Machine Learning Based Transmission Scheduling For LoRaWANs

by

Mohammed Ali Alenezi

A thesis submitted to Queen Mary University of London for the degree of

Doctor of Philosophy

School of Electronic Engineering and Computer Science
Queen Mary University of London
United Kingdom

October 2021

TO MY FAMILY...

Abstract

This thesis addresses the problem of excessive packet collision rate in highly dense Long-Range Wide Area Networks (LoRaWAN). The outcomes of this research are reduced collision rate, transmission delay and energy consumption, and enhanced packet delivery rate. A novel framework for packet transmissions is proposed for LoRaWAN protocol. The proposed framework includes:

- Unsupervised learning clustering algorithm that aims to reduce the packet collision rate by classifying the random nature of LoRaWAN nodes packet transmissions into different clusters.

- Supervised learning for packet dynamic transmission Priority Scheduling Technique (PST) that allows the gateway to prioritise the transmissions from nodes located in different clusters. The dynamic transmission *PST* aims at preserving the limited resources of LoRaWAN battery-powered nodes by eliminating unnecessary packet transmissions while maintaining a fair trade-off between the packet delivery rate and energy consumption.

- Deep learning scheme for collision avoidance in ultra-dense networks. This scheme allows the gateway to predict the number of collisions and implement a hybrid procedure to alternate between LoRaWAN transmission classes in order to avoid packet collisions.

The simulation results under different network scenarios show reduced packet collision rate, total transmission delay and total energy consumption, and also enhanced packet delivery rate.

Acknowledgments

First and foremost, I would like to thank the government of the Kingdom of Saudi Arabia for the opportunity to pursue my research programme.

It has been a great honour to work alongside such an elite team, for which I would like to express my sincerest gratitude to my research supervision panel. Firstly, to my primary supervisor Dr. Kok Keong Chai for his limitless support and continual guidance throughout my research progress. Secondly, to Prof. Yue Chen and Dr. Jesus Requena for their influential feedback and advice.

I owe a deep sense of gratitude to Prof. Talib Alukaidey for his genuine support and long-standing ties before and during my research at Queen Mary University of London. Also a special thank you to Dr. Atm Alam for his generous efforts and enlightenment throughout my research.

The deepest heartfelt thank you to my father Maj. Gen. Ali Alenezi who taught me the meaning of perseverance, persistence and discipline, to my mother Bahiah Alsharif who taught me the value of seeking knowledge and for always believing in me, to my wife, to my brothers and sisters for the love they have given. Their presence and moral support has made it all possible, for which I am forever indebted to you all.

Lastly, thank you to my friends and colleagues for being part of this once in a lifetime experience.

Contents

List of Figures

List of Tables

List of Abbreviations

ABP	Activation By Personalisation
ACK	Acknowledgment
ADR	Adaptive Data Rate
BW	Bandwidth
CA-PST	Collision-Aware Priority Scheduling Technique
CID	Command Identifier
Colli	Collisions
con.	conservative
CR	Coding Rate
CRC	Cyclic Redundancy Check
CSMA	Carrier-Sense Multiple Access
CSS	Chirp Spread Spectrum
DL	Deep Learning
DoS	Denial-of-Service
DR	Data Rate
DT	Decision Tree
FEC	Forward Error Correction
FSK	Frequency-Shift Keyin
GRU	Gated Recurrent Unit
GW	Gateway
IoT	Internet of Things

ISM	Industrial, Scientific and Medical
ITD	Initial Transmission Delay
K-Means	Unsupervised Learning Clustering Algorithm
LBT	Listen Before Transmit
LoRa	Long-Range
LoRaWAN	Long-Range Wide Area Networks
LPWAN	Low-Power Wide Area Networks
LSTM	Long Short-Term Memory
MAC	Medium Access Control
MSE	Mean Square Error
ncon.	non-conservative
PDR	Packet Delivery Rate
Pr	Priority
PST	Priority Scheduling Technique
R^2	coefficient of determination
R_c	Chip rate
R_s	Symbol rate
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RSSI	Received Signal Strength Indicator
RTD	Retransmission Delay
SF	Spreading Factors
SSM	State Space Model
TEC	Total Energy Consumption
ToA	Time on Air
TSCH	Time Slotted Channel Hopping
TTD	Total Transmission Delay
WCSS	Within-Cluster Sum of Square
WMAN	Wireless Metropolitan Area Network

Chapter 1

Introduction

The Low-Power Wide Area Networks (LPWAN) technologies have been increasingly researched and deployed as a promising solution for serving Internet of Things (IoT) applications. Long-Range (LoRa) technology via its Long-Range Wide Area Network (LoRaWAN) protocol [1], has shown a very attractive platform due to its low energy consumption and wide area coverage. However, one main drawback associated with LoRaWAN is the vulnerability to a high packet collision rate. This is due to the adoption of ALOHA communication protocol, where LoRa nodes initiate packet transmissions without the presence of Listen Before Transmit (LBT) protocol [2, 3].

As a result, LoRaWAN efficiency suffers a depreciation, particularly on network's energy consumption and transmission delay. In order to compensate for the absence of LBT protocol, LoRaWAN provides different Spreading Factors (SF) based on the LoRa physical layer Chirp Spread Spectrum (CSS) technique to allow simultaneous packet transmissions. Alternating between different *SF* comes at the expense of higher transmission power and time-on-air, which can be an ideal solution for small-scale networks [4, 5, 6, 7]. However, adopting LoRaWAN to serve dense applications remains an open challenge.

1.1 Research Motivation

LoRa physical layer modulations relies on the *CSS* technique [8], which quantifies how many chirps are pulsed per second. Using *CSS* technique, LoRa provides a wide area communication coverage for a range of more than 10 *km*. In addition, using *CSS* increases the robustness against noise and external interferences. The Medium Access Control (MAC) LoRaWAN protocol exploits the *CSS* by providing the *SF* feature to further boost the communication efficiency. The transmissions using different *SF*, between *SF*7 and *SF*12, vary in terms of data per chirp per second [1]. This allows the receiver to distinguish between simultaneous transmissions according to the used *SF* [9]. The packet transmission delay is the duration of transmitting a packet from the sender to the receiver. In an ideal environment the packet transmission delay is mainly effected by the *SF*, the transmission power and the packet size [10]. Given a typical IoT packet size of 50-300 bytes, an IoT battery-powered device using LoRaWAN has an expected lifetime of up to 6 years [11], provided there are infrequent daily transmissions [12, 13]. This explains the wide interest of adopting LoRaWAN in IoT applications [14, 15, 16]. However, as formerly mentioned the LoRaWAN efficiency is still an open challenge especially in dense applications.

This led to the strive of a number of research bodies and industrial organisations to challenge the efficiency of LoRaWAN. For example, Rachkidy *et al.* [17] proposed a collision resolution technique that allows LoRa gateway to decode the overlapped packet signals and hence, corrupted received packets. While Liao *et al.* [18] introduced a multi-hop based concurrent transmission technique in order to mitigate the probability of simultaneous packets transmissions of LoRa nodes. In addition, Zhu *et al.* [19] proposed a tree based clustering algorithm to enhance LoRaWAN capacity. Their scheme particularly exploits the variety in *SF* communication reliability by allocating different *SF* to different clusters. Based on the *SF* allocation, clusters with less *SF* reliability off-load traffic to clusters with higher *SF* reliability via multi-hop relay. Although, the collision rate has been enhanced in the aforementioned schemes [17, 18, 19], this

comes at the expense of compromising the transmission delay or the energy consumption. LoRaWAN and other LPWAN technologies are solutions for IoT applications which usually have limited resources. Since LoRaWAN adopts ALOHA protocol, the number of packet collisions increase proportionally to the number of nodes within the network. This introduces packet retransmission attempts from the nodes side, which in return increases the transmission delay and energy consumption. Hence, in this thesis the main focus is to reduce the packet collision rate in LoRaWAN and maintain relatively low transmission delay and energy consumption.

1.2 Research Scope and Objectives

The scope of the thesis is to address the issue of high packet collision rate when using LoRaWAN to serve dense and ultra-dense applications. The main challenges are the excessive packet collision rate, inefficient Total Transmission Delay (TTD), Total Energy Consumption (TEC) and Packet Delivery Rate (PDR). In order to efficiently implement LoRaWAN it is necessary to define the target application. This thesis considers using LoRaWAN as a wireless communication solution for serving an early warning weather monitoring system. Given the unlabelled data delivered by the nodes (sensors), the limited resources for the IoT devices (battery-powered) and the random transmission behaviour of LoRaWAN due to adopting ALOHA protocol, the objectives of this thesis are:

- Develop a new algorithm using unsupervised learning clustering algorithm (K-Means) to reduce the collision rate. This is achieved by partitioning LoRaWAN nodes into different clusters based on the the unlabelled data transmitted from the nodes to the gateway. K-Means is particularly chosen to be applied at the gateway level. This is to lift the computational burdens from the nodes level.

- Develop a dynamic packet transmission Priority Scheduling Technique (PST) based on supervised learning to preserve the energy consumption. The dynamic transmis-

sion *PST* enhances the gateway's decision making process in favour of minimising the energy consumption in the network. This is achieved by classifying different clusters with different transmission priorities. Based on which, only the nodes that are located in clusters classified as high transmission priority enjoy better transmission capabilities. While the nodes that are located in lower transmission priority clusters only have the necessary transmission capabilities. Although the mechanism of the dynamic transmission *PST* focuses on preserving the energy consumption, it offers a fair trades-off between *PDR*, *TTD* and *TEC*.

- Develop a deep learning based prediction scheme for collision avoidance. The packet collision prediction scheme is particularly aimed at enhancing LoRaWAN's reliability in terms of *PDR* when serving ultra-dense networks where the number of nodes are scaled up to 5000 in a limited area. In particular, the gateway implements a hybrid procedure, in which the nodes operate on one of LoRaWAN transmission classes *A* or *C*[1]. This allows better control over the actively transmitting cluster in favour of ensuring higher levels of *PDR* at ultra-dense application.

1.3 Contributions

The main contributions of this thesis are summarised as follows:

1. The reduction of the excessive packet collision rate associated with LoRaWAN. This is achieved via classifying the unpredicted transmissions' nature of LoRaWAN into an organised manner that allows better resource management.

2. A dynamic transmission *PST* based on supervised learning to enhance the network's transmission delay and energy consumption while maintaining relatively acceptable levels of the packet delivery rate. This is performed in two folds:

[1]Note that LoRaWAN provides three different classes for the end-device to join the network. First is class *A*, which is the most energy efficient, where the nodes initiate transmissions without prior sensing to the channel status and open a temporary receive window following each transmission. Second is class *B*, where nodes listen to periodic beacons from the gateway. Third is class *C*, which is the most energy inefficient, where nodes listen continuously to the gateway. More details of LoRaWAN end-device classes are discussed in chapter 2 and published in [20].

(a) First, it allocates a unique transmission priority to each of the clusters in the network ranging from high to low. Based on the cluster's priority, the corresponding nodes are assigned specific transmission intervals by the gateway.

(b) Second, it provides two transmission modes specifically designed to prevent excessive energy consumption that is caused by unnecessary transmission attempts. This is followed by a decision making process using the Naive Bayes classifier algorithm in order to decide the best transmission mode for each cluster based on the cluster's transmission needs.

3. A Collision-Aware Priority Scheduling Technique (CA-PST) aimed at enhancing the packet delivery rate in ultra-dense networks. This technique exploits the Long Short-Term Memory (LSTM) deep learning model to predict the number of packet collisions. Based on the collision predictions, the gateway using the *CA-PST* instructs the nodes that are located in high transmission priority clusters to switch from LoRaWAN transmission class *A* to class *C*. In return, the corresponding nodes follow specific transmission intervals to avoid packet collisions.

1.4 Authour's Publications

Journal Papers

1. M. Alenezi, K. K. Chai., Y. Chen. and S. Jimaa, "Ultra-dense LoRaWAN: Reviews and challenges," in *IET Communications*, vol. 14, pp. 1361-1371, doi: 10.1049/iet-com.2018.6128

2. M. Alenezi, K. K. Chai, A. S. Alam, Y. Chen and S. Jimaa, "Unsupervised Learning Clustering and Dynamic Transmission Scheduling for Efficient Dense LoRaWAN Networks," in *IEEE Access*, vol. 8, pp. 191495-191509, 2020, doi: 10.1109/ACCESS.2020.3031974.

3. M. Alenezi, K. K. Chai, A. S. Alam, Y. Chen and S. Jimaa, "Collision Avoidance

for Ultra-Dense LoRaWAN Based on Deep Learning," in *IEEE Internet of Things Journal*, (Submitted).

Conference Papers

1. M. Alenezi, K. K. Chai, S. Jimaa and Y. Chen, "Use of Unsupervised Learning Clustering Algorithm to Reduce Collisions and Delay within LoRa System for Dense Applications," 2019 *International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2019, pp. 1-5, doi: 10.1109/WiMOB.2019.8923515.

1.5 Thesis Structure

Chapter 2 provides the fundamental details of LoRa technology and LoRaWAN protocol. It also presents a set of papers addressing a number of issues in LoRaWAN networks. Moreover, it explores a set of studies that have implemented various machine learning techniques in LoRaWAN.

Chapter 3 introduces the proposition of the unsupervised learning clustering algorithm for reducing the collision rate in LoRaWAN.

Chapter 4 covers the proposed dynamic transmission *PST* based on clustering and supervised learning for preserving energy within LoRaWAN networks.

Chapter 5 reveals the proposed deep learning based Collision Aware transmission Priority Scheduling Technique *CA-PST* to enhance *PDR* levels in ultra-dense LoRaWAN networks.

Chapter 6 concludes this thesis and gives an insight to the future work that can potentially be carried out based on the work done in this thesis.

Chapter 2

Background and State-of-the-Art

2.1 LoRa Overview

LoRa is a long-range wireless communication technology proposed by LoRa Alliance and developed by Semtech [1]. It resides in Wireless Metropolitan Area Network (WMAN) using LPWAN protocol as shown in Figure 2.1. LoRa is aimed at providing wireless network services to resource-limited devices (battery-powered), with low data rate (up to 50 *kbps*) and long distance (up to 10 *km*).

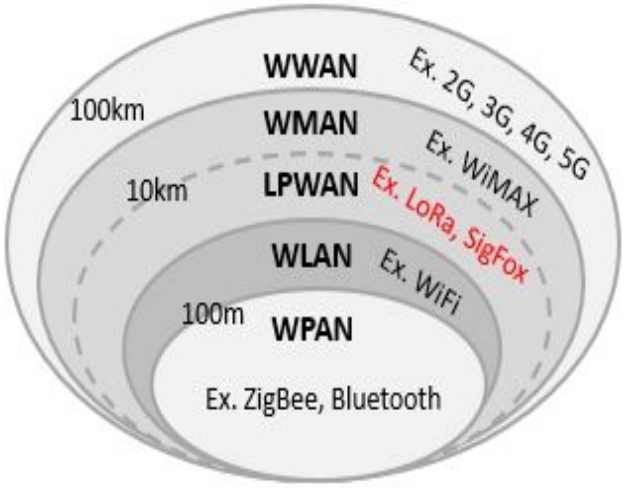


Figure 2.1: LPWAN vs. Other Wireless Protocols Coverage Range

Industrial, Scientific and Medical (ISM) bands of the world are shown in Figure 2.2. The band over North America is 915 MHz while over Europe is 868 MHz [21]. LPWAN technologies can be approved if it is capable of providing the allowed band in a specific geographical area [22].

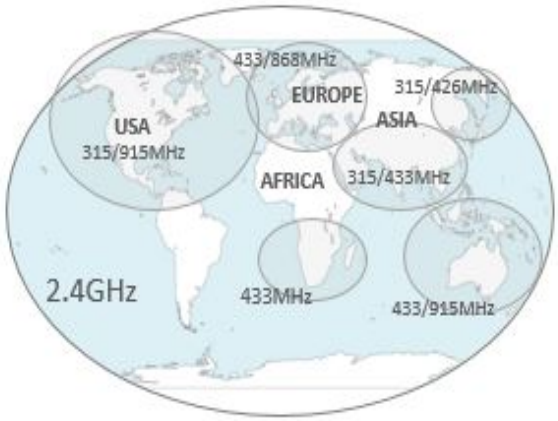


Figure 2.2: World ISM Bands

Since sensors can be served via a number of different wireless protocols, it is necessary to compare the pros and cons of different wireless protocols especially in terms of coverage, data rates, energy efficiency and cost of implementation. Table 2-A provides a comparison of technologies that are able to provide network connectivity to a set of sensors. Table 2-A is based on long range, low data rate, low power and cost. It indicates that "LPWAN technology is suited for connecting devices that need to send small amounts of data over a long range, while maintaining long battery life" [23].

Within LPWAN there are three technologies that use different modulation techniques. These are SigFox [24], LoRa [1] and NB-IoT [25]. The modulation techniques of SigFox and LoRa are proprietary while NB-IoT is 3GPP [26, 27, 28]. The list of the three products are displayed in Table 2-B.

IoT applications could be categorised as fixed or mobile [29, 30]. Examples of fixed IoT applications are street lights and farming. Examples of mobile IoT are vehicles and cattle. For fixed application SigFox, LoRaWAN and NB-IoT are able to operate good. However,

Table 2-A: Comparisons of Existing Wireless Protocols

-	Bluetooth	BLE	ZigBee	WiFi	Cellular M2M	LPWAN
Long Range (>10km)	×	×	×	×	✓	✓
Low Data Rate (<5k bit/s or 20 to 256 bytes per message)	✓	✓	✓	×	×	✓
Low Power (to last 5 to 10 years on a single battery)	✓	✓	✓	×	×	✓
Low Cost	✓	✓	✓	✓	×	✓

Table 2-B: Comparisons of Existing LPWAN Protocols

	SigFox	LoRaWAN	NB-IoT
Mobility	Poor	Good	Poor
Availability	Europe	Worldwide	Worldwide
Cost Radio Modules	<$5	<$10	<$12
Uplink/Downlink	Uplink	Both	Both

for fixed applications LoRaWAN has one interesting feature, which is the Adaptive Data Rate (ADR) in the physical layer [31, 32, 33]. *ADR* maximises battery life, range and network capacity for each end-device. Section 2.1.4 provides in-depth details of *ADR* mechanism. SigFox and NB-IoT do not have this feature.

SigFox operates on EU 868 frequency band [34] hence, the unpopularity in the USA market. LoRaWAN and NB-IoT provide uplink and downlink transmissions, while SigFox provides only one-way communication (uplink). Therefore, SigFox cost is the lowest. Table 2-B shows that LoRaWAN is the best amongst other protocols in reference to mobility, worldwide availability, radio module cost and uplink/downlink capabilities [35].

LoRa network protocol is shown in Figure 2.3 as depicted from [1]. The Sensor, which is a processor based device, transmits and receives, among other details, the sensing values to LoRa gateway via the antenna physical layer. Similarly, the Gateway Host receives and transmits to the sensors via the antenna physical layer. The last block of Figure 2.3 is the Network Server, which routes backward and forward messages from sensors to

a specific application. In addition, the Gateway Host could be interfaced to a Network Server via Ethernet, 3G or WiFi protocols.

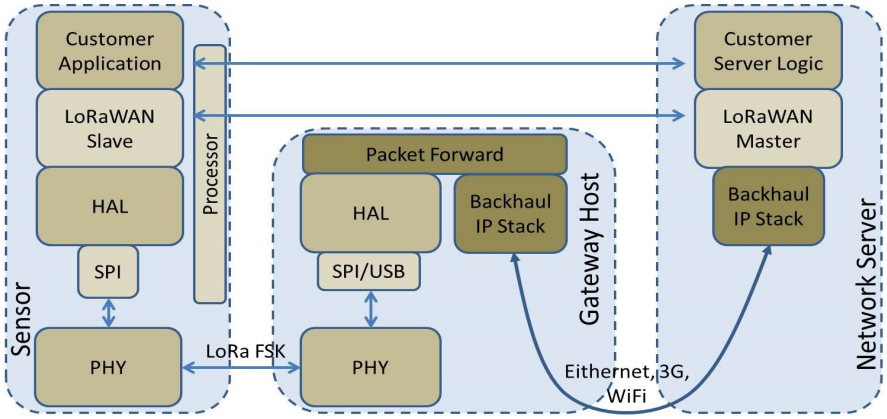


Figure 2.3: LoRa Network Protocol - LoRa Gateway Host, Sensor and Network Server Block Diagram [1]

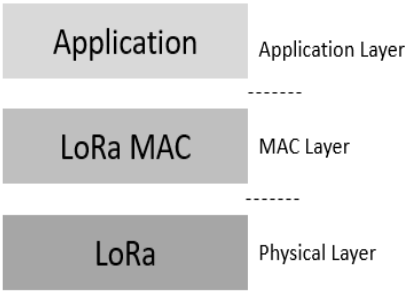


Figure 2.4: LoRa Three Layers

As shown in Figure 2.4, LoRa consists of three layers [1]. The first layer is the physical layer or modulation layer based on Chirp Spread Spectrum (CSS) technique. In addition, the physical layer offers four regional ISM bands, including EU and US allowed bands. The SX1276/77/78/79 chips transceivers, produced by Semtech, form the physical layer for LoRa. The second layer is the Media Access Control (MAC) layer protocol (LoRaWAN) with specific access network architecture. MAC protocol manages uplink and downlink between gateways and sensors. LoRaWAN provides three different classes; class A, class B and class C devices. Class A, is the lowest power sensor system for applications where sensors are pure ALOHA based. Class B, where the sensor receives a time synchronised beacon from the gateway. Class C is continuous listening where

sensor use more power to operate but it offers the lowest latency for server to sensor communications. The third layer is the application layer.

LoRa adopts star topology, where a set of LoRa sensors are connected wireless to LoRa gateway. Sensors communicate in their time or frequency slot with LoRa gateway. The radio link between LoRa gateway and LoRa sensors can be very long which leads to less battery life. However, LoRa sensors are able to rest between message transmissions, which means LoRa sensors consume less energy. Star topology is mainly aimed for LPWAN. It is also widely used in WiFi and mobile (cellular) networks.

LoRaWAN supports both uplink and downlink messaging. A message from a sensor to a gateway is referred to as an uplink message, while a message from a gateway to a sensor is called a downlink message.

LoRa deploys its own modified Chirp Spread Spectrum (CSS) modulation. It also deploys a Frequency-Shift Keying (FSK) modulation with higher data rate (up to 50 kbps) [10, 36].

Semtech [37] has produced number of digital baseband chips for outdoor and indoor LoRaWAN gateways. LoRaWAN is well suited to cover connectivity over, for example, a large building using star network topology. However, the market is geared towards using LoRa for applications that deploy over thousands of nodes in a limited area to be connected to a gateway. Hence, the terms Dense Networks where the number of end-devices scale up to 1000 and Ultra-Dense Networks where the number of end-devices exceed 1000, in a limited geographical area (km^2) [38, 39, 40].

2.1.1 LoRa Physical Layer

For the ease of understanding the design, programming and simulation/modelling of LoRa technology, terminologies such as Chip, Symbol, Chirp, Spreading Factor (SF), Coding Rate (CR) and Data Rate (DR) are described in this section.

2.1.1.1 Chips, Symbols, Chirps and Spreading Factor

A Chip is a pulse that sweeps from f_{Low} ($\frac{-BW}{2}$) to f_{High} ($\frac{+BW}{2}$), where BW is Bandwidth. A number of Chips forms a Symbol. For example, assume a Symbol value is between 1 and 128, this number would be one of the combinations of $2^7 = 128$ chips. Figure 2.5 illustrates an example of symbol values within 128 chips.

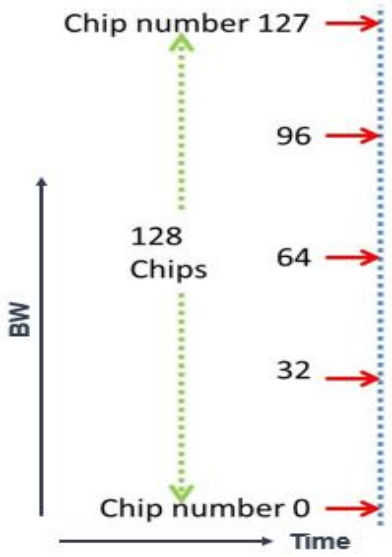


Figure 2.5: An Example of Symbol Values Within 128 Chips

The Chip rate (R_c) and Symbol rate (R_s) are expressed in Equations (2.1) and (2.2) below.

$$R_c = ChipRate = Chips/Second \tag{2.1}$$

$$R_s = SymbolRate = Symbols/Second \tag{2.2}$$

The Chip is equivalent to one pulse of the Bandwidth (BW). Hence, the following Equation (2.3).

$$R_c = \frac{1}{BW} \tag{2.3}$$

There are 2^{SF} combinations of symbols that can be transmitted over the BW, where $SF = \{7, 8, 9, 10, 11, 12\}$. Therefore, the R_s from (2.2) can be expressed in terms of BW

as shown in Equation (2.4).

$$R_s = \frac{BW}{2^{SF}} \tag{2.4}$$

To estimate the processing gain of R_c and R_s, the term Spreading Factor (SF) was emerged. LoRa has been designed for Symbols with group of sizes N of chips $N \in \{128, 256, 512, 1024, 2048, 4096\}$. Equation (2.5) shows SF in terms of N.

$$SF = \log_2(N) \tag{2.5}$$

To gain the highest throughput with the lowest power consumption but for short distances, SF should be set to 7. While at $SF12$, the distance is at the maximum but the data rate is at the lowest [1].

LoRa modulation deploys *CSS*. In *CSS* technique, first developed for radar applications in the 1940's [41, 42], the frequency of the generated Chirps varies linearly with time to provide low cost, low power and resilience to interference based solution. So Chirp (sweep) is a signal of continuously increasing frequency (up-chirp), a ramp from frequency minimum $f_{min} = 0kHz$ to frequency maximum $f_{max} = 127kHz$ as shown in Figure 2.6, or continuously decreasing frequency (down-chirp) $f_{max} = 127kHz$ to $f_{min} = 0kHz$ as shown in Figure 2.7.

Figure 2.8 is an example of cyclically-shifted up-chirp data Symbol 96 which requires spreading factor of 7, as it is one of the combinations of 2^7.

In the Symbol example in Figure 2.8, $\{96\ Decimal = 1100000\ Binary\}$ is a modulated data. The number of raw bits that can be encoded by this symbol is 7, which means $SF = 7$. The sweep signal is divided in two $SF = 2^7 = 128$ chips. The Symbol starts from chip 96 and ends with chip 128 and cyclically-shifted from chip 0 to chip 95.

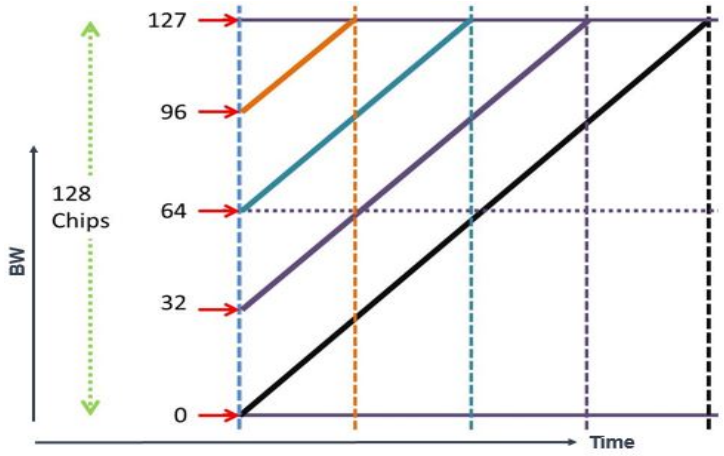


Figure 2.6: Up-Chirp with $SF = 7$ (The symbol is one of the combinations of $2^7 = 128$)

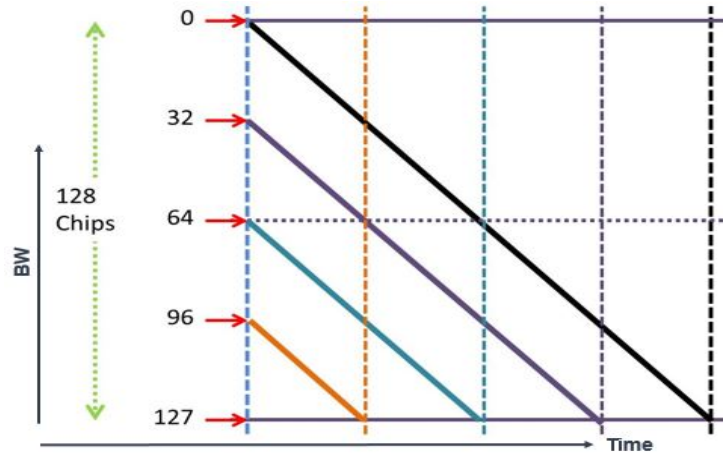


Figure 2.7: Down-Chirp with $SF = 7$ (The symbol is one of the combinations of $2^7 = 128$)

2.1.1.2 Coding Rate and Data Rate

Some of the data bits are lost due to interference. Error correction bits recover the original lost data bits. LoRa uses Forward Error Correction (FEC) scheme to avoid a costly re-transmission. *FEC* requires error correction bits (redundant bits) to be added to the data. Although *FEC* reduces data throughput, it also increases the sensitivity of the receiver [43, 44].

LoRa defined a set of values which are referred to as $Code \in \{1, 2, 3, 4\}$ to calculate the Coding Rate (CR) based on Equation (2.6).

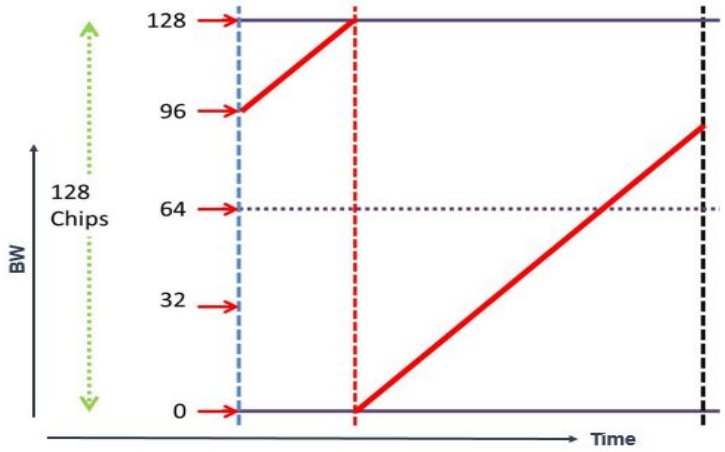


Figure 2.8: An Example of Cyclically-Shifted Up-Chirp Data Symbol

$$CR = \frac{4}{4 + Code} \tag{2.6}$$

Hence $CR = \{4/5, 4/6, 4/7, 4/8\}$. So for $SF7$, Figure 2.9 shows the redundant bits relative to the data bits.

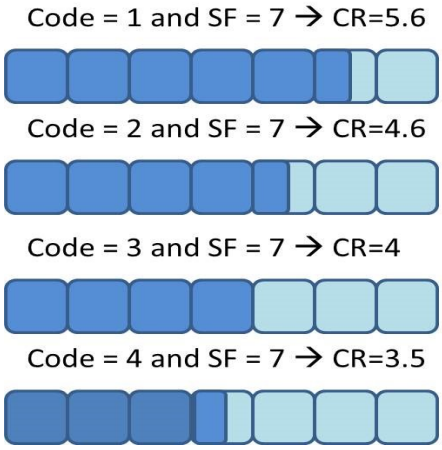


Figure 2.9: Coding Rate Example for $SF7$

CR could maximise the data rate if less code bits are used. However, more redundant bits sent to the receiver, LoRa will consume more power [45]. DR is given as in Equation (2.7).

$$DR = SF \times \frac{R_c}{2^{SF}} \times \frac{4}{4 + CR} \tag{2.7}$$

The values of DR for bandwidths from 125 kHz to 500 kHz with $SF7$ to $SF12$ are discussed in details in LoRa Patent [46].

