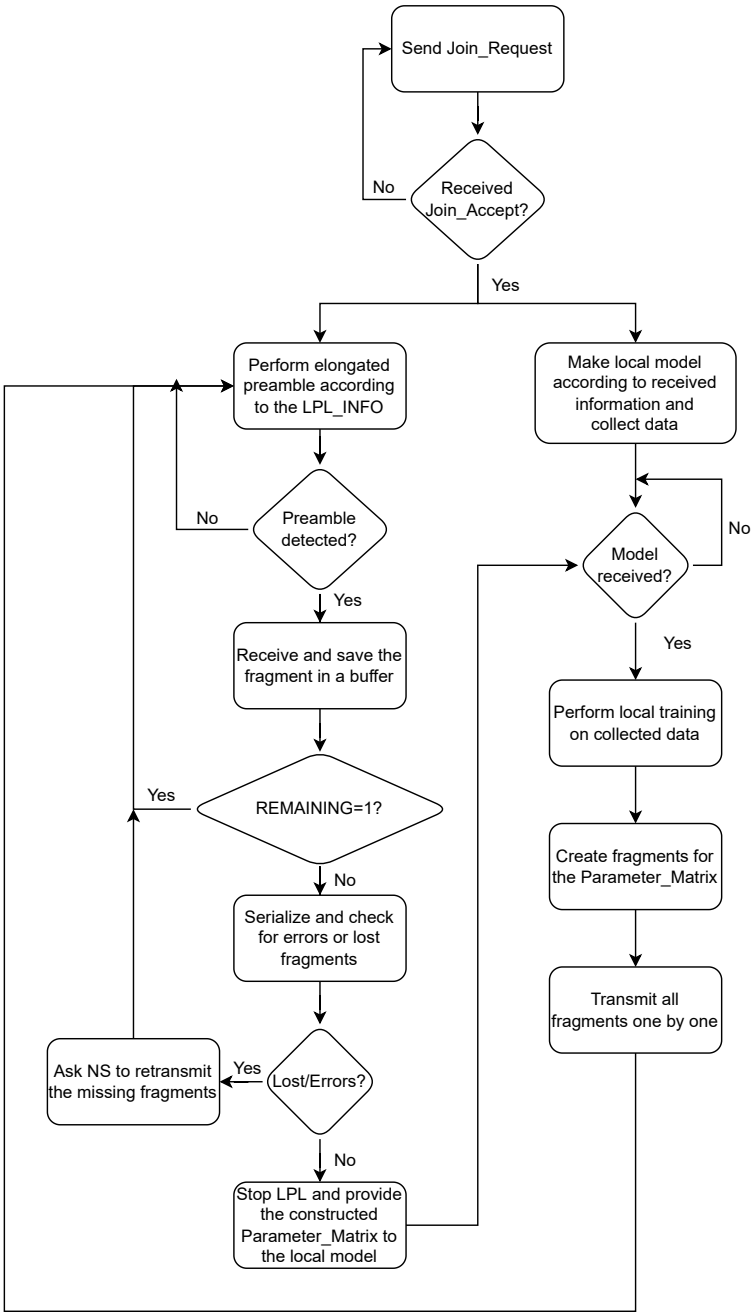


Figure 8.4: Flow of FL-LoRaMAC on end device side

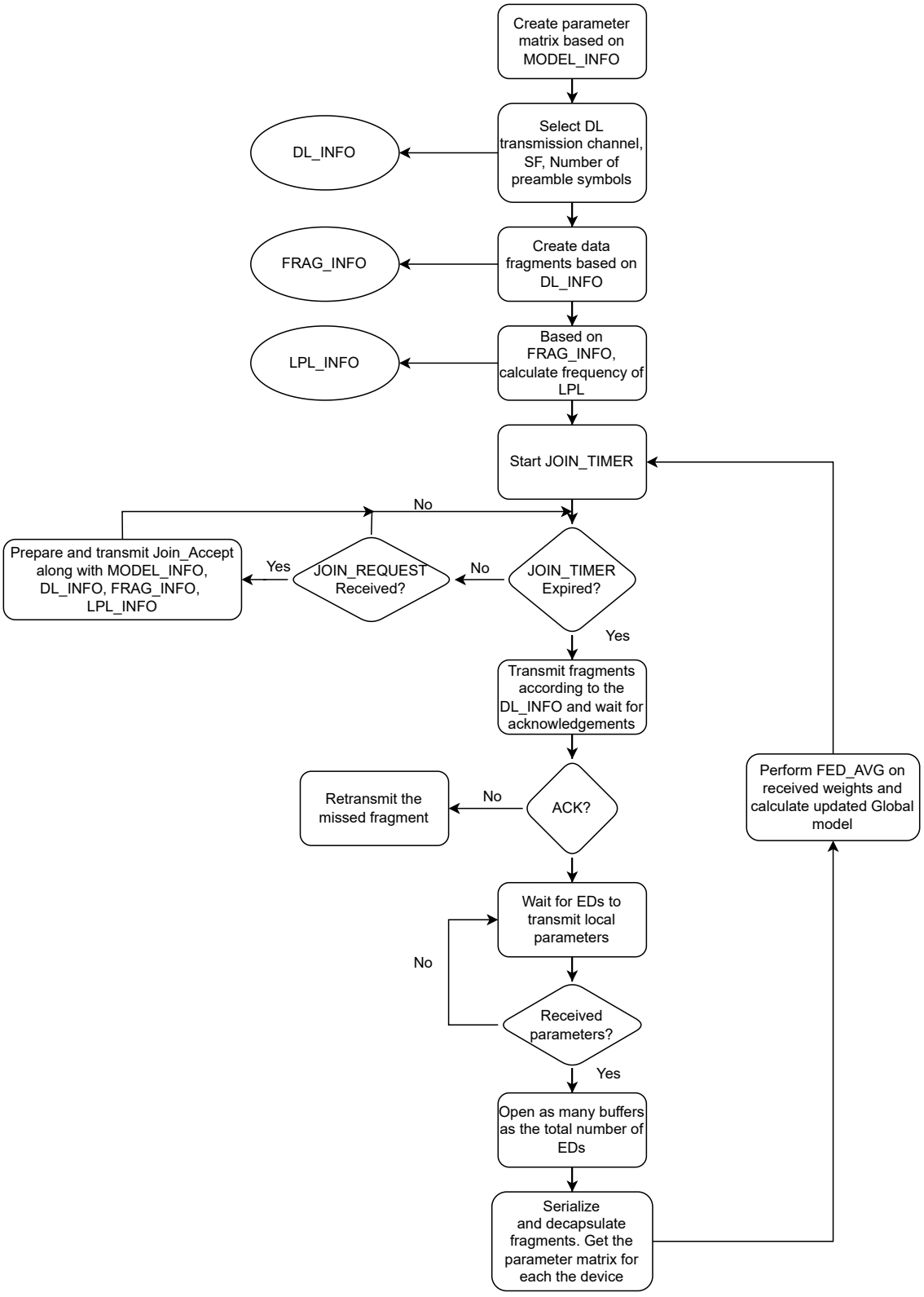


Figure 8.5: Flow of FL-LoRaMAC on network server side

As shown in figure 8.5, when the network server starts receiving uplinks from any of the end devices, it reserves a buffer for the end device's local model parameters. Once all the fragments are received from the end device, the fragments are serialized based on the frame numbers. It is possible that some of the fragments from the end devices may be lost in the RF environment due to collisions with other devices' communication. Using the frame numbers the network server can detect the loss of fragments. During serialization, all the information except the model parameters is removed and if any of the frames are missing, the parameters that the lost frame would have contained are set to zero. After this serialization, we get the same flattened matrix (with losses) from which the end devices created the uplink frames.

Once the local updates from all the participating end devices are received, the server performs federated averaging on them. The output of the averaging provides an updated and intuitively better global model. Following the same procedure of flattening and creating fragments for transmission, this global model is again sent to the end devices for subsequent training rounds.

8.5 Optimizing Communication Bandwidth

The artificial intelligence models are quite large, in terms of the number of parameters in the weight matrix. Since LoRa is a low bandwidth and low data rate communication technology, transmission of the model parameters to the server will require a large amount of time and may consume all the network bandwidth. Hence the total transmission time and communication bandwidth used need to be optimized without significantly affecting the performance of the model.

8.5.1 Model Pruning

In a neural network, not all parameters are significant. In other words, not all the connections between neurons of fully connected layers contribute equally to the model's prediction. In the ECG anomaly detection scenario explored in section 8.6,

the number of model parameters is reduced to 635 using the PCA algorithm. However, not all of these parameters are significant. The pruning technique can be used to reduce the complexity and time of execution by removing the insignificant connections between neurons. This is done by modifying the parameter values related to those connections to zero. The output of pruning is a sparsified weight matrix.

Typically pruning is accompanied by quantization or compression or both in order to get a smaller parameter matrix in terms of bytes but this also increases the computation cost. In this particular scenario, training is being performed on resource-constrained devices. Applying these methods to those devices will result in significant overhead. Instead, we may choose not to communicate these pruned parameters or send them with lower priority.

8.5.2 QoS-based assignment of SF for differential priorities

As mentioned in pruning, not all parameters are significant to the model training. Significance can be established based on the magnitude of the model parameters. The pruning algorithm sparsifies the parameter matrix by pruning the parameters closer to zero. Hence, it can be assumed that a higher magnitude means more significance.

If all the parameters are sent using the same priority level, in other words, there is an equal chance of data being lost for all. If the high-significance parameters are lost, then either retransmissions or more communication rounds will be required by the system to reach convergence. In order to reduce the number of collisions, the collision domain for the packets carrying parameters of high significance can be changed. This can be achieved by employing the QoS-based SF distribution [20] described in chapter 6.

