Implementing Efficient and Multi-Hop Image Acquisition In Remote Monitoring IoT systems using LoRa Technology

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Tonghao Chen

©Tonghao Chen, August/2019. All rights reserved.

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Abstract

Remote sensing or monitoring through the deployment of wireless sensor networks (WSNs) is considered an economical and convenient manner in which to collect information without cumbersome human intervention. Unfortunately, due to challenging deployment conditions, such as large geographic area, and lack of electricity and network infrastructure, designing such wireless sensor networks for large-scale farms or forests is difficult and expensive. Many WSN-appropriate wireless technologies, such as Wi-Fi, Bluetooth, Zigbee and 6LoWPAN, have been widely adopted in remote sensing. The performance of these technologies, however, is not sufficient for use across large areas. Generally, as the geographical scope expands, more devices need to be employed to expand network coverage, so the number and cost of devices in wireless sensor networks will increase dramatically. Besides, this type of deployment usually not only has a high probability of failure and high transmission costs, but also imposes additional overhead on system management and maintenance.

LoRa is an emerging physical layer standard for long range wireless communication. By utilizing chirp spread spectrum modulation, LoRa features a long communication range and broad signal coverage. At the same time, LoRa also has low power consumption. Thus, LoRa outperforms similar technologies in terms of hardware cost, power consumption and radio coverage. It is also considered to be one of the promising solutions for the future of the Internet of Things (IoT). As the research and development of LoRa are still in its early stages, it lacks sufficient support for multi-packet transport and complex deployment topologies. Therefore, LoRa is not able to further expand its network coverage and efficiently support big data transfers like other conventional technologies. Besides, due to the smaller payload and data rate in LoRa physical design, it is more challenging to implement these features in LoRa. These shortcomings limit the potential for LoRa to be used in more productive application scenarios.

This thesis addresses the problem of multi-packet and multi-hop transmission using LoRa by proposing two novel protocols, namely Multi-Packet LoRa (MPLR) and Multi-Hop LoRa (MHLR). LoRa's ability to transmit large messages is first evaluated in this thesis, and then the protocols are well designed and implemented to enrich LoRa's possibilities in image transmission applications and multi-hop topologies. MPLR introduces a reliable transport mechanism for multi-packet sensory data, making its network not limited to the transmission of small sensor data only. In collaboration with a data channel reservation technique, MPLR is able to greatly mitigate data collisions caused by the increased transmission time in laboratory experiments. MHLR realizes efficient routing in LoRa multi-hop transmission by utilizing the power of machine learning. The results of both indoor and outdoor experiments show that the machine learning based routing is effective in wireless sensor networks.

Acknowledgements

First and foremost, I would like to show my deepest gratitude to my supervisors, Dr. Derek Eager and Dr. Dwight Makaroff, who are respectable, responsible and resourceful scholars. Both of them have provided me with valuable guidance in every stage of the writing of this thesis and earlier paper. Without their enlightening instruction, impressive kindness and patience, I could not have completed my thesis. Their keen and vigorous academic observation enlightens me not only in this thesis but also in my future career. Thanks a billion.

Here I also want to thank those who have been encouraging me and supporting me all the time. Without their trust and help, I could not have the strong motivations to urge me working hard on this dissertation. Thank you all.

Contents

List of Tables

List of Figures

List of Abbreviations

AODV	Ad Hoc On-Demand Distance Vector
ARQ	Automatic Repeat Request
BVACK	Bit Vector Acknowledgement
BW	Bandwidth
CRC	Cyclic Redundancy Check
CSMA	Carrier Sense Medium Access
CSS	Chirp Spread Spectrum
DSDV	Destination-Sequenced Distance-Vector
DTN	Delay Tolerant Network
FEC	Forward Error Correction
LPWAN	Low-Power Wide-Area Network
IoT	Internet of Things
JPEG	Joint Photographic Experts Group
MHLR	Multi-Hop LoRa
MLP	Multi-layer Perceptron
MPLR	Multi-Packet LoRa
MTU	Maximum Transport Unit
ROC	Receiver Operating Characteristic
RPi	Raspberry Pi
RSSI	Received Signal Strength Indicator
SF	Spreading Factor
SGD	Stochastic Gradient Descent
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
WSN	Wireless Sensor Network

Chapter 1

Introduction

In the past decade, with the rapid development of battery and wireless communication technology, the concept of the Internet of Things has been widely popularized and realized. Internet of Things, or IoT, is the interconnection of various small computing units and sensor/actuator devices to the Internet [72], thus enabling a wide range of the communicable network to achieve common purposes, such as data acquisition and system automation. IoT systems, at the time of writing, have already been applied in a variety of different scenarios. Some well-known applications are, for instance, smart home automation system [81], precision agriculture [20], and water quality monitoring and controlling [72]. The emergence of these IoT-based applications has made our daily lives and work easier and more convenient, and is affecting the way we communicate with things.