2.1.2 LoRa Packet Structure

The LoRa modem employs two types of packet formats, explicit and implicit modes. It also comprises of three elements; preamble, optional header and data payload. Figures 2.10 and 2.11 show the explicit and implicit header modes respectively. Both modes are discussed in details in [45].

Figure 2.10: Explicit Header Mode

Figure 2.11: Implicit Header Mode

If the payload, coding rate and Cyclic Redundancy Check (CRC) presence are fixed or known in advance, then the implicit mode could be deployed. This mode reduces the transmission time. Figure 2.11 shows the implicit header mode. More in-depth details on this mode can be found in [45].

2.1.3 LoRa Packet Time on Air (ToA)

As shown in Figure 2.12, Time on Air (ToA) is a unit to measure the transmission time of LoRa packet. A LoRa packet consists of preamble and payload symbols. Hence, the packet's ToA (T_{packet}) is the sum of preamble duration ($T_{preamble}$) and payload duration ($T_{payload}$) as shown in Figure 2.13.

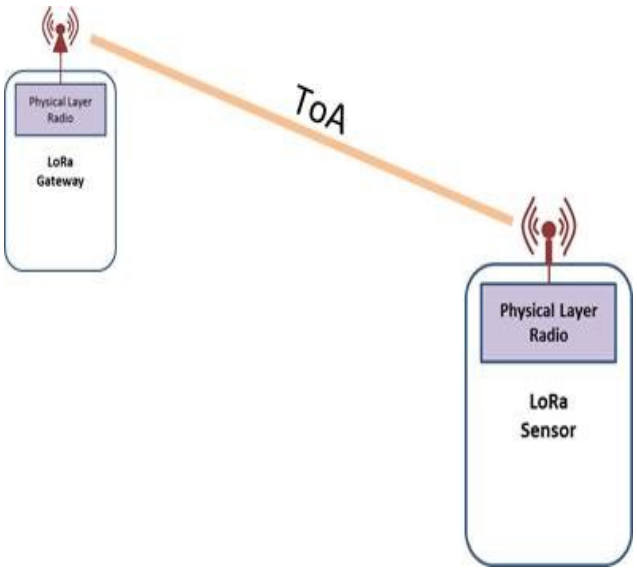


Figure 2.12: LoRa Packet Time on Air

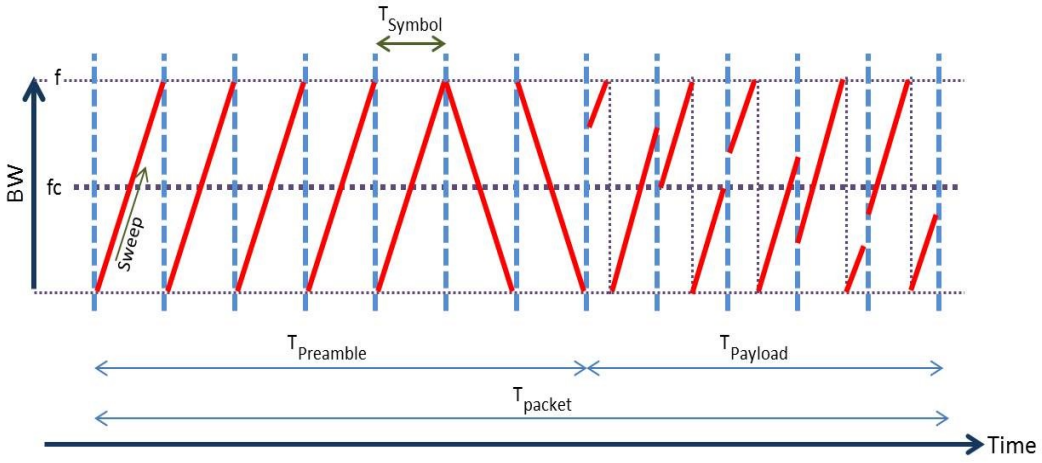


Figure 2.13: Total Packet ToA (T_{packet}) is the Sum of Preamble Duration ($T_{preamble}$) and Payload Duration ($T_{payload}$)

$T_{preamble}$ is a function of T_s and by using Equation (2.2), the symbol period can be defined as shown in Equation (2.8)

$$T_s = 1/R_s \tag{2.8}$$

From [45], $T_{preamble}$ is given by Equation (2.9).

$$T_{preamble} = (n_{preanble} + 4.25) \times T_s \tag{2.9}$$

The length of preamble ($n_{preamble}$) is programmable. The data sheet in [45] describes two registers; *RegPreambleMsb* and *RegPreambleLsb*, and shows that their functions in LoRa mode is dedicated to store the length of preamble. The 4.25 in Equation 2.9 is the number of symbols added to accommodate for the variance in the payload lengths in the explicit and implicit modes.

Again from [45], $T_{payload}$ is extracted as shown in Equation (2.10)

$$T_{payload} = n_{payload} \times T_s \tag{2.10}$$

From the same reference [45], $n_{payload}$ can be calculated using Equation (2.11)

$$n_{payload} = 8 + max\left(\frac{8PL - 4SF + 28 + 16CRC - 20IH}{4(SF - DE)} \times (CR + 4), 0\right) \tag{2.11}$$

where 8 is the default number of symbols, PL is the number of bytes of payload and SF is the spreading factor. IH is the implicit header mode when (0) or explicit header mode when (1). DE is Low data rate, disabled when (0) or enabled when (1). CRC is not present in Payload when (0) or CRC is present in Payload when (1).

CR is the programmed coding rate from 1 to 4. Therefore, the total packet time on air T_{packet} is given by Equation (2.12).

$$T_{packet} = T_{preamble} + T_{payload} \tag{2.12}$$

2.1.4 Adaptive Data Rate

Adaptive Data Rate (ADR) is an interesting feature and crucial for the IoT infrastructure. It allows for high network performance and enforces scalability. With *ADR*, LoRa network is able to maximise battery life, range and network capacity for each end-device. "LoRa network allows the end-devices to individually use any of the possible data rates and *TxPower*. This feature is used by the LoRaWAN to adapt and optimize the data rate and *TxPower* of static end-devices." [1].

The *ADR* mechanism involves a set of LoRa commands and parameters. *ADR* commands are *LinkADRReq* and *LinkADRAns*. ADR parameters are *ADRParamSetupReq* and *ADRParamSetupAns*. These commands and parameters are embedded in LoRa frame format as shown in Figure 2.14.

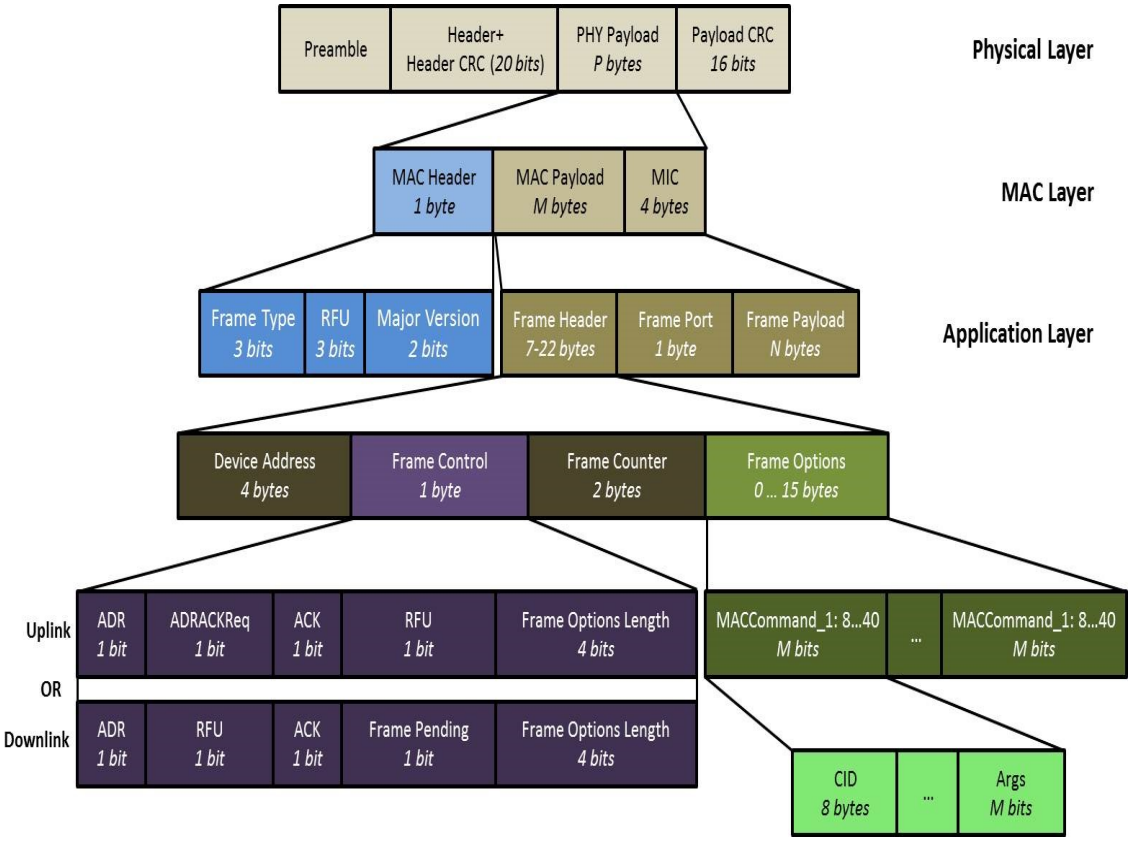


Figure 2.14: LoRa Frame Format

Table 2-C: 6 out of 30 MAC Commands – The Last Two Commands Set ADR [1]

CID	Command	Transmitted by		Short Description
		End-Device	Gateway	
0x01	*ResetInd*	×		Used by an ABP device to indicate a reset to the network and negotiate protocol version
0x01	*ResetConf*		×	Acknowledges ResetInd command
0x02	*LinkCheckReq*	×		Used by an end-device to validate its connectivity to a network
0x02	*LinkCheckAns*		×	Answer to LinkCheckReq command. Contains the received signal power estimation indicating to the end-device the quality of reception (link margin)
0x03	*LinkADRReq*		×	Requests the end-device to change data rate, transmit power, repetition rate or channel
0x03	*LinkADRAns*	×		Acknowledges the LinkADRReq

Figure 2.14 shows that LoRa protocol consists of layers to include Physical layer, MAC layer and Application layer. *LinkADReq* and *LinkADRAns* are MAC commands as specified by the LoRaWAN specifications [1]. However, each *Req/Ans* has the same Command Identifier (CID). They differ if the message is uplink or downlink as seen in Table 2-C. Therefore, one of them should be used at one time. *FOpts.* shown in Figure 2.14, is used to piggyback MAC commands on a data message. *Args* are the optional arguments of the command.

For a better explanation, ADR procedure is illustrated as shown in Figure 2.15. The network server (via *LinkADRReq* command) requests an end-device to perform a rate

adaptation.

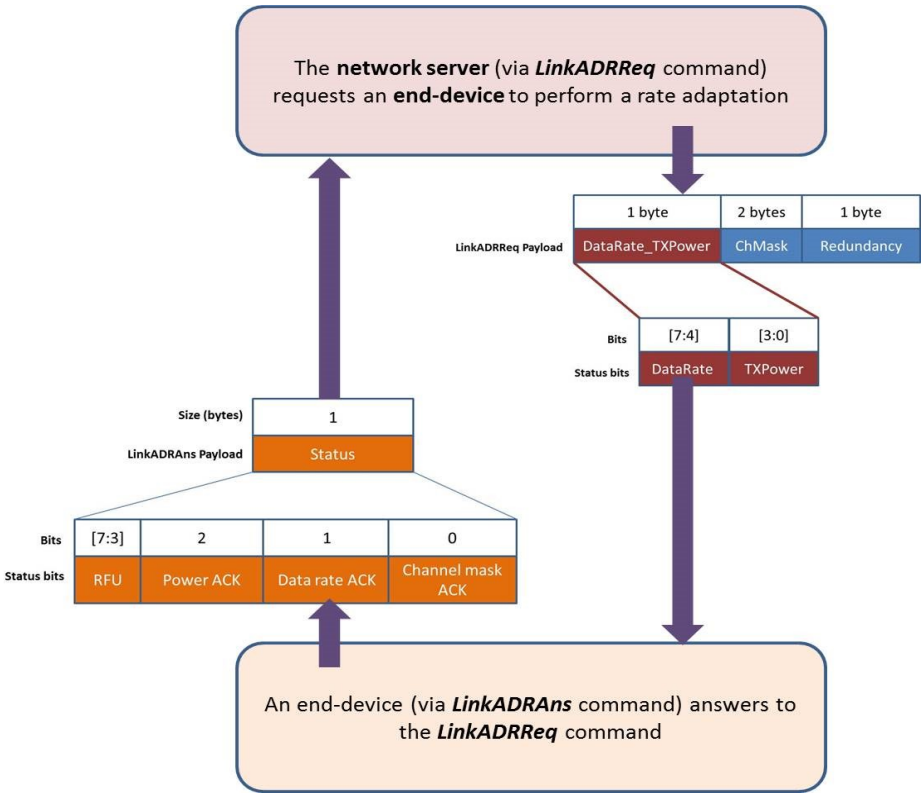


Figure 2.15: ADR Procedure

From Figure 2.15, an end-device (via *LinkADRAns* command) answers to the *LinkADRReq* command with acknowledgment. Bits 1 and 2 for *LinkADRAns* indicate if *TxPower* and *DataRate* have been set or not. Therefore, when the bits 1 and 2 set to (0), the command is discarded and the end-device state is not changed. When set to (1), the data rate and *TxPower* were successfully set.

Note that the *LinkADRReq* is used for two purposes. First is for ADR (requesting data-rate and *TxPower*). Second is for channel-reconfiguration.

Algorithm 1 shows ADR algorithm with a procedure to calculate the margin by the network server if an end-device requested ADR. If the ADR algorithm is executed, then the network server will be able to enhance the data rate and power consumption of the sending device by adopting the most efficient data rate.

Algorithm 1 ADR Algorithm at Pseudo Code Level

1: Check ADR flag in uplink message
2: **if** ADR flag is set to 1 **then**
3: NS collects recent uplink messages to calculate data rate
4: NS calculates margin
$margin = SNR_{measured} - SNR_{limit} - MARGIN_{default}$
5: **else**
ADR flag is not set
6: **end if**
7: **if** calculated margin is large (± 10) **then**
8: NS calculates new margin value based on optimised data rate
9: **else**
use received data rate
10: **end if**

where NS is the network server and SNR is the Signal to Noise Ratio. The margin is the range of the expected SNR value on which assigning a new ADR is based.

2.2 State-of-the-Art

This section reviews a set of papers addressing issues related to the capacity, scalability, reliability, transmission delay, coverage, interference, energy efficiency and packet collisions in LoRaWAN networks. It also explores the use of various machine learning techniques, all for the purpose of enhancing LoRaWAN performance. The section concludes with identifying the technical concerns that are bridged by the proposed chapters of this work.

LoRa end-devices use LoRaWAN class A, class B and class C protocol. Analysing these classes reveals capacity and scalability issues as described in Section 2.2.1. In Section 2.2.2 a mathematical model has been reported to measure throughput to determine the reliability of packets transmission within LoRaWAN. Section 2.2.3 reviews a set of studies that address the impact of adopting ADR technique and LoRa's different modulation classes on the transmission delay. In Section 2.2.4 a set of papers tackling issues related to the energy consumption in LoRaWAN and wireless sensors networks are reviewed. Although the CSS offers high immunity to interference, LoRaWAN is

known to have packet collision issues especially in dense networks. Therefore, Section 2.2.5 explores a set of studies on the deployment of LoRaWAN in urban and rural areas, and reviews the collision behaviour. Section 2.2.6 reviews a set of research efforts that were focused on incorporating the various machine learning algorithms for enhancing LoRaWAN performance. This is concluded in Section 2.3 by shedding light on how the work carried out in this thesis bridges the research gaps identified based on the reviewed studies.

2.2.1 Capacity and Scalability

Adelantado et al. [10] carried out analysis of LoRaWAN class B end-devices. Reliability is achieved by acknowledgment (ACK) received for data frames received. In LoRaWAN specifications [1] transmitting data frames takes place over one of the two receiving windows ($RX1$ and $RX2$). This means a delay as result of the time-off period following each transmission receive window. Such behaviour in LoRaWAN network raises the question of how feasible class A and class B end-devices are for ultra-reliable services using LoRaWAN. Augustin and *et al.* [47] deployed a technique to avoid the delay happening as a result of waiting for the receiving windows by not sending packets more than the smallest maximum payload size (which was 36 bytes according to their simulation parameters). However, such a solution has a severe impact on the capacity, resulting in lower throughput and higher Time-on-Air (ToA). Of course class C end-devices provides a constant listing behaviour but at the expense of very high power consumption.

Mikhaylov [48] carried out analysis on the capacity and scalability of LoRaWAN network. Referring to a scenario with an ideal low transmission traffic model (8 bytes packet a day), LoRaWAN gateway using 3×125 kHz channels can serve thousands of end-devices. Of course the number end-devices is vastly reduced in a much denser traffic model. For example, increasing the traffic model to a 1 byte packet every 30 *seconds*, the gateway can only serve up to 357 end-devices. End-devices with a shorter distance to the gateway (up to 2.4 km using $DR5$) have more data transmission rate (2 $kbit/s$), while further

end-devices have lower transmission rate (100 *bit/s*). Note that LoRaWAN adheres to end-device duty cycle restrictions (EU regulations [49]). More LoRaWAN scalability scenarios and numeric figures are available in [50].

2.2.2 Reliability

Within LoRaWAN, a packet transmission has a serious drawback to the technology. In regards to transmission drawback, Sørensen *et al.* [51] proposed analytical models that estimate the impact of offered loads on packet error rate. Their models evaluate and estimate the maximum throughput and maximum loads for reliable packets transmission within LoRaWAN.

Augustin and *et al.* [47] carried out a study of LoRaWAN for IoT. In their study, LoRa's physical and data link layer performance was evaluated by field tests and simulations. From the perspective of a single device maximal throughput, they conducted a test with six 125 *KHz* channels using *SF* of 7-12. Considering a size of 13 bytes MAC header, 51 bytes packet size was the maximum payload allowed to be transmitted between the end-device and the network server. The diagnoses revealed the receive windows as limiting factors as the device following the initial transmission has to wait for the two downlink receive windows to be done before attempting to send another packet. Thus, a limitation to scaled-up applications with a large number of devices that send data on a regular traffic basis (couple of times a day). The proposed solution to the aforementioned limitation is to avoid sending more than the smallest maximum payload size (36 bytes in their simulation). However, such a solution has a severe impact on LoRaWAN capacity and results in lower throughput.

2.2.3 Transmission Delay

In line with the clustering based routing algorithms, Liu and Chang [52] proposed a clustering rerouting scheme based on the assumption that nodes are scattered unevenly. Their assumption is reflected on their clustering being performed based on an unequal

number of nodes in each cluster. They utilise a probabilistic model in order to determine which node has the energy capability to perform the tasks of multi-hoping for other nodes in need within the same cluster. Therefore, the nodes within the cluster has the ability to communicate with the gateway through the the cluster head. The node acting as a cluster head has the ability to opt out from serving as a cluster head when the energy level reaches a specified threshold. Although the proposed scheme enhanced the transmission delay in comparison to similar approaches reported in [53, 54, 55], it relies on a decent level of energy at the nodes level in limited-resources environment. The assumption of energy availability in an IoT environment (battery-powered) could result in a network failure.

As the functionality of LoRaWAN is highly dependent on resource allocation the technology uses an Adaptive Data Rate (ADR) technique. In practice, *ADR* aims to achieve right first time reception between end-device and the gateway through basic minimum *SF* selection. However, Cuomo *et al.* [56] recognised two sophisticated *SF* allocation algorithms, *EXPLoRa-SF* and *EXPLoRa-TA*. The algorithms show a reduction in interference between clusters of end-devices with varying *SF* through improved time-on-air. More specifically, *EXPLoRa-SF* attempts to equally assign redundantly high *SF* groups across multiple base stations that are restricted solely by their Received Signal Strength Indicator (RSSI). Although high *SF* provide long-range coverage, they increase interference and collisions through greater time-on-air. Hence, *EXPLoRa-TA* works by assigning different *SF* to end-device groups to ensure each group has an equal amount of time-on-air. They coined the term "ordered waterfiling". It was observed that both algorithms prevailed over *ADR* at improving throughput in highly loaded systems of end-devices distributed 200 meters from the gateway.

LoRaWAN has three classes of communication, class *A*, *B* and *C*, listed in descending order of energy consumption. Delobel and *et al.* [57] selected class *B* to study the energy efficiency of downlink communication (performance) as it is optimised for this purpose. The downlink communication is confirmed through an acknowledgment (ACK)

mechanism. Failure to receive ACK will trigger a retransmission, which accumulates delays. The expected delay time is analytically computed in their proposed Markov chain model. However, it exposes further flaws within the application of class B. The limitations include; the gateway duty-cycle, conflict between classes A & B, and delay before ACK sub-band availability.

The first limitation of duty-cycle is apparent to Delobel and *et al.* [57], where the gateway is prevented from sending ACK for a large number of confirmed uplinks, for which it has delays of up to 98.13s before the use of the next ping slot could be seen. Nonetheless, they assumed all data frames could be acknowledged by gateways in which all ping slots could be used.

The second limitation in LoRaWAN specifications is the conflict between class A and B. Since class A devices transmissions are random, Delobel and *et al.* [57] prevented other class B devices from transmissions during designated ping slots from the gateway (beacons). By adopting this approach, their scheme was using Markov chain based model, which increased the data-rate and reduced time-on-air frames. Moreover, the delay time was further improved by increasing the number of sub-bands together with increasing the ping period, which in return allow more frame transmissions and less delays.

2.2.4 Energy Efficiency

Energy efficiency has always been within the interest scope of researchers, especially with technologies designed for serving IoT applications. In this regard, Kavitha and Suseendran in [58] proposed a priority based adaptive scheduling algorithm for IoT sensor systems where several performance aspects were taken into consideration. One main issue that the proposed algorithm tackles is the energy consumption in wireless sensor networks. The scheduling algorithm is based on preset delay and energy requirements. Based on these requirements a given packet can only transmit when there is a free slot for transmission. In particular, the algorithm introduces a queuing procedure where packets

queue before initiating transmissions. This procedure is mainly utilised to reduce the amount of transmissions and hence reduce the total energy consumption of the network. Their work is also inspired by similar techniques presented in [59, 60, 61, 62].

Rubel and *et al.* [63] proposed a clustering based priority management scheme to reduce the overall energy consumption in a wireless sensor network. In specific the scheme classifies data received from nodes in different delay requirements. Based on each classification the scheme allows sensors to initiate communication with the base station. Their scheme trades off the quality-of-service in serving each class of nodes.

In regards to the energy efficiency of IoT devices, Ogundile and *et al.* [64] investigated the energy consumption constraint in wireless sensor networks and proposed a clustering based routing algorithm. Their work takes into consideration that in some scenarios the sensors tend to consume more power attempting to initiate transmissions with the base station. Therefore, they proposed a clustering algorithm where each group of nodes utilise predefined nodes (cluster heads) within their cluster to reach the base station. This multi-hop technique showed an improvement to the energy efficiency. However, such an approach can negatively impact the overall performance in the case of busy multi-hops. Another common problem associated with the rerouting approaches is that the nodes selected for multi-hoping are vulnerable to having a short lifetime. Hence, the reliability is still an open issue. Similar approaches are also adopted in [65, 66, 67].

Different from the aforementioned clustering based routing algorithms, Liu and Chang in [52] proposed a clustering rerouting scheme based on the assumptions that nodes are scattered unevenly. Their assumption resulted in their clustering being performed based on an unequal number of nodes in each cluster. They utilised a probabilistic model in order to determine which node has the energy capability to perform the tasks of multi-hoping for other nodes in need within the same cluster. Then the node has the ability to opt out from serving as a cluster head at when the energy level reaches a specified threshold. Although there proposed scheme outperformed those in [53, 54, 55], yet it has a negative impact on the the total network delay.

2.2.5 Collision Rate

In terms of latency, Sørensen and *et al.* [51], with Markov model of the jockeying queue, they created a matrix A, which contains all state transition probabilities. Matrix A can be used to evaluate the steady state probabilities, P, by solving the linear system $A \cdot P = 0$. Matrix A facilitated the adoption of a Markov model for LoRaWAN device behaviour in the sub-band selection. For the sake of simplicity, they assumed a model of only class A devices which transmits a fixed payload size. In addition, all devices are assumed to successfully join the network in which there will be no acknowledgement messages needed (downlink) and hence no retransmissions.

Furthermore, for simplicity, they adopted the queuing theory of $M/M/c$ queue in their model, taking into account that the mean queue length within $M/M/c$ is twice that of $M/D/c$. In addition they adopted jockeying $M/M/c$ queue and carried out a comparison of their model in terms of latency against the use of only $M/D/c$ queue. Their approach showed lower latency results.

Following LoRaWAN capacity evaluation, Augustin and et al. [47] carried out a simulation of intense packets transmissions at a single data point. The results showed that the maximum use is 18% of the channel capacity for 0.48 link load. 60% of packets dropped due to collisions (collisions happen when two packet transmissions time overlap). Confirmed messages sent by devices as a collision solution is not practical, as it will result in retransmitting packets several times and thus causing more delay.

Huang and *et al.* [68] mentioned that regulatory and aggregated duty cycling limitation within LoRaWAN uplink was investigated from the perspectives of latency and collision rate. The proposed models tend to analyse the latency and collisions of packets taking into account sub-channel selection and combining. Results showed that sub-band with the highest duty cycle provide low latency even in the case of very high loads but with very high collision rate. Vice versa, sub-band with the lowest duty cycle results in high latency even in the case of low loads but very low collision rate. They conclude with

disproportionate relation between latency and collision rate, meaning enhancing one could have severe impact on the other.

Although LoRaWAN is designed for serving a large number of devices with minimal need for packets transmissions per day, Rizzi [69] carried out an industrial scenario experiment using LoRaWAN. They adopted a similar approach to Time Slotted Channel Hopping (TSCH) where using different *SF* were used in the same time-slot. Their approach resulted in providing each device with one communication opportunity per minute and minimised collision rate due to the accurate time and channel scheduling of *TSCH*.

Although Mikhaylov and *et al.* [70] empirically investigated that the *SF*s used within LoRaWAN are not fully orthogonal, however the transmission can be decoded successfully when its power is greater than any other interfering power. Sørensen and *et al.* [51] set out a simple rule in their model that all channels and *SF* are orthogonal, meaning any two or more transmissions take place over the same channel, *SF* and time cause a collision. For that reason, their scheme has multi-channel ALOHA random access (same as the case in [10], [47] and [48]) representing LoRaWAN 6 *SF* in 6 ALOHA channels (orthogonal) within a sub-band. Again for simplicity, they neglected acknowledgement messages meaning there is no downlink and hence no retransmissions. Using their scheme, they were able to quantify the outage caused by collisions by calculating the traffic load and hence the collision probability.

Neumann and *et al.* [71] ran an experiment based on an indoor deployment of a gateway and single LoRa Mote (LoRa tool designed to demonstrate specific LoRa modem capabilities). Both packet loss and packet error where measured based on the mote data transmission from various locations (different floors over different distances). In terms of packet loss and packet error, results vary based on the end-device location. Results showed that devices with the lowest data rate (*DR0*) located on different floors have packet loss decreasing as the device moves further from the gateway. The gateway experienced a packet loss of 25% when the device used *DR0* at a close distance. Vice versa, using higher data rates (*DR5*) showed an increase in the packet loss at further distances.

The gateway experienced packet loss of 27% when the device used *DR5* at a further distance (for example, the building basement).

2.2.6 Machine Learning in LoRaWAN

LoRaWAN is meant for serving IoT applications, where low latency is not usually a critical requirement [20]. Hence, the simplicity in LoRaWAN protocol, which makes it suitable for IoT serving applications given the limited resources. However, this results in a major drawback in LoRaWAN, which is the severe packet collision rate especially as the served networks scale up. In return, this results in a serious degradation to LoRaWAN performance and thus, to its reliability. The different features LoRaWAN protocol provides for example, *SF* and *CR*, enhance its flexibility and suitability for being adapted according to the needs of the served application [1]. In addition, these features encouraged a number of research efforts to use them as elements to improve LoRaWAN in different aspects using various machine learning algorithms and techniques.

For example, Cui and Joe [72] have proposed an enhanced packet collision prediction scheme based on the Long Short-Term Memory (LSTM) model. Despite the high prediction accuracy *LSTM* model provided, LoRaWAN random transmission behaviour requires an online training schemes to achieve a practical prediction process. This has motivated Cui and Joe [72] to combine *LSTM* with a State Space Model (SSM) and propose an enhanced Long Short-Term Memory Extended Kalman Filter (LSTM-EKF) scheme. Their proposed scheme showed relatively higher prediction accuracy in comparison to the original *LSTM* model. However, the prediction process is highly dependant on the input parameters, which are chosen to include LoRaWAN protocol features such as different *SF*s, *CR* and the class of end-device communication. Hence, the prediction accuracy remains a function of the pre-inputs selection process leading to a very high computation overhead.

Acknowledging the randomness of LoRaWAN transmissions, Cuomo and *et al.* [73] proposed nodes profiling scheme based on the unsupervised learning clustering algorithm

(K-Means). Considering two gateways within the proximity of the nodes, the profiling scheme aims to predict the duplication in nodes transmissions by grouping packets that have similar transmission characteristics. Based on the duplication prediction, a traffic prediction is carried out by combining the Decision Tree (DT) and *LSTM* models for the purpose of enhancing the resource allocation. Although the unsupervised learning clustering algorithm is the preferable machine learning classifying tool in low power networks due to its simplicity, the number of clusters could play a vital role making it very complex to implement. Hence, analysis of the optimal number of clusters is essential to achieve the optimal clustering accuracy. This is highly dependant on the application parameters used in the clustering process.

Exploiting the variety of LoRaWAN features as parameters in machine learning tools, Sandoval *et al.* [74] proposes a configuration update scheme to the nodes based on Reinforcement Learning (RL) to maximise the throughput of each node individually. The configuration process relies on categorising packets received from the nodes into different importance scales. Their scheme reserves LoRaWAN parameters that ensure robust transmission for nodes classified as important source of information. These nodes receive configuration updates from the gateway to elevate their individual throughput, whereas, the gateway using the *RL*-based scheme refrains from updating nodes classified as a lower importance source of information. In other words, the gateway using the proposed scheme learns how to intelligently elevate the chance of allowing certain nodes to successfully transmit at the expense of other nodes, all based on prior importance classifications of the nodes.

Similarly, Aihara *et al.* [75] use *RL* by proposing Q-learning model combined with Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) to mitigate the collisions in LoRaWAN and enhance the network *PDR*. In their proposed scheme, the number of successfully received packets form the nodes to the gateway is defined as a reward function that their scheme learns to maximise. They have identified the main cause of collisions to be simultaneous transmissions. Hence, to mitigate the collision

rate, their schemes allows the gateway to learn which nodes are prone to simultaneous transmissions. Hence, the gateway allows the target nodes to transmit over different channels using *CSMA/CA*. However despite the high energy consumption needs when using such techniques, the *PDR* is still a function of the number of available channels. Hence, this can cause a negative impact especially when the network scales up.