For ease of understanding, two levels of priorities for the packets carrying model parameters are considered.

- **High priority:** High priority is given to the packets carrying any number of significant parameters in the weight matrix. These parameters must be delivered

to the receiver with the highest probability of success (say >90%).

- **Low priority:** Low priority is given to the packets carrying the non-significant and less significant parameters in the weight matrix. These parameters can be delivered with the highest possible probability of success that the network can provide.

There can be other ambient devices in the network running some other applications, transmitting packets with higher priorities selected based on some criteria. Using the network configuration, the maximum number of devices that can be supported by each spreading factor can be found and the spreading factors can be allocated to different packets according to the top-down approach mentioned in section 6.4.1.

8.6 Results and Discussions

To evaluate the performance of the proposed framework and the efficiency of federated learning to make accurate predictions, the approaches were applied to an ECG anomaly detection application. A LoRa network simulator was used to simulate the networking layers. Federated learning was implemented in Python which ran on top of the networking layer.

All the codes for evaluation ran on a machine configured with Intel(R) Core(TM) i5-8250U CPU clocked at 1.60GHz, no GPU, and a single 8 GB DDR4 RAM. The ECG-5000 dataset was used for training and testing purposes. The details of the dataset are discussed in section 8.3.

The training hyper-parameters used are tabulated in table 8.2 unless otherwise specified. These hyper-parameters were found via experimentation discussed in section 8.6.3 and provide the best performance for the representative application. The model details are provided in table 8.1.

Table 8.1: Autoencoder model details

Parameter	**Value**
Number of layers	3
Layers Type	Dense
Neurons in Input layer	20
Neurons in Hidden layer & activation	15; ReLU
Neurons in Output layer & activation	20; Sigmoid
Optimizer	Adam
Loss Function	MAE

Table 8.2: Federated learning: Training hyperparameters

Parameter	**Value**
Number of PCA components	20
Batch Size	10
Communication rounds	3
Epochs per Communication Rounds	100

8.6.1 Network Setup and System Model

A total of 'X' number of nodes were considered to be placed uniformly in a circular LoRa cell of radius 'R'. The gateway was assumed to be placed at the center of the circular cell. It was assumed that at any moment the gateway has enough resources available to demodulate any number of valid receptions. The network server was connected to the application server. The application server was assumed to contain the patent untrained global model and it is also responsible for performing the federated averaging on parameters received from the nodes.

Out of the total of X devices, 5 devices were considered to be running the ECG anomaly detection application. All 5 of those devices will be taking part in the federated learning process.

As discussed earlier, the ECG-5000 dataset is used for training and testing an autoencoder. The original ECG-5000 dataset is a 20-hour-long ECG downloaded from Physionet. The dataset was then pre-processed to extract each heartbeat and interpolation was used to make each heartbeat equal in length [109]. The dataset contains

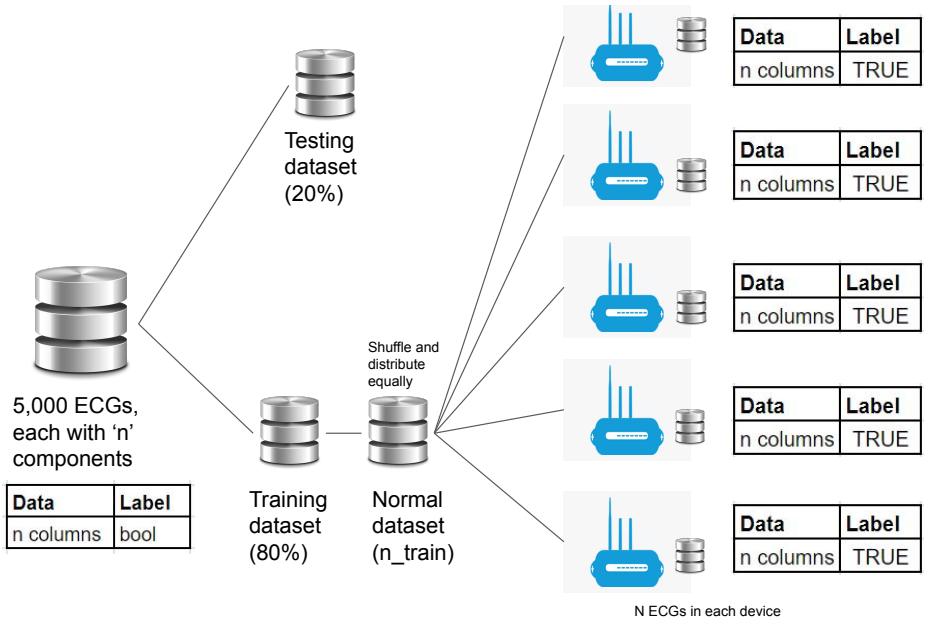


Figure 8.6: Illustration of dataset division

5,000 labeled samples of ECG data. Each sample consists of 140 data points and a label. The value of the label is either 0 or 1 representing abnormal or normal heartbeats respectively. The training dataset consisted of 80% of the whole dataset and the remaining 20% was used for testing purposes. To simulate the real-world scenario, the data samples from the training subset were equally and randomly distributed among the end nodes participating in learning as shown in figure 8.6. In real world scenario, this data will be collected by the devices themselves.

The autoencoder originally consisted of 3 fully connected layers with 140 neurons in the input layer, 32 neurons in the hidden layer, and 140 neurons in the output layer. In this experiment, the Mean Absolute Error (MAE) function was used to compute the training losses and ADAM was used as the optimizer to update network weights during training.

8.6.2 Federated Learning vs Machine Learning

This section presents the performance analysis of federated learning compared to machine learning as discussed in section 8.3.

8.6.2.1 Volume of Traffic

Here we evaluate and compare the volume of traffic for machine learning, federated learning, and federated learning with PCA.

Machine Learning: In this scenario, all the data points need to be sent to the central server. Each data sample consists of 140 data points (8 bytes each) and a label (4 bytes) so each data sample is $(140*8+4 =)1124$ bytes long. There is a total of 5,000 such data samples. Hence, in the case of machine learning, $(5000 * 1124 =)4,496,000$ bytes or 4.5MB. of data needs to be communicated.

Federated Learning: In federated learning, the data stays on the local machine, only the model parameters need to be communicated. The model used for this experiment had a total of 9,132 parameters. The dataset was divided equally among 5 devices. Each of these parameters consists of 4 bytes. Hence, each device will send 36,528 bytes in one communication round. That adds up to 182,640 bytes for 5 devices in one communication round. The model was able to converge to a similar performance as of machine learning in 3 communication rounds with 100 epochs in each communication round. Hence, the total data sent by 5 devices in 3 communication rounds equals 547,920 bytes. The server will also send the updated global model after performing federated averaging in each communication round. The total data sent by the server to end devices equals 109,584 bytes. Hence the total volume of data that needs to be communicated in the case of federated learning is 657,504 bytes or 657.5 KB.

Federated learning with PCA: PCA was employed as the dimensionality reduction algorithm similar to [110] along with federated learning. Using PCA, each data sample was reduced to 20 components. In other words, each data sample that earlier consisted of 140 data points was reduced to 20 data points. By doing so, the number of neurons in the input and output layer also got reduced from 140 to only 20. 32 neurons were kept in the hidden layer (although, they can also be reduced).

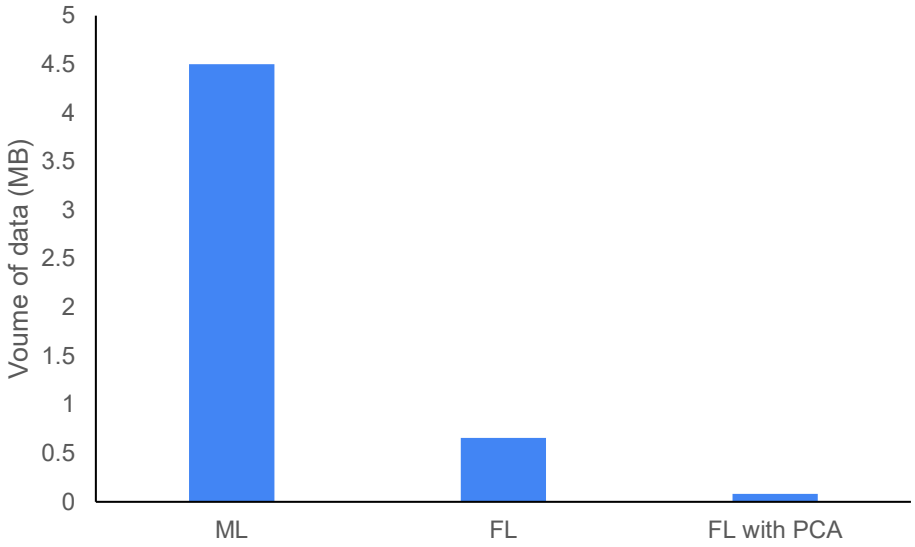


Figure 8.7: Traffic volume for machine learning (100 Epochs), federated learning (3 comm. rounds and 100 epochs), and federated learning with PCA (20 components, 3 comm. rounds and 100 epochs) approach.

The resultant model comprised of 1,332 model parameters. Following the similar calculations done earlier for federated learning without PCA, the total volume of data traffic communicated among 5 devices and the server, in 3 communication rounds in the case of federated learning along with PCA equals 83,916 bytes or 83.9 KB.

The volume of data for all the scenarios is plotted in figure 8.7. It can be observed that the volume of data traffic in the case of federated learning is significantly reduced when compared with the typical machine learning approach. The traffic volume was further reduced using the PCA dimensionality reduction algorithm.

8.6.2.2 Computation Time

The computation time measured in this scenario only consists of model training time only. It doesn't include data pre-processing time or communication time. All experiments were performed on a CPU-based machine with 8 GB of RAM.