1.1 Agricultural IoT Systems

IoT systems are highly sought after in remote sensing and have been explored as an effective technology for improving agricultural production management and maintenance [43] [73]. As shown in Figure 1.1, by deploying small battery-powered devices with sensors in agricultural fields, data such as temperature, moisture, and soil nutrient levels can be periodically and remotely collected. Each sensed data item can then be either sent back to a gateway device or obtained by field users and eventually uploaded to a server for remote users to access. Such systems have also greatly lowered the cost of agricultural management and reduced human intervention [22]. Similar systems can also be applied to livestock farming [52] and wildlife conservation [7], etc. The IoT systems are successful because of their convenience and flexibility. Before the emergence of IoT systems, collecting this type of data was done manually. This was considered cumbersome and tedious, and often required massive human and material resources. After entering the era of Big Data, it is expected that a larger amount of data will be collected and stored to help people better understand the environmental conditions and other state changes in remote areas. Nonetheless, this also comes with more frequent and broader data collection. Due to the low efficiency and flexibility of manual acquisition, the manual data acquisition method can no longer meet the varying requirements of the massive data collection task. As an alternative option, the use of IoT systems can reduce human intervention by deploying small devices for collecting data. Commonly, an IoT system only needs to be deployed and configured once during

the construction, and it then can operate for a long time. Even though each IoT device is powered by a battery, the lifetime of a typical IoT system is usually around a few years, depending on the power consumption of IoT devices [48].

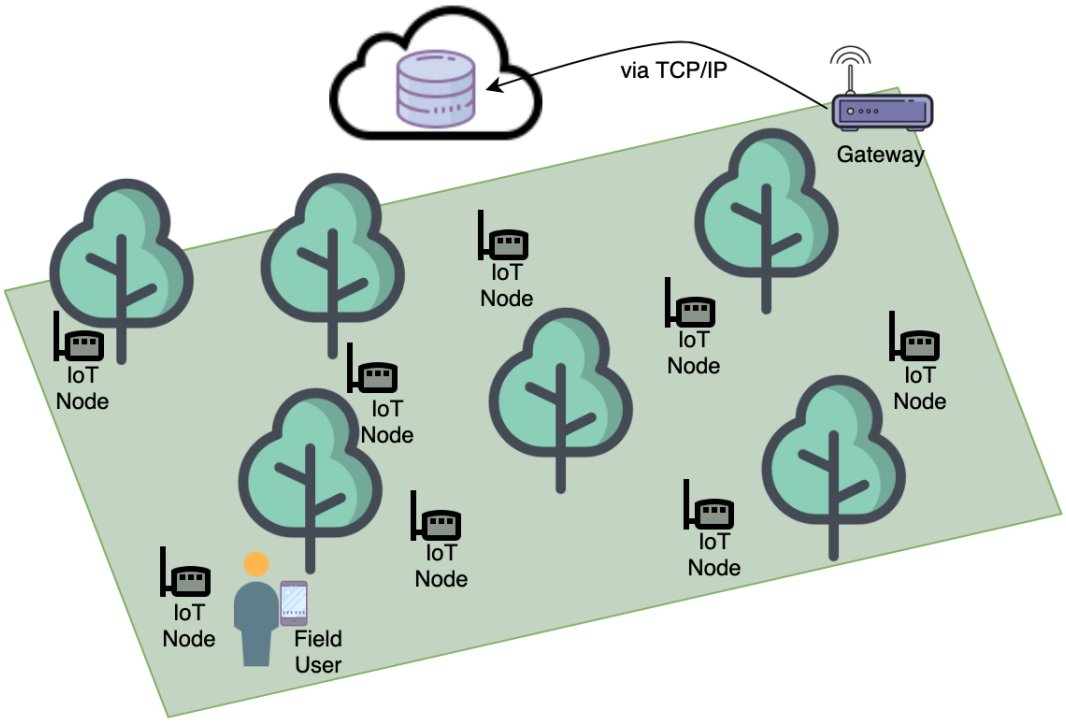


Figure 1.1: Context View: Precision Agriculture[1]

The data in IoT systems also is time related. Even in a delay-tolerant acquisition project, data collection within reasonable delivery latency and accurate data reporting is essential and determines the success of the entire project. For those projects that require real-time performance, such as environmental monitoring and status reporting, if the data is received after its expiration date, the data loses its value. Manual data collection often cannot achieve a good timeliness. In order to reduce the travel cost and maximize the harvest of a single acquisition, technicians/scientists often drive kilometres to multiple locations for collection and bring all the data back at once after all the collection points have been visited. It is time-consuming and even the first collected data must wait until the collection task is over before it can be returned. Thus, the timeliness cannot be guaranteed. Conversely, there is no such trouble in using IoT systems, since both the data collection and transmission are proactive. Combining with a variety of other features, IoT systems for data acquisition in precision agriculture and environmental sensing are extremely popular now and are considered to be a highly promising solution.