2.3 Chapter Summary

The root cause to many of the problems addressed in the reviewed set of studies is the packet collision rate associated with LoRaWAN random transmissions. Although machine learning was the base to many of the proposed performance enhancement approaches, it can be easily accompanied with negative consequences especially in terms of energy consumption. These negative consequences are usually due to the very high computational burdens at the end-devices level. For example, it was noticed when reviewing the aforementioned set of studies [52, 53, 54, 55, 56, 57, 72, 73, 74, 75] that the energy consumption was de-prioritised if not completely neglected. The fact that IoT devices are resource-limited (e.g. battery-powered) makes it necessary to consider the energy limitations. Hence, the work in Chapters 3, 4 and 5 take the energy limitations into account.

Chapter 3

Clustering in LoRaWAN

3.1 Overview

The random LoRaWAN transmissions behaviour leads to high packet collision rate that can exceed 90% at dense networks where the number of nodes reach up to 1000. This is mainly due to the adoption of ALOHA protocol in LoRaWAN where the nodes initiate transmissions to the gateway regardless of the channel status. This results in an increased number of retransmission attempts by the nodes, which in return increases the network's total transmission delay and energy consumption. Hence, it is essential to reduce the amount of possible simultaneous transmissions from the nodes in order to mitigate the collision rate. This can be achieved by partitioning the nodes into different clusters. Hence, this chapter proposes the use of unsupervised learning clustering algorithm (K-Means) in LoRaWAN, and investigates the impact of different number of clusters on the network's collision rate. Simulations of the network performance show that the proposed scheme results in a significant reduction to the packet collision rate, which in return enhances the transmission delay and energy consumption.

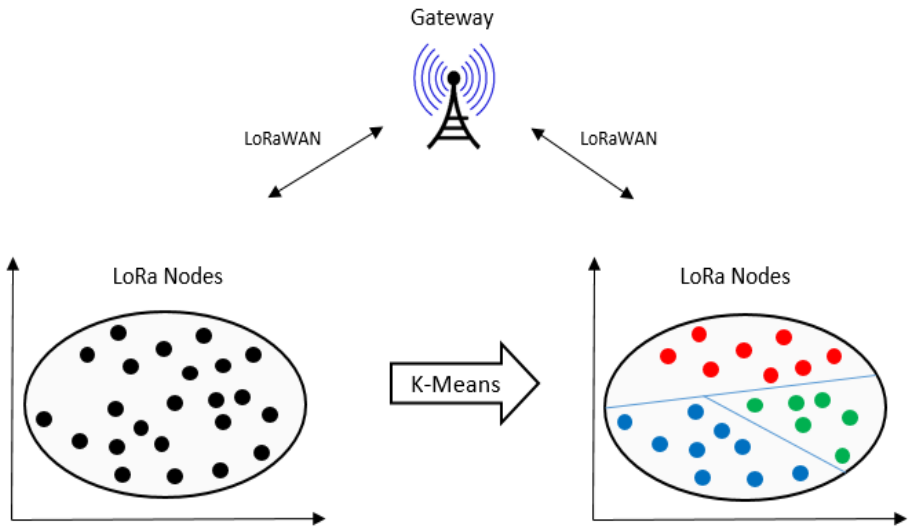


Figure 3.1: Random Distribution of LoRa Nodes Using LoRaWAN Protocol to Communicate with the Gateway Before and After K-Means Clustering

3.2 System Model

The system model in Figure 3.1 shows randomly distributed nodes $n \in \{n_1, n_2, \cdots, n_i\}$, where $1 \leq i \leq 1000$ that communicate with one gateway (GW) using LoRaWAN protocol. Each node transmits a set of unique values. These values can be adjusted to any practical application. The GW normalises these values and processes them as entities for performing K-Means clustering. As in Figure 3.1, the nodes fall into different number of clusters making the GW able to implement a simple transmission Priority Scheduling Technique (PST). The simple transmission PST is aimed at regulating the order of transmissions from the nodes based on the transmission priority of the corresponding cluster. Hence, a lower number of nodes transmitting at same time.

3.3 Unsupervised Learning Clustering Algorithm (K-Means)

The core propose of choosing K-Means [76, 77] for clustering is due to its simple implementation at the GW level without involving the nodes in the clustering process. This is to ensure minimal energy burden at the nodes level. The nodes fall within different clusters following K-Means clustering algorithm. These clusters are based on a set of

values $S = \{s_1, \ldots, s_k\}$ where S is in the space of a positive integer number k.

In K-Means the number of clusters has to be pre-defined. Therefore, considering the scale of the network the number of clusters is chosen to be $k = \{3, 4, 5\}$. This is to evaluate the impact of introducing a different number of clusters on the collision rate. The core purpose is to reduce the collision rate.

The clustering process begins by defining the number of cluster. Since the clustering is based on the set of values transmitted from the nodes to the gateway, these set values are normalised into a value of z for each node. This is followed by initiating a cluster center value c_j that are randomly chosen for the available values of z. The aim is to classify nodes with the minimum Euclidean distance from a specific c_j into the same cluster following Equation (3.1).

$$\arg\min_{c_j} dist(c_j, z_{n_i}) \tag{3.1}$$

where $dist(c_j, z_{n_i})$ is the Euclidean distance between each cluster's center c_j and the node's normalised value z_{n_i}. Note that c_j value and therefore, the Euclidean distance are updated following each node classification following Equation (3.2) and (3.3). This is to minimise the difference between c_j and other nodes' values of z.

$$d(c_j, z_{n_i}) = \sqrt{(c_{j_{(A)}} - z_{n_{i(A)}})^2 + (c_{j_{(B)}} - z_{n_{i(B)}})^2} \tag{3.2}$$

where A and B are the original set of values transmitted by each n_i to the GW. $z_{n_{i(A)}}$ and $z_{n_{i(B)}}$ are the normalised values of the original A and B values. $c_{j_{(A)}}$ and $c_{j_{(B)}}$ are the initial cluster center values.

$$c_j = \frac{1}{\mid c_j \mid} \sum_{n \in k} n \tag{3.3}$$

The same procedure above happens with each node of n_i. Since c_j is updated following each node clustering process, a number of iterations are needed to ensure the conver-

gence of c_j. The convergence of c_j indicates the classification of each node to the closest cluster in terms of the value z. In the considered scenario of this work, a total of thirty iterations were performed to reach the converged stage of each cluster. Sections 3.3.1, 3.3.2 and 3.3.3 provides analysis of using a different number of clusters $k = \{3, 4, 5\}$ for this work. Note that the minimum limit of the number of clusters is one, which has no impact on reducing the number of simultaneous transmissions as the typical network with no clustering is naturally falling in the form of one cluster. Although two clusters were included in the analysis of network performance, it was not included in the detailed comparisons in the following sections as the variations are insignificant. Therefore, the comparisons of number of clusters started from three clusters onward. Note that the values of A and B are listed prior to the normalisation for better visualisation of the considered scenario. The data set in this work is based on a forest wildfire early warning application hence, the value A represents the atmospheric humidity while value B represents weather temperature.

3.3.1 Three Clusters Analysis ($k = 3$)

In this scenario three clusters are chosen to evaluate the reduction of the collision rate in comparison to different number of clusters. By assigning an initial clustering values (Table 3-A) to be the initial centroids for each of the three clusters, the clustering process begins following Equations (3.1), (3.2) and (3.3) above. Therefore, nodes are assigned to the cluster with the nearest value of A and B.

Table 3-A: Initial Cluster Centers ($k = 3$)

-	Clusters		
-	1	2	3
A (Atmospheric humidity %)	50	70	30
B (Temperature)	30	44	45

Note that the values in Table 3-A were the initial values; following the convergence of c_j, Table 3-B shows the final centroid values for each cluster and the number of nodes classified to each cluster.

Table 3-B: Final Cluster Centers ($k = 3$)

-	Clusters		
-	1	2	3
A (Atmospheric humidity %)	49	63	37
B (Temperature)	37	38	37
Number of nodes in each cluster	338	334	328

The total number of nodes considered in the system model is 1000. From Table 3-B, this number was divided into three clusters, where cluster one consists of 338, cluster two consists of 334 and cluster three consists of 328 nodes following the clustering process, and an average of 20 runs.

The bar chart in Figure 3.2 illustrates the clusters based on the mean values of A and B for the nodes within each cluster. Note that in a simple transmission priority arrangement, the clusters are assigned different transmission priorities corresponding to the relationship between the values A and B. In other words, the lower the difference between the values, the higher the transmission priority assigned to the corresponding cluster. The main purpose of this consecutive transmission style of the clusters is to prevent simultaneous transmissions from nodes in different clusters, and therefore reducing the total collision rate.

3.3.2 Four Clusters Analysis ($k = 4$)

In this scenario four clusters are chosen to evaluate the reduction of the collision rate in comparison to different number of clusters. Similar to ($k = 3$), initial clustering values (Table 3-C) are assigned to be the initial centroids for each of the four clusters, the clustering process begins following Equations (3.1), (3.2) and (3.3) above.

Note that the values in Table 3-C were the initial values; following the convergence of c_j, Table 3-D shows the final centroid values for each cluster and the number of nodes classified to each cluster.

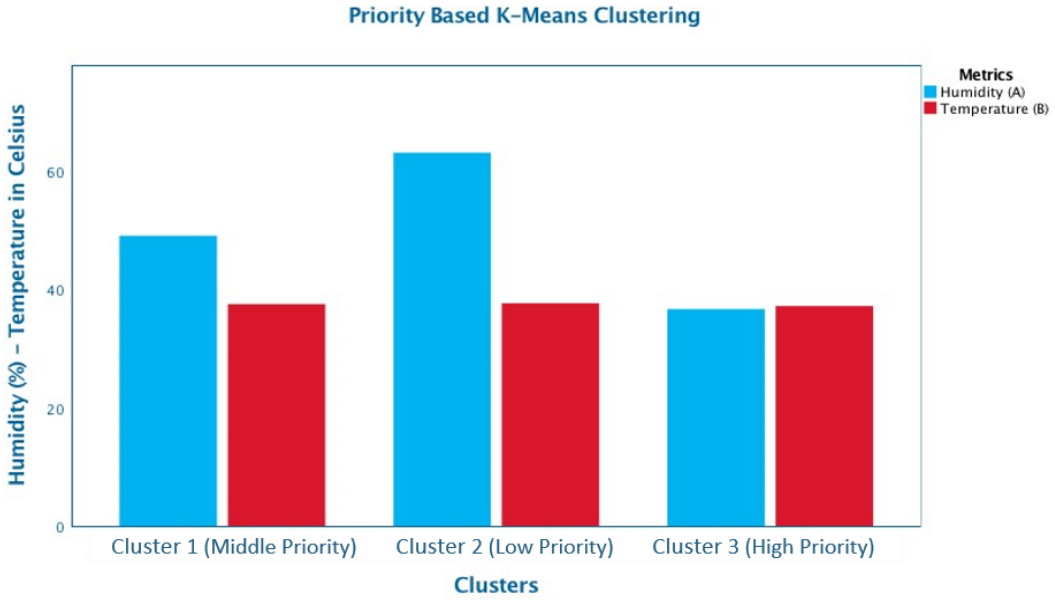


Figure 3.2: Simple Transmission Priority Scheduling Technique Based on K-Means Clustering ($K = 3$) Bar Chart

Table 3-C: Initial Cluster Centers ($k = 4$)

-	Clusters			
-	1	2	3	4
A (Atmospheric humidity %)	56	70	30	43
B (Temperature)	45	31	44	30

Table 3-D: Final Cluster Centers ($k = 4$)

-	Clusters			
-	1	2	3	4
A (Atmospheric humidity %)	54	65	35	45
B (Temperature)	39	37	37	37
Number of nodes in each cluster	239	255	257	249

The total number of nodes considered in the system model is 1000. From Table 3-D, this number was divided into four clusters, where cluster one consists of 239 nodes, cluster two consists of 255, cluster three consists of 257, cluster four consists of 249, following the clustering process, and an average of 20 runs. Figure 3.3 illustrates the clusters with different values of A and B.

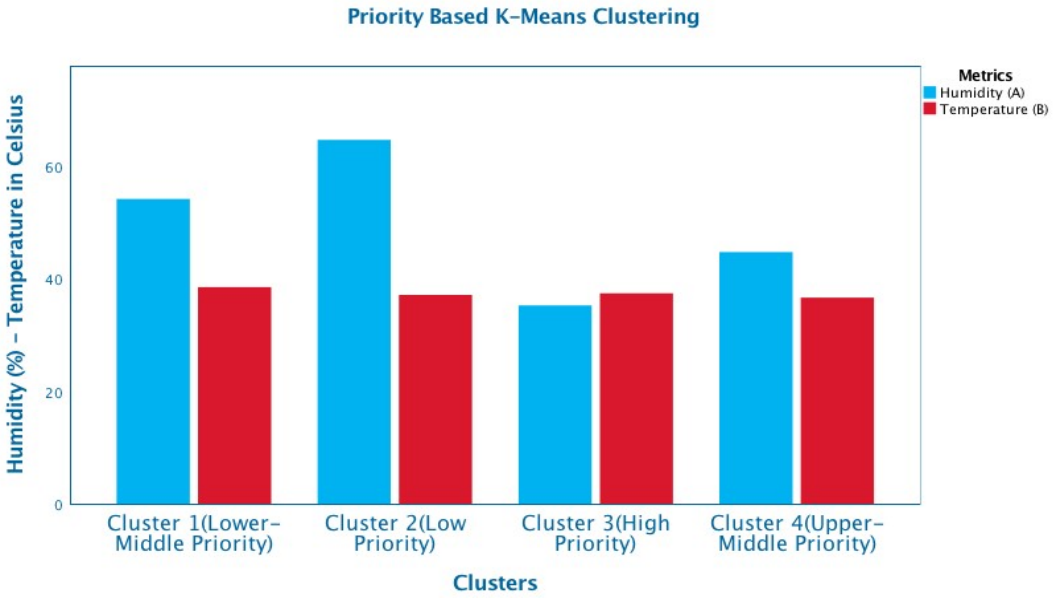


Figure 3.3: Simple Transmission Priority Scheduling Technique Based on K-Means Clustering ($K = 4$) Bar Chart

Table 3-E: Initial Cluster Centers ($k = 5$)

-	Clusters				
-	1	2	3	4	
A (Atmospheric humidity %)	41	50	30	60	69
B (Temperature)	30	45	44	31	44

3.3.3 Five Clusters Analysis ($k = 5$)

Similar to the above two clustering scenarios. This work takes clustering even further by partitioning the nodes into five clusters. In this scenario five clusters are chosen to evaluate the reduction of the collision rate in comparison to different number of clusters. Similar to ($k = 3, 4$), initial clustering values (Table 3-E) are assigned to be the initial centroids for each of the five clusters, the clustering process begins following the same Equations (3.1), (3.2) and (3.3).

Note that the values in Table 3-E were the initial values; following the convergence of c_j, Table 3-F shows the final centroid values for each cluster and the number of nodes classified to each cluster.

Table 3-F: Final Cluster Centers ($k = 5$)

-	Clusters				
-	1	2	3	4	5
A (Atmospheric humidity %)	44	49	35	57	65
B (Temperature)	34	41	37	35	38
Number of nodes in each cluster	177	178	256	178	211

The total number of nodes considered in the system model is 1000. From Table 3-F, this number was divided into four clusters, where cluster one consists of 177 nodes; cluster two consists of 178, cluster three consists of 256, cluster four consists of 178 and cluster five consists of 211 nodes, following the clustering process, and an average of 20 runs. The bar chart in Figure 3.4 shows the clusters with different values of A and B, based on which the the cluster with the lowest difference between A and B is granted the highest transmission priority.

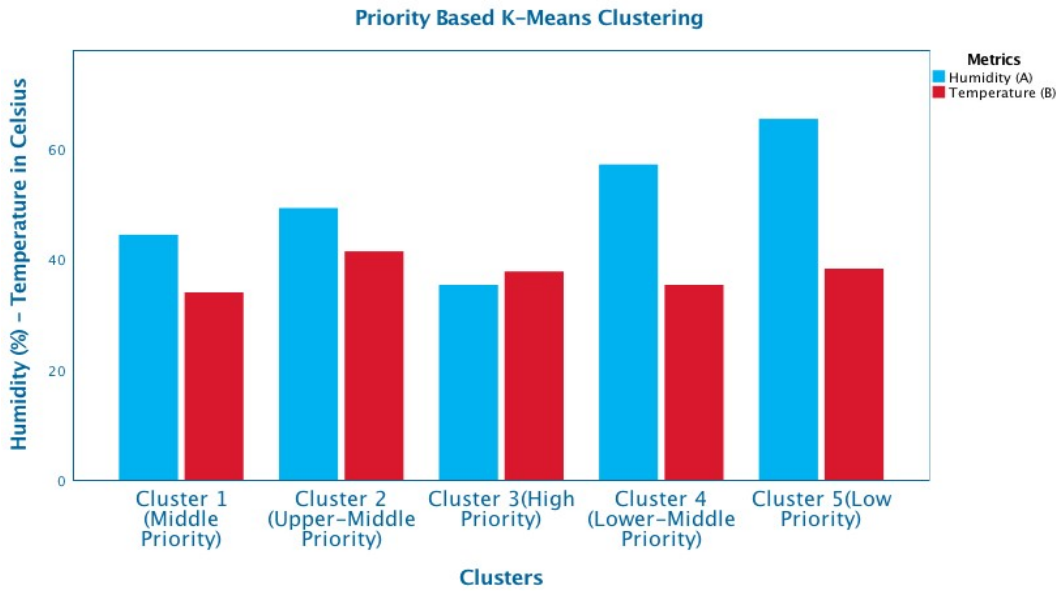


Figure 3.4: Simple Transmission Priority Scheduling Technique Based on K-Means Clustering ($K = 5$) Bar Chart

The purpose of increasing the number of clusters is to evaluate the impact of the number of clusters on the collisions and compare it to a typical (non-clustered) LoRaWAN

network. The discussion below which is based on the simulation results proves that the higher the number of clusters, the lower the collisions. Hence, the evaluations are taken further to assess the impact of the clustering technique on the network total transmission delay and the total energy consumption which is mainly effected by the collisions and the retransmissions of collided packets.

3.4 Problem Formulations

3.4.1 Packet Collisions in LoRaWAN

In the defined system model there are 1000 nodes assumed to be transmitting randomly to the gateway. Each node transmits one packet following ALOHA protocol. A collision happens when two nodes transmit at the same time over the same frequency. According to Poisson distribution, the probability P of a collision to happen is given by Equation (3.4):

$$P = e^{-2G} \tag{3.4}$$

where G is the rate of packet transmission attempts per node. All nodes are assumed to be transmitting over one channel using one spreading factor ($SF7$) with a bandwidth of 125 kHz and a Coding Rate (CR) of 1, which maximise the actual packet bits at the expense of the Forward Error Correction (FEC) redundant bits. In LoRaWAN typical deployment each node transmits whenever there is a ready to send packet regardless of any other transmission occupying the channel (ALOHA style) [1]. This transmission behaviour results in a collision rate of up to 90% at a LoRaWAN network with 1000 nodes (shown in Figure 3.7, Section 3.5).

On the other hand, the proposed clustering based simple transmission PST via regulating the nodes transmissions based on the corresponding cluster's priority reduces the collision rate. The reduction to the collision rate is a result of reducing the number of nodes in each cluster. By using Equation (3.5), the proposed technique reduces the total collision

rate to 58% when $k = 3$, 45% when $k = 4$ and 41% when $k = 5$ in comparison to the collision rate in typical LoRaWAN. Figures 3.8, 3.9 and 3.10 in Section 3.5 show the collision rate at each cluster of C_k.

$$Total\ Collision\ Rate = \frac{C_{1_{colli}} + C_{2_{colli}} + \cdots + C_{k_{colli}}}{C_k} \tag{3.5}$$

where *colli* is the number of collisions in the corresponding cluster of C_k.

3.4.2 Total Transmission Delay

For further evaluation, this section addresses the impact of applying K-Means clustering on the total transmission delay. The transmission delay in LoRaWAN is a function of the number of bits and the bit rate per second [8]. It is proportional to the number of bits within a packet and it is calculated as in Equation (3.6) below

$$Transmission\,Delay = \frac{Number\ of\ bits}{Bitrate} \tag{3.6}$$

where Bitrate is given by Equation (3.7)

$$Bitrate = \frac{SF \times BW}{2^{SF}} \times \frac{4}{4 + CR} \tag{3.7}$$

where SF is the Spreading Factor and it is fixed to $SF7$, $BW = 125kHz$ and CR is the Coding Rate and is set to $CR = 1$. Note that $CR \in \{4/5, 4/6, 4/7, 4/8\}$ is the ratio of the actual data bits to the redundant bits and is represented by $CR = \{1, 2, 3, 4\}$, respectively. Using these parameters insures a maximum successful transmissions in LoRaWAN given a limited area [1]. More details of the BW, SF and CR are given in [20].

Note that for simulation practicality, collided packets are allowed one retransmission per packet. The transmissions follow LoRa SX1272 LoRa model specifications [45]. This is to validate the simulation performance against practical experiments held in [78, 79, 47].

According to LoRaWAN specifications [1], a packet collision results in a retransmission attempt. Hence, Initial Transmission Delay (ITD) and Retransmission Delay (RTD) are taken into consideration in the simulation process. The simple transmission *PST* for different clusters introduces waiting times, which means a cluster with lower priority waits until transmissions from nodes in higher priority clusters are complete.

For better explanation, it is important to give details of the transmission delay concept in the considered LoRaWAN system model. This takes place via the following two examples (A) and (B) to explain how the transmission delay differs in typical LoRaWAN (non-clustering) and the proposed K-Means clustering based simple transmission *PST*. Values used in (A) and (B) are simplified for the purpose of clarity. Simulation values are different and revealed in the performance evaluations section.

- **Example A (no clustering):**

  Assume there are 1000 nodes, with each node sending 1 packet and the transmission time is 1 second for each node. Hence,

  $ITD = 1000\ nodes \times 1\ second = 1000\ seconds$

  However, there are collided packets that need to be retransmitted again. In such a network size, the collision rate is 90%. This means 900 packets need to be retransmitted. Hence,

  $RTD = 900\ nodes \times 1\ second = 900\ seconds$

  Thus, The Total Transmission Delay (TTD) can be calculated as the following:

  $TTD = ITD + RTD = 1000\ seconds + 900\ seconds = 1900\ seconds$

- **Example B (four clusters):**

  Assume there are 1000 nodes divided evenly into four clusters. This means 250 nodes in each cluster. Assume there is a collision of 48% in each cluster. This

means 120 collided packets of the total 250 nodes in each cluster. Overall, there are four clusters of 250 transmitting nodes each with 120 collided packets. Let the Initial Transmission Delay in each cluster C be denoted as $C_{k_{ITD}}$. Similarly, let Retransmission Delay due to collisions in each cluster be denoted as $C_{k_{RTD}}$ and Total Transmission Delay of the cluster k be $C_{k_{TTD}}$, where k is the number of clusters and $k = \{1, 2, 3, 4\}$. Hence, $C_{k_{TTD}}$ in each cluster of k can be calculated as in the following equations:

$$C_{1_{TTD}} = C_{1_{ITD}} + C_{1_{RTD}}$$
$$C_{1_{TTD}} = 250 \times 1\,second + 120 \times 1\,second = 250s + 120s = 370s$$

$$C_{2_{TTD}} = C_{1_{TTD}} + C_{2_{ITD}} + C_{2_{RTD}}$$
$$C_{2_{TTD}} = 370s + 250 \times 1\,second + 120 \times 1\,second = 370s + 250s + 120s = 740s$$

$$C_{3_{TTD}} = C_{2_{TTD}} + C_{3_{ITD}} + C_{3_{RTD}}$$
$$C_{3_{TTD}} = 740s + 250 \times 1\,second + 120 \times 1\,second = 740s + 250s + 120s = 1110s$$

$$C_{4_{TTD}} = C_{3_{TTD}} + C_{4_{ITD}} + C_{4_{RTD}}$$
$$C_{4_{TTD}} = 1110s + 250 \times 1\,second + 120 \times 1\,second = 740s + 250s + 120s = 1480s$$

From examples (A) and (B), the total transmission delay in example A resembling typical LoRaWAN with no clustering applied is 1900 *seconds*. While the total transmission delay in the network after applying four clusters is 1480 *seconds*. Hence, the proposed K-Means clustering based simple transmission *PST* reduces the delay more than that of typical LoRaWAN. Figure 3.5 illustrates the impact on collisions and transmission delay before (example A) and after (example B) applying K-Means clustering. Figure 3.6 shows the expected outcome of the total transmission delay in both examples. Simulations were carried out to prove the aforementioned examples and the results are shown in the

simulation results section of this chapter.

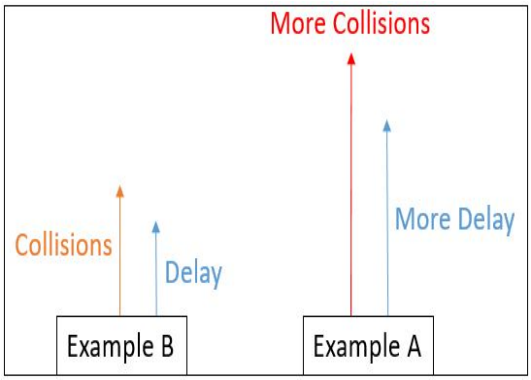


Figure 3.5: Collision Rate and Transmission Delay Trends in Typical LoRaWAN (A) Vs. Proposed K-Means Clustering (B)

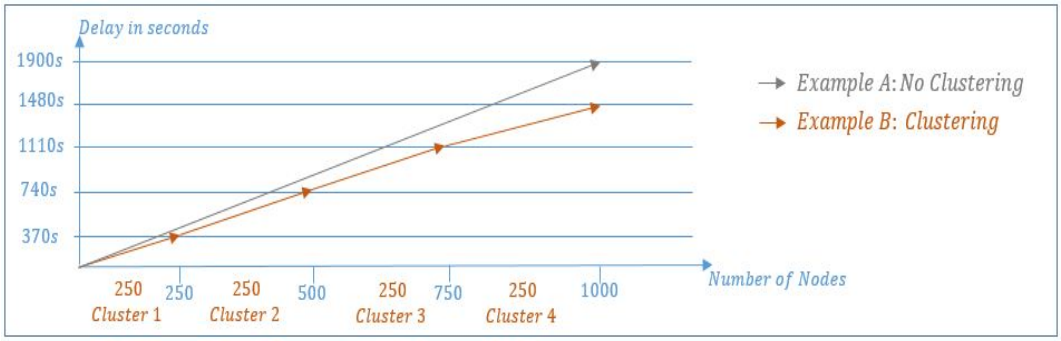


Figure 3.6: Transmission Delay in Typical LoRaWAN (example A) Vs. Proposed K-Means Clustering based Simple Transmission Priority Scheduling Technique (example B)

From examples (A) and (B), TTD is calculated following a set of Equations (3.8), (3.9) and (3.10).

$$C_{HP_{TTD}} = C_{HP_{ITD}} + C_{HP_{RTD}} \tag{3.8}$$

$$C_{LP_{TTD}} = C_{HP_{TTD}} + C_{LP_{ITD}} + C_{LP_{RTD}} \tag{3.9}$$

$$TTD = C_{HP_{ITD}} + C_{LP_{TTD}} \tag{3.10}$$

where $C_{HP_{TTD}}$ is transmission delay of clusters with a higher transmission priority, $C_{HP_{ITD}}$ is the delay caused by initial transmission in higher transmission priority clus-

ters, and $C_{HP_{RTD}}$ is the delay caused by the retransmissions of collided packets in higher transmission priority clusters. Similarly, $C_{LP_{TTD}}$ is transmission delay of clusters with a lower transmission priority, $C_{LP_{ITD}}$ is the delay caused by initial transmission in lower transmission priority clusters, and $C_{LP_{RTD}}$ is the delay caused by the retransmissions of collided packets in lower transmission priority clusters. TTD is the total transmission delay of the network including all clusters. Note that transmission delay in lower priority clusters always contain the transmission delay of clusters with higher priority, as lower priority clusters wait until transmissions in higher priority clusters are over.

3.4.3 Energy Consumption

As for the energy consumption and since LoRa SX1272 model is adopted in this work, the energy evaluations are validated against LoRa SX1272 model used in an experiment carried out in [78]. This work considers the energy consumed by transmitting clusters and the energy consumed by non-transmitting clusters following the Equations (3.11) and (3.12), respectively:

$$EC_{Tr} = P_{Tr} \times D_{Tr} \tag{3.11}$$

where EC_{Tr} is the energy consumed by transmitting clusters, P_{Tr} is transmission power in watts per node/packet and D_{Tr} is the transmission duration in seconds per node/packet.

$$EC_{nonTr} = P_{nonTr} \times D_{nonTr} \tag{3.12}$$

where EC_{nonTr} is the energy consumed by non-transmitting clusters, P_{nonTr} is the power consumption per standby nodes in non-transmitting clusters and D_{nonTr} is the transmission duration expected for the standby nodes in non-transmitting clusters.

From Equations (3.11) and (3.12), the total energy consumption $E_c^{(i)}$ in Equation (3.13) is calculated as the sum of individual cluster's energy consumption as expressed in Equations (3.14), (3.15), (3.16) and (3.17). It is assumed that only one scenario with four

clusters are considered ($k = 4$) in this analysis.

$$E_c^{(i)} = EC_{Tr}^{(i)} + \sum_{j=0, j\neq i}^{K} EC_{nonTr}^{(i)} \tag{3.13}$$

where $i \in \{1, 2, 3, 4\}$ is the cluster number. The energy consumption in each cluster is calculated as follows:

$$E_c^{(1)} = EC_{Tr}^{(1)} + EC_{nonTr}^{(2)} + EC_{nonTr}^{(3)} + EC_{nonTr}^{(4)} \tag{3.14}$$

$$E_c^{(2)} = EC_{nonTr}^{(1)} + EC_{Tr}^{(2)} + EC_{nonTr}^{(3)} + EC_{nonTr}^{(4)} \tag{3.15}$$

$$E_c^{(3)} = EC_{nonTr}^{(1)} + EC_{nonTr}^{(2)} + EC_{Tr}^{(3)} + EC_{nonTr}^{(4)} \tag{3.16}$$

$$E_c^{(4)} = EC_{nonTr}^{(1)} + EC_{nonTr}^{(2)} + EC_{nonTr}^{(3)} + EC_{Tr}^{(4)} \tag{3.17}$$

where $E_c^{(1)}$ is the energy consumption during cluster 1 transmissions, $EC_{Tr}^{(1)}$ is energy consumed by transmitting nodes in cluster 1, $EC_{nonTr}^{(2)}$ is energy consumed by non-transmitting nodes in cluster 2, $EC_{nonTr}^{(3)}$ is energy consumed by non-transmitting nodes in cluster 3 and $EC_{nonTr}^{(4)}$ is energy consumed by non-transmitting nodes in cluster 4. $E_c^{(2)}$, $E_c^{(3)}$ and $E_c^{(4)}$ are the energy consumption during clusters 2, 3 and 4 transmissions respectively, each of which includes the energy consumed by both the transmitting node and the non-transmitting nodes within the corresponding cluster. A similar approach is adopted for calculating energy consumption in other scenarios with a different number of clusters ($k = \{2, 3, ..., 100\}$).

3.5 Simulation Results and Performance Evaluations

This section reveals the impact of applying K-Means clustering on the collision rate and the Packet Delivery Rate (PDR) in LoRaWAN. Moreover, it shows an evaluation of the total transmission delay within the network in comparison to typical LoRaWAN and a collision resolution scheme proposed in [80]. Furthermore, it reveals the evaluation of the optimal number of clusters in terms of energy consumption efficiency.