Machine Learning: In the case of machine learning, the training took 100 epochs to converge. The entire training phase took 62 seconds to be completed.

Federated Learning: For federated learning, all devices performed the train-

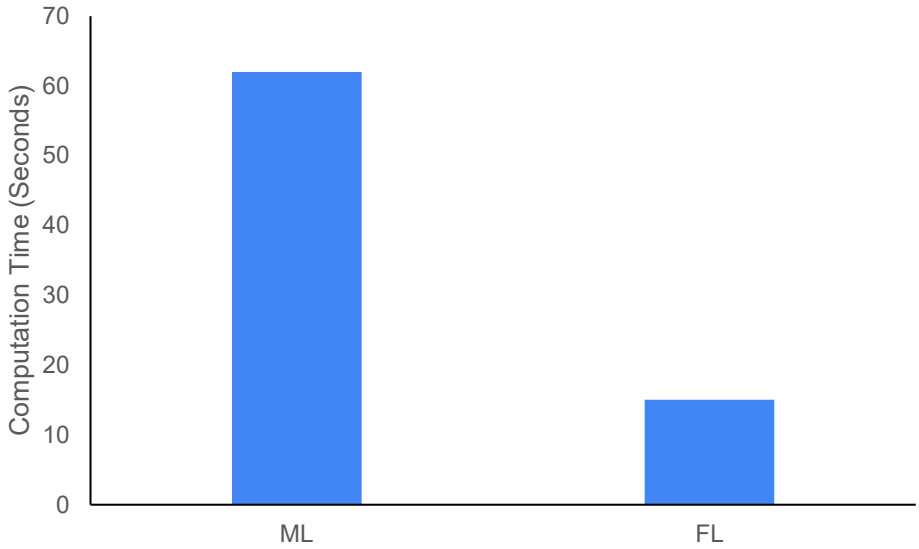


Figure 8.8: Computation time for machine learning (100 Epochs) and federated learning (3 comm. rounds and 100 epochs) approach.

ing of model parameters in parallel. During the training phase, one communication round on average took 5 seconds. The federated averaging algorithm running on the server took 0.016 seconds. Hence, one communication round took a total of 5.016 seconds for computation. Hence, three communication rounds took 15.048 seconds in computation. The results are plotted in figure 8.8.

8.6.2.3 Trained Model Performance

The autoencoder was trained individually using typical machine learning as well as using federated learning principles. Also, since each data sample consists of a large number of data points. A dimension reduction algorithm can be very employed that can reduce the number of data points in each data sample and helps to optimize system performance. Therefore we also implemented Principal Component Analysis (PCA) along with federated learning in a separate experiment. In the case of machine learning, the model was trained for 100 epochs. In federated learning, the model was trained for 3 communication rounds and each communication round consisted of 100 epochs. The performance of models trained with different methods is tabulated in table 8.3.

Table 8.3: Performance of models trained with different methods

	TN	FP	FN	TP
Machine learning	404	7	10	579
Federated learning	407	4	16	573
Federated learning with PCA	407	4	20	569

The table provides true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). The table below describes their meaning in an ECG anomaly detection scenario:

TP: The number of samples that are actually normal and are predicted as normal

TN: The number of samples that are actually abnormal, and are predicted as abnormal

FP: The number of samples that are actually abnormal but are wrongly predicted as normal

FN: The number of samples that are actually normal but are wrongly predicted as abnormal

It is intuitive that for ECG anomaly detection, TN, and TP should be maximum and FP should be minimum. FN is not of much importance in this scenario as a false alarm is better than a missed anomaly but it should also be minimum.

From table 8.3, it can be observed that for the representative application, the model trained with federated learning outperforms the model trained by machine learning. When the PCA algorithm is used along with federated learning the performance of the model is very similar to that of the model trained without the PCA algorithm with a slight increase in false alarms.

It can be observed from the performance analysis that federated learning takes less computational time compared to machine learning as it trains models in a decentralized way. Federated learning also provides the benefit of requiring substantially less amount of data to be communicated while practically providing the same quality of

performance.

8.6.3 Selection of Hyper-Parameters for FL-LoRaMAC

This section presents the experiments performed in order to choose the hyperparameters used for training the model.

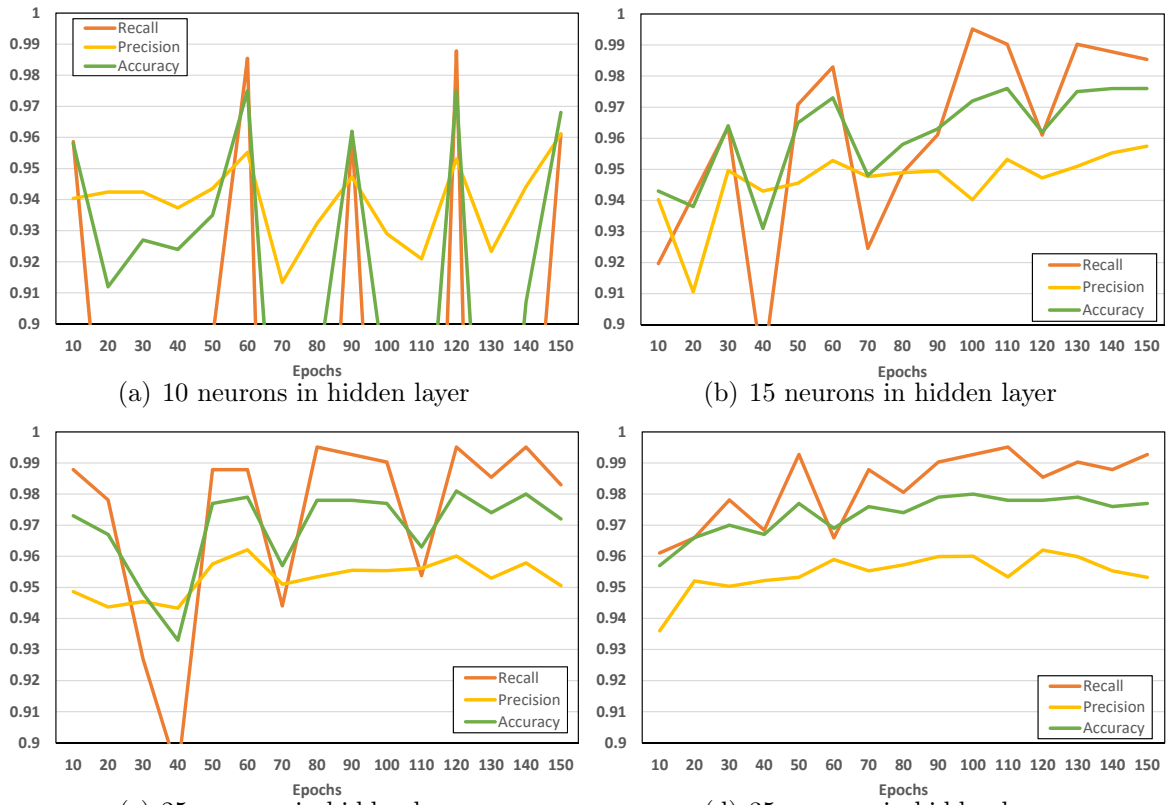
As discussed earlier, in order to construct a model we need to know the number of layers in the model and the number of neurons in each layer. Also for training purposes, we need to know the number of epochs and communication rounds.

All the hyper-parameters were selected after extensive experimentation. In section 8.6.2, we observed that the dimension reduction algorithm significantly reduces the volume of data that needs to be communicated while maintaining a similar performance level. Hence, PCA is incorporated in FL-LoRaMAC.

Figure 8.9, illustrates the performance of the model for the different numbers of hidden layer neurons in the model and with varying numbers of epochs on which the model is trained for. This experiment was conducted with 20 PCA components and 3 communication rounds. It can be observed from the plots that with 15 neurons in the hidden layer and 100 epochs, the maximum recall of 99.5% can be achieved with 94% precision and 97.2% accuracy. Similar performance can be observed with the higher number of neurons in the hidden layer but as we increase the number of hidden layer neurons the number of parameters in the weight matrix also increases. This increases the communication cost of the system. However, we can reduce the communication cost by reducing the number of neurons in the hidden layer to 10. This will reduce the number of parameters in the weight matrix reducing the communication cost but sacrificing the performance to have a recall of 98.5% with 95.5% precision and 97.5% accuracy. Since using 15 neurons in the hidden layer provides us with the optimum performance in this scenario, hence we will choose 15 hidden neurons in the neural network.

Figure 8.10, illustrates the performance of the model for different numbers of epochs

for which the model is trained with a varying number of PCA components. This experiment was conducted considering 15 neurons in the hidden layer and 3 communication rounds. It can be observed from the plots that with 20 PCA components and 100 epochs, the maximum recall of 99.5% can be achieved with 94% precision and 97.2% accuracy. It can be further observed that in almost all the experiments performed with different numbers of epochs, the overall performance is at its peak with 20 PCA components. Hence, 20 PCA components were selected to be used for the system in this scenario.



(a) 10 neurons in hidden layer

(b) 15 neurons in hidden layer

(c) 25 neurons in hidden layer

(d) 35 neurons in hidden layer

Figure 8.9: Performance of model trained with varying number of epochs and a) 10 neurons b) 15 neurons c) 25 neurons d) 35 neurons in hidden layer with 3 communication rounds and 20 PCA components

Similarly, other hyper-parameters were selected via experimentation and are tabulated in table 8.1 and 8.2.