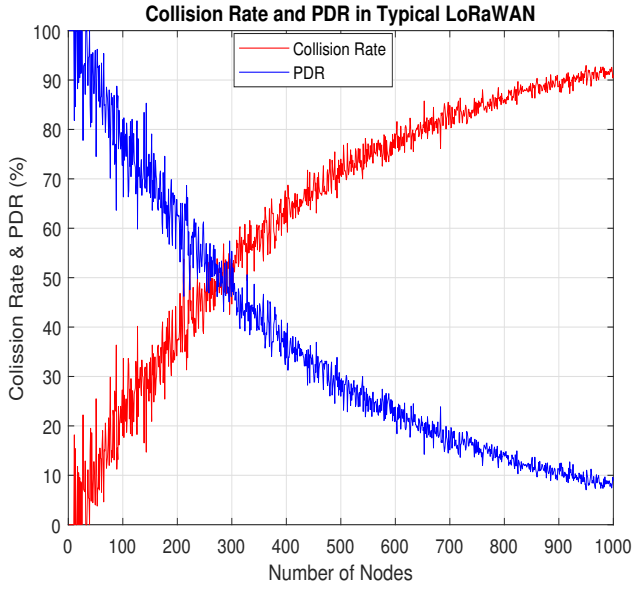


Figure 3.7: Collision Rate and PDR in Typical LoRaWAN

3.5.1 Collision and Packet Delivery Rates in Typical LoRaWAN vs. Different Number of Clusters

3.5.1.1 Typical LoRaWAN

In typical LoRaWAN, the collision rate is excessively high due to simultaneous transmissions attempts from different nodes. This is a result of the adoption of ALOHA protocol where nodes initiate transmissions regardless of the channel status. To evaluate the packet collision behaviour in LoRaWAN, a simulation LoRaWAN engine is built and the number of transmitting nodes are scaled from 1 to 1000 nodes. From Figure 3.7, it is shown that the collision rate using typical LoRaWAN at 1000 nodes exceeds 90%. This illustrates the problem of adopting LoRaWAN to serve dense applications hence the proposed clustering for LoRaWAN. Note that in the simulation, the transmission environment is assumed to be idle, where the packet loss can only be a result of packet collision. Thus, *PDR* follows a mirror trend of the collision rate.

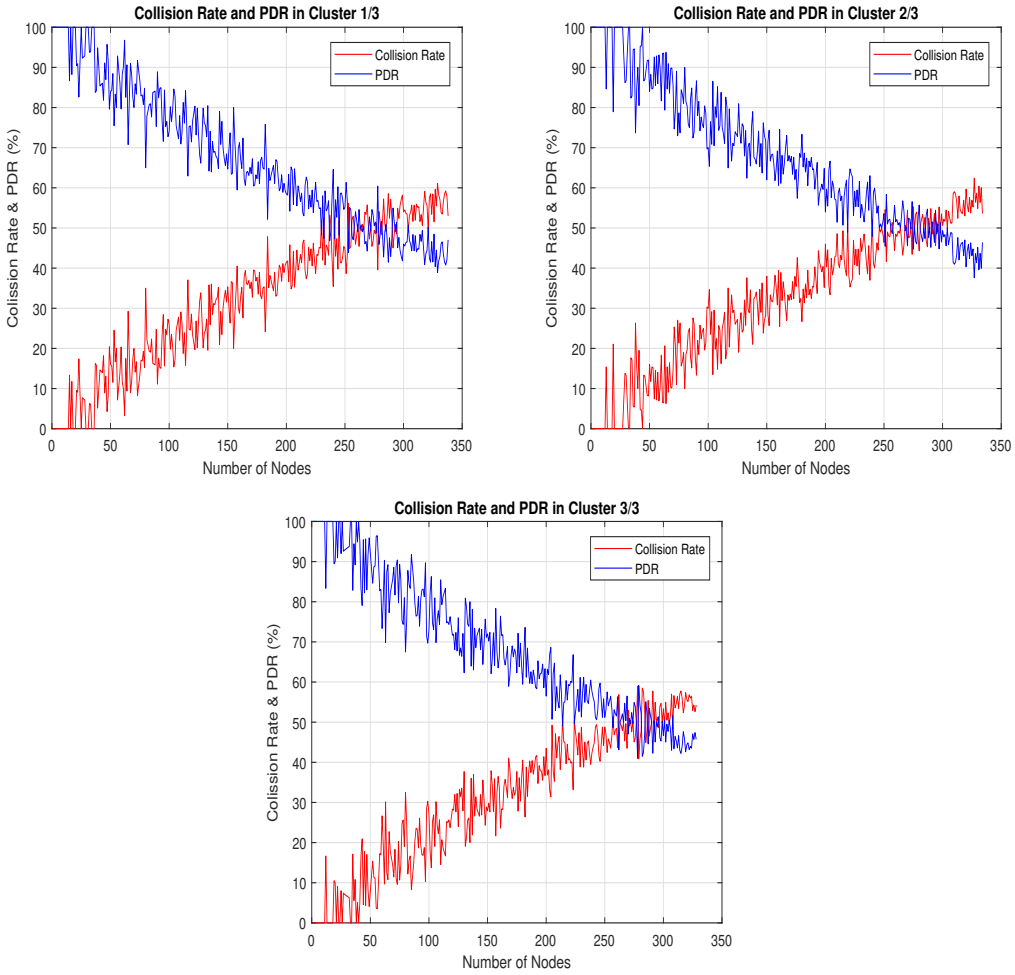


Figure 3.8: Collision Rate and PDR in LoRaWAN with Three Clusters ($k = 3$)

3.5.1.2 Three Clusters ($k = 3$)

From Figure 3.8, it is noticed that the collision rate in each cluster significantly decreased. This is a result of reducing the number of simultaneously transmitting nodes. Note that nodes within each cluster transmit at different times from nodes in other clusters following a simple transmission priority scheduling technique. The total collision rate for the network is reduced from 90% to 58%.

The number of nodes vary in each cluster with 338, 334 and 328 nodes in clusters one, two and three, respectively. The transmissions from the nodes follow LoRaWAN class *A* transmission protocol.

3.5.1.3 Four Clusters ($k = 4$)

The number of clusters are increased in order to have a better insight to the impact of the number of clusters on the collision rate and *PDR*. From Figure 3.9, it is noticed that the collision rate in each cluster further decreased. This is a result of further reduction to the number of simultaneously transmitting nodes in each cluster. Note that nodes within each cluster transmit at different times from nodes in other clusters following a simple transmission priority scheduling technique. The total collision rate for the network is further reduced to 45% in comparison to 58% at $k = 3$.

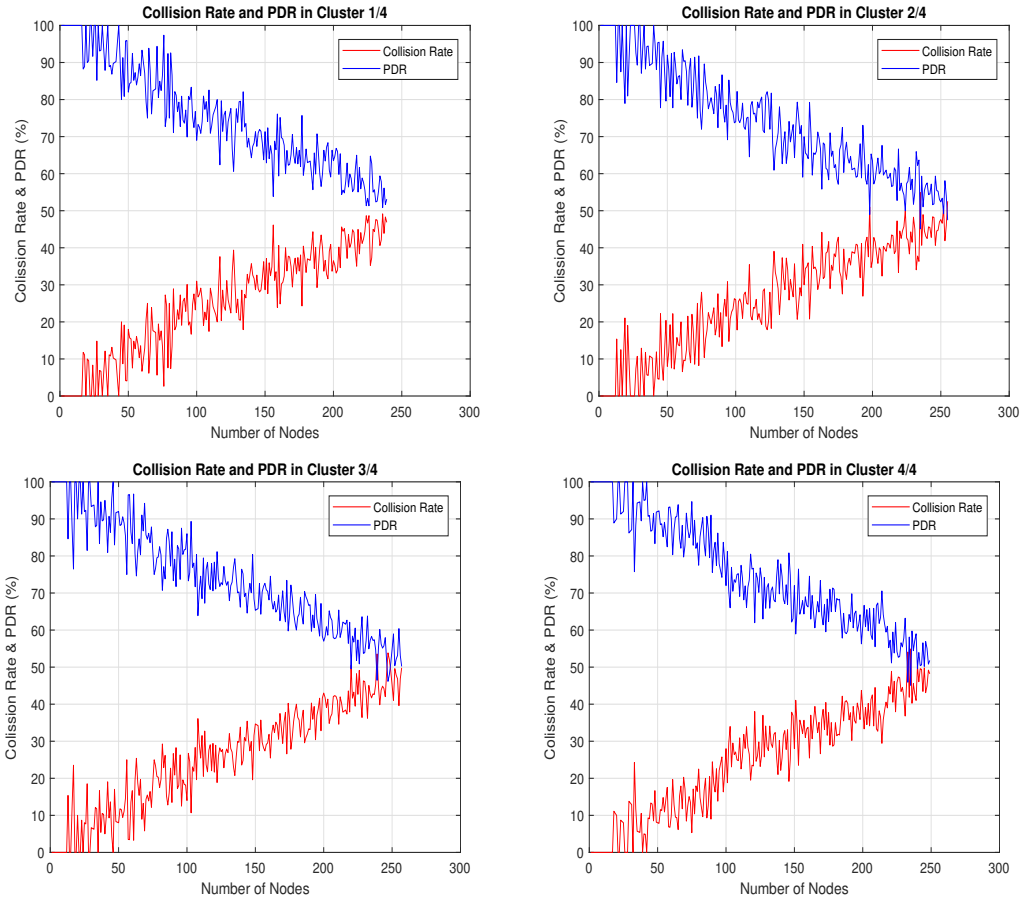


Figure 3.9: Collision Rate and PDR in LoRaWAN with Four Clusters ($k = 4$)

Similarly the number of nodes vary in each cluster with 239, 255, 257 and 249 nodes in clusters one, two, three and four, respectively. In a similar manner, the transmissions from the nodes follow LoRaWAN class *A* transmission protocol.

3.5.1.4 Five Clusters ($k = 5$)

When increasing the number of clusters to $k = 5$, it is noticed in Figure 3.10 that the collision rate in each cluster further decreased in comparison to a lower number of clusters. This is a result of further reduction to the number of simultaneously transmitting nodes in each cluster. Similarly, the nodes within each cluster transmit at different times from nodes in other clusters following a simple transmission priority scheduling technique. The total collision rate for the network is further reduced to 41% in comparison to 45% at $k = 4$ and 58% at $k = 3$.

Similarly the number of nodes vary in each cluster with 177, 178, 256, 178 and 211 nodes in clusters one, two, three, four and five, respectively. The transmissions from the nodes follow LoRaWAN class *A* transmission protocol.

It is noticed that the collision rate is reduced linearly as the number of clusters increases. This is a result of the continuous reduction of the number of nodes in each cluster. However, increasing the number of clusters increases the complexity of the clustering process. Hence, it is essential to find the optimal number of clusters that provides the best trade-off between reducing the collision rate, providing fairly reduced transmission delay and energy consumption. Since LoRaWAN is a low power orientated wireless technology, the optimal number of clusters is to be found based on the most appropriate total energy consumption. Section 3.5.2 reveals the process of obtaining the optimal number of clusters using the Elbow Method [81, 82, 83]. While Section 3.5.3 reveals the impact of reducing the number of clusters on the total transmission delay within the network.

3.5.2 Optimal Number of Clusters Analysis using Elbow Method

As for the unsupervised clustering algorithm (K-Means), the number of clusters are predefined where nodes are clustered based on the Euclidean distance from each cluster's centroid. Although the effect of increasing the number of clusters on the collision rate of the network is evaluated, this work also takes into account obtaining the most appropriate

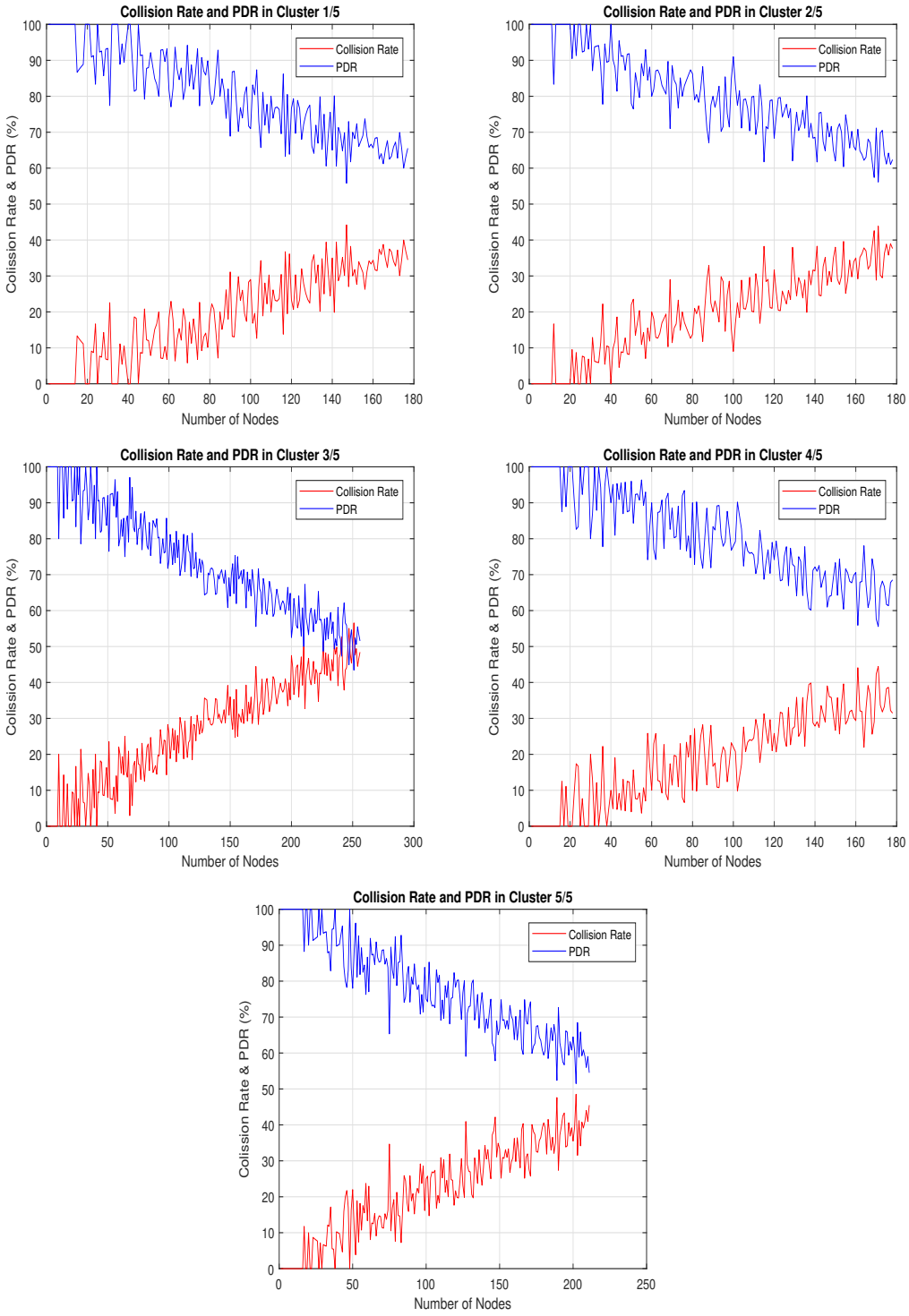


Figure 3.10: Collision Rate and PDR in LoRaWAN with Five Clusters ($k = 5$)

number of clusters in terms of the energy consumption. One well-known technique of finding the optimal number of clusters when using K-Means is the Elbow Method [81, 82, 83]. The Elbow Method as the name indicates is widely used with continuously increasing or decreasing behaviours. Specifically it is the process of analysing a scree behaviour to find the point (elbow of the curve) where the most significant variation happens.

Since the collision rate is a decreasing function of the number of clusters, the Elbow Method is adopted in the analysis of the collision behaviour. Increasing the number of clusters decreases the number of nodes in each cluster. In return, less simultaneous transmissions and hence, less packet collision rate and less retransmission attempts. Thus, the transmission delay and energy consumption caused by the retransmission attempts are directly effected by the number of clusters in the network. However, increasing the number of clusters also increases the complexity of the clustering process. Since the energy consumption is vital in LoRaWAN, it is essential to find the optimal number of clusters that provides the most appropriate energy consumption. To find the optimal number of clusters, simulations of the energy consumption for the considered network size of 1000 nodes are carried out under different number of clusters $k = \{1, 2, \cdots, 100\}$. The purpose is to analyse the energy consumption behaviour under different number of clusters.

Figure 3.11 shows a scree plot of E_c where it is continuously decreasing as the number of clusters increase. However, in the considered network scenario it is noticed that the steepest E_c decrease point is at $k = 4$. Afterwards, E_c continues to gradually decrease with a very slight trend in relation to increasing the number of clusters. E_c gradient behaviour forms an Elbow point at $k = 4$. Since the energy consumption after $k = 4$ is not significantly decreasing, $K = 4$ is adopted as the most appropriate number of clusters in this case.

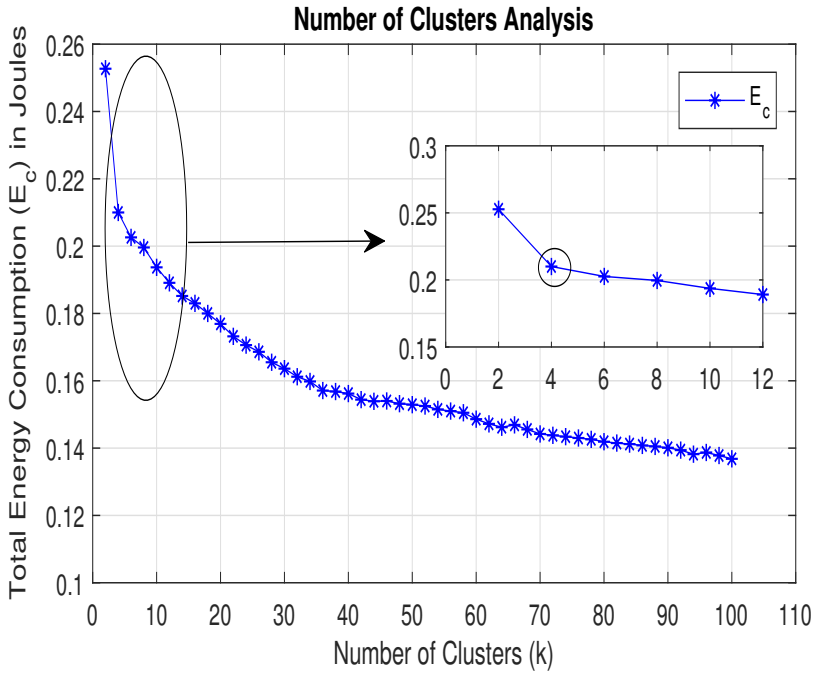


Figure 3.11: Analysis of Total Energy Consumption at a Different Number of Clusters - The Elbow Point Forms At ($k = 4$) Indicating The Most Appropriate Number of Clusters

3.5.3 Total Transmission Delay

The simulation results in Figure 3.12 show a total transmission delay comparison between a typical LoRaWAN network, the proposed clustering based simple transmission *PST* and the collision resolution scheme proposed in [80]. The choice of the work in [80] as a benchmark is to validate the implementation of class *A* LoRaWAN protocol against class *B* and evaluate the impact in terms of transmission delay especially when scaling up the network density to 1000 nodes.

From Figure 3.12, it can be noticed that *TTD* in typical LoRaWAN when serving a network of 1000 nodes reaches up to almost 70s. This is due to number of factors such as the initial transmissions of the considered 1000 nodes, the retransmissions attempts and the waiting time between the initial transmission and retransmissions attempts, all as a result of the high collision rate shown earlier in Figure 3.7. The proposed K-Means clustering based simple transmission *PST* is evaluated under a different number

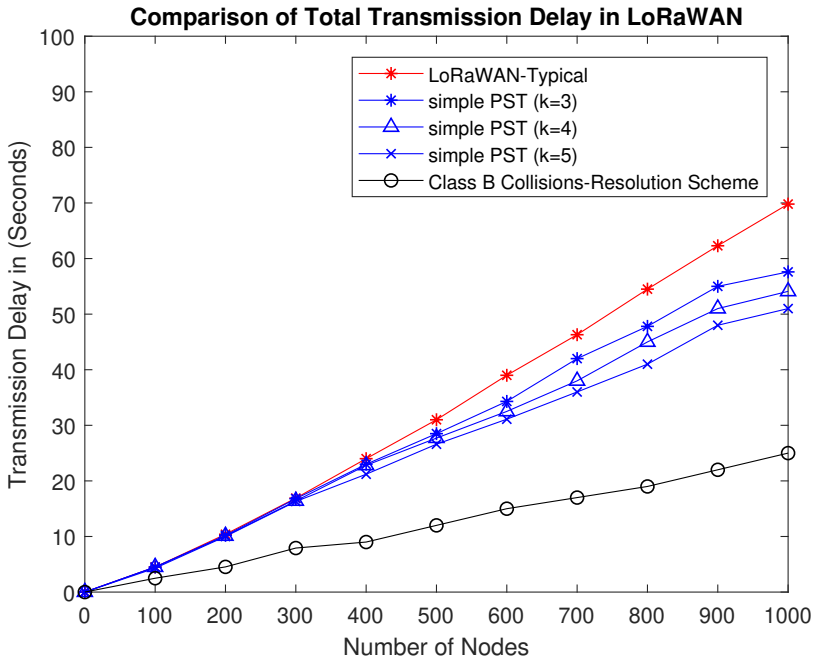


Figure 3.12: The Impact of Reducing the Number of Clusters on The Network Total Transmission Delay

of clusters. When $k = 3$ *clusters*, the delay is reduced to 57.6s. This reduction is a result of reducing the collision rate in each of the clusters and therefore, reducing the number of nodes competing at the same time as shown earlier in Figure 3.8. When $k = 4$ *clusters*, the delay is further reduced to 54.1s. Again this is a result of minimising the collision rate as shown earlier in Figure 3.9. When $k = 5$ *clusters*, the delay is reduced further to 51s due to the reduction of the number of nodes in each cluster. Therefore, it has the lowest collision rate as shown earlier in Figure 3.10. It is noticed that increasing the number of clusters leads to minimising the number of nodes in each cluster and hence, minimising the collision rate in the network. As in Equations (3.8), (3.9) and (3.10), retransmissions of collided packets have a severe impact on the total delay. Hence reducing collision rate results in reducing TTD in a similar manner to reducing the energy consumption.

Comparing against the periodic beacons based collision resolution scheme proposed by Rachkidy and *et.al.* in [80], the transmission delay is enhanced more than that of typical LoRaWAN and the proposed simple transmission *PST*. However, their technique is

Table 3-G: Performance Comparisons in Terms of Collision Rate, Packet Delivery Rate and Transmission Delay

-	Total Collision Rate (%)	Total *PDR* (%)	TTD	
Number of Nodes	1000 nodes	1000 nodes	300 nodes	1000 nodes
Typical LoRaWAN (both simulation and in [84])	90%	10%	17s	70s
Proposed $k = 3$ Technique	58%	42%	17s	58s
Proposed $k = 4$ Technique	45%	55%	16s	54s
Proposed $k = 5$ Technique	41%	59%	16s	51s
Periodic Beacon Based Technique [80]	-	-	8s	25s

based on periodic beacons according to class B of LoRaWAN protocol. Devices operating on class B listen to beacons produced by the gateway and transmit only when there is an available beacon. This makes class B far more power consuming, which defeats the core purpose of using LoRaWAN for serving dense resource-limited IoT devices (battery-powered). Table 3-G shows a comparison performance evaluation of all simulated scenarios.

It is noticed that in Table 3-G the transmission delay at a smaller network scale (300 nodes) has very small variations. However, the variations increase when scaling up the network size (up to 1000 node). This is due to the increased number of nodes within each cluster and hence, more packet collisions. The work carried out has shown that lower collision rate is accompanied with higher *PDR* and lower transmission delays. Hence, more applicability of adopting LoRaWAN for dense IoT applications using the proposed clustering based simple transmission *PST*.

3.6 Chapter Summary

The packet collisions in LoRaWAN play a significant role especially when serving dense networks with up to 1000 nodes. This drawback is due to the random transmissions from the nodes, leading to a packet collision rate that exceeds 90%. Hence, introducing the clustering to LoRaWAN have significantly reduced the collision rate. The collision rate was reduced to 58% at $k = 3$ clusters. Increasing the number of clusters further reduced the collision rate to 45% and 41% at $k = 4$ and $k = 5$, respectively. In return PDR is improved as a result of reducing the collision rate. Although, increasing the number of clusters results in a lower collision rate, it was essential to obtain the optimal number of clusters in order to limit the clustering process complexity. Since LoRaWAN is aimed at serving resource-limited IoT devices (battery-powered), the optimal number of clusters ($k = 4$) was found based on the most appropriate total energy consumption of the network. Introducing the clustering to LoRaWAN reduced the number of retransmission attempts caused by packet collisions. Hence, the total transmission delay when $k = 4$ is also reduced by 23% in comparison to typical LoRaWAN.

Chapter 4

Dynamic Transmission Priority Scheduling Technique

4.1 Overview

Despite the benefits of LoRaWAN protocol for IoT applications, it still suffers from excessive random and simultaneous transmissions due to the adoption of ALOHA protocol. Therefore, resulting in a severe packet collision rate as the network scales up. This leads to continuous retransmission attempts, which in return increases the transmission delay and energy consumption. Thus, this chapter reveals the proposition of a dynamic transmission Priority Scheduling Technique (PST) based on the unsupervised learning clustering algorithm to enhance the network's transmission delay, energy consumption and packet delivery rate. Particularly, the LoRa gateway classifies the nodes into different transmission priority clusters. While the dynamic transmission *PST* allows the gateway to configure the transmission intervals for the nodes according to the transmission priorities of the corresponding clusters. Furthermore, the dynamic transmission *PST* involves a decision making property to assess the necessity of the re-transmission attempts of collided packets in each cluster. The purpose is to preserve the energy consumption via

eliminating unnecessary transmissions. This work allows scaling up the network density while maintaining low packet collision rate and significantly enhancing the transmission delay & the energy consumption. Simulation results show that the proposed work outperforms the typical LoRaWAN and recent clustering & scheduling schemes. Therefore, the proposed work is well suited for dense applications in LoRaWAN.

4.2 System Model, Problem Statement and Formulation

The system model, the considered dense application scenario and the impact of packet collision rate are revealed in Section 4.2.1. This is followed by formulations of the Total Transmission Delay (TTD) and Total Energy Consumption (TEC) in Section 4.2.2. Where Section 4.2.3 introduces the unsupervised learning clustering algorithm (K-Means) and reveals the optimal number of clusters analysis. The notations used in the rest of the chapter are presented in Table 4-A.

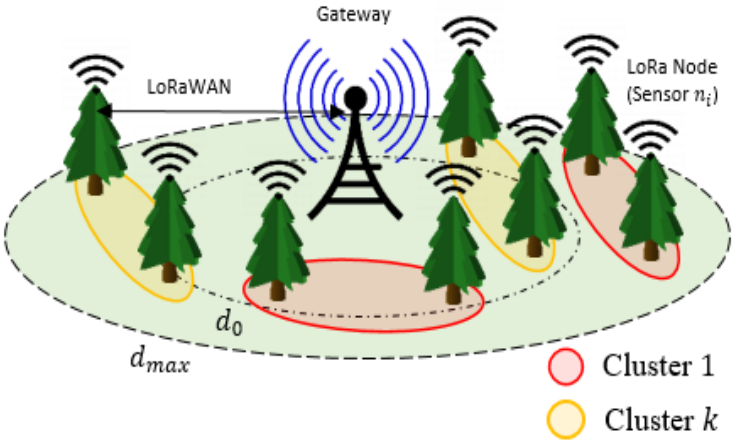


Figure 4.1: Dense Application Resembling a Forest Scenario using LoRaWAN

4.2.1 System Model

The system model shown in Figure 4.1 resembles a forest scenario with one gateway (GW) and randomly distributed nodes $n \in \{n_1, n_2, \ldots n_i\}$, where $1 \leq i \leq 1000$. The LoRa nodes n_i are configured following LoRa SX1272 model. This is to validate n_i's performance against practical experiments carried out in [78, 79, 47]. The nodes are

Table 4-A: List of Notations

Notation	**Description**
A_{C_K}	Average value of A transmitted by all n_i in the corresponding C_K
A_{n_i}, B_{n_i}	A,B values transmitted by n_i
A_{nC_K}	A value transmitted by n_i in the corresponding C_K
B_{nC_K}	B value transmitted by n_i in the corresponding C_K
B_{C_K}	Average value of B transmitted by all n_i in the corresponding C_K
C_{Pr}	Cluster's transmission priority
C_{HPr}	Higher transmission priority cluster
C_{LPr}	Lower transmission priority cluster
C_K	Clusters of $K \in \{1, 2, ..., 5\}$
c_j	Cluster's center point
C_k	Clusters of $k \in \{1, 2, 3, ..., 30\}$
$D_{R_{colli}}$	Daley of n_i's retransmission of collided packets
$D_{R_{ch}}$	Daley of n_i's retransmission of lost packets
D_{I_T}	Daley of n_i's initial packet transmission
$D_{C_{HPr}}$	Delay in of higher priority clusters
$D_{C_{LPr}}$	Delay in of lower priority clusters
D_{active}	Duration of n_i in active mode
D_{idle}	Duration of n_i in idle mode
d	n_i's distance from GW
D_{C_K}	Total transmission delay of a cluster of K
d_0	Distance from GW (500 $meters$)
d_{max}	Maximum distance from GW ($3km$)
E_{active}	Energy consumption of active transmitting cluster ($Joules$)
E_{idle}	Energy consumption of idle cluster ($Joules$)
G	Rate of packet transmission attempts per node
GW	LoRa gateway
I_T	Initial packet transmission
K_{opt}	Optimal number of clusters, ($K = 5$)
$n_{C_{HPr}}$	n_i in higher priority clusters
$n_{C_{LPr}}$	n_i in lower priority clusters
$n_{i_{Pr}}$	i^{th} node transmission priority
n_{C_K}	Node corresponding to a cluster of K
n_i	i^{th} LoRa node (sensor)
n_{i_d}	i^{th} node distance from GW
Pr	Transmission priority, $Pr \in \{LP, LMP, MP, UMP, HP\}$
R_{colli}	Retransmission due to a packet collision
R_{ch}	Retransmission due to a packet loss
S_n	Transmission slot
T_n	Transmission time interval
T_m	Transmission mode
$T_{m_{con.}}$	Conservative transmission mode
$T_{m_{ncon.}}$	Non-conservative transmission mode
$Th_{con.}$	Threshold value of z for $T_{m_{con.}}$
$Th_{ncon.}$	Threshold value of z for $T_{m_{ncon.}}$
v_{ab}	the normalisation value of A & B
z_{C_K}	Average value of z in cluster of K
z_{n_i}	Difference between n_i's values A & B

stationary and communicate with the GW following class A LoRaWAN protocol, while the GW communicates back through a temporary receive window that opens following

each transmission from n_i [1].

Note that this work is based on applying a clustering algorithm, which usually incorporates the conventional solution of deploying multiple GWs in order to provide transmission alternatives for the nodes in different clusters. However, this is not the case in this work for a number of reasons. Firstly, deploying multiple GWs introduces a set of problems, some of which are multipath propagation, interferences and nodes transmission duplication just like the problem reported in [73]. Secondly, multiple GWs are usually ideal in protocols where the the low-latency and ultra-reliability requirements are critical, for example, cellular networks ($5G$). However, this is not always the case in low power protocols like LoRaWAN, especially when the energy resources are limited [1]. Therefore, considering more than one GW could deviate the scope of this work away from evaluating the feasibility of adopting LoRaWAN for severing dense IoT applications as a worse-case scenario. Finally, mainly due to the *CSS* technique in LoRa modulation, a LoRa GW is reported in several studies and experiments to be capable of serving thousands of nodes [85, 86, 87]. Hence, the complexity in this work lies in reducing the collision rate and therefore, enhancing the *PDR* in LoRaWAN while maintaining relatively low *TEC* and *TTD*, all using a single GW.

For that, this work scenario considers two sets of random values (A) and (B) that are assigned to each node n_i following random-uniform distribution[1]. The node n_i transmits these values to the GW, where the clustering is formed. Based on these values the node is assigned to the corresponding cluster. This is for the purpose of evaluating the wildfire possibility within the covered area.

In [47, 79], two different experimental projects using LoRaWAN were carried out to evaluate the channel condition impact on the *PDR*. Both were carried out in urban environments where obstacles are highly deployed between the nodes and the GW. Both showed that the node's distance n_{i_d} from the GW has a great impact on the *PDR*.

[1] These values can be adjusted according to any application. As for the forest scenario adopted in this work, the values A and B represent atmospheric humidity and weather temperature, respectively.

Specifically, n_i located at a distance (d) more than 500 meters (d_0) away from the GW ($n_{i_d} \geq d_0$) experiences a bad channel condition, where the PDR ranges between 50% to 90%. On the other hand, the PDR is guaranteed more than 90% when $n_{i_d} \leq d_0$.

Based on the considered forest scenario and the results in [47, 79], only one fifth of the nodes are distributed within a range of d_0 from the GW. These nodes are assumed to have a good channel condition with a PDR more than 90%. The rest of the nodes are distributed at distances range from d_0 up to 3000 meters (d_{max}). These nodes have bad channel condition with a PDR that can be deprived down to 50%. In other words, for a more realistic system model only one fifth of the nodes have good PDR of more than 90% while the rest are vulnerable to packet loss.

In addition, the nodes communicate with the GW using class A of LoRaWAN protocol, spreading factor ($SF7$) and coding rate of 4/5. These parameters are particularly chosen to provide the maximum data rate, lowest transmission delays and lowest energy consumption for the network. Although the LoRaWAN network with the most reliable $SF7$ provides the best performance in terms of data rate, transmission delay and energy consumption [1, 20], the LoRaWAN network still under-performs in certain scenarios due to high collision rate, especially when the network is dense. Hence, given the system model is dense at a limited area (up to d_{max} around the GW), the analysis in this work is based on using $SF7$.

The main objective of this work is to enhance the TTD and TEC. This is achieved by reducing the excessive packet collision rate associated with LoRaWAN due to the adoption of ALOHA protocol communication in class A LoRaWAN [88]. Packet collisions happen when two packets are transmitted at the same time over the same frequency using the same SF [1, 20]. When a collision happens, the node keeps attempting to retransmit until an acknowledgement from the GW is received, which results in increasing the TTD and eventually the TEC.

Since LoRaWAN adopts ALOHA protocol for communications between the nodes and the

GW, the node transmits packets whenever there are ready to transmit data, regardless of the channel status. Hence, following Poisson distribution, the probability P of a packet collision to happen is given as in Equation (4.1):

$$P = e^{-2G} \tag{4.1}$$

where G is the rate of packet transmission attempts per node. Hence, having more nodes transmitting at the same time increases the probability of a packet collision. Simulations are carried out in Section 4.4 to show the proportional relationship between the number of nodes and the total collision rate.

Given the limited resources and random transmission behaviour of LoRaWAN nodes, it is essential to minimise the number of simultaneous transmission. Considering the given application scenario with the unlabelled data associated with the nodes, an effective method to reduce the simultaneous transmissions is to partition the nodes into different clusters. Assuming sufficient resources for the gateway, K-Means can be adopted to perform clustering of the nodes based on their transmitted data. Therefore, the gateway applies the dynamic transmission *PST* to regulate the nodes transmissions without exhausting the nodes limited resources in the transmission intervals configuration process.

Since the aim is to provide a dynamically control the transmissions between the nodes and the *GW*, formulations of *TTD* and *TEC* are essential in order to evaluate the network performance. The following subsection reveals *TTD* and *TEC* as functions of K number of clusters (C_K). In addition, the formulations and analysis show that *TEC* is proportional to *TTD*.

4.2.2 Problem Formulations

4.2.2.1 Total Transmission Delay (*TTD*)

In the proposed dynamic transmission *PST* (Section 4.3), the *GW* assigns different transmission priorities Pr to K number of clusters C_K, where $K = \{1, 2, 3, ..., k\}$. The

nodes in a lower Pr cluster wait until transmissions from nodes in higher Pr clusters are satisfied. This introduces waiting times in lower transmission priority clusters. Hence, TTD can be given as in Equation (4.2):

$$TTD(K) = \sum_{j=1}^{K} (D_{C_1}, D_{C_2}, ..., D_{C_K}), \tag{4.2}$$

where K is the number of clusters in the network, D_{C_K} is the total transmission delay of n in a cluster of C_K and is given as in Equation (4.3):

$$D_{C_K} = \sum_{i=1}^{n_{C_K}} X(i), \tag{4.3}$$

where n_{C_K} is the total number of all n_i in the corresponding C_K; $X(i) = (D_{I_T} + D_{R_{colli}} + D_{R_{ch}})$; D_{I_T} is the delay of the initial transmission I_T of each n_i; $D_{R_{colli}}$ is the delay of the retransmission caused by n_i's collided packet (R_{colli}); $D_{R_{ch}}$ is the delay of the retransmission caused by n_i's lost packet due to bad channel condition (R_{ch}). Note that the GW is assumed to be able to distinguish between I_T, R_{colli} and R_{ch}.

Since the transmissions from n_i in the lower Pr clusters C_{LPr} wait until transmissions from n_i in the higher Pr clusters C_{HPr} are satisfied, the delay of $D_{C_{LPr}}$ is given as in Equation (4.4):

$$D_{C_{LPr}} = D_{C_{HPr}} + \sum_{i=1, i \notin C_{HPr}}^{n_{C_{LPr}}} X(i), \tag{4.4}$$

where $D_{C_{HPr}}$ is the delay of all n_i in higher Pr clusters and $n_{C_{LPr}}$ is n_i in the corresponding C_{LPr} clusters.

4.2.2.2 Energy Consumption

The GW using the proposed dynamic transmission PST (detailed in Section 4.3) regulates the transmissions from n_i in different clusters of C_K to the GW based on the

corresponding transmission *Pr*. Hence, each of C_K is either at an active or idle transmission status. When a cluster of C_K is at an active status, the corresponding n_{C_K} are allowed transmissions. Otherwise, the cluster is at an idle transmission status, and no transmissions from the corresponding n_{C_k}. Note that only one cluster of C_K can be active at a time. Hence, *TEC* as a function of the number of clusters *(K)* can be given as in Equation (4.5):

$$TEC(K) = E_{active} + \sum_{j=1, E_{active} \notin j}^{K} E_{idle}^{(j)} \tag{4.5}$$

where E_{active} and E_{idle} are the energy consumption in *Joules* of all n_{C_K} in the corresponding active and idle clusters of C_K, respectively. E_{active} and E_{idle} are given in Equations (4.6) and (4.7), respectively.

$$E_{active} = \sum_{i=1}^{n_{C_k}} (P_T \times D_{active}), \tag{4.6}$$

$$E_{idle} = \sum_{i=1}^{n_{C_k}} (P_{idle} \times D_{idle}), \tag{4.7}$$

where P_T and D_{active} are the transmission's power and duration of n_i in an active cluster of C_K. While P_{idle} and D_{idle} are the power consumption and the duration of standby n_i in the other idle clusters of C_K.

From Sections 4.2.2.1 and 4.2.2.2, the *TTD* and *TEC* are proportionally impacted by the number of n_i's initial transmissions and retransmissions of collided or lost packets. In other words, minimising *TTD* eventually results in minimising *TEC*. Hence, from Equations (4.2) and (4.5), the objective function of obtaining the minimum value of *TTD* at a given number of clusters K can then be represented as in Equation (4.8), subject to a number of constraints:

$$\min_{k} TTD(K) \tag{4.8}$$

$$S.T.$$

$$\min TEC(K) \tag{4.9}$$

$$Pr = K \tag{4.10}$$

$$R_{colli} = \begin{cases} 1, & \text{in case of a collision} \\ 0, & \text{else} \end{cases} \tag{4.11}$$

$$R_{ch} = \begin{cases} R_{colli} + 1, & d_0 < n_{i_d} < d_{max} \\ R_{colli}, & d_0 > n_{i_d} \end{cases} \tag{4.12}$$

where constraint (4.9) denotes the proportionality of TEC to TTD at a given number of clusters K. Pr in constraint (4.10) is the transmission priority assigned to each cluster of K. The process of assigning Pr to each cluster is revealed in Section 4.3. R_{colli} in constraint (4.11) is the retransmission of collided packets and it is limited to one retransmission per node. This is to retain the practicality of simulations given the considered high number of nodes. R_{ch} in constraint (4.12), is the retransmission of lost packets due to bad channel condition for nodes n_i located at distances further than d_0 from the GW.

4.2.3 Unsupervised Learning Clustering Algorithm (K-Means)

The evaluation of the impact of adopting the unsupervised clustering algorithm K-Means on the total collision rate is revealed in Chapter 3 [89]. In fact, it was noticed that the TTD and TEC are decreasing function of the number of clusters. However, in K-Means, the number of clusters k is a predefined value. Hence, in this work the optimal number of clusters is obtained according to the most efficient performance of TTD and TEC against the number of clusters k.

The partition of the nodes takes place by minimising the within-cluster sum of square (WCSS) of the given data set $z_n = \{z_{n_1}, z_{n_2}, ...z_{n_i}\}$, where z_{n_i} is the difference between the values A and B, which are transmitted by n_i as explained in the system model.

Since the values A and B can be measured in different units, a normalisation is needed to obtain the value of z. In other words, nodes with almost similar values of z are grouped together forming one cluster. Note that the clustering is based on the values of z_n. This means that nodes at different locations from the GW can belong to the same cluster, see Figure 4.1. The objective partitioning function can then be represented as in Equation (4.13):

$$\arg\min_{c_j} \sum_{j=1}^{k} \sum_{z_{n_i} \in C_k} \|z_{n_i} - c_j\|^2 \tag{4.13}$$

where c_j is an initial value of z fixed to form the center point of the corresponding cluster C_k. Note that the clustering is formed at the GW level. The GW is assumed to be aware of z_n from previous successful transmissions. z_{n_i} is then updated at the GW upon each successful transmission from the corresponding n_i.

K-Means is a suitable unsupervised machine learning clustering tool for networks with limited resources. This is due to the simplicity of performing the clustering process provided unlabelled data[90]. However, since the number of clusters play a vital role in the clustering process, implementing K-Means can be very complex in the case of excessively diversified data. Hence, it is very important to define objectives that can be used to evaluate the optimal number of clusters.

In order to obtain the optimal number of clusters, extensive simulations were carried out to evaluate TTD and TEC performance at a different number of clusters k, where $(0 \leq k \leq 30)$. Based on Equations (4.2) and (4.5), Figure 4.2 shows that TTD and therefore, TEC are decreasing functions of k. On one hand, it is noticed that TTD and TEC sharply decrease until $(k = 5)$. This is mainly due to the reduction in the number of nodes within each cluster and hence the reduction in the collision rate caused by nodes transmitting at the same time. Note that collided packets get retransmission attempts, which impact both TTD and TEC in an almost symmetrical manner. On the other hand, at $(5 \leq k)$, TTD and TEC start to regain their values forming convex curves. This is due to the gradually fading impact of the retransmissions caused by packet collisions

R_{colli}, and the increasing impact of the retransmissions caused by packet loss due to bad channel conditions R_{ch}. Thus, from Figure 4.2, the optimal number of clusters K_{opt} in this scenario is at $(k = 5)$, where TTD and TEC at the bottom points of the convex curves forming the lowest values. In the case of different traffic model, the process of obtaining the objective functions of minimum TTD and TEC is executed in a similar manner. Therefore, the optimum number of clusters could vary depending on the traffic model from the optimum number specified in Figure 4.2. However, the variance follows the aim of minimising TTD and TEC as per Equation 4.8.

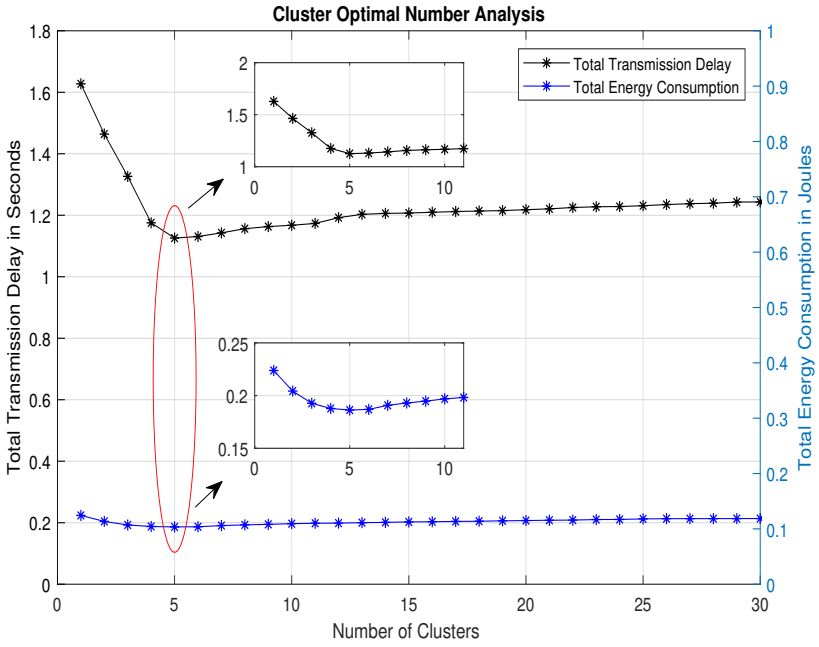


Figure 4.2: Simulation Analysis of The Optimal Number of Clusters in Terms of TTD and TEC - The Convex Point Forms At (k=5) Indicating The Optimal Number of Clusters

4.3 Proposed Dynamic Transmission Priority Scheduling Technique

The dynamic transmission Priority Scheduling Technique (PST) takes place at the GW level where it schedules transmissions from the nodes according to different transmission priorities assigned to the different clusters that are obtained by K-Means. Initially, a

set of values z_{n_i} transmitted to the GW from each n_i is processed to partition n_i to different clusters. Following the clustering formation, each n_i is assigned to a cluster of C_K. Let z_{C_K} denote the average value of all z_{n_i} within the same cluster of C_K. Based on z_{C_K}, the GW using the dynamic transmission PST designates different transmission priorities (Pr) to the different C_K. Since the optimal number of clusters is $K_{opt} = 5$, the transmission priorities range between lowest, lower-middle, middle, upper-middle and highest, where $Pr = K_{opt}$ and $Pr \in \{LP, LMP, MP, UMP, HP\}$, respectively.

Following the transmission priority designation process to each cluster of C_K, the dynamic transmission PST provides two transmission modes to trade-off TTD and TEC for further PDR gain according to each cluster transmission priority C_{Pr}. Given the density in the network, the GW using the Naive Bayes classifier [91] determines the probability of each n_i to transmit using a certain transmission mode.

4.3.1 Transmission Priority Scheduling

For better elaboration, it is necessary to explain the details of the considered scenario in this work. The GW assigns n_i that has the highest value of z_{n_i} to the highest transmission Pr cluster. To reiterate, z is the difference between the values A & B, where A_{n_i} & B_{n_i} represent the atmospheric humidity and weather temperature values transmitted by n_i, respectively. Since A & B can be measured in different units, a normalisation is needed to obtain the value of z. There is a number of normalisation methods [92, 93, 94, 95], which vary in terms of the considered values. Considering the scenario adopted in this work, A & B are given as upward and downward attributes[2]. Hence, the enhanced max-min normalisation method is adopted for adjusting the normalisation value

[2]Referring the used data set, in a forest scenario it is less likely for a wildfire to happen when the atmospheric humidity (A) is high, while the wildfire possibility increases with lower values of A. Hence, the value A is considered as an upward attribute. Vice versa, it is less likely for a wildfire to happen when the weather temperature (B) is low and the possibility increases with higher values of B. Hence, the value B is considered as a downward attribute.

$v_{ab} = f(A, B)$ and is given as in Equation (4.14):

$$v_{ab} = \begin{cases} \text{for upward attributes :} \\ 1 - \dfrac{|A_{n_i} - max(A_{n_{C_K}})|}{(max(A_{n_{C_K}}) - min(A_{n_{C_K}}))} \\ \\ \text{for downward attributes :} \\ 1 - \dfrac{|B_{n_i} - min(B_{n_{C_K}})|}{(max(B_{n_{C_K}}) - min(B_{n_{C_K}}))} \end{cases} \tag{4.14}$$

where $A_{n_{C_K}}$ and $B_{n_{C_K}}$ represent A and B values reported by n_i in the corresponding cluster C_K.

The designation process of the Pr level to each cluster of C_K follows Equation (4.15):

$$\max(Pr) = \max(z_{C_K}) \tag{4.15}$$

where z_{C_K} denotes the average value of z in the corresponding cluster C_K. Note that $z_{C_K} = A_{C_K} - B_{C_K}$, where A_{C_K} and B_{C_K} denote the average values of A and B in the corresponding cluster C_K, respectively.

For further illustration, the transmission cycle used in the simulations of this work is depicted in Figure 4.3 to show the Pr designation process. From Figure 4.3, the Pr is proportional to z_{C_K}. This means the higher value of z_{C_K} is assigned higher Pr. Based on the cluster transmission priority C_{Pr}, the GW configures transmissions from the corresponding nodes accordingly. In other words, the GW allows transmissions from n_i in higher Pr clusters $n_{C_{HPr}}$, where it blocks transmissions from n_i in lower Pr clusters $n_{C_{LPr}}$.

This strict condition introduces a network under-performance for some cases. For example, given a dense application, the transmissions from C_{LPr} can be blocked due to the presence of excessive and unnecessary transmissions from C_{HPr} that may not be

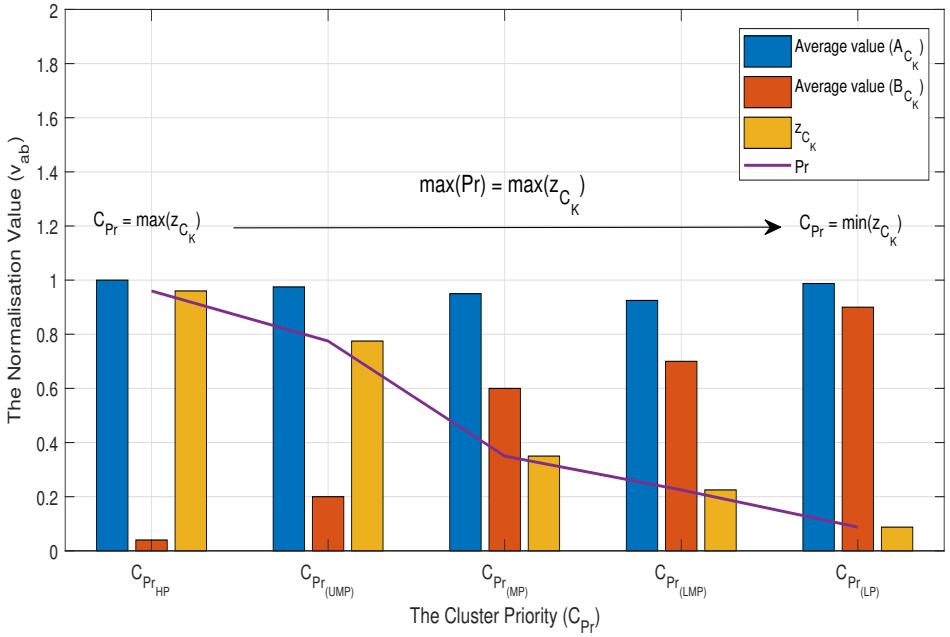


Figure 4.3: Transmission Priority Designation to Clusters of C_K Based on The Corresponding Value of z_{C_K}

desirable. In order to achieve further performance gain, the proposed dynamic transmission PST is performed under two transmission modes (T_m): conservative ($con.$) and non-conservative ($ncon.$).

Based on the following assumptions, Algorithm 1 shows the process of the transmission initiations from each n_i according to its corresponding cluster.

Assumptions:

- GW already has z_{n_i} for all the nodes from previous successful transmissions
- z_{n_i} at the GW are updated upon each successful transmission
- Each n_i transmit one packet an hour to the GW unless configured otherwise
- Each n_i is allowed only one retransmission in the case of a collision R_{colli}
- Each n_{i_d} at $d_0 \leq d \leq d_{max}$, is allowed one retransmission in the case of a

packet loss R_{ch} due to bad channel condition

– The environment is idle, where there are no inter communications exist and the channel duty-cycle constraint is neglected

Algorithm 1 Transmission Priority Scheduling Process

At the *GW* **level**
Initialize: TTD, TEC, n_i, z_{n_i}, $n_{i_{Pr}}$, $Pr \in \{LP, LMP, MP, UMP, HP\}$,
To achieve $min\,TTD$ & $min\,TEC$;
1: **for** n_i **do**
2: **if** $n_i \in HP$ **then**
3: $n_i := n_{i_{HP}}$ and $n_{i_{HP}}$ transmission = 1;
4: **else if** $n_i \in UMP$ and $(n_{i_{HP}}) = 0$ **then**
5: $n_i := n_{i_{UMP}}$ and $n_{i_{UMP}}$ transmission = 1;
6: **else if** $n_i \in MP$ and $(n_{i_{HP}}, n_{i_{UMP}}) = 0$ **then**
7: $n_i := n_{i_{MP}}$ and $n_{i_{MP}}$ transmission = 1;
8: **else if** $n_i \in LMP$ and $(n_{i_{HP}}, n_{i_{UMP}}, n_{i_{MP}}) = 0$ **then**
9: $n_i := n_{i_{LMP}}$ and $n_{i_{LMP}}$ transmission = 1;
10: **else if** $n_i \in LP$ and $(n_{i_{HP}}, n_{i_{UMP}}, n_{i_{MP}}, n_{i_{LMP}}) = 0$ **then**
11: $n_i := n_{i_{LP}}$ and $n_{i_{LP}}$ transmission = 1;
12: **else**
13: $n_{i_{Pr}}$ transmission = 0;
14: **end if**
15: **end for**

4.3.2 Transmission Modes Options

The two transmission modes T_m (*con.* and *ncon.*) are provided by the dynamic transmission *PST* to trade-off *PDR* with *TTD* and *TEC*. T_m is defined based on whether or not retransmissions of collided packets R_{colli} in each C_K are permitted. In other words, the purpose is to allow the *GW* to assess whether there is a need for R_{colli}, hence control the *PDR* accordingly. The *GW* decides which mode to operate for each C_K based on threshold values (Th) assumed to be provided by a third party (e.g local authority). These Th are $Th_{con.}$ for *con.* mode and $Th_{ncon.}$ for *ncon.* mode. Furthermore, $Th_{con.}$ and $Th_{ncon.}$ contain a set of values of z that act as limits. The *GW* uses these limits in order to determine the probability of using one of the two T_m by each C_K.

In *con.* mode, the *GW* allows retransmissions of collided packets (R_{colli}). Note that for

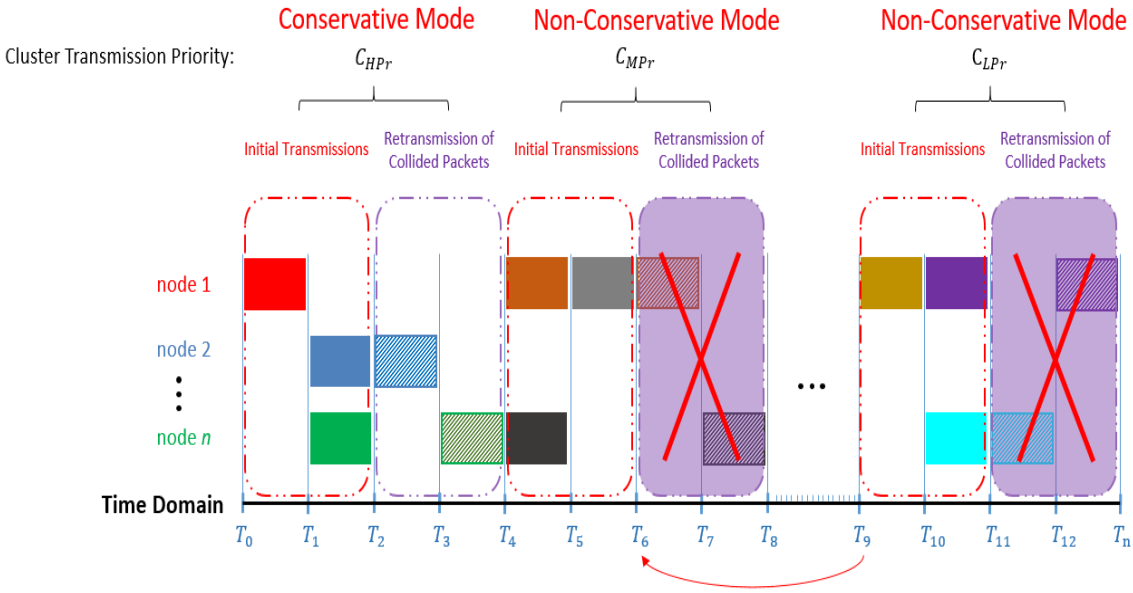


Figure 4.4: Transmission Modes Control - Higher Priority Clusters are Allowed Retransmissions for Better *PDR* Levels and Lower Priority Clusters are Detained From Retransmissions to Preserve Energy and Enhance Transmission Delay

simulation practicality the n_i with collided packet is allowed one retransmission attempt. In the case of any further collisions, the collided packets will be dropped. Operating the *con.* mode elevates *PDR* at the expense of higher *TTD* and *TEC*. On the other hand, R_{colli} in *ncon.* mode is not allowed. Operating the *ncon.* mode minimises *TTD* and *TEC* at the expense of lower *PDR*. Simulations results in Section 4.4 show that both transmission modes T_m maintain acceptable *PDR* in comparison to other techniques. Figure 4.4 illustrates the difference between both *con.* and *ncon.* transmission modes. While Algorithm 2 shows the dynamic transmission *PST* process of alternating between the two transmission modes according to Th values.

4.3.3 Naive Bayes Classifier Algorithm

Considering a dense application with a massive amount of transmissions from n_{C_K} to *GW*, the process of classifying C_K to a certain T_m can be time inefficient. For this reason the *GW* applies the Naive Bayes classifying algorithm to efficiently determine

Algorithm 2 Dynamic Transmission *PST* Transmission Modes

At the *GW* **level**
Initialize: z_{C_K}, $T_{m_{con.}}$, $T_{m_{ncon.}}$, I_T, R_{colli}, R_{ch}, n_{i_d}, d_0, $Pr \in \{LP, LMP, MP, UMP, HP\}$
1: **for** $T_{m_{con.}}(HP, UMP)$ **do**
2: $I_T = 1$;
3: $R_{colli} = 1$;
4: **if** $n_{i_d} > d_0$ **then**
5: $R_{ch} = 1$;
6: **else**
7: $I_T = 0$;
8: $R_{colli} = 0$;
9: $R_{ch} = 0$;
10: **end if**
11: **end for**
12: **for** $T_{m_{ncon.}}(MP, LMP, LP)$ **do**
13: $I_T = 1$;
14: $R_{colli} = 0$;
15: **if** $n_{i_d} > d_0$ **then**
16: $R_{ch} = 1$;
17: **else**
18: $I_T = 0$;
19: $R_{colli} = 0$;
20: $R_{ch} = 0$;
21: **end if**
22: **end for**

the probability of n_{C_K} transmissions using either $T_{m_{con.}}$ or $T_{m_{ncon.}}$. Where $T_{m_{con.}}$ and $T_{m_{ncon.}}$ denote *con.* and *ncon.* transmission modes, respectively. According to Th values, the GW classifies each cluster of C_K to a certain T_m based on the average value of z_{C_K} following Equation (4.16):

$$P(T_{m_{con.}}|z_{C_K}) \geq P(T_{m_{ncon.}}|z_{C_K}) \tag{4.16}$$

where $P(T_{m_{con.}}|z_{C_K})$ is the posterior probability of a cluster of C_K to transmit using $T_{m_{con.}}$ and is given as in Equation (4.17):

$$P(T_{m_{con.}}|z_{C_K}) = \frac{P(z_{C_K}|T_{m_{con.}})P(T_{m_{con.}})}{P(z_{C_K})} \tag{4.17}$$

where $P(z_{C_K}|T_{m_{con.}})$ denote the posterior probability of z_{C_K} conditioned on $T_{m_{con.}}$; $P(T_{m_{con.}})$ is the prior probability of $T_{m_{con.}}$; and $P(z_{C_K})$ is the prior probability of z_{C_K}.

Table 4-B: Likelihood Occurrence Pattern Table

C_K	$T_{m_{con.}}$ $(+\alpha)$	$T_{m_{ncon.}}$ $(+\alpha)$
C_1	1 (2)	0 (1)
C_2	1 (2)	0 (1)
C_3	0 (1)	1 (2)
C_4	0 (1)	1 (2)
C_5	0 (1)	1 (2)
$P(T_m)$	$P(T_{m_{con.}}) = \frac{2}{2+3} = 0.4$	$P(T_{m_{ncon.}}) = \frac{3}{3+2} = 0.6$

Since there are two transmission modes $T_{m_{con.}}$ and $T_{m_{ncon.}}$, a cluster of C_K communicates with GW using one of them. Hence, the corresponding T_m is denoted by 1 when used by a cluster and 0 when not in use. The out of use T_m denoted by 0 can cause inaccurate probability results when using the product function in Equation (4.18). Hence, α is added to avoid such confusion in the Naive Bayes classifying process, where $\alpha = 1$.

In a similar approach, Equation (4.17) is applied to obtain the posterior probability of $P(T_{m_{ncon.}}|z_{C_K})$.

z_{C_K} is an independent value that varies in each cluster of K, which can result in a high computational complexity when obtaining $P(T_{m_{con.}}|z_{C_K})$. In order to reduce the computation complexity, the posterior probability $P(z_{C_K}|T_{m_{con.}})$ can be calculated is in Equation (4.18):

$$\prod_{K=1}^{K_{opt}} P(z_{C_K}|T_{m_{con.}}) = P(T_{m_{con.}}) \times P(z_{C_K}) \tag{4.18}$$

using Equation (4.18), the communications data set in Figure 4.3 is utilised as a training set to construct the likelihood occurrence pattern given in Table 4-B. This is to determine the probability of classifying a cluster of C_K to a certain T_m. In particular, there are K number of clusters, where $K = \{1, 2, ..., K_{opt}\}$. Lets assume a threshold value for the *con.* mode, $Th_{con.} \geq 0.5$, where the value 0.5 represent z_{C_K}. Hence, all clusters with $z_{C_K} \geq 0.5$ are considered as higher priority clusters that more likely need to communicate with the GW using $T_{m_{con.}}$. The rest of the clusters are more likely to communicate with the GW using $T_{m_{ncon.}}$. Thus, based on the Bayesian theorem, the probability of having a cluster assigned to using $T_{m_{con.}}$ is given as in $P(T_{m_{con.}})$ and $P(T_{m_{ncon.}})$. Note that the

classifying process is performed by the GW upon each new transmission cycle, where the T_m probability is determined according to the new values of z_{C_K} reported by each n_i. Figure 4.5 illustrates the dynamic transmission PST using Naive Bayes classifier.