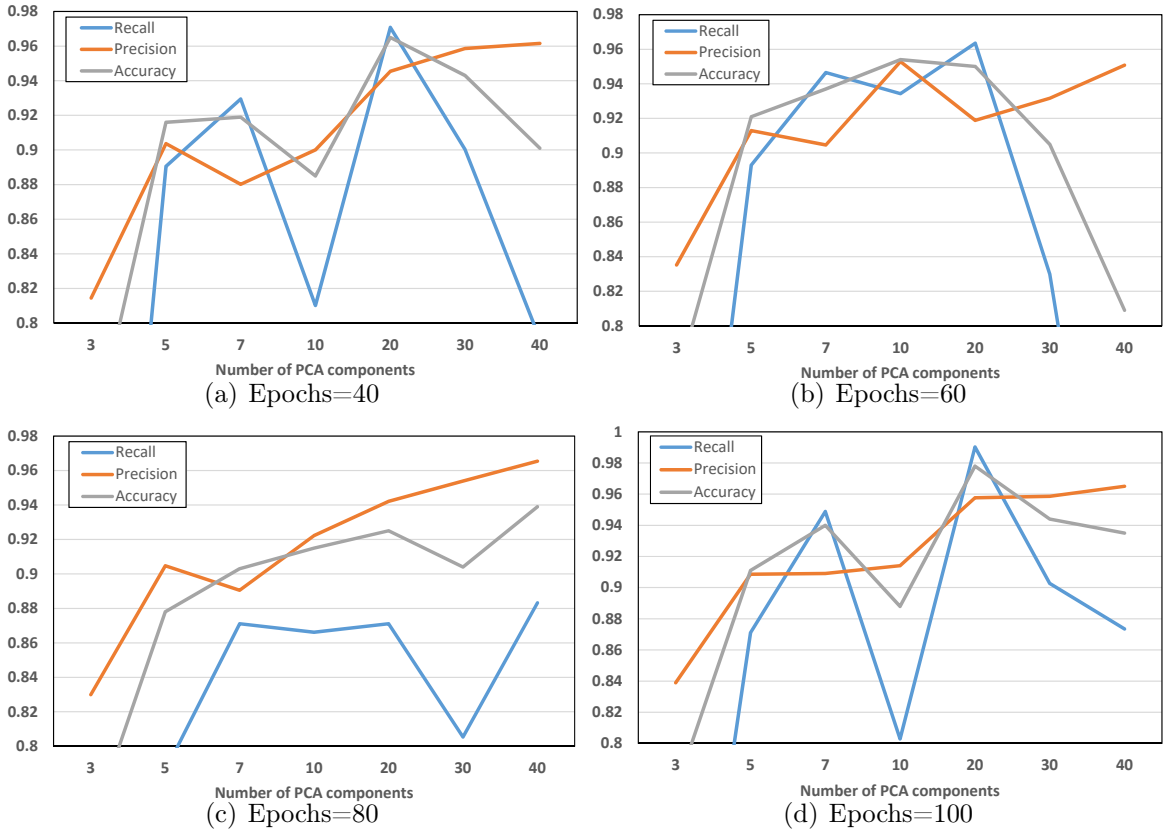


Figure 8.10: Performance of model trained with varying number of PCA components and a) Epochs=20 b) Epochs=40 c) Epochs=60 d) Epochs=80 e) Epochs=100 with 3 communication rounds and 15 neurons in hidden layer

8.6.4 Model Performance Vs Communication Loss for FL-LoRaMAC

In order to evaluate the performance of the system, it was assumed that the devices had successfully joined the network and were also configured according to specifications mentioned in section 8.4.2. Also, the end nodes had been supplied with the training data (collected in the real scenario). The application server transmits the untrained global model as downlink messages to the end devices taking part in the training. This transmission of the global model was acknowledged by the end devices. Hence, all fragments of the global model were received by all the end devices. The end nodes trained the model based on the local data. The training was performed for a certain duration and after that, the gradient updates were transmitted to the

application server as fragments. Each end device was assumed to transmit one fragment every 60 seconds. Also, each of these fragments was 28 bytes long. Considering SF7, the time on air for each packet was 62 milliseconds. During this transmission, fragments would have collided with either data fragments from other participating devices or data from non-participating devices. The probability of successful transmission was calculated using Poisson's distribution. If a collision took place then that fragment was considered to be lost. The system treats these lost fragments according to the specifications in section 8.4.3. Due to this loss, the model performance would deteriorate in terms of making predictions on the testing dataset. Recall, precision, and accuracy metrics were used for evaluating model performance to make predictions. The recall is defined as the ratio between the number of samples correctly predicted as positive to the total number of positive samples. In the case of ECG anomaly detection, recall can be formulated as

$$Recall = \frac{Number\ of\ samples\ correctly\ predicted\ as\ abnormal}{Total\ abnormal\ samples} \tag{8.1}$$

Precision is defined as the ratio of positive samples to the total number of positive predictions that the model made. In the case of ECG anomaly detection, precision can be formulated as

$$Precision = \frac{Total\ number\ of\ abnormal\ samples}{Number\ of\ samples\ predicted\ to\ be\ abnormal} \tag{8.2}$$

Finally, accuracy is defined as the ratio of the total number of correct predictions to the total number of predictions. In the case of ECG anomaly detection, accuracy can be formulated as

$$Accuracy = \frac{Number\ of\ samples\ correctly\ predicted\ (abnormal\ or\ normal)}{Total\ number\ of\ predictions} \tag{8.3}$$

All these evaluation metrics were calculated for varying loss rates from the implementation. The results are plotted in figure 8.11.

When no packet was lost in communication, the performance of the model was highest with a recall of 99.27% and a precision of 95.77%. As the loss rate increases the performance of the model should degrade but till a loss rate of 50%, the system was able to maintain the recall at 93.6% and precision at 89.5%. The Federated averaging mechanism was responsible to keep the performance stable and hence it did not degrade steeply. During communication, even if one of the transmissions of a certain model parameter was received by the server, the averaging algorithm preserves the notion of that parameter in the updated global model. Hence, the end devices instead of getting a 'zero' due to the lost model parameters, receive the averaged value. This helps the model to converge faster and reap the benefits of federated learning. Once, the system starts to lose more packets, meaning that not even a single transmission for some parameters is able to reach the server, then the model might need more time to converge or does not converge at all.

8.6.5 Model Training Time with FL-LoRaMAC

The time duration for computing gradient updates during the training phase was measured experimentally and the communication time was calculated mathematically for the training hyperparameters used.

$$T_{Train} = T_{DU} + N * (T_C + T_{Up} + T_{FAv} + T_D) \tag{8.4}$$

Equation 8.4 was used to calculate the total training time. Where T_{DU} is the average time taken by the server to transmit all the untrained global model parameters, N is the number of communication rounds used, T_C is the average time taken to compute local model updates, T_{Up} is the average time taken by end devices to transmit local updated model parameters, T_{FAv} is the average time taken by the federated

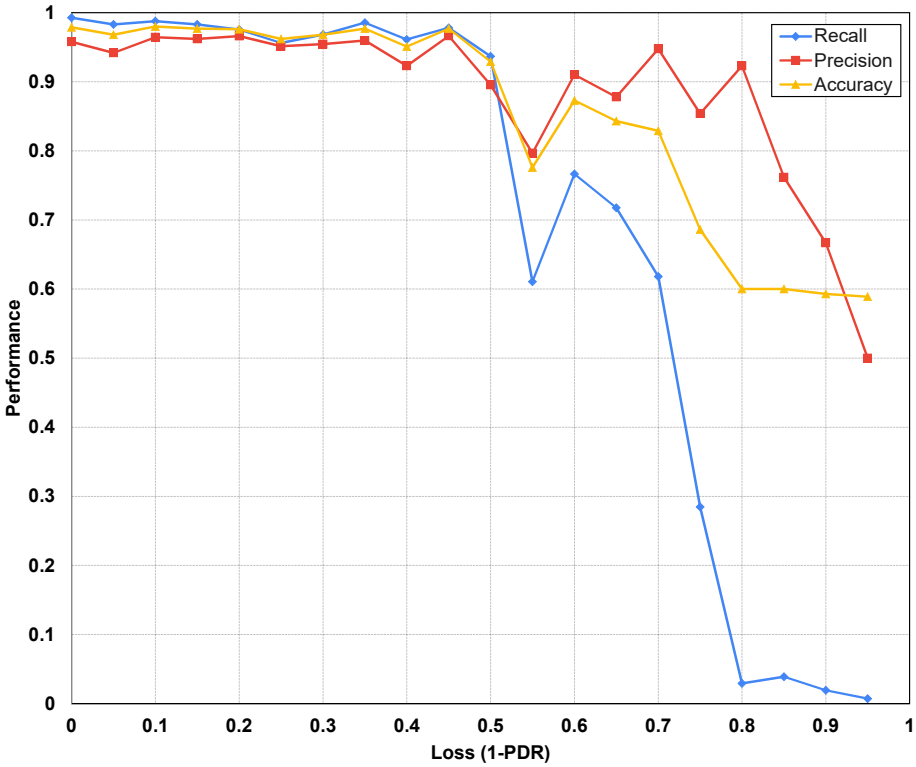


Figure 8.11: Performance Vs loss

averaging algorithm to compute the updated global model from local updates, and T_D is the average time taken to transmit the updated global model parameters.

All the calculations for communication time were made considering a spreading factor of SF7. The same can be calculated if other spreading factors were to be used.

To calculate communication time, the total number of model parameters that need to be communicated was determined. The model has 635 parameters and which equates to 2,540 bytes. Each fragment was considered to be 28 bytes long. Hence, there were 91 fragments that need to be communicated from each end device in each communication round. Considering, each end device transmits one packet every 60 seconds, T_{Up} will be 5460 seconds.