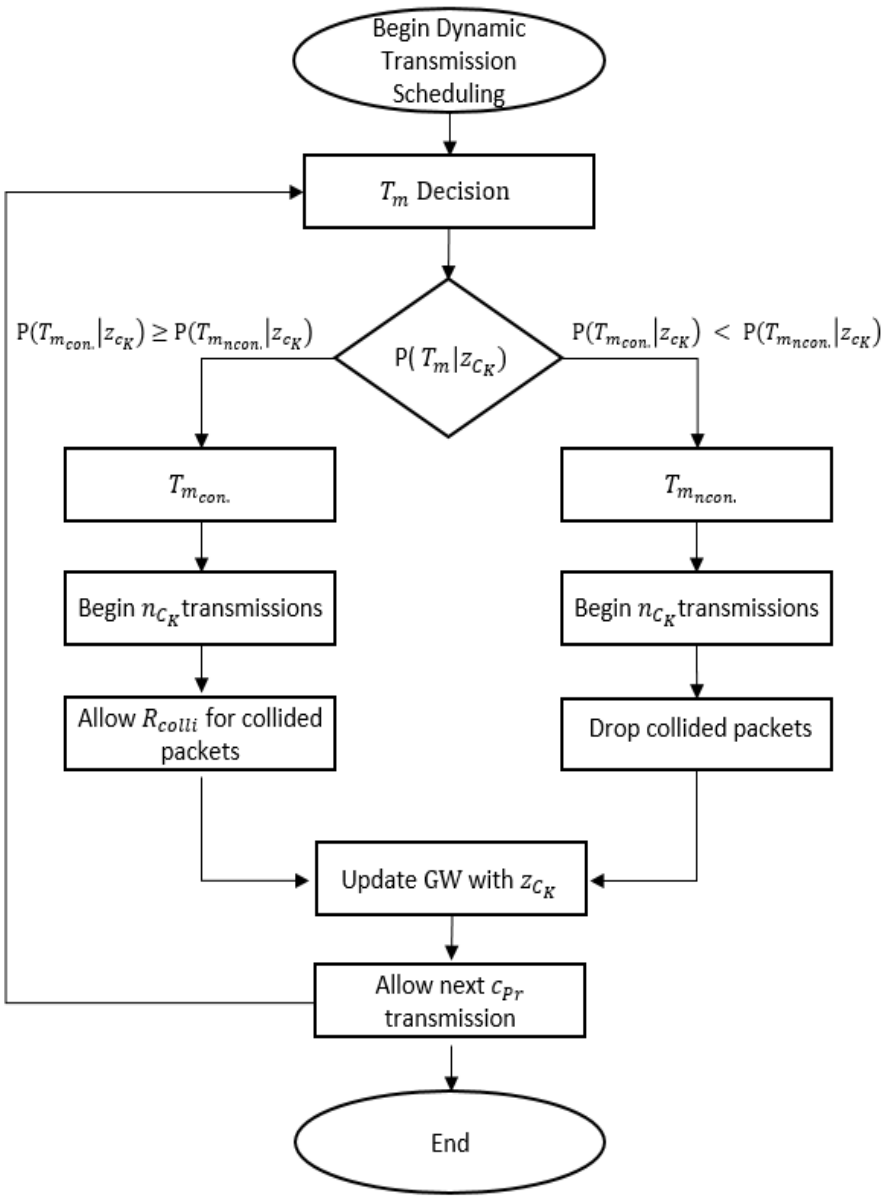


Figure 4.5: Naive Bayes Classifier in Dynamic Transmission PST

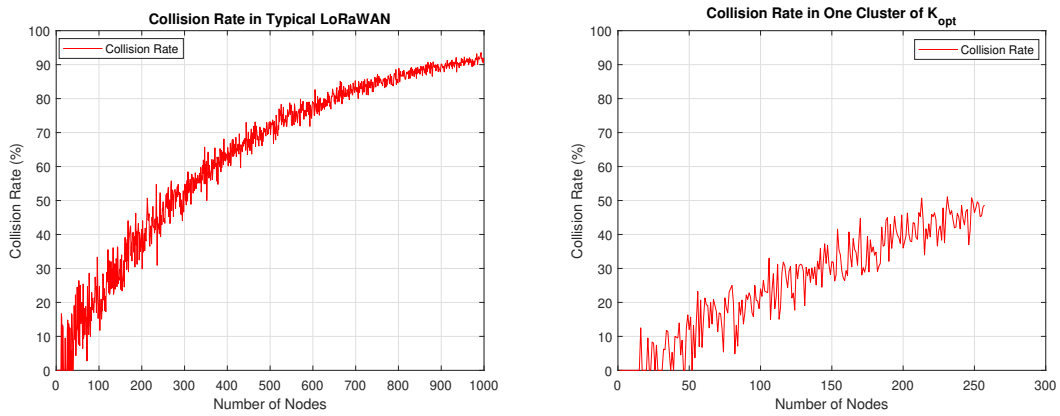


Figure 4.6: Collision Rate in Typical LoRaWAN Vs. Collision Rate in One Cluster of K_{opt}

4.4 Simulation Results and Performance Evaluations

The impact of the proposed dynamic transmission PST on the network performance is evaluated via simulations following the parameters in Table 4-C.

Table 4-C: Simulation Parameters

Parameter	**Value**
Protocol	LoRaWAN (v1.1)
Number of nodes n_i	1000
Payload size	25 $Bytes$
SF	7
CR	4/5
BW	125 kHz
Channel frequency	915 MHz
Power consumption per active n_i	0.1 W
Power consumption per idle n_i	0.072 W
Transmission duration	0.036 s
Optimal number of clusters	$K_{opt} = 5$

It is shown that LoRaWAN is vulnerable to severe collision rate especially when serving a high number of nodes. This is due to the fact that the nodes adopt ALOHA style communication in its LoRaWAN protocol of class A [1, 20]. As shown in Figure 4.6, the total collision rate in typical LoRaWAN network serving up to 1000 n_i is up to 91%. Having introduced the optimal number of clusters K_{opt}, Figure 4.6 shows only one of the five clusters in the network, hence the heterogeneity in the number of nodes

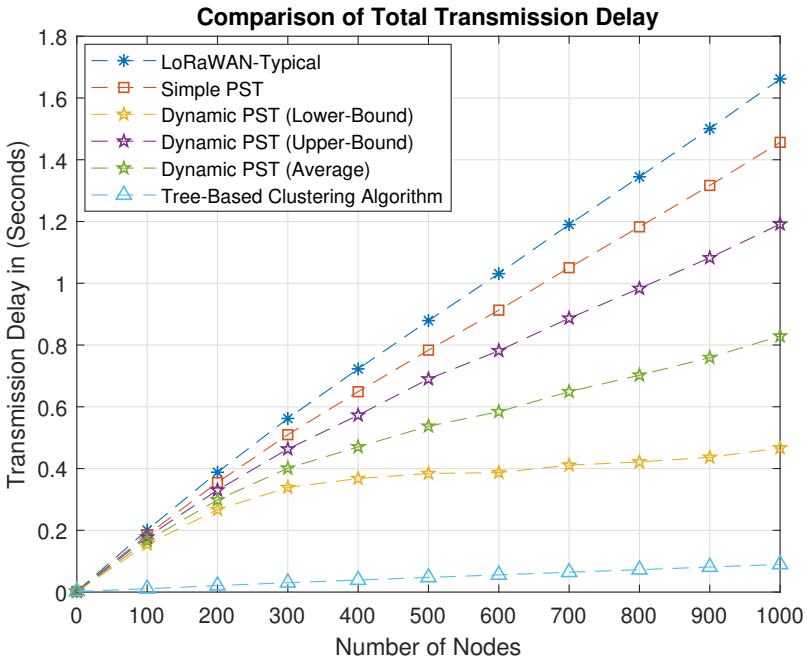


Figure 4.7: Comparison of The Total Transmission Delay Over Different Network Density

between the two figures. Other clusters of K_{opt} follow the same trend and hence, the total collision rate when $k = 5$ is vastly reduced to an average of 38% in comparison to typical LoRaWAN. Note that the more clusters introduced to the system result in less nodes simultaneously transmitting, which in return reflects in less packet collisions. However, this comes at the expense of inefficient *TTD* and *TEC* due to the impact of other factors such as the bad channel conditions (as discussed in Section 4.2.3, Figure 4.2).

In regards to the *TTD*, *TEC* and *PDR*, a comparison of the proposed dynamic transmission *PST* is carried out against a typical LoRaWAN network, the simple *PST* proposed in Chapter 3 [89], and the tree-based clustering algorithm scheme proposed in [19]. The benchmark in [19] is specifically chosen to validate the proposed work in terms of energy efficiency especially in a resource-limited environment such as LoRaWAN.

When observing Figure 4.7 it is noticed that the typical LoRaWAN shows the least efficient *TTD*. This is due to the adoption of star topology accompanied with the ALOHA

protocol communication in LoRaWAN. Where the nodes initiate transmissions to the *GW* regardless of any other transmission occupying the channel. For this reason the collision rate is excessively high especially when scaling up the network. As a result, retransmission attempts are much higher, which in return increase the *TTD* of the network. Note that due to the high number of nodes considered in this scenario and for simulation practicality, the retransmissions of collided packets are limited to one per node. The simple *PST* comes second after typical LoRaWAN with slight improvement to *TTD*. This improvement is mainly due to the clustering where the number of nodes in each cluster is reduced. This results in a lower number of transmission attempts and according to Equation (4.1), the probability of the packet collision to happen is significantly impacted by the transmission attempts which is proportionally related to the number of nodes transmitting at the same time. Although the number of nodes in each cluster is significantly decreased, there are still collisions that happen where each collision results in another retransmission attempt regardless of the necessity of the retransmission.

Therefore, the proposed dynamic transmission *PST* outperformed both typical LoRaWAN and simple *PST*. This is due to the ability to alternate between *con.* and *ncon.* transmission modes. Note that in the simulation results, the dynamic transmission *PST* is represented as *con.* (Upper-Bound) when the majority of clusters are transmitting using $T_{m_{con.}}$, while it is represented as *ncon.* (Lower-Bound) when the majority of clusters are transmitting using $T_{m_{ncon.}}$. It can be noticed that the *TTD* at *con.* is high in comparison to that of *ncon.* mode. This is because the nodes are located in a cluster that is assigned a $T_{m_{con.}}$ and has the chance to initiate a retransmission for each collided packet. Vice versa, the *TTD* at *ncon.* is relatively low due to the strict retransmission condition which results in each collided packet to be dropped.

The tree-based clustering algorithm proposed in [19] shows the best *TTD* amongst all approaches. This is due to the rerouting approach, which results in avoiding a packet collision by utilising multi-hop technique that relay the packet to the *GW* through other

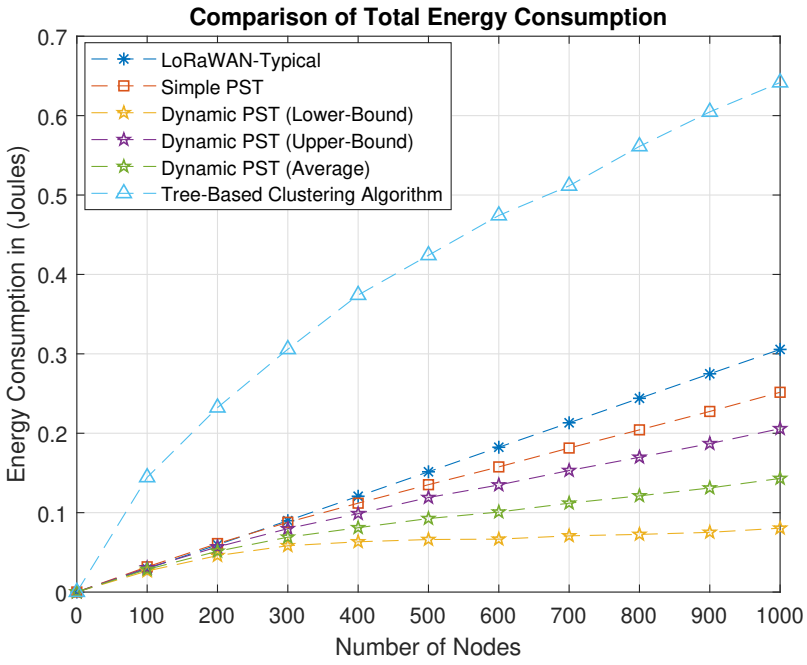


Figure 4.8: Comparison of The Total Energy Consumption Over Different Network Density

routes using neighbouring nodes. However, this comes at the expense of much higher *TEC* as shown in Figure 4.8, which defeats the whole purpose of using LoRaWAN as a low power technology.

Referring back to Section 4.2.2.2, *TEC* is directly affected by the number of collisions and hence the number of retransmissions as a consequence. For that, it can be noticed that *TEC* is generally proportional to *TTD*. In Figure 4.8, *con.* mode (Upper-Bound) shows more energy consumption when comparing to *ncon.* mode (Lower-Bound), however despite it consuming more *TEC*, adopting *con.* mode provides better *PDR* in comparison to *ncon.* mode as shown in Figure 4.9. This comes as a result of allowing nodes in clusters that are transmitting using *con.* to initiate retransmissions of collided packets. Hence, it can be noticed that when *PDR* in *con.* is high, the *TEC* and therefore *TTD* are high, whereas the opposite in *ncon.* mode. From Figure 4.9, the simple *PST* outperforms the proposed dynamic transmission *PST*, however this comes at the expense of more *TTD* and *TEC*. While the dynamic transmission *PST* at *con.* mode shows better *PDR* in

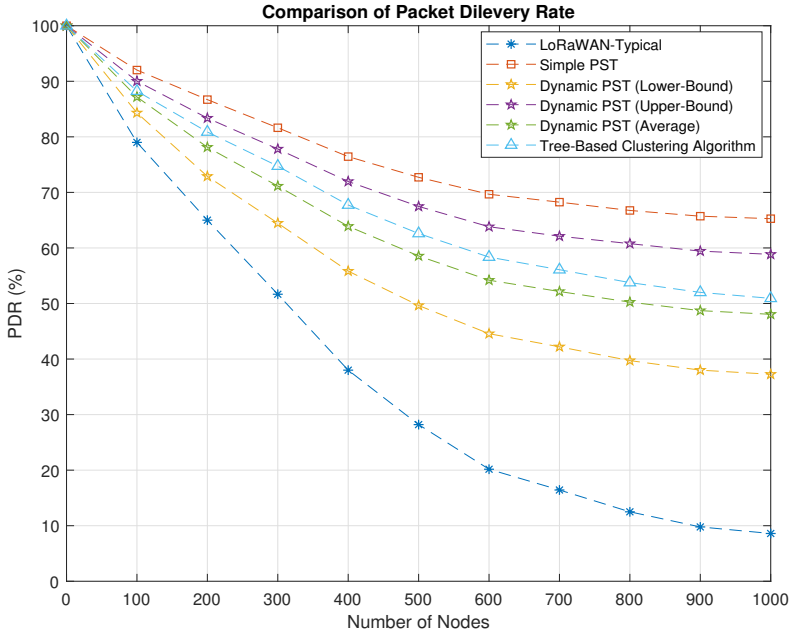


Figure 4.9: Comparison of The Packet Delivery Rate Over Different Network Density

Table 4-D: Performance Comparisons of The Dynamic *PST* Against Other Considered Schemes

-	TTD	TEC	PDR
LoRaWAN-Typical (Comparison Baseline)	**1.66 Seconds**	**0.3 Joules**	**8.6%**
Simple PST	12% (⇓)	18% (⇓)	65% (⇑)
Tree-Based Clustering Algorithm	94% (⇓)	116% (⇑)	51% (⇑)
Dynamic PST (Average)	50% (⇓)	53% (⇓)	48% (⇑)

* (⇑): indicates increased by value from the comparison baseline.
* (⇓): indicates decreased by value from the comparison baseline.
* The objective is to increase *PDR*; decrease *TTD* and *TEC*.

comparison to the tree based clustering algorithm.

From Table 4.4 and given the trade-off between *TTD*, *TEC* and *PDR*, the proposed dynamic transmission *PST* shows more suitability for being adopted in LoRaWAN to serve dense IoT applications.

4.5 Chapter Summary

The use of machine learning techniques can lead to inefficient energy consumption when applied to low power networks with limited resources. This is because machine learning techniques usually require coordination between the end-nodes and the gateway. However, the dynamic transmission *PST* allows maintaining low energy consumption and transmission delay due to its ability of providing the nodes with specific transmission modes that appropriately fit the corresponding clusters transmission priorities and needs. Therefore, the dynamic transmission *PST* provides a fair trade-off between *TTD*, *TEC* and *PDR*. For example, considering typical LoRaWAN performance as a baseline comparison and given the same network density of 1000 nodes, it is noticed that the dynamic transmission *PST* reduced *TTD* and *TEC* by 53% and 50%, respectively. While the simple *PST* reduced *TTD* and *TEC* by 12% and 17%, respectively. Although the dynamic transmission *PST* shows slightly lower *PDR* levels of 48% in comparison to 65% in the simple *PST*, it provides much better performance in terms of *TTD* and *TEC*. On the other hand, the tree-based clustering approach sharply reduced *TTD* by almost 94%. However, this comes at the expense of an extravagant increase to *TEC* by more than 116% in comparison to typical LoRaWAN. Such a compromise of *TEC* may defeat the core purpose of using LoRaWAN as a low power transmission protocol. This leads to a conclusion that the dynamic transmission *PST* is the most convenient amongst other considered approaches especially when serving dense IoT applications.

Chapter 5

Packet Collision Prediction for Ultra-Dense LoRaWAN

5.1 Overview

This chapter introduces a deep learning based Collision Aware Priority Scheduling Technique (CA-PST). This is aimed at assisting LoRaWAN to handle ultra-dense networks where the number of nodes reach up to 5000. *CA-PST* allows LoRa gateway to configure the nodes with one of LoRaWAN transmission protocol classes A and C based on predicting the number of packet collisions. The nodes located at higher transmission priority clusters have the chance to communicate with the gateway using class C and therefore, avoid packet collisions. While nodes in lower transmission priority clusters operate using class A. This technique offers a fair trade-off between the Packet Delivery Rate (PDR) and Total Energy Consumption (TEC). Simulation results show that using *CA-PST* ensures LoRaWAN's reliability in terms of *PDR* especially in ultra-dense networks.

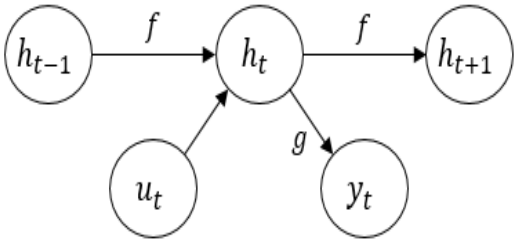


Figure 5.1: SSM Structure

5.2 Collision Prediction Model

This chapter focuses on the use of LoRaWAN to serve ultra-dense applications with the number of nodes within the network range between $1000 \leq n_i \leq 5000$. The main purpose is to achieve acceptable levels of *PDR* while keeping fair trade-off with *TEC*. In order to achieve that, this chapter proposes *CA-PST*, which predicts the possible packet collisions in a given network and therefore, allows the gateway to schedule the nodes n_i transmissions to achieve the best possible *PDR* taking into account the nature of limited resources in IoT applications.

In order to achieve accurate predictions it is necessary for the prediction model to understand the behaviour of the packet collision rate in LoRaWAN. Therefore, it is mandatory to exploit previous traffic information where the proposed *CA-PST* can rely on in its transmission scheduling decisions. Exploiting previous information to predict future trends is known as time-series predictions. There are a number of well-known methods that process time-series predictions, the simplest of which is the State Space Model (SSM) [96, 97]. As shown in Figure 5.1, *SSM* is a one time step dependant. The output y_t at a time step t is derived from two assumptions which are the hidden state h_t influenced by the external input u_t with the transfer function g; where the hidden state h_t is based on a one time step previous state h_{t-1} with the transfer function f. Thus, merely observing the previous one time step state is not sufficient enough in complex scenarios with sequential time-series problems incorporating multiple inputs that influence the output.

In packet collision predictions there are a number of internal factors (parameters) that vary from one time step to another, and hence have a great influence on the output (packet collisions). Therefore, it is necessary to exploit more powerful machine learning models such as the well-known deep learning model Recurrent Neural Network (RNN) [98, 99]. The non-linearity capabilities of *RNN* allows handling more complex sequential data with multiple previous time steps. *RNN* models such Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM) have additional internal cells extending their capabilities to accurately handle longer term dependency sequences of data [100, 101]. In the considered scenario, the number of collisions is an output denoted by y_t and is dependant on a sequence of time series inputs denoted by $\{x_{t-s}, x_{t-s+1}, \cdots, x_{t-1}\}$; where x_t is the input in a given time step t. In *RNN*, the information of the past time steps are processed in a hidden state. The forward hidden state can be a result of a linear connection of multiple previous hidden states. The forward process is given as in Equation (5.1)

$$h_t = tanh(W[x_t, h_{t-1}] + b) \tag{5.1}$$

where W and b are the weight matrix and the bias vector, respectively.

However, a common problem in *RNN* is known as information explosion or gradient vanishing [102]. This a result of a long term dependency, where the inputs influencing a certain hidden state remain part of all the forward hidden states. *LSTM* overcomes this problem via the internal gates, specifically the forget gate that filters out the unnecessary inputs from continuing to the forward hidden states [72].

Given the considered data set used in the scenario of this chapter, *LSTM* is adopted in the proposed *CA-PST*, the following sections details the structure of *LSTM* and evaluates its performance against other *RNN* models.

5.2.1 Long Short-Term Memory (LSTM) and Model Architecture

Unlike traditional *RNN* cell structure [98], where only cells with repetitive *tanh* function are processed forward, *LSTM* has a number of additional gates in each cell. These gates are input, forget and output, in which the purpose is to control the information flow. Just as in other deep learning models, there are two types of training; online and offline. The online training is where the weights are updated following each sample of data fed into the model. While in offline training, the weights are only updated when receiving a data batch. It is worth mentioning that for the considered scenario and the utilised data set in this chapter, only offline training is applied. Similar to *simpleRNN* model, in *LSTM* model the non-linearity functions do not effect the cell state moving across the constructed chain. However, the information in the hidden states that are processed forward through the chain of cells are controlled by *LSTM* gates. The input gate i_t controls the information that can be added form the previous hidden state h_t to the forward cell state c_t and therefore, the forward hidden state h_t. In a similar manner, the forget gate f_t controls the information that should be blocked from being added to the cells c_t and h_t. While the output gate o_t controls the information that should be processed to h_t. Figure 5.2 explains the *LSTM* cell, while the forward process is given as in the following Equations:

$$f_t = \sigma(W_f[x_t, h_{t-1}] + b_f) \tag{5.2}$$

$$i_t = \sigma(W_i[x_t, h_{t-1}] + b_i) \tag{5.3}$$

$$\tilde{c_t} = tanh(W_c[x_t, h_{t-1}] + b_c) \tag{5.4}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tilde{c_t} \tag{5.5}$$

$$o_t = \sigma(W_o[x_t, h_{t-1}] + b_o) \tag{5.6}$$

$$h_t = o_t \cdot tanh(c_t) \tag{5.7}$$

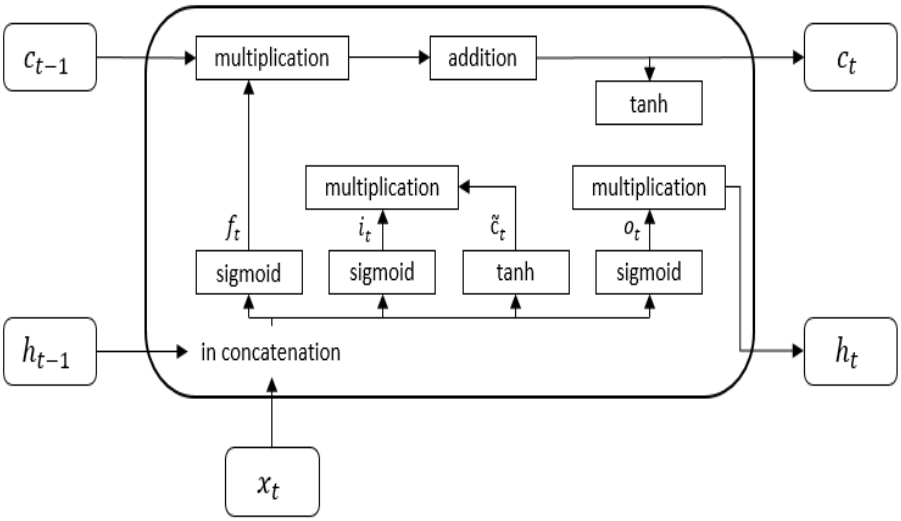


Figure 5.2: LSTM Cell - Showing The Input Gate i_t; Forget Gate f_t; and Output Gate o_t

where σ is the logistic sigmoid function; W is the weights; b is the bias vectors; $\tilde{c}_t$ is the intermediate state vector; and x_t is the input vector.

Constructing a neural network of *LSTM* cells can be achieved by feeding the forward cells of c_t and h_t as inputs to the next *LSTM* cell. *LSTM* based neural network models vary in terms of the inputs and the outputs. The shape of the network could consist of one input to one output, one-to-many, many-to-one and many-to-many. In this work, the constructed neural network follows a many-to-one model. A number of previous time steps are considered in the prediction process, hence the output of the model can be given as in Equations (5.8) and (5.9)

$$[h_{t-s}, h_{t-(s+1)}, \cdots, h_{h-1}] = LSTM(x_{t-s}, x_{t-(s+1)}, \cdots, x_{h-1}) \tag{5.8}$$

$$\hat{y_t} = w_t^T [h_{t-s}, h_{t-(s+1)}, \cdots, h_{h-1}] + b_t \tag{5.9}$$

where s is the length of the time steps.

Training this model follows the well-known back propagation through time method [103]. The purpose of constructing this model is to predict the number of collisions based on a

Table 5-A: Features of the Data Set

Index	**Features**
1	$Colli$
2	n_i
3	C_k
4	Packet Size (bytes)
5	Number of Packets
6	SF
7	CR (1 to 4)/5

set of features that influence the collisions in the network. The features are fed into the model as an input vector x_t at each time step t. As listed in Table 5-A, these parameters are the number of nodes in the network n_i, number of clusters C_k, transmitted packet size, the number of packets transmitted per each node of n_i, and SF and CR used in the transmissions between the n_i and GW. Note that the clustering process is the same as in Chapters 3 and 4, where SF is chosen to provide the maximum data rate, lowest transmission delays, and lowest energy consumption for the network. The weights and the parameters of the training were chosen based on evaluating the best validation performances as detailed in the following performance evaluations section.

5.2.2 Traffic Model and Data Set

In *LSTM*, the data set used in training the model plays a significant role in constructing a convenient neural network. The sole purpose of using *LSTM* is to predict the packet collisions using given network features. As aforementioned, a packet collision happens when two or more packets are transmitted at the same time using the same SF and CR. The considered number of nodes in this work is scaled up to 5000 n_i. All n_i are configured following the LoRa SX1272 model. Where n_i performance is validated against practical experiments carried out in [79, 47]. The transmissions between n_i and GW take place once every hour. While the period of the transmission window lasts for 120 *seconds* [1].

The data set consists of a number of features that were specifically chosen due to their direct effect on the packet collisions. These features are the number of nodes in the network n_i, number of clusters C_k, transmitted packet size, the number of packets trans-

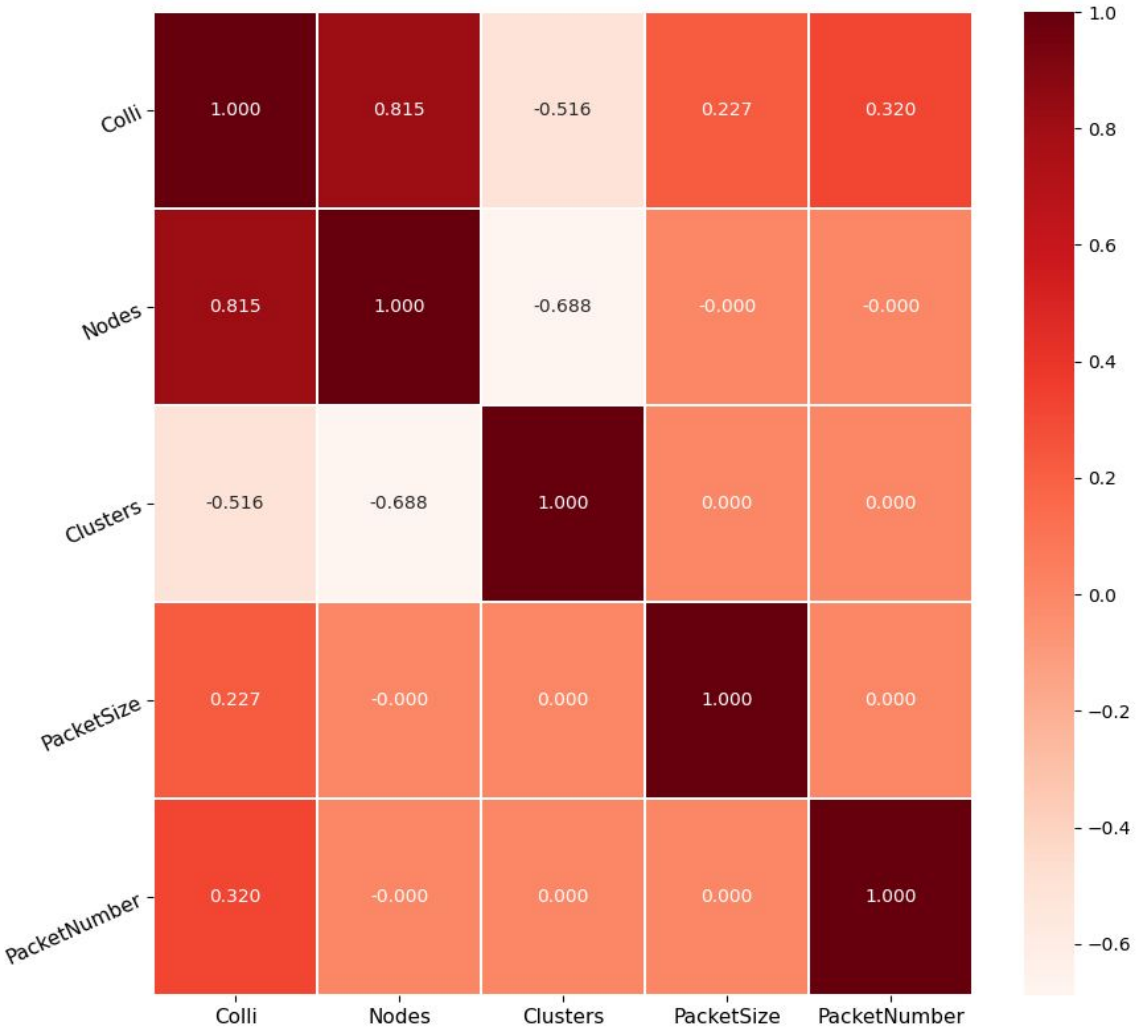


Figure 5.3: Heat Map for Data Set Features Correlations - Darker Colour Indicates Higher Correlation and Lighter Colour Indicates Lower-to-Inverse Correlation

mitted per each n_i, SF, and CR. A validation of the data set is then carried out to evaluate the diversity of the selected features. Usually a data set is considered diverse when there is low to no correlation between the selected features [104, 105]. For this purpose a heat map is generated to visualise the correlations between features to features and features to output. In Figure 5.3, the correlation scales between 1 and -1, where 1 is fully correspondent and -1 is inversely correspondent. Note that the feature is always correspondent to itself.

From Figure 5.3, the number of packet collisions (*Colli*) is highly correlated to the number of nodes. In other words, the higher the number of nodes the higher the number of collisions. Similarly, the packet size and the number of packets transmitted by the nodes also have a positive correspondence to the number of collisions. On the other hand, the number of clusters has an inverse correlation to the number of collisions, this is due to the lower number of nodes that transmit at the same time (detailed in Chapter 3). For the same reason, it can be noticed that there is an inverse correlation between the number of clusters and number of nodes.

The data set is generated using a LoRa SX1272 model communication simulation engine built in MATLAB for the purpose of carrying out this work. While the *LSTM* model is constructed in Python using Keras library and Tensor-Flow backend version (2.4.1). The data set is split into 70% training and 30% test. Higher portion of the data set is dedicated for training to expose the model to high varieties of samples and hence, the ability to handle more differences in samples over time.

5.2.3 Performance Evaluations

Constructing an *LSTM* model can be achieved via three main parameters. These parameters are the number of layers l, the size of the hidden states m and the length s of the time steps t. The main purpose of building this model is to predict the number of collisions based on a previous set of information fed into the model. Hence, manipulating the *LSTM* parameters is highly dependant on the type of data set used in the training process. In order to achieve the best prediction performance it is necessary to evaluate the model performance under different parameter setups. As the considered prediction problem is a regression problem, the validation metrics chosen to evaluate the model performance are the Mean Square Error (MSE) and the coefficient of determination (R^2). A lower value of *MSE* indicates better prediction output, while the R^2 is scaled between 0 and 1. A value of R^2 closer to 1 indicates better explanation of the data, while 0 indicates regression model incapability of explaining the data set. *MSE*

and R^2 are calculated as in Equations (5.10) and (5.11), respectively.

$$MSE = \frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2 \tag{5.10}$$

$$R^2 = 1 - \frac{\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{T}(y_t - \bar{y}_t)^2} \tag{5.11}$$

where y_t is the true output value at a time step t; $\tilde{y}_t$ is the predicted output value; and $\bar{y}_t$ is the mean value of the true output y_t.

Following the evaluations, the best performance model of *LSTM* is then compared to other predictions models such as the *GRU* and the conventional *simpleRNN* models. The size of the hidden states m is scaled as $m = \{10, 30, 50\}$, while the number of layers l is scaled up as $l = \{1, 2, 3, 4\}$. The length s of the time steps t is $s = \{5, 15, 25\}$. Note that in all models the well-known adaptive stochastic algorithms *Adam* and *RMSprop* are used due to its optimising properties for achieving convergences especially in regression problems [106, 107].

From Table 5-B, it is noticed that increasing l results initially in decreasing *MSE* loss. This is until $l = 3$, where the best performance of *LSTM* is situated. When increasing l further to $l = 4$, it is noticed that *MSE* loss regain increasing. When observing the performance under a different number of hidden states m, it is generally noticed that an excessively high number of hidden states can result in complicating the training process, and therefore higher *MSE* loss in the prediction. The best value of *MSE* is when $l = 3$ and $m = 30$. Tuning the model using these parameters is also reflected on R^2 where the model is performing at its best in explaining the data.

Figure 5.4 shows the performance of *LSTM* at its best tuning parameters in terms of *MSE* loss when validated against a testing set.

In a similar manner, the two other models *GRU* and the conventional *simpleRNN* were

Table 5-B: Performance Evaluation of the Parameters Used in *LSTM*

Parameters (number of layers l, size of hidden states m)	Evaluation metrics	
	MSE(10^{-3})	R^2
$l = 1, m = 10$	0.69523	0.98912
$l = 1, m = 30$	0.66391	0.98775
$l = 1, m = 50$	0.66827	0.98627
$l = 2, m = 10$	0.57324	0.98821
$l = 2, m = 30$	0.56395	0.99075
$l = 2, m = 50$	0.52687	0.99173
$l = 3, m = 10$	0.47875	0.99308
$l = 3, m = 30$	**0.42753**	**0.99365**
$l = 3, m = 50$	0.45136	0.99349
$l = 4, m = 10$	0.49021	0.99254
$l = 4, m = 30$	0.46234	0.99327
$l = 4, m = 50$	0.48529	0.99278

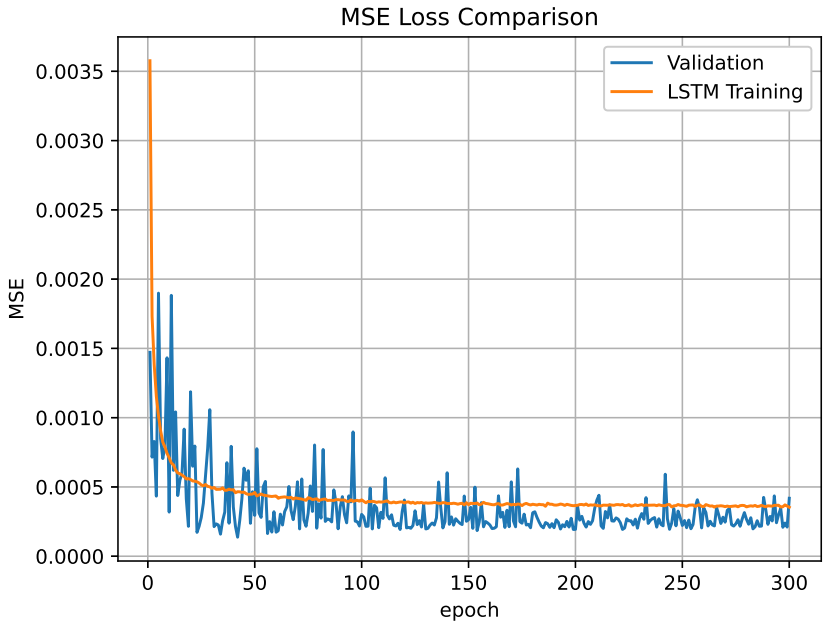


Figure 5.4: Validation of *MSE* Loss in *LSTM* (training vs. validation)

constructed for the purpose of predicting the number of collisions in the network. For fair comparison, the models were tuned using the same parameters l, m and s. The best performance values and the tuning parameters are listed in Table 5-C. As for *simpeRNN*, it is noticeable that the longer the length of time steps considered in the training process the lower the prediction performance gets. This could be a reason of the explosion

Table 5-C: Performance Evaluations of *LSTM*, *GRU* and *simpleRNN*

Model	$s = 5$		$s = 15$		$s = 25$	
	MSE(10^{-3})	R^2	MSE(10^{-3})	R^2	MSE(10^{-3})	R^2
SimpleRNN	**1.37924**	**0.98729**	1.38247	0.98652	1.39184	0.98569
GRU	0.95948	0.98873	**0.95207**	**0.98914**	0.95241	0.98876
LSTM	0.42886	0.99154	**0.42753**	**0.99365**	0.42791	0.99279

problems [102] related to long-term dependencies. While in both *LSTM* and *GRU*, the longer the time steps lengths does not always result in better performance. There are a number of reasons for the prediction deterioration accompanied with longer lengths of time steps. Some of which are the overfitting problems [108] and the generalisation avoidance dropout mechanism [109]. In general, the prediction in *LSTM* is much better than that of *GRU* and *simpleRNN*. In comparison to *LSTM*, the *MSE* loss in *GRU* is 55% more, while the *simpeRNN MSE* loss is 69% more than *LSTM*. Therefore, *LSTM* is adopted in this work for predicting the number of the collisions in the network.

Figure 5.5 shows the performance of *GRU* at its best tuning parameters in terms of *MSE* loss when validated against a testing set. While Figure 5.6 shows the performance of the conventional *simpleRNN* model. Finally, R^2 in all models are shown in Figure 5.7, where *LSTM* shows the closest to 1. Therefore, since *LSTM* shows the best performance in terms of explaining the data, *LSTM* is adopted for predicting the packet collisions in this work.

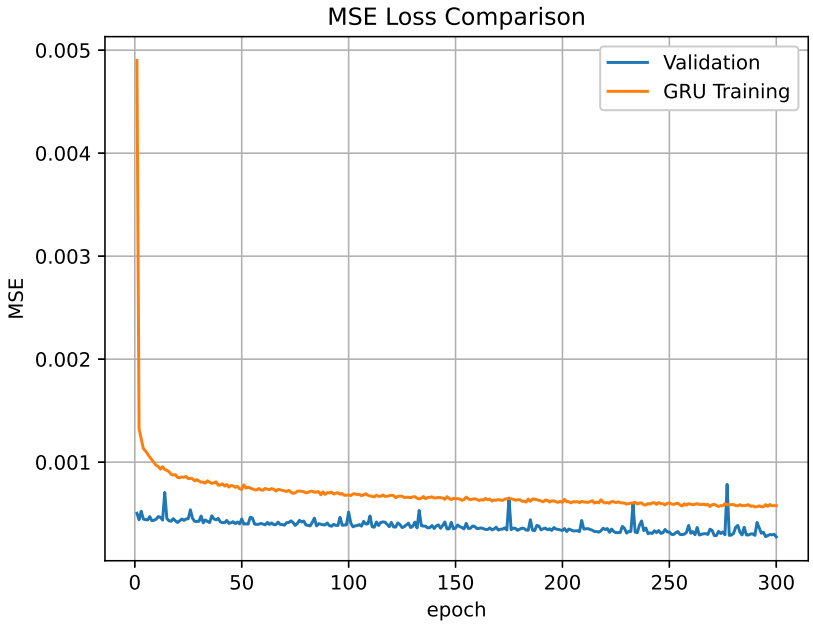


Figure 5.5: Validation of *MSE* Loss in *GRU* (training vs. validation)

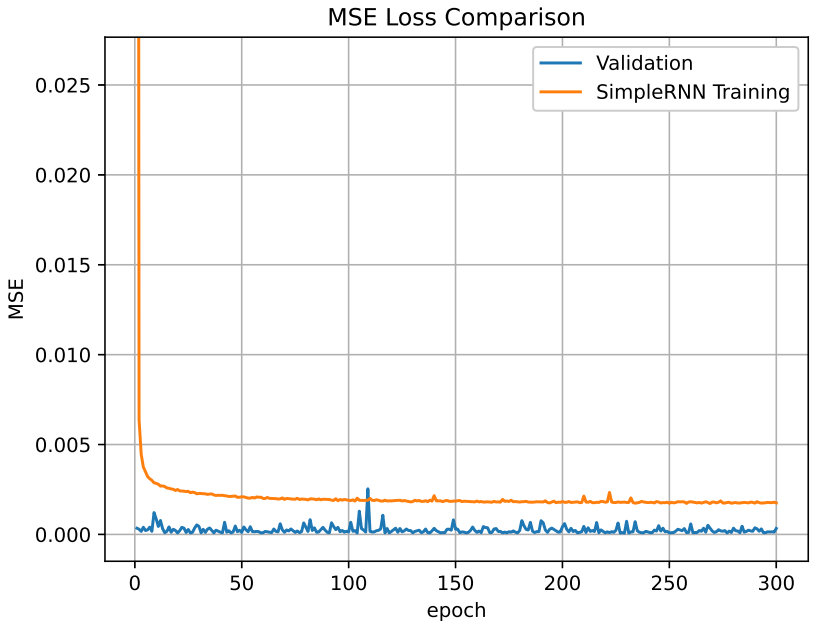


Figure 5.6: Validation of *MSE* Loss in *SimpleRNN* (training vs. validation)

Figure 5.7: Coefficient of Determination (R^2) (*LSTM* vs. *GRU* vs. *SimpleRNN*)

5.3 Collision Aware Transmission Priority Scheduling Technique (CA-PST)

In the considered scenario, n_i initiate transmissions to one *GW* following class *A* LoRaWAN. When the number of n_i exceeds 1000 in the network, it is noticed that *PDR* suffers from serious deterioration. This is due to the excessive number of collisions caused by simultaneous transmissions. In spite of applying the unsupervised learning clustering to reduce the amount of nodes competing on the channel at the same time, yet in ultra-dense network the number of n_i is still high in each cluster. Therefore, the transmissions attempts are increased leading to higher levels of *TTD* and *TEC*. Thus, the proposition of the *CA-PST*, in which it takes place at the *GW* level.

Following the flow chart in Figure 5.8, when $n_i \geq 1000$ and the collision rate exceeds a specific threshold, the *GW* runs a prediction of the number of collisions in the network prior to the initiation of the nodes transmissions period. Then *GW* determines the

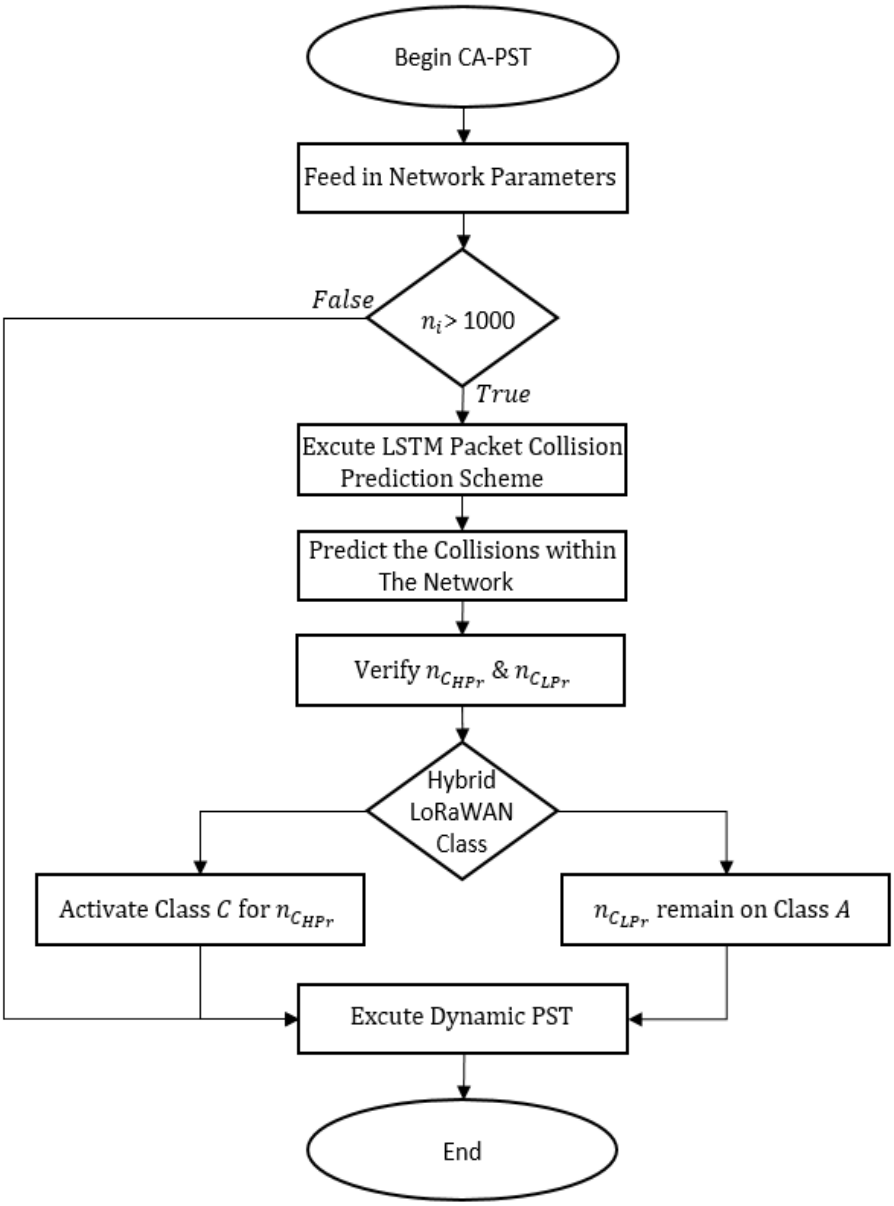


Figure 5.8: *CA-PST* Flow Chart

clusters with the higher transmission priority needs $C_{k_{HPr}}$ according to the given application's specific threshold assumed to be known beforehand to the GW. Other clusters in the network are classified as lower transmission priority needs $C_{k_{LPr}}$. The GW then instructs nodes in higher transmission priority clusters $n_{C_{HPr}}$ to operate using class C following LoRaWAN protocol. The reason of choosing class C and not class B is that despite the slotted-ALOHA style transmissions in class B where the nodes initiate trans-

missions during a periodic beacon, the transmissions during the beacon remain random. In other words, although the nodes transmissions are limited to the beacon period, still the transmissions during the beacon period are random. Hence, the risk of simultaneous transmission remains present and therefore, packet collisions possibilities, while the aim in *CA-PST* is to eliminate collisions caused by simultaneous transmissions. Note that unlike LoRaWAN classes A and B, class C is a continuously listening class where nodes have the capabilities to initiate transmissions at a specified time slot following a command received by GW. Although, class C consumes much more energy than class A, the nodes are only operating using class C during the transmission period. While nodes in lower transmission priority clusters remain operating on class A and are treated just as in the dynamic transmission *PST* (detailed in Chapter 4). This hybrid adoption of LoRaWAN classes allows better control over the nodes transmissions especially in ultra-dense applications where n_i surpasses 1000 in a limited geographical area.

5.3.1 Performance Metrics Formulations

The proposed *CA-PST* adopts a hybrid technique, where nodes switch between LoRaWAN classes A and C. Therefore, TEC is expected to be increased due to higher energy consumption by nodes operating on a LoRaWAN class C. However, considering ultra-dense applications, allowing more collisions to take place due to using class C will eventually lead to higher energy consumption as a result of the retransmission attempts. Therefore, it is necessary to understand how the network TEC is effected. Since TEC is directly effected by the transmissions, hence both TTD and TEC are formulated as in the following sections.

5.3.1.1 Total Transmission Delay

In the proposed *CA-PST*, the GW assigns different transmission priorities Pr to K number of clusters C_K, where $K = \{1, 2, 3, ..., k\}$. The nodes in a lower Pr cluster wait until transmissions from nodes in higher Pr clusters are satisfied. This introduces waiting times in lower transmission priority clusters. Hence, TTD can be given as in

Equation (5.12):

$$TTD(K) = \sum_{j=1}^{K}(D_{C_1}, D_{C_2}, ..., D_{C_K}), \tag{5.12}$$

where K is the number of clusters in the network, D_{C_K} is the total transmission delay of n in a cluster of C_K and is given as in Equation (5.13):

$$D_{C_K} = \sum_{i=1}^{n_{C_K}} X(i), \tag{5.13}$$

where n_{C_K} is the total number of all n_i in the corresponding C_K; $X(i) = (D_{I_T} + D_{R_{colli}} + D_{R_{ch}})$; D_{I_T} is the delay of the initial transmission I_T of each n_i; $D_{R_{colli}}$ is the delay of the retransmission caused by n_i's collided packet (R_{colli}); $D_{R_{ch}}$ is the delay of the retransmission caused by n_i's lost packet due to bad channel condition (R_{ch}). Note that the GW is assumed to be able to distinguish between I_T, R_{colli} and R_{ch}.

Since the transmissions from n_i in the lower priority Pr clusters C_{LPr} wait until transmissions from n_i in the higher Pr clusters C_{HPr} are satisfied, the delay of $D_{C_{LPr}}$ is given as in Equation (5.14):

$$D_{C_{LPr}} = D_{C_{HPr}} + \sum_{i=1, i \notin C_{HPr}}^{n_{C_{LPr}}} X(i), \tag{5.14}$$

where $D_{C_{HPr}}$ is the delay of all n_i in higher Pr clusters and $n_{C_{LPr}}$ is n_i in the corresponding C_{LPr} clusters.

5.3.1.2 Total Energy Consumption

The GW using the proposed *CA-PST* regulates the transmissions from n_i in different clusters of C_k to the GW based on the corresponding transmission Pr. Hence, each of C_k is classified either C_{HPr} or C_{LPr} status. When a cluster of C_k is at C_{HPr} status, the corresponding $n_{C_{HPr}}$ are allowed transmissions using class C LoRaWAN. Otherwise

the cluster is at C_{LPr} transmission status and transmissions from the corresponding n_{C_k} follow class A. Note that only one cluster of C_k can be actively transmitting at a time. Hence, TEC as a function of the number and type of clusters can be given as in Equation (5.15):

$$TEC = E_{n_{C_{HPr}}} + \sum_{j=1, C_{HPr} \notin j}^{K} E_{C_{LPr}}^{(j)} \tag{5.15}$$

where $E_{n_{C_{HPr}}}$ and $E_{n_{C_{LPr}}}$ are the energy consumption in *Joules* of all n_{C_k} in the corresponding HPr and LPr clusters of C_k, respectively. $E_{n_{C_{HPr}}}$ and $E_{n_{C_{LPr}}}$ are given in Equations (5.16) and (5.17), respectively.

$$E_{n_{C_{HPr}}} = \sum_{i=1}^{n_{C_{HPr}}} (P_{T_{HPr}} \times D), \tag{5.16}$$

$$E_{n_{C_{LPr}}} = \sum_{i=1}^{n_{C_{LPr}}} (P_{T_{LPr}} \times D), \tag{5.17}$$

where $P_{T_{HPr}}$ is the power consumed for a packet transmission by n_i in an C_{HPr} of C_k. While $P_{T_{LPr}}$ is power consumed for a packet transmission by n_i in the other C_{LPr} clusters of C_k. The duration of the transmission is denoted by D.

5.4 Simulation Results

In order to evaluate the performance of the proposed *CA-PST*, this section reveals simulation results that compares the *CA-PST* against other techniques. The simulations follow LoRaWAN protocol [1] and the list of parameters in Table 5-D.

The simulations were carried out to evaluate the impact of the proposed technique on *TTD*, *TEC* and *PDR*. Figure 5.9 is divided into three rows and five columns. The first row in Figure 5.9(a) shows *TTD*, Figure 5.9(b) shows *TEC* and Figure 5.9(c) shows *PDR* performances. While the first column to the left hand-side of Figure 5.9 shows

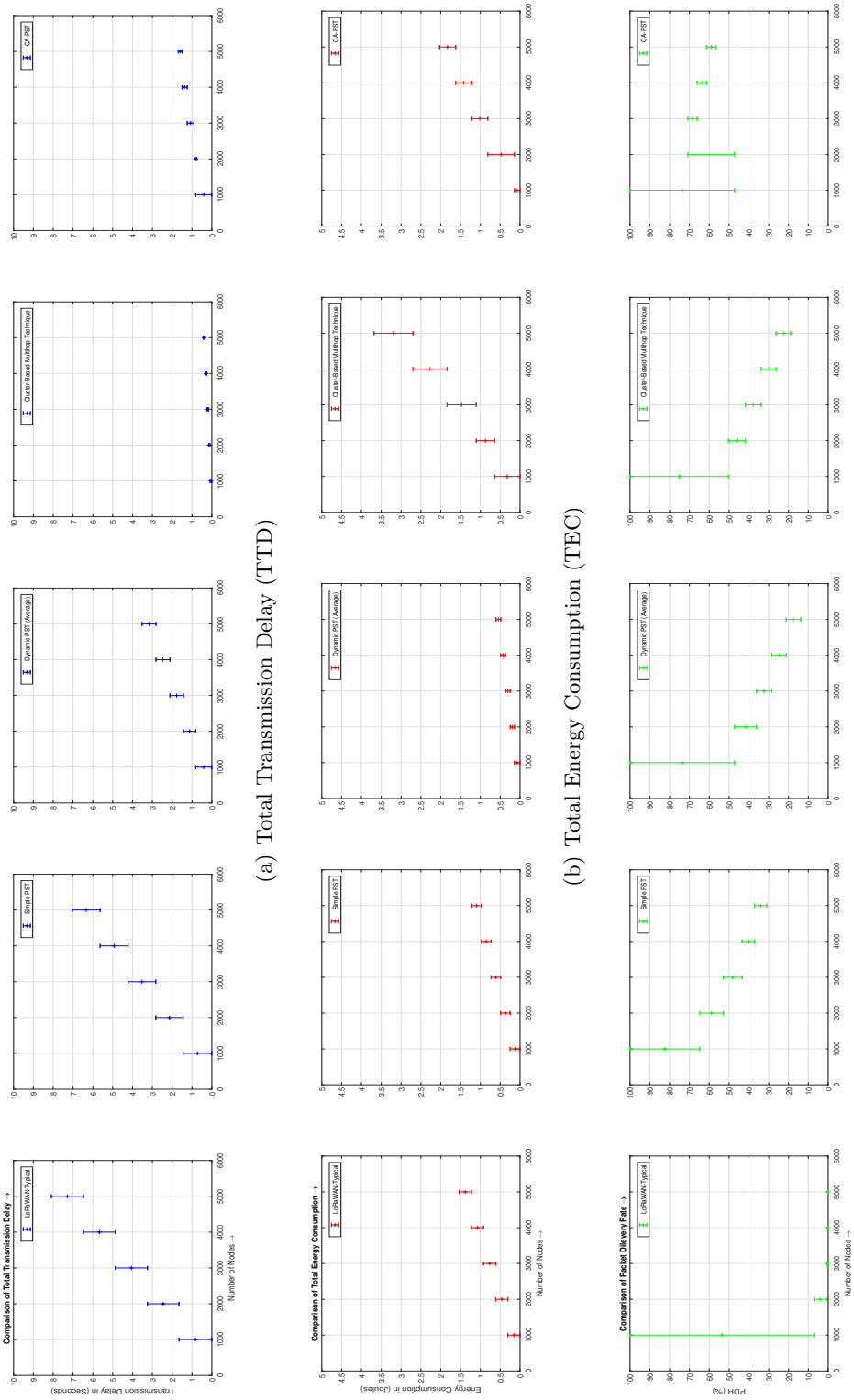
Table 5-D: Simulation Parameters

Parameter	**Value**
Protocol	LoRaWAN (v1.1)
Number of nodes n_i	$i = \{1, \cdots, 5000\}$
Payload size	25 - 200 $Bytes$
SF	7
CR	4/5
BW	125 kHz
Channel frequency	915 MHz
Active class C n_i power consumption	0.153 W
Active class A n_i power consumption	0.1 W
Idle class A n_i power consumption	0.072 W
Transmission duration	0.036 s
Number of clusters k	$k = \{2, \cdots, 30\}$

the performance of the conventional LoRaWAN, followed by the simple *PST* proposed in Chapter 3 [89], dynamic transmission *PST* proposed in Chapter 4 [110], clustering-based multihop scheme proposed in [19] and the proposed *CA-PST*, respectively.

For convenient visualisations to the very high number of values and due to the high number of nodes considered in the simulations, 2-values box-plot [111] is adopted in Figure 5.9. In the sub-figures of Figure 5.9(a) and (b), the over-all trend of the plots is ascending, hence the top hinges represent the higher number of nodes, while the bottom hinges represent the previous number in x-axis. The margin between the two hinges values of each box-plot represent the values in between. For example, in the first sub-figure of Figure 5.9(a), the box-plot at the x-axis value of 1000 represent the values for the nodes from $n_i = 1$ (the bottom hinge) up to $n_i = 1000$ (the top hinge). Vice versa, the plots trend in the sub-figures of Figure 5.9(c) is descending, hence the top hinges represent the lower number of nodes (previous number on the x-axis), while the bottom hinges represent the higher number of nodes. For example, in the first sub-figure of Figure 5.9(c), the box-plot at the x-axis value of 1000 represent the values for the nodes from $n_i = 1$ (the top hinge) up to $n_i = 1000$ (the bottom hinge).

With holding Typical LoRaWAN at $n_i = 5000$ as a baseline for comparisons, it is noticed that all schemes perform better in terms of *TTD*. This is due to reducing the packet col-



(a) Total Transmission Delay (TTD)

(b) Total Energy Consumption (TEC)

(c) Packet Delivery Rate (PDR)

Figure 5.9: Comparison of *TTD*, *TEC* and *PDR*

lisions, which in return reduces the need for packet retransmissions. The simple *PST* enhances *TTD* by 13% in comparison to typical LoRaWAN. While the dynamic transmission *PST* enhances *TTD* by 57%. *CA-PST* comes second best with reducing *TTD* by 84% from that of typical LoRaWAN, this is due to the hybrid functionality of switching nodes in higher transmission priority cluster to operate using class *C* LoRaWAN. The best performance in terms of *TTD* is achieved using the cluster-based multihop scheme, which reduces *TTD* by 95%. However, observing the performance in terms of *TEC*, the cluster-based multihop scheme compromises the energy consumption, and is the worst amongst all techniques with increasing the energy consumption by 141% in comparison to that of typical LoRaWAN. This defeats the purpose of using LoRaWAN as a low power technology aimed at serving IoT applications with limited resources. It is noticed that the best scheme in terms of *TEC* is the dynamic transmission *PST* scheme, however at the expense of very low *PDR*. When observing the performance in terms of *PDR*, it is noticed that all schemes almost decline at an ultra-dense number of nodes ($n_i \geq 1000$). The *CA-PST* shows an excellent performance with a *PDR* level of up to 58%. All other schemes show very low *PDR* with levels below 30%. By a comprehensive observation, it can be concluded that *CA-PST* provides the best trade-off between *TTD*, *TEC* and *PDR* at ultra-dense applications (up to 5000 nodes), while the dynamic transmission *PST* performs the best in dense applications with nodes less than 1000. Note that *CA-PST* performs exactly the same as dynamic transmission *PST* when $n_i \leq 1000$. This is because *CA-PST* activates dynamic transmission *PST* at lower scale networks. Thus, the heterogeneity in the performance trend when $n_i \geq 1000$.

5.5 Chapter Summary

LoRaWAN is a low power technology with a simple transmission protocol design aimed at serving resource-limited IoT applications (battery-powered devices). Due to its simple transmission protocol, LoRaWAN specifically suffers from excessive collision rate especially when serving ultra-dense applications with up to 5000 nodes. The *PDR* in typical

LoRaWAN drops below 10% when the number of nodes exceeds 1000. Considering the other schemes performance at 5000 nodes, the best level of *PDR* is at 30%. This slight enhancement of *PDR* in the other schemes comes at the expense of either higher *TTD*, *TEC* or both. On the contrary, the proposed *CA-PST* enhanced the *PDR* up to 57%. Although the *CA-PST* increased *TEC* by 33% in comparison to typical LoRaWAN, the *TTD* was significantly reduced by 84%. The increase in *TEC* is due to providing the nodes located in higher transmission priority clusters with the ability to transmit using class *C* of LoRaWAN protocol, in which the nodes only transmit following instructions from the gateway, hence a packet collision avoidance. Therefore, the *CA-PST* performs the best in terms of *PDR* in comparison to the other schemes.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The work in this thesis particularly focuses on enhancing LoRaWAN performance while taking into account LoRaWAN's devices limited resources nature. In particular, when carrying out the work in Chapters 3, 4 and 5 the energy consumption was always a key factor that was kept at the lowest possible levels. The main drawback in LoRaWAN protocol is the excessive packet collisions. This is due to the adoption of ALOHA protocol where nodes initiate transmissions regardless of the channel status.

In Chapter 3, the unsupervised learning clustering algorithm is adopted as a backbone for the proposed simple *PST*. The aim is to reduce the simultaneous transmission by the end-nodes. Based on that a simple *PST* is introduced where clusters are granted unique transmission priorities. Although nodes in higher transmission priority clusters are allowed to initiate transmissions prior to nodes in lower priority cluster, however, nodes in lower priority clusters still have the chance to transmit regardless of the importance of the transmission. This led to reducing the *TEC* by 12% and *TTD* by 17%, in comparison to typical LoRaWAN, while increasing *PDR* levels up to 65%. Note that the number of nodes considered was $n_i = 1000$. However, a more precise transmission decision making

process was needed to further enhance *TEC* and *TTD*, while keeping acceptable levels of *PDR*. This was achieved by the introduction of the dynamic transmission *PST* in Chapter 4.

The dynamic transmission *PST* was introduced mainly to preserve the energy consumption in the network. It introduces two transmission modes, and adopts the Naive Bays Classifier algorithm for accurate transmission modes allocations to the nodes based on the corresponding cluster transmission priority. It also introduces a strict transmission termination process to the nodes located in lower transmission priority clusters. This is because when analysing the transmission behaviour in simple *PST*, it was noticed that nodes located in lower priority clusters have less importance, hence classified as lower priority. Yet, these nodes still initiate transmissions to the gateway regardless of the actual importance of the information they are attempting to deliver. Therefore, unnecessary energy consumption and transmission delays are introduced to the network with insignificant yield to *PDR* levels. Hence, the aim is to eliminate unnecessary waste of resources by nodes classified as lower priorities without compromising *PDR* levels. Comparisons against typical LoRaWAN shows a reduction of *TEC* by 50% and *TTD* by 53%. The *PDR* was increased up to 48%, leading to slightly lower performance of the dynamic transmission *PST* against the simple *PST* in terms of *PDR*, however, significantly better performance in terms of *TEC* and *TTD*. Thus, overall trade-off between *TTD*, *TEC*, and *PDR* promotes the dynamic transmission *PST* to be the best amongst other considered techniques for serving LoRaWAN dense networks.