The size of the frame was chosen to be 28 bytes because, in the US, LoRa technology has a dwell time restriction of 400 milliseconds. Each packet duration must be at most the dwell time. In the case of SF10, the duration of a 28 bytes packet is 400

milliseconds. Hence, the size of the fragment was chosen to be 28 bytes for the simplicity of implementation. Although, in the case of lower spreading factors, more data can be sent in one packet. However, a longer packet means a higher probability of collision when there are multiple devices transmitting, which is the case during end devices sending local updates.

When the global model is sent by the server, only one gateway will be transmitting. Hence, the maximum number of bytes that can be packed in a packet and can be sent more frequently with negligible probability of packet loss due to collisions. Considering gateway transmits using SF7, each packet can have 200 bytes with 100 ms of the preamble. Hence, the gateway can send all 635 parameters with 13 packets, transmitting one packet every 10 seconds. Considering no packets were lost during transmission, T_{DU} as well as T_D will be 130 seconds.

The value of T_C and T_{FAv} was measured to be 4.95 seconds and 0.016 seconds respectively. By putting all the values in equation 8.4, the total training in seconds was calculated to be 16914.9 seconds or 4.7 hours.

$$T_{Train} = 130 + (3 * (5460 + 130 + 4.95 + 0.016))Seconds \tag{8.5}$$

It can be observed that the communication of model parameters takes the most time in the training phase. This communication time can be reduced if less number of communication rounds will be used but doing so will deteriorate the performance of the model. This trade-off can be exploited depending on the nature of the application.

8.6.6 Energy Consumption of FL-LoRaMAC vs Legacy

As discussed earlier, the federated learning can be implemented using legacy LoRaWAN protocol using Class-C mode of operation for end devices. However, doing so will incur enormous energy costs. This section compares the energy consumption of the proposed FL-LoRaMAC framework to the Class-C operation of the legacy

LoRaWAN protocol.

The uplink transmissions for FL-LoRaMAC are similar to that of the legacy protocol. Hence the energy consumption for uplink transmissions will be the same for both cases. However, for downlink, the end devices according to FL-LoRaMAC will conserve energy by putting its radios to sleep.

The energy consumed by the LoRa transceiver in the end device to receive one downlink packet in FL-LoRaMAC can be calculated using the equation 8.6

$$E_T = E_S + E_{Un} + E_{Sl} \tag{8.6}$$

Where, E_S is the energy consumed when the device starts listening and actually receives the downlink packet, E_{Un} is the energy consumed when the device starts listening and doesn't receive any data, and E_{Sl} is the energy consumed by the transceiver module in sleep mode.

The energy consumption can be found by knowing the operating voltage of the transceiver, the current drawn, and the duration for which the current is drawn. The SX1276 LoRa module operates on 3.3V, draws 10.8 mA (I_R) during receive mode and 0.0002 mA (I_{Sl}) in sleep mode [111]. The gateway transmits downlinks once every 10 seconds. Consider the end devices opening receive window every 100 milliseconds for 5 milliseconds and then going to sleep if it does not detect any LoRa preamble. E_S, E_{Un}, and E_{Sl} can be found according to equations 8.7-8.9.

$$E_S = V * I_R * T_R = 3.3V * 10.8mA * 400ms \tag{8.7}$$

$$E_{Un} = V * I_R * T_R = 3.3V * 10.8mA * 480ms \tag{8.8}$$

$$\begin{aligned} E_{Sl} &= V * I_{Sl} * T_{Sl} \\ &= 3.3V * 0.0002mA * (1000 - 480 - 400)ms \end{aligned} \tag{8.9}$$

Where T_R and T_{Sl} are the time duration of receiving and duration of sleep respectively. Using equation 8.6, the energy consumption of the LoRa transceiver for 1 downlink packet is 0.0087 mWH. Using SF7, 13 packets need to be sent in the downlink for transmitting all model parameters in one communication round. Hence energy consumed by the transceiver to receive all the parameters in one communication round is 0.1131 mAH.

In the case of legacy Class-C operation, the transceiver module always stays in receive mode. It will use the standard 8-symbol preamble, not the elongated preamble so a data packet of SF7 can have 255 bytes. In this scenario, 10 packets will be sufficient to transmit all 635 model parameters. The energy consumed by the transceiver to receive all model parameters will be $10 * (3.3V * 10.8mA * 10000s) = 0.99$ mAH. Hence, legacy Class-C devices will consume approximately 9 times more energy than the proposed FL-LoRaMAC for receiving model parameters.

8.6.7 Performance Vs Pruning

While performing model pruning, a sparsification parameter specifies what percentage of the model parameters will be pruned. The performance of the model was evaluated for varying sparsification percentages and plotted in figure 8.12.

From the plot, it can be observed that with the value of recall fixed by varying the model threshold, as the matrix gets more and more sparse, the model precision degrades. Sparsity is a hyper-parameter and it can be observed that for the representative application, till 60% sparsity, the model performance is not affected much, hence even if 60% of the model parameters are pruned the model will still perform well. If the framework chooses not to send those pruned parameters, this will result in at least 60% savings in bandwidth without compromising the performance of the system.

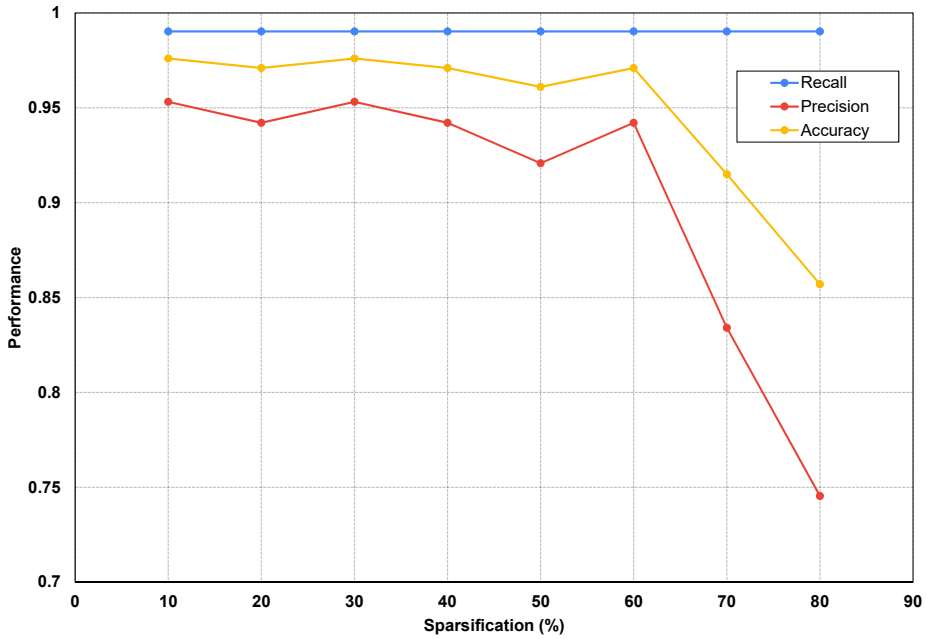


Figure 8.12: Performance Vs sparsification

8.6.8 Performance comparison with and without QoS-based SF distribution

For comparing the performance of the system with and without QoS-based SF distribution principles, ambient devices were also considered in the network to be running some other applications, transmitting packets with higher priorities selected based on some criteria. It was assumed that the probability of success requirements for high-priority packets was 90%. Assuming that other ambient end devices are transmitting data every 60 seconds, similar to the devices running the representative application. Also, it can be assumed that at any time the network has 10% high-priority packets out of the total traffic. Using this network configuration, the maximum number of devices that can be supported by each spreading factor were found and tabulated in table 8.4. The spreading factors were then allocated to different packets according to the top-down approach mentioned in section 6.4.1. The packet delivery ratios were calculated using Poisson's distribution for a varying number of total devices in the network and are plotted in figure 8.13.

Using the PDR values for the varying total number of devices illustrated in figure