The Collision-Aware Priority Scheduling Technique (*CA-PST*) was specifically introduced to enhance the reliability of LoRaWAN in terms of *PDR* at ultra-dense applications where the number of nodes are scaled up to $n_i = 5000$. In ultra-dense applications, excessively higher levels of *TTD* and *TEC*, and much lower levels of *PDR* were noticed as the number of nodes surpasses 1000. Considering $n_i = 5000$, *PDR* in all considered schemes showed levels that are lower than 30%. This is due to a higher number of nodes within each cluster, thus, higher collision rate. Hence, the introduction of *CA-PST* in

Chapter 5. *CA-PST* is a hybrid scheme that regulates the transmission of nodes when $n_i \geq 1000$ based on a prior number of packet collisions prediction scheme using *LSTM*. Where the nodes located in higher transmission priority clusters switch to LoRaWAN class *C* transmission protocol and follow the gateway transmission instructions. On the other hand, the dynamic transmission *PST* is activated for transmission scheduling when $n_i \leq 1000$. This has increased *TEC* by 33% while *TTD* sharply reduced by 84%. In return the *PDR* level is elevated up to 57% in comparison to less than 30% in other schemes. Thus, although the energy consumption is slightly compromised, *CA-PST* performs the best at ultra-dense applications given a trade-off between *TTD*, *TEC* and *PDR*.

It is worth mentioning that although the work in this thesis was carried on LoRaWAN wireless protocol, it was designed to be applicable for implementation in other wireless protocols provided some tuning and modifications to specific parameters in order to fulfill the technical requirements of the target wireless protocol.

6.2 Future Work

LoRaWAN is a perfect solution for serving IoT applications. Usually in IoT applications no ultra-low latency requirements nor dense transmission traffic models are expected. However, the main concern in IoT applications is the energy consumption, due to the limited energy resources at the end-devices level (nodes are battery-powered). Hence, a significant importance was given to the energy consumption while carrying out the work in this thesis. Carrying out this work unveiled important areas that can potentially add to the feasibility of enhancing LoRaWAN performance for serving dense IoT application. Some of the open research areas are discussed in the following.

- **Security in LoRaWAN:** The wireless communication medium that is utilised to transmit packets between the gateway and the nodes takes place using open ISM radio bands. Therefore, it is essential to maintain secure communication links

that can prevent cybersecurity threats and attacks such as Denial-of-Service (DoS). Additionally, LoRaWAN protocol is mainly designed to fit the limited resources nature of IoT applications especially in terms of energy consumption given that most IoT devices are battery-powered. Hence, it is very important to design a secure scheme that meets the limited capabilities of LoRaWAN in dense networks. Although LoRaWAN protocol provides different security properties such as confidentiality, authentication, integrity as well as network joining mechanism, the gap still remains unfulfilled in terms of designing lightweight robust and secure mechanism that meets the unique LoRaWAN requirements.

- **Blockchain in LoRaWAN:** LoRaWAN by default adopts the star topology as a mean of communication between the gateway and the nodes. In other words, the information received from the nodes are relayed to the server via a single gateway. Although the traffic loads in typical IoT applications are usually low, it is critically important to consider the possibility of a single point of failure to occur after LoRaWAN being deployed. One of the mostly used methodologies to address such a threat in today's state-of-the-art is the Blockchain technology. This could be implemented via uploading transactions periodically to a Blockchain cloud. Of course this involves reviewing many steps that are essential in implementing Blockchain for example, the decision making for control policies needed for granting authorisation and issuing credentials for the purpose of data access. Moreover, Blockchain can play a significant role in improving the scalability of the network. Hence, this could be a potential area for future research, which will have a great impact in improving LoRaWAN.

- **Mobility in LoRaWAN:** Most IoT applications are stationary or have limited mobility needs. However, there are applications that are significantly based on mobile and roaming capabilities. Although LoRaWAN protocol supports mobility, the adoption of Adaptive Data Rate (ADR) mechanism could result in elevating the packet loss ratio in highly mobile applications. This is due to the increased

amount of requests and acknowledgements needed between the gateway and nodes to achieve the best data rate possible. Hence, there is a room for improving mobility especially in line with ADR mechanism in LoRaWAN.

References

[1] LoRa Alliance. *LoRaWAN™ Specification v1.1*. URL: `https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/` (visited on 01/12/2021).

[2] Jorge Ortın, Matteo Cesana, and Alessandro Redondi. "Augmenting LoRaWAN performance with listen before talk". In: *IEEE Transactions on Wireless Communications* 18.6 (2019), pp. 3113–3128.

[3] Quy Lam Hoang et al. "A Real-Time LoRa Protocol for Industrial Monitoring and Control Systems". In: *IEEE Access* 8 (2020), pp. 44727–44738.

[4] Lidia Pocero et al. "A Comparative Study of LoRa and IEEE 802.15. 4-based IoT Deployments inside School Buildings". In: *IEEE Access* (2020).

[5] Emiliano Sisinni et al. "LoRaWAN range extender for Industrial IoT". In: *IEEE Transactions on Industrial Informatics* 16.8 (2019), pp. 5607–5616.

[6] Qihao Zhou et al. "Design and Implementation of Open LoRa for IoT". In: *IEEE Access* 7 (2019), pp. 100649–100657.

[7] Hafiz Husnain Raza Sherazi et al. "Energy-efficient LoRaWAN for Industry 4.0 Applications". In: *IEEE Transactions on Industrial Informatics* (2020).

[8] Semtech. "LoRa Modulation Basics". In: *Camarillo, CA, USA* (2015).

[9] Jothi Prasanna Shanmuga Sundaram, Wan Du, and Zhiwei Zhao. "A Survey on LoRa Networking: Research Problems, Current Solutions and Open Issues". In: *IEEE Communications Surveys & Tutorials* (2019).

[10] Ferran Adelantado et al. "Understanding the limits of LoRaWAN". In: *IEEE Communications magazine* 55.9 (2017), pp. 34–40.

[11] Lluis Casals et al. "Modeling the energy performance of LoRaWAN". In: *Sensors* 17.10 (2017), p. 2364.

[12] Ericsson. *Massive IoT in the city.* URL: https://www.ericsson.com/en/mobility-report/articles/massive-iot-in-the-city (visited on 05/18/2019).

[13] Jun Xu et al. "Narrowband internet of things: Evolutions, technologies, and open issues". In: *IEEE Internet of Things Journal* 5.3 (2017), pp. 1449–1462.

[14] Rashmi Sharan Sinha, Yiqiao Wei, and Seung-Hoon Hwang. "A survey on LPWA technology: LoRa and NB-IoT". In: *Ict Express* 3.1 (2017), pp. 14–21.

[15] Huang-Chen Lee and Kai-Hsiang Ke. "Monitoring of large-area IoT sensors using a LoRa wireless mesh network system: Design and evaluation". In: *IEEE Transactions on Instrumentation and Measurement* 67.9 (2018), pp. 2177–2187.

[16] Mariusz Slabicki, Gopika Premsankar, and Mario Di Francesco. "Adaptive configuration of LoRa networks for dense IoT deployments". In: *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*. IEEE. 2018, pp. 1–9.

[17] N. El Rachkidy, A. Guitton, and M. Kaneko. "Collision Resolution Protocol for Delay and Energy Efficient LoRa Networks". In: *IEEE Transactions on Green Communications and Networking* 3.2 (June 2019), pp. 535–551. ISSN: 2473-2400. DOI: 10.1109/TGCN.2019.2908409.

[18] C. Liao et al. "Multi-Hop LoRa Networks Enabled by Concurrent Transmission". In: *IEEE Access* 5 (2017), pp. 21430–21446. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2755858.

[19] Guibing Zhu et al. "Improving the capacity of a mesh lora network by spreading-factor-based network clustering". In: *IEEE Access* 7 (2019), pp. 21584–21596.

[20] Mohammed Alenezi et al. "Ultra-dense LoRaWAN: Reviews and challenges". In: *IET Communications* 14.9 (2020), pp. 1361–1371. DOI: https://doi.org/10.1049/iet-com.2018.6128.

[21] Kais Mekki et al. "A comparative study of LPWAN technologies for large-scale IoT deployment". In: *ICT express* 5.1 (2019), pp. 1–7.

[22] Ladislav Polak and Jiri Milos. "Performance analysis of LoRa in the 2.4 GHz ISM band: coexistence issues with Wi-Fi". In: *Telecommunication Systems* 74.3 (2020), pp. 299–309.

[23] Cristian González Garcıa et al. *Protocols and Applications for the Industrial Internet of Things*. IGI Global, 2018.

[24] Sigfox. *Sigfox Radio specifications v1.5*. URL: `https://build.sigfox.com/sigfox-device-radio-specifications` (visited on 04/18/2020).

[25] 3GPP TR 36.763 V17.0.0. *Study on Narrow-Band Internet of Things (NB-IoT), (Release 17)*. URL: `https://www.3gpp.org/specifications/specifications` (visited on 12/18/2020).

[26] Kais Mekki et al. "Overview of cellular LPWAN technologies for IoT deployment: Sigfox, LoRaWAN, and NB-IoT". In: *2018 ieee international conference on pervasive computing and communications workshops (percom workshops)*. IEEE. 2018, pp. 197–202.

[27] Hussein Mroue et al. "MAC layer-based evaluation of IoT technologies: LoRa, SigFox and NB-IoT". In: *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*. IEEE. 2018, pp. 1–5.

[28] Mads Lauridsen et al. "Coverage comparison of GPRS, NB-IoT, LoRa, and SigFox in a 7800 km^2 area". In: *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*. IEEE. 2017, pp. 1–5.

[29] Soraya Sinche et al. "A survey of IoT management protocols and frameworks". In: *IEEE Communications Surveys & Tutorials* 22.2 (2019), pp. 1168–1190.

[30] Li Da Xu, Wu He, and Shancang Li. "Internet of things in industries: A survey". In: *IEEE Transactions on industrial informatics* 10.4 (2014), pp. 2233–2243.

[31] Shengyang Li, Usman Raza, and Aftab Khan. "How agile is the adaptive data rate mechanism of LoRaWAN?" In: *2018 IEEE Global Communications Conference (GLOBECOM)*. IEEE. 2018, pp. 206–212.

[32] Joseph Finnegan, Ronan Farrell, and Stephen Brown. "Analysis and enhancement of the LoRaWAN adaptive data rate scheme". In: *IEEE Internet of Things Journal* 7.8 (2020), pp. 7171–7180.

[33] Rachel Kufakunesu, Gerhard P Hancke, and Adnan M Abu-Mahfouz. "A survey on Adaptive Data Rate optimization in LoRaWAN: Recent solutions and major challenges". In: *Sensors* 20.18 (2020), p. 5044.

[34] Yukimasa Nagai et al. "Sub-1 GHz Frequency Band Wireless Coexistence for the Internet of Things". In: *IEEE Access* 9 (2021), pp. 119648–119665. DOI: 10.1109/ACCESS.2021.3107144.

[35] Ashish Kumar Sultania, Farouk Mahfoudhi, and Jeroen Famaey. "Real-Time Demand Response Using NB-IoT". In: *IEEE Internet of Things Journal* 7.12 (2020), pp. 11863–11872. DOI: 10.1109/JIOT.2020.3004390.

[36] Muhammad Hanif and Ha H. Nguyen. "Frequency-Shift Chirp Spread Spectrum Communications with Index Modulation". In: *IEEE Internet of Things Journal* (2021), pp. 1–1. DOI: 10.1109/JIOT.2021.3081703.

[37] Semtech. *LoRa Platform for IoT*. 2021. URL: https://www.semtech.com/lora.

[38] David López-Pérez et al. "Towards 1 Gbps/UE in Cellular Systems: Understanding Ultra-Dense Small Cell Deployments". In: *IEEE Communications Surveys Tutorials* 17.4 (2015), pp. 2078–2101. DOI: 10.1109/COMST.2015.2439636.

[39] Mahmoud Kamel, Walaa Hamouda, and Amr Youssef. "Ultra-Dense Networks: A Survey". In: *IEEE Communications Surveys Tutorials* 18.4 (2016), pp. 2522–2545. DOI: 10.1109/COMST.2016.2571730.

[40] Mary A. Adedoyin and Olabisi E. Falowo. "Combination of Ultra-Dense Networks and Other 5G Enabling Technologies: A Survey". In: *IEEE Access* 8 (2020), pp. 22893–22932. DOI: 10.1109/ACCESS.2020.2969980.

[41] Hwan Hur and Hyo-Sung Ahn. "A circuit design for ranging measurement using chirp spread spectrum waveform". In: *IEEE Sensors Journal* 10.11 (2010), pp. 1774–1778.

[42] Robert Scholtz. "The origins of spread-spectrum communications". In: *IEEE Transactions on communications* 30.5 (1982), pp. 822–854.

[43] Ahmed Badr et al. "Perfecting Protection for Interactive Multimedia: A survey of forward error correction for low-delay interactive applications". In: *IEEE Signal Processing Magazine* 34.2 (2017), pp. 95–113. DOI: 10.1109/MSP.2016.2639062.

[44] Francisco Sandoval, Gwenael Poitau, and François Gagnon. "Optimizing Forward Error Correction Codes for COFDM With Reduced PAPR". In: *IEEE Transactions on Communications* 67.7 (2019), pp. 4605–4619. DOI: 10.1109/TCOMM.2019.2910811.

[45] Semtech. *SX1272/73 Datasheet*. URL: https://www.semtech.com/products/wireless-rf/lora-core/sx1272 (visited on 12/20/2020).

[46] Nicolas Sornin and Ludovic Champion. *Signal concentrator device*. US Patent 9,794,095. Oct. 2017.

[47] Aloÿs Augustin et al. "A study of LoRa: Long range & low power networks for the internet of things". In: *Sensors* 16.9 (2016), p. 1466.

[48] Konstantin Mikhaylov, Juha Petaejaejaervi, and Tuomo Haenninen. "Analysis of capacity and scalability of the LoRa low power wide area network technology". In: *European Wireless 2016; 22th European Wireless Conference*. VDE. 2016, pp. 1–6.

[49] Martijn Saelens et al. "Impact of EU duty cycle and transmission power limitations for sub-GHz LPWAN SRDs: An overview and future challenges". In: *EURASIP Journal on Wireless Communications and Networking* 2019.1 (2019), pp. 1–32.

[50] Juha Petäjäjärvi et al. "Evaluation of LoRa LPWAN technology for remote health and wellbeing monitoring". In: *2016 10th International Symposium on Medical Information and Communication Technology (ISMICT)*. IEEE. 2016, pp. 1–5.

[51] René Brandborg Sørensen et al. "Analysis of latency and MAC-layer performance for class a LoRaWAN". In: *IEEE Wireless Communications Letters* 6.5 (2017), pp. 566–569.

[52] F. Liu and Y. Chang. "An Energy Aware Adaptive Kernel Density Estimation Approach to Unequal Clustering in Wireless Sensor Networks". In: *IEEE Access* 7 (2019), pp. 40569–40580. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2902243.

[53] Rajeev Kumar and Dilip Kumar. "Hybrid swarm intelligence energy efficient clustered routing algorithm for wireless sensor networks". In: *Journal of sensors* 2016 (2016).

[54] Saurav Ghosh, Sanjoy Mondal, and Utpal Biswas. "A dominating set based modified LEACH using Ant Colony Optimization for data gathering in WSN". In: *2016 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)* (2016), pp. 390–396.

[55] Haojun Teng et al. "Adaptive transmission power control for reliable data forwarding in sensor based networks". In: *Wireless Communications and Mobile Computing* 2018 (2018).

[56] Francesca Cuomo et al. "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)*. IEEE. 2017, pp. 1–8.

[57] François Delobel, Nancy El Rachkidy, and Alexandre Guitton. "Analysis of the delay of confirmed downlink frames in Class B of LoRaWAN". In: *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*. IEEE. 2017, pp. 1–6.

[58] K. Kavitha and G. Suseendran. "Priority based Adaptive Scheduling Algorithm for IoT Sensor Systems". In: *2019 International Conference on Automation, Computational and Technology Management (ICACTM)*. Apr. 2019, pp. 361–366. DOI: 10.1109/ICACTM.2019.8776691.

[59] Vipan Kakkar. "Scheduling Techniques for Operating Systems for Medical and IoT Devices: A Review". In: *Global Journal of Computer Science and Technology* (2017).

[60] J Sathish Kumar, Mukesh A Zaveri, and Meghavi Choksi. "Task based resource scheduling in iot environment for disaster management". In: *Procedia computer science* 115 (2017), pp. 846–852.

[61] Bilal Afzal et al. "Energy efficient context aware traffic scheduling for IoT applications". In: *Ad Hoc Networks* 62 (2017), pp. 101–115.

[62] T. Kim, D. Qiao, and W. Choi. "Energy-Efficient Scheduling of Internet of Things Devices for Environment Monitoring Applications". In: *2018 IEEE International Conference on Communications (ICC)*. May 2018, pp. 1–7. DOI: 10.1109/ICC.2018.8422174.

[63] M. S. I. Rubel, N. Kandil, and N. Hakem. "Priority management with clustering approach in Wireless Sensor Network (WSN)". In: *2018 Sixth International Conference on Digital Information, Networking, and Wireless Communications (DINWC)*. Apr. 2018, pp. 7–11. DOI: 10.1109/DINWC.2018.8356987.

[64] O. O. Ogundile et al. "Energy-balanced and energy-efficient clustering routing protocol for wireless sensor networks". In: *IET Communications* 13.10 (2019), pp. 1449–1457. ISSN: 1751-8636. DOI: 10.1049/iet-com.2018.6163.

[65] Huu Phi Tran et al. "A Two-Hop Real-Time LoRa Protocol for Industrial Monitoring and Control Systems". In: *IEEE Access* 8 (2020), pp. 126239–126252.

[66] Nikolaos A Pantazis, Stefanos A Nikolidakis, and Dimitrios D Vergados. "Energy-efficient routing protocols in wireless sensor networks: A survey". In: *IEEE Communications surveys & tutorials* 15.2 (2012), pp. 551–591.

[67] K. Suwandhada and K. Panyim. "ALEACH-Plus: An Energy Efficient Cluster Head Based Routing Protocol for Wireless Sensor Network". In: *2019 7th International Electrical Engineering Congress (iEECON)*. Mar. 2019, pp. 1–4. DOI: 10.1109/iEECON45304.2019.8938948.

[68] Junwei Huang et al. "Two-stage resource allocation scheme for three-tier ultra-dense network". In: *China Communications* 14.10 (2017), pp. 118–129.

[69] Mattia Rizzi et al. "Using LoRa for industrial wireless networks". In: *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*. IEEE. 2017, pp. 1–4.

[70] Konstantin Mikhaylov, Juha Petäjäjärvi, and Janne Janhunen. "On LoRaWAN scalability: Empirical evaluation of susceptibility to inter-network interference". In: *2017 European Conference on Networks and Communications (EuCNC)*. IEEE. 2017, pp. 1–6.

[71] Pierre Neumann, Julien Montavont, and Thomas 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)*. IEEE. 2016, pp. 1–8.

[72] Shengmin Cui and Inwhee Joe. "Collision prediction for a low power wide area network using deep learning methods". In: *Journal of Communications and Networks* 22.3 (2020), pp. 205–214.

[73] Francesca Cuomo et al. "Predicting LoRaWAN Behavior: How Machine Learning Can Help". In: *Computers* 9.3 (2020), p. 60.

[74] Ruben M Sandoval, Antonio-Javier Garcia-Sanchez, and Joan Garcia-Haro. "Optimizing and updating LoRa communication parameters: A machine learning approach". In: *IEEE Transactions on Network and Service Management* 16.3 (2019), pp. 884–895.

[75] Naoki Aihara et al. "Q-learning aided resource allocation and environment recognition in LoRaWAN with CSMA/CA". In: *IEEE Access* 7 (2019), pp. 152126–152137.

[76] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[77] Kristina P. Sinaga and Miin-Shen Yang. "Unsupervised K-Means Clustering Algorithm". In: *IEEE Access* 8 (2020), pp. 80716–80727. DOI: 10.1109/ACCESS.2020.2988796.

[78] Lluis Casals et al. "Modeling the energy performance of LoRaWAN". In: *Sensors* 17.10 (2017), p. 2364.

[79] Dong-Hoon Kim, Eun-Kyu Lee, and Jibum Kim. "Experiencing LoRa Network Establishment on a Smart Energy Campus Testbed". In: *Sustainability* 11.7 (2019), p. 1917.

[80] Nancy El Rachkidy, Alexandre Guitton, and Megumi Kaneko. "Collision Resolution Protocol for Delay and Energy Efficient LoRa Networks". In: *IEEE Transactions on Green Communications and Networking* 3.2 (2019), pp. 535–551.

[81] Robert L Thorndike. "Who belongs in the family". In: *Psychometrika*. Citeseer. 1953.

[82] MA Syakur et al. "Integration K-means clustering method and elbow method for identification of the best customer profile cluster". In: *IOP Conference Series: Materials Science and Engineering*. Vol. 336. 1. IOP Publishing. 2018, p. 012017.

[83] Fan Liu and Yong Deng. "Determine the number of unknown targets in Open World based on Elbow method". In: *IEEE Transactions on Fuzzy Systems* (2020).

[84] Jetmir Haxhibeqiri et al. "LoRa scalability: A simulation model based on interference measurements". In: *Sensors* 17.6 (2017), p. 1193.

[85] Jansen C Liando et al. "Known and unknown facts of LoRa: Experiences from a large-scale measurement study". In: *ACM Transactions on Sensor Networks (TOSN)* 15.2 (2019), pp. 1–35.

[86] Roger Pueyo Centelles et al. "LoRaMoto: A communication system to provide safety awareness among civilians after an earthquake". In: *Future Generation Computer Systems* (2020).

[87] Natchaya Chungsawat and Peerapon Siripongwutikorn. "Predicting Application Performance in LoRa IoT Networks". In: *Proceedings of the 11th International Conference on Advances in Information Technology*. 2020, pp. 1–7.

[88] JU Duncombe. "Infrared navigation—Part I: An assessment of feasibility". In: *IEEE Trans. Electron Devices* 11.1 (1959), pp. 34–39.

[89] Mohammed Alenezi et al. "Use of Unsupervised Learning Clustering Algorithm to Reduce Collisions and Delay within LoRa System for Dense Applications". In: *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE. 2019, pp. 1–5.

[90] Jingjing Wang et al. "Thirty years of machine learning: The road to Pareto-optimal wireless networks". In: *IEEE Communications Surveys & Tutorials* (2020).

[91] Irina Rish et al. "An empirical study of the naive Bayes classifier". In: *IJCAI 2001 workshop on empirical methods in artificial intelligence*. Vol. 3. 22. 2001, pp. 41–46.

[92] A. S. Alam et al. "Optimal Datalink Selection for Future Aeronautical Telecommunication Networks". In: *IEEE Transactions on Aerospace and Electronic Systems* 53.5 (Oct. 2017), pp. 2502–2515. ISSN: 2371-9877. DOI: 10.1109/TAES.2017.2701918.

[93] Lusheng Wang and David Binet. "MADM-based network selection in heterogeneous wireless networks: A simulation study". In: *2009 1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology* (2009), pp. 559–564.

[94] Lusheng Wang and Geng-Sheng GS Kuo. "Mathematical modeling for network selection in heterogeneous wireless networks—A tutorial". In: *IEEE Communications Surveys & Tutorials* 15.1 (2012), pp. 271–292.

[95] Lusheng Wang and David Binet. "TRUST: A Trigger-Based Automatic Subjective Weighting Method for Network Selection". In: *2009 Fifth Advanced International Conference on Telecommunications* (2009), pp. 362–368.

[96] Anthony Stathopoulos and Matthew G Karlaftis. "A multivariate state space approach for urban traffic flow modeling and prediction". In: *Transportation Research Part C: Emerging Technologies* 11.2 (2003), pp. 121–135.

[97] Chunjiao Dong et al. "A spatial–temporal-based state space approach for freeway network traffic flow modelling and prediction". In: *Transportmetrica A: Transport Science* 11.7 (2015), pp. 547–560.

[98] Mowei Wang et al. "Machine learning for networking: Workflow, advances and opportunities". In: *Ieee Network* 32.2 (2017), pp. 92–99.

[99] Yibo Zhou et al. "A deep-learning-based radio resource assignment technique for 5G ultra dense networks". In: *IEEE Network* 32.6 (2018), pp. 28–34.

[100] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[101] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[102] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[103] Paul J Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[104] Qingchen Zhang et al. "A survey on deep learning for big data". In: *Information Fusion* 42 (2018), pp. 146–157.

[105] MD Zakir Hossain et al. "A comprehensive survey of deep learning for image captioning". In: *ACM Computing Surveys (CsUR)* 51.6 (2019), pp. 1–36.

[106] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. "On the convergence of adam and beyond". In: *arXiv preprint arXiv:1904.09237* (2019).

[107] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[108] Heng Shi, Minghao Xu, and Ran Li. "Deep learning for household load forecasting—A novel pooling deep RNN". In: *IEEE Transactions on Smart Grid* 9.5 (2017), pp. 5271–5280.

[109] Gaofeng Cheng et al. "An Exploration of Dropout with LSTMs." In: *Interspeech*. 2017, pp. 1586–1590.

[110] Mohammed Alenezi et al. "Unsupervised Learning Clustering and Dynamic Transmission Scheduling for Efficient Dense LoRaWAN Networks". In: *IEEE Access* 8 (2020), pp. 191495–191509. DOI: 10.1109/ACCESS.2020.3031974.

[111] Robert McGill, John W Tukey, and Wayne A Larsen. “Variations of box plots”. In: *The American Statistician* 32.1 (1978), pp. 12–16.

Appendix A

Simulations Environments

MATLAB is used to build a communication engine based on LoRa SX1272 model for the purpose of carrying out the work in this thesis. While Python is used together with Keras library and Tensor-Flow backend version (2.4.1) to construct and train the prediction models.

The following are snapshots of the significant parts of the codes used to produce the simulation results presented in this thesis.

```
maxnrofdevices = 1000; %5000 in ultra-densenetwork
devicestepsize = 1;


%Parameters for different clusters:

maxnrofdevicesC1 = maxnrofdevices/numcluster1;


devicestepsizeCX = 1;
nrofdevicesCX = devicestepsizeCX:1:maxnrofdevicesCX;

nrofdevices = devicestepsize:devicestepsize:maxnrofdevices;
nrofdevicesC = devicestepsize:devicestepsize:maxnrofdevicesC;

nrofpackets = 1;
duration_lora = [07 5478 81]; %SF/Datarate/step
durationpack = duration_lora(:,3);

numofbits = lora_duration(:,2);
CR = 1
bitrate = (duration_lora(:,1))*(freqspan/(2^duration_lora(:,1)))*(4/(4+CR));
delay = (numofbits/bitrate); %in Seconds
```

Figure 1.1: Initialisation of The Number of Nodes, Packets, Step Size, Bits, CR, SF and Bitrate.

```
for nrCX = nrofdevicesCX
    ftCX = zeros(slotsnum, channelsnumr);
    colissionCX = zeros(nrCX, packetsnumr);
    ftnodedelayCX = zeros(slotsnum, channelsnumr);
    nodedelayCX = zeros(nrCX, packetsnumr);
    ftnodeEECX = zeros(slotsnum, channelsnumr);

    sf = randi([start_channel end_channel], [nrCX packetsnumr]);
    %time = randi([1 floor(slotsnum - (durationpack*packetsnumr)/timeinterval)], [nr 1]);


    for i = 1:nrCX
        AvailableSlot = floor((slotsnum - (durationpack(sf(i))*packetsnumr)/timeinterval) * rand(1,1));
```

Figure 1.2: Initialisation of the Collisions, Transmission Slots, Transmission Duration and The Transmission Process.

```
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout, GRU, Bidirectional
from keras.optimizers import SGD
import math
from sklearn.metrics import mean_squared_error
```

Figure 1.3: Importing The Libraries Needed For Constructing RNN Models.

```
dataset = pd.read_csv('5000-Data.csv')
dataset.head()
Target = pd.read_csv('5000-Test.csv')

print(dataset)


# Checking for missing values
training_set = dataset.iloc[:, 0:1].values
test_set = Target.iloc[:, 0:1].values

print(training_set)
print(test_set)
```

Figure 1.4: Importing The Data Set For Training The Model.

```
# Scaling the training set
sc = MinMaxScaler(feature_range=(0,1))
training_set_scaled = sc.fit_transform(training_set)
test_set_scaled = sc.fit_transform(test_set)

# Since LSTMs store long term memory state, we create a data structure with 60 timesteps and 1 output
# So for each element of training set, we have 60 previous training set elements
X_train = []
y_train = []

n_future = 1
n_past = 14

for i in range(n_past, len(training_set_scaled) - n_future +1):
    X_train.append(training_set_scaled[i - n_past:i, 0:training_set_scaled.shape[1]])
    y_train.append(training_set_scaled[i + n_future - 1:i + n_future, 0])

##for i in range(60,2769):
##    X_train.append(training_set_scaled[i-60:i,0])
##    y_train.append(training_set_scaled[i,0])
X_train, y_train = np.array(X_train), np.array(y_train)

# Reshaping X_train for efficient modelling
X_train = np.reshape(X_train, (X_train.shape[0],X_train.shape[1],1))
```

Figure 1.5: Scaling The Data Set For Better Data Explanation To The Training Model

```
#$$$$#$$$$#$$$$ LSTM #$$$$#$$$$#$$$$#$$$$
##########################################
##########################################

# The LSTM architecture
regressor = Sequential()
# First LSTM layer with Dropout regularisation
regressor.add(LSTM(units=5, return_sequences=True, input_shape=(X_train.shape[1],1)))
regressor.add(Dropout(0.2))
# Second LSTM layer
regressor.add(LSTM(units=5, return_sequences=True))
regressor.add(Dropout(0.2))
# Third LSTM layer
regressor.add(LSTM(units=5, return_sequences=True))
regressor.add(Dropout(0.2))
# Fourth LSTM layer
regressor.add(LSTM(units=5))
regressor.add(Dropout(0.2))
# The output layer
regressor.add(Dense(units=1))

# Compiling the RNN
regressor.compile(optimizer='rmsprop',loss='mean_squared_error')
# Fitting to the training set
regressor.fit(X_train,y_train,epochs=1,batch_size=32)
```

Figure 1.6: Constructing The *LSTM* Model

```
#$$$$#$$$$#$$$$$ GRU #$$$$$#$$$$#$$$$#$$$
########################################
########################################

# The GRU architecture
regressorGRU = Sequential()
# First GRU layer with Dropout regularisation
regressorGRU.add(GRU(units=5, return_sequences=True, input_shape=(X_train.shape[1],1), activation='tanh'))
regressorGRU.add(Dropout(0.2))
# Second GRU layer
regressorGRU.add(GRU(units=5, return_sequences=True, input_shape=(X_train.shape[1],1), activation='tanh'))
regressorGRU.add(Dropout(0.2))
# Third GRU layer
regressorGRU.add(GRU(units=5, return_sequences=True, input_shape=(X_train.shape[1],1), activation='tanh'))
regressorGRU.add(Dropout(0.2))
# Fourth GRU layer
regressorGRU.add(GRU(units=5, activation='tanh'))
regressorGRU.add(Dropout(0.2))
# The output layer
regressorGRU.add(Dense(units=1))

# Compiling the RNN
regressorGRU.compile(optimizer=SGD(lr=0.01, decay=1e-7, momentum=0.9, nesterov=False),loss='mean_squared_error')
# Fitting to the training set
regressorGRU.fit(X_train,y_train,epochs=1,batch_size=150)
```

Figure 1.7: Constructing The *GRU* Model