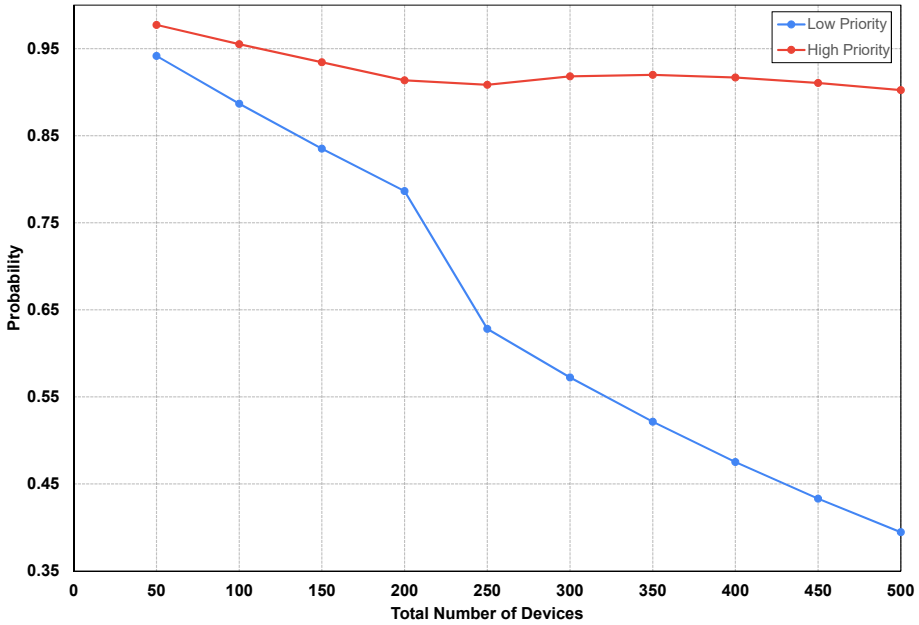


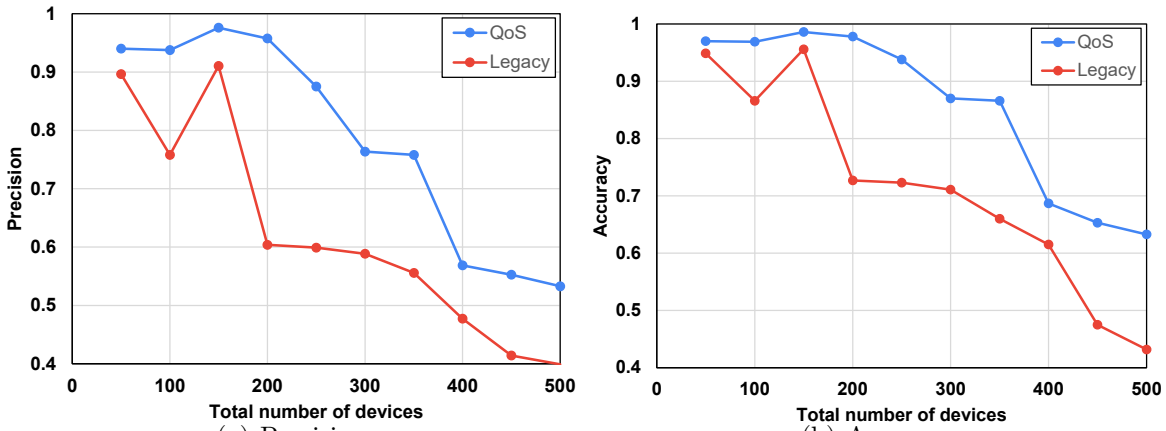
Figure 8.13: Probability of success for high and priority devices for varying total number of devices

Table 8.4: Maximum number of devices for various spreading factors with 90% or greater probability of success

Spreading Factors	**Max. number of devices (D)**
SF7	50
SF8	27
SF9	15
SF10	8

8.13, the model for the ECG anomaly detection application was trained and tested. The significance criteria was chosen to be 80% of the maximum and minimum values. In other words, if the value of the parameters is greater than 80% value of the maximum value or less than 80% value of the minimum value, that particular model parameter is considered to be significant. The performance of the model trained by applying the pruning technique along with QoS-based distribution and without applying QoS distribution is plotted in figure 8.14.

For the comparison, the value of recall was kept fixed at 99.02%, and the values of precision and accuracy were determined against the recall. This was achieved by varying the threshold for the prediction. It can be observed that the number of



(a) Precision (b) Accuracy

Figure 8.14: Performance comparison of the trained model with QoS and legacy (a) Precision (b) Accuracy with the varying total number of devices in network

devices in the network increases the performance of the model degrades. However, the model trained by considering the differential priority of packets provides better performance than the one trained by considering the same priority for all. Providing better success rates to high-priority packets that contain more significant parameters helps the model to converge faster when compared to the legacy approach.

8.7 Conclusions

The chapter introduces artificial intelligence tools for IoT applications and a novel framework to enable on-device learning for LoRa-based devices. Machine learning principles can prove to be very beneficial for IoT applications but due to the challenges imposed by IoT and LoRaWAN ecosystems such as data privacy, low bandwidth, etc., traditional machine learning techniques can not be efficiently applied. However, a decentralized learning technique known as federated learning can be used while maintaining data privacy. The performance of models trained via traditional machine learning and federated learning principles was analyzed. This analysis proves that for the representative application federated learning can be viably employed in place of machine learning. However, federated learning requires independent bidirectional

communication. The legacy LoRaWAN protocol fails to satisfy the communication requirements of federated learning, hence FL-LoRaMAC has been proposed to fill this gap. The performance of the framework was evaluated by implementing the framework on a representative application. The results show that the framework was able to successfully support all the design requirements of federated learning even with communication losses in the system. The proposed framework also incorporates bandwidth optimization to enable efficient resource utilization.

CHAPTER 9: CONCLUSIONS AND FUTURE WORKS

The main purpose of this research was to improve the performance of LoRa networks for smart city deployments.

We proposed several approaches to improve the performance of LoRa networks. We started by experimentally evaluating the performance of LoRaWAN networks, which included evaluating the transmission range, power consumption, and scalability of the LoRa/LoRaWAN networks. We observed a transmission range of more than 3 miles with over 50% successful transmissions. It was also found that a LoRa end device can achieve a life expectancy of more than 7 years on a standard Li-ion 2800mAH battery. The scalability evaluation showed us that although the gateway can theoretically handle hundreds of thousands of devices, in the real world as the number of devices increases by a few hundred, the performance degrades. The main reason behind this performance degradation is collisions among the transmissions.

Based on the fact that collisions are the main reason for performance degradation, we proposed an approach to improve the scalability of LoRaWAN networks by spreading factor distribution. The proposed approach exploits the orthogonality property of spreading factors i.e., devices configured on the same channel but different spreading factors do not collide with each other. This property was used to configure devices on various spreading factors and performance improvement was determined. We also found the optimum fraction of devices that when configured on various spreading factors maximize the network performance.

As the LoRa networks will be deployed in IoT and smart cities scenarios, the network must be able to satisfy the underlying QoS requirements for such applications. The LoRaWAN protocol considers all devices to be the same and serves all devices

equally. Due to this, QoS requirements are not satisfied at all times. Our proposed approach extends our previous work and makes use of the orthogonality of spreading factors. By using the proposed approach, the network was successfully able to satisfy the QoS requirements. It was also found that by using SFA-1, the average performance of the network will also improve.

Finally, the dissertation expanded to incorporate AI tools in LoRa-based IoT networks. AI tools have the capability to improve and gain more insights into IoT applications. This can have the potential to enable abundant opportunities for the research community and industry. In order to understand the implementation, benefits, and communication requirements of machine learning, we developed a traffic congestion prediction model. The model was able to predict the congestion with 90% accuracy, which can be of tremendous use to drivers. However, it was found that the machine learning principles require all sensor data to be sent to the central server where the training takes place.

Hence, the typical machine learning tools face hindrances to be employed in IoT and LoRa ecosystems due to low bandwidth available and privacy concerns. Our proposed framework FL-LoRaMAC enables federated learning principles in IoT and LoRa ecosystems that are proven to resolve the challenges faced by typical machine learning principles while achieving similar performance. The framework provided the necessary communication requirements required to employ federated learning in LoRa-based IoT platforms. Several approaches to optimize bandwidth usage were also exploited. The use of QoS-based SF allocation for communicating gradient updates also proved to improve the performance.

9.1 Future Works

The dissertation considers LoRa networks with all transmissions taking place on a single channel, assuming a single-channel gateway was used. Typically, LoRaWAN gateways are available as 1-channel, 8-channel, 16-channel, and 64-channel gateways.

When a multi-channel gateway is used, each device performs channel hopping for every uplink packet in a sequential round-robin fashion. Also, the LoRaWAN networks deployed in smart city scenarios have multiple gateways connected to the networks forming a star-of-stars topology. The potential areas of future research in this field are:

- In a multi-channel gateway scenario, the channels are allocated sequentially for every uplink transmission. Exploring an efficient channel-hopping approach based on channel characteristics toward improving the performance of the network could be an interesting research area. AI principles can also be employed to dynamically allocate the channels for each uplink transmission.

- With the help of the optimum fraction of device distribution on various SFs (Chapter 5), the average packet delivery rates are improved. By configuring devices on higher spreading factors, the packets become more susceptible to interference. Considering a multi-channel gateway scenario, the devices can be configured on different channels but with spreading factors that are optimum according to the ADR mechanism.

- In addition to the proposed QoS-based SF allocation approaches discussed in Chapter 6, allocating different sets of channels to the transmissions according to the application requirements in a multi-channel gateway scenario is left unexplored in this work. Such channel allocation schemes have the potential to benefit smart city IoT applications.

- The framework discussed in Chapter 8, considers all devices running the application taking part in the learning process. Typically, only a set of devices have the most information and should be able to train the learning model. Training all devices without considering the degree of information content can increase computational and communication costs. The selection mechanism for devices

with the best information and channel conditions for participating in federated learning is also a much-unexplored territory and can be of significant importance for federated learning-based smart city applications.

These future works have the potential to further improve the performance of the LoRa networks.

REFERENCES

[1] "LoRa symbol generation (accessed on 11/28/2021)." [Online] Available: http://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html.

[2] "ECG waveform (accessed on 11/8/2022)." [Online]. Available: https://www.ems1.com/ems-products/medical-monitoring/articles/quiz-interpreting-cardiac-waveforms-r3tCWbZuBkI5Ol9M/.

[3] K. Garg, C. Goswami, R. Chhatrawat, S. Kumar Dhakar, and G. Kumar, "Internet of things in manufacturing: A review," *Materials Today: Proceedings*, vol. 51, pp. 286–288, 2022. CMAE'21.

[4] Sullivan AN, Lachman ME. Behavior Change with Fitness Technology in Sedentary Adults: A Review of the Evidence for Increasing Physical Activity. Front Public Health. 2017 Jan 11;4:289. doi: 10.3389/fpubh.2016.00289. PMID: 28123997; PMCID: PMC5225122.

[5] K. Gnana Sheela and K. Jilna, "An intelligent energy management system using IoT," *Materials Today: Proceedings*, vol. 24, pp. 1903–1908, 2020. International Multi-conference on Computing, Communication, Electrical Nanotechnology, I2CN-2K19, 25th 26th April 2019.

[6] O. Ribeiro, L. Gomes, and Z. Vale, "IoT-based human fall detection system," *Electronics*, vol. 11, no. 4, 2022.

[7] A. A. Brincat, F. Pacifici, S. Martinaglia, and F. Mazzola, "The internet of things for intelligent transportation systems in real smart cities scenarios," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pp. 128–132, 2019.

[8] K. Bakthavatchalam, B. Karthik, V. Thiruvengadam, S. Muthal, D. Jose, K. Kotecha, and V. Varadarajan, "IoT framework for measurement and precision agriculture: Predicting the crop using machine learning algorithms," *Technologies*, vol. 10, no. 1, 2022.

[9] C. Rosenberg, "Challenges in multi-hop networks," in *2006 2nd Conference on Next Generation Internet Design and Engineering, 2006. NGI '06.*, pp. 1 pp.–xvii, 2006.

[10] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 855–873, 2017.

[11] S. Andreev, O. Galinina, A. Pyattaev, M. Gerasimenko, T. Tirronen, J. Torsner, J. Sachs, M. Dohler, and Y. Koucheryavy, "Understanding the IoT connectivity landscape: a contemporary m2m radio technology roadmap," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 32–40, 2015.

[12] "RP002-1.0.2 LoRaWAN regional parameters (accessed on 11/28/2021)." [Online]. Available: https://lora-alliance.org/wp-content/uploads/2020/11/RP_2 − 1.0.2.*pdf*.

[13] B. Reynders, W. Meert, and S. Pollin, "Range and coexistence analysis of long range unlicensed communication," in *2016 23rd International Conference on Telecommunications (ICT)*, pp. 1–6, 2016.

[14] "Semtech: Analog and Mixed-Signal Semiconductors (Accessed on 03/14/2023)." [Online] Available: https://www.semtech.com/.

[15] "LoRa Alliance (accessed on 11/28/2021)." [Online]. Available: https://lora-alliance.org/.

[16] K. Mikhaylov, J. Petaejaejaervi, and T. Haenninen, "Analysis of capacity and scalability of the LoRa low power wide area network technology," in *European Wireless 2016; 22th European Wireless Conference*, pp. 1–6, 2016.

[17] B. Vejlgaard, M. Lauridsen, H. Nguyen, I. Z. Kovacs, P. Mogensen, and M. Sorensen, "Coverage and capacity analysis of Sigfox, LoRa, GPRS, and NB-IoT," in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, pp. 1–5, 2017.

[18] S. Aggarwal and A. Nasipuri, "Survey and performance study of emerging LP-WAN technologies for IoT applications," in *2019 IEEE 16th International Conference on Smart Cities: Improving Quality of Life Using ICT IoT and AI (HONET-ICT)*, pp. 069–073, 2019.

[19] S. Aggarwal and A. Nasipuri, "Improving scalability of lorawan networks by spreading factor distribution," in *SoutheastCon 2021*, pp. 1–7, 2021.

[20] S. Aggarwal and A. Nasipuri, "QoS based spreading factor assignment for lorawan networks in iot applications," in *SoutheastCon 2022*, pp. 46–53, 2022.

[21] F. Salahdine, S. Aggarwal, and A. Nasipuri, "Short-term traffic congestion prediction with deep learning for LoRa networks," in *SoutheastCon 2022*, pp. 261–268, 2022.

[22] "P.M. John Burns, Selcuk Kirtay, "Future use of licence exempt radio spectrum", Plum Consulting, Tech. Rep., 2015." [Online]. Available: http://www.plumconsulting.co.uk/pdfs/Plum July 2015 Future use of Licence Exempt Radio Spectrum.pdf.

[23] "Weightless SIG. (accessed on 11/28/2021)." [Online]. Available: https://www.weightless-alliance.org/.

[24] "Inegu formerly known as On-Ramp (accessed on 11/28/2021)." [Online]. Available: https://www.ingenu.com/technology/rpma/lpwa/.

[25] "Communications system, us patent 8,406,275." Inventor: Francois Sforza Available: https://patents.google.com/patent/US8406275B2/en.

[26] "LoRaWAN specification (accessed on 10/10/2022)." [Online]. Available: https://lora-alliance.org/lorawan-for-developers/.

[27] "LoRa and LoRaWAN: Technical overview (accessed on 03/14/2023)." [Online]. Available: https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/.

[28] A. Berni and W. Gregg, "On the utility of chirp modulation for digital signaling," *IEEE Transactions on Communications*, vol. 21, no. 6, pp. 748–751, 1973.

[29] "Semtech's simple rate adaptation recommended algorithm (accessed on 11/28/2021).." [Online]. Available: https://www.thethingsnetwork.org/forum/uploads/default/original/2X/7/7480e044aa93a54a910dab8ef0adfb5f515d14a1.pdf.

[30] "The Things Network (accessed on 11/28/2021).." [Online]. Available: https://www.thethingsnetwork.org/.

[31] "What is artificial intelligence?." Accessed on 03/14/2023 https://www-formal.stanford.edu/jmc/whatisai.pdf.

[32] "Computing machinery and intelligence." A. M. Turing (1950) https://redirect.cs.umbc.edu/courses/471/papers/turing.pdf.

[33] T. M. Mitchell, "Machine learning," 1997.

[34] J. Alzubi, A. Nayyar, and A. Kumar, "Machine learning from theory to algorithms: an overview," in *Journal of physics: conference series*, vol. 1142, p. 012012, IOP Publishing, 2018.

[35] I. Borg and P. J. Groenen, *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[36] T. Kurita, "Principal component analysis (PCA)," *Computer Vision: A Reference Guide*, pp. 1–4, 2019.

[37] I. T. Jolliffe, *Principal component analysis for special types of data*. Springer, 2002.

[38] "Linear discriminant analysis, explained." Accessed on 03/14/2023 https://towardsdatascience.com/linear-discriminant-analysis-explained-f88be6c1e00b.

[39] Choudhary, T., Mishra, V., Goswami, A. et al. A comprehensive survey on model compression and acceleration. Artif Intell Rev 53, 5113â5155 (2020). https://doi.org/10.1007/s10462-020-09816-7.

[40] Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J. (2016). Pruning Convolutional Neural Networks for Resource Efficient Inference. ArXiv. /abs/1611.06440.

[41] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Pettissalo, "On the coverage of LPWANs: Range evaluation and channel attenuation model for LoRa technology," in *2015 14th International Conference on ITS Telecommunications (ITST)*, pp. 55–59, 2015.

[42] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 60–67, 2016.

[43] A. Augustin, J. Yi, T. Clausen, and W. Townsley, "A study of LoRa: Long range low power networks for the internet of things," *Sensors*, vol. 16, p. 1466, Sep 2016.

[44] P. Neumann, J. Montavont, and T. Noel, "Indoor deployment of low-power wide area networks (LPWAN): A LoRaWAN case study," in *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1–8, 2016.

[45] J. Petajajarvi, K. Mikhaylov, M. Hamalainen, and J. Iinatti, "Evaluation of LoRa LPWAN technology for remote health and wellbeing monitoring," in *2016 10th International Symposium on Medical Information and Communication Technology (ISMICT)*, pp. 1–5, 2016.

[46] S. Hosseinzadeh, H. Larijani, K. Curtis, A. Wixted, and A. Amini, "Empirical propagation performance evaluation of LoRa for indoor environment," in *2017 IEEE 15th International Conference on Industrial Informatics (INDIN)*, pp. 26–31, 2017.

[47] J. Haxhibeqiri, A. Karaagac, F. Van den Abeele, W. Joseph, I. Moerman, and J. Hoebeke, "LoRa indoor coverage and performance in an industrial environment: Case study," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–8, 2017.

[48] J. Petajajarvi, K. Mikhaylov, M. Pettissalo, J. Janhunen, and J. Iinatti, "Performance of a low-power wide-area network based on LoRa technology: Doppler robustness, scalability, and coverage," *International Journal of Distributed Sensor Networks*, vol. Vol. 13, pp. 1–16, 03 2017.

[49] T. Voigt, M. Bor, U. Roedig, and J. Alonso, "Mitigating inter-network interference in LoRa networks," in *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*, EWSN â17, (USA), p. 323â328, Junction Publishing, 2017.

[50] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa low-power wide-area networks scale?," in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '16, (New York, NY, USA), p. 59â67, Association for Computing Machinery, 2016.

[51] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello, "Impact of LoRa imperfect orthogonality: Analysis of link-level performance," *IEEE Communications Letters*, vol. 22, no. 4, pp. 796–799, 2018.

[52] B. Reynders, W. Meert, and S. Pollin, "Range and coexistence analysis of long range unlicensed communication," in *2016 23rd International Conference on Telecommunications (ICT)*, pp. 1–6, 2016.

[53] K. Mikhaylov, J. Petajajarvi, and J. Janhunen, "On LoRaWAN scalability: Empirical evaluation of susceptibility to inter-network interference," in *2017 European Conference on Networks and Communications (EuCNC)*, pp. 1–6, 2017.

[54] L. Feltrin, C. Buratti, E. Vinciarelli, R. De Bonis, and R. Verdone, "LoRaWAN: Evaluation of link and system-level performance," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2249–2258, 2018.

[55] F. Van den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Scalability analysis of large-scale LoRaWAN networks in NS-3," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2186–2198, 2017.

[56] A.-I. Pop, U. Raza, P. S. Kulkarni, and M. Sooriyabandara, "Does bidirectional traffic do more harm than good in LoRaWAN based LPWA networks?," *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1–6, 2017.

[57] J. Haxhibeqiri, F. Van den Abeele, I. Moerman, and J. Hoebeke, "LoRa scalability: A simulation model based on interference measurements," *Sensors*, vol. 17, no. 6, 2017.

[58] M. Centenaro, L. Vangelista, and R. Kohno, "On the impact of downlink feedback on LoRa performance," in *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–6, 2017.

[59] M. Capuzzo, D. Magrin, and A. Zanella, "Confirmed traffic in LoRaWAN: Pitfalls and countermeasures," in *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, pp. 1–7, 2018.

[60] B. Reynders, W. Meert, and S. Pollin, "Power and spreading factor control in low power wide area networks," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2017.

[61] J.-T. Lim and Y. Han, "Spreading factor allocation for massive connectivity in LoRa systems," *IEEE Communications Letters*, vol. 22, no. 4, pp. 800–803, 2018.

[62] F. Cuomo, M. Campo, A. Caponi, G. Bianchi, G. Rossini, and P. Pisani, "EXPLoRa: Extending the performance of lora by suitable spreading factor allocations," in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1–8, 2017.

[63] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of LoRa networks for dense IoT deployments," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, 2018.

[64] S. Li, U. Raza, and A. Khan, "How agile is the adaptive data rate mechanism of LoRaWAN?," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 206–212, 2018.

[65] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin, "Improving reliability and scalability of LoRaWANs through lightweight scheduling," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1830–1842, 2018.

[66] M. Rizzi, A. Depari, P. Ferrari, A. Flammini, S. Rinaldi, and E. Sisinni, "Synchronization uncertainty versus power efficiency in LoRaWAN networks," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 4, pp. 1101–1111, 2019.

[67] J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Low overhead scheduling of LoRa transmissions for improved scalability," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3097–3109, 2019.

[68] T. Polonelli, D. Brunelli, and L. Benini, "Slotted ALOHA overlay on LoRaWAN - a distributed synchronization approach," in *2018 IEEE 16th International Conference on Embedded and Ubiquitous Computing (EUC)*, pp. 129–132, 2018.

[69] T.-H. To and A. Duda, "Simulation of LoRa in NS-3: Improving LoRa performance with CSMA," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–7, 2018.

[70] M. Centenaro and L. Vangelista, "Boosting network capacity in LoRaWAN through time-power multiplexing," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6, 2018.

[71] R. Almeida, R. Oliveira, D. Sousa, M. Luis, C. Senna, and S. Sargento, "A multi-technology opportunistic platform for environmental data gathering on smart cities," in *2017 IEEE Globecom Workshops (GC Wkshps)*, pp. 1–7, 2017.

[72] R. Duan, X. Chen, and T. Xing, "A QoS architecture for IoT," in *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, pp. 717–720, 2011.

[73] A. Dvornikov, P. Abramov, S. Efremov, and L. Voskov, "QoS metrics measurement in long range IoT networks," in *2017 IEEE 19th Conference on Business Informatics (CBI)*, vol. 02, pp. 15–20, 2017.

[74] I. Awan, M. Younas, and W. Naveed, "Modelling QoS in IoT applications," in *2014 17th International Conference on Network-Based Information Systems*, pp. 99–105, 2014.

[75] Simjanoska M, Gjoreski M, Gams M, Madevska Bogdanova A. Non-Invasive Blood Pressure Estimation from ECG Using Machine Learning Techniques. Sensors (Basel). 2018 Apr 11;18(4):1160. doi: 10.3390/s18041160. PMID: 29641430; PMCID: PMC5949031.

[76] Raju KB, Dara S, Vidyarthi A, Gupta VM, Khan B. Smart Heart Disease Prediction System with IoT and Fog Computing Sectors Enabled by Cascaded Deep Learning Model. Comput Intell Neurosci. 2022 Jan 10;2022:1070697. doi: 10.1155/2022/1070697. PMID: 35047027; PMCID: PMC8763532.

[77] A. Anwar and A. Raychowdhury, "Autonomous navigation via deep reinforcement learning for resource constraint edge nodes using transfer learning," *IEEE Access*, vol. 8, pp. 26549–26560, 2020.

[78] "Multitech mDot long range LoRa modules (mtdot series)." Accessed on 03/14/2023 https://www.multitech.com/brands/multiconnect-mdot.

[79] "Datasheet of multitech conduit (accessed on 03/14/2023)." [Online]. Available: https://www.multitech.com/documents/publications/datasheets/86002193.pdf.

[80] N. Bisnik and A. A. Abouzeid, "Queuing network models for delay analysis of multihop wireless ad hoc networks," *Ad Hoc Networks*, vol. 7, no. 1, pp. 79–97, 2009.

[81] S. Lee, B. Han, and M. Shin, "Robust routing in wireless ad hoc networks," in *Proceedings. International Conference on Parallel Processing Workshop*, pp. 73–78, 2002.

[82] C. Schurgers and M. Srivastava, "Energy efficient routing in wireless sensor networks," in *2001 MILCOM Proceedings Communications for Network-Centric Operations: Creating the Information Force (Cat. No.01CH37277)*, vol. 1, pp. 357–361 vol.1, 2001.

[83] "Datasheet of SX1272 LoRa module by semtech (accessed on 03/14/2023)." [Online]. Available: https://www.semtech.com/uploads/documents/SX1272 DS V4.pdf.

[84] "Documentation of network simulator-3 (accessed on 03/14/2023)." [Online]. Available: https://www.nsnam.org/documentation/.

[85] "LoRaWAN simulation model for NS-3 (accessed on 01/24/2021)." [Online]. Available: https://github.com/signetlabdei/lorawan.

[86] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: the rising stars in the iot and smart city scenarios," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 60–67, 2016.

[87] A. Dvornikov, P. Abramov, S. Efremov, and L. Voskov, "QoS metrics measurement in long range IoT networks," in *2017 IEEE 19th Conference on Business Informatics (CBI)*, vol. 02, pp. 15–20, 2017.

[88] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of LoRaWAN," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[89] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello, "Impact of lora imperfect orthogonality: Analysis of link-level performance," *IEEE Communications Letters*, vol. 22, no. 4, pp. 796–799, 2018.

[90] "US vehicle registration statistics." Accessed on 03/14/2023 https://hedgescompany.com/automotive-market-research-statistics/auto-mailing-lists-and-marketing/.

[91] J. C. Falcocchio and H. S. Levinson, "The costs and other consequences of traffic congestion," in *Road Traffic Congestion: A Concise Guide*, pp. 159–182, Springer, 2015.

[92] M. Akhtar and S. Moridpour, "A review of traffic congestion prediction using artificial intelligence," *Journal of Advanced Transportation*, vol. 2021, 2021.

[93] S. A. A'ssri, F. H. Zaman, and S. Mubdi, "The efficient parking bay allocation and management system using LoRaWAN," in *2017 IEEE 8th Control and System Graduate Research Colloquium (ICSGRC)*, pp. 127–131, IEEE, 2017.

[94] Y. Tian, K. Zhang, J. Li, X. Lin, and B. Yang, "LSTM-based traffic flow prediction with missing data," *Neurocomputing*, vol. 318, pp. 297–305, 2018.

[95] "California department of transportation." Accessed on 03/14/2023 https://pems.dot.ca.gov/.

[96] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for internet of things: Recent advances, taxonomy, and open challenges," *IEEE Communications Surveys & Tutorials*, 2021.

[97] "Future of industry ecosystems: Shared data and insights (accessed on 03/14/2023)." [Online]. Available: https://blogs.idc.com/2021/01/06/future-of-industry-ecosystems-shared-data-and-insights/: :text=IDC%20estimates%20there%20will%20be,the%20importance%20of%20expanding%20their.

[98] N. Majumdar, S. Shukla, and A. Bhatnagar, "Survey on applications of internet of things using machine learning," in *2019 9th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pp. 562–566, 2019.

[99] P. P. Shinde and S. Shah, "A review of machine learning and deep learning applications," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, pp. 1–6, 2018.

[100] M. Chen, Y. Hao, K. Hwang, L. Wang, and L. Wang, "Disease prediction by machine learning over big data from healthcare communities," *IEEE Access*, vol. 5, pp. 8869–8879, 2017.

[101] "Rajkomar, a., oren, e., chen, k. et al. scalable and accurate deep learning with electronic health records. npj digital med 1, 18 (2018). https://doi.org/10.1038/s41746-018-0029-1." https://www.nature.com/articles/s41746-018-0029-1citeas.

[102] K. Shailaja, B. Seetharamulu, and M. A. Jabbar, "Machine learning in healthcare: A review," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 910–914, 2018.

[103] "Microsoft: AI for health (accessed on 03/14/2023)." [Online] Available: https://www.microsoft.com/en-us/ai/ai-for-health.

[104] "Tempus: Data-driven precision medicine (accessed on 03/14/2023)." [Online] Available:https://www.tempus.com/.

[105] "Beta bionics (accessed on 03/14/2023)." [Online] Available:https://www.betabionics.com/.

[106] "Insitro (accessed on 03/14/2023)." [Online] Available:https://insitro.com/.

[107] "Electrocardiogram (ECG or EKG)." [Online] Available:https://www.mayoclinic.org/tests-procedures/ekg/about/pac-20384983: :text=During%20an%20ECG%20%2C%20up%20to,a%20monitor%20or%20on%20paper.

[108] "Can a smartwatch with ECG capability really warn you about an irregular heartbeat? (accessed on 03/14/2023)." [Online] Available: https://www.houstonmethodist.org/blog/articles/2022/jan/can-a-smartwatch-with-ecg-capability-really-warn-you-about-an-irregular-heartbeat/: :text=The%20ECG%20technology%20in%20a,sense%20of%20your%20heart's%20rhythm.

[109] "ECG-5000 dataset (accessed on 11/8/2022)." [Online]. Available: http://www.timeseriesclassification.com/description.php?Dataset=ECG5000.

[110] R. J. Martis, U. R. Acharya, K. Mandana, A. Ray, and C. Chakraborty, "Application of principal component analysis to ECG signals for automated diagnosis of cardiac health," *Expert Systems with Applications*, vol. 39, no. 14, pp. 11792–11800, 2012.

[111] "SX1276 datasheet (accessed on 03/14/2023)." [Online] Available: https://www.mouser.com/datasheet/2/761/sx1276-1278113.pdf.