[1]Recreated From Bajceta *et al.* [8].

1.2 Wireless Sensor Network based IoT systems

The use of wired networks on remote agricultural land is typically not feasible, due to the lack of reliable network and electricity infrastructure. Wireless sensor network (WSN) technology is considered to be an effective and economical solution for agricultural IoT systems [16]. A WSN is comprised of spatially distributed autonomous sensor-equipped devices to monitor physical or environmental conditions and transmit the sensor data back to the data warehouse via wireless media, so that the data can be stored for further analysis [5]. Most IoT systems for remote sensing rely on WSN-based data acquisition systems. The basic components of an IoT remote sensing system are shown in Figure 1.2. Wireless networks are responsible for transmitting data collected from each sensor node to the gateway device. Unlike conventional wireless networks, devices in WSNs are typically powered by batteries. Without rechargeable capacity, the total electricity of each device in the WSN is fixed and the lifetime is limited. Therefore, in the design of low-budget WSNs, power consumption and device lifetime are often considered as key factors. A low-power WSN can operate for months to years without recharging or battery replacement. Alternatively, small solar panels can be equipped on the sensing device to provide charging capability. This approach increases the cost of hardware and is geographically constrained.

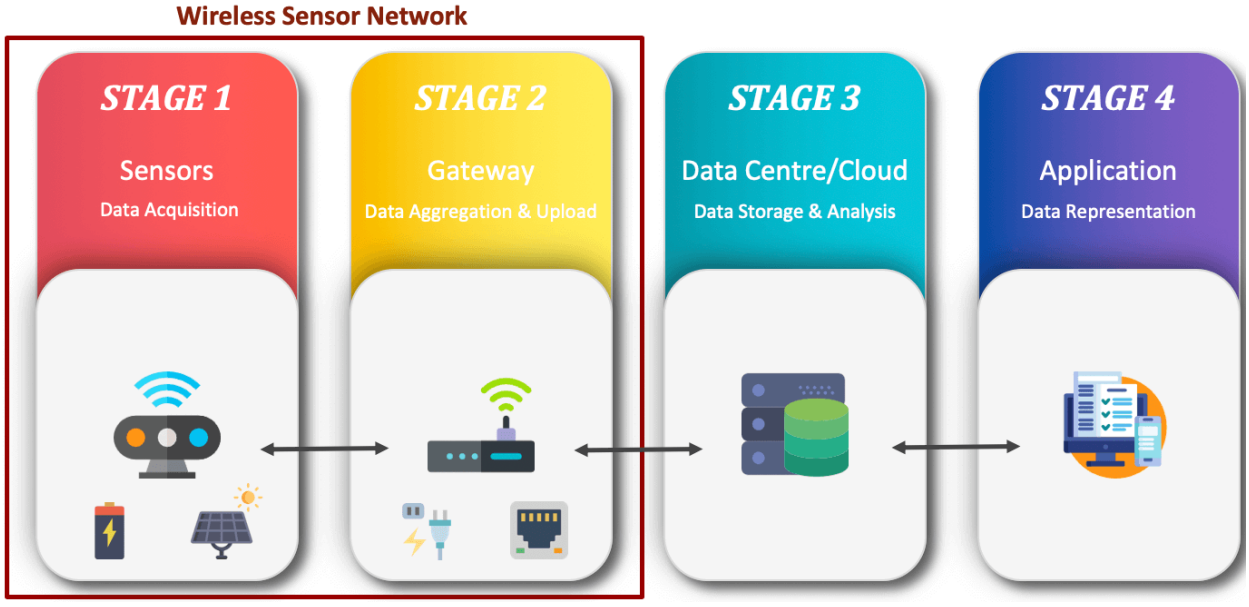


Figure 1.2: Procedures of IoT-Based Remote Sensing

WSNs have been extensively researched over the past two decades, while various related physical layer standards and protocols have been proposed. Among them, there are many well known physical layer standards, for example, Wi-Fi and Bluetooth [80]. Also, IEEE 802.15.4-based low-power-consumption wireless standards have also been prevalent in recent years, such as Zigbee [15, 43, 64] and 6LoWPAN [73]. Each of these standards is associated with one or more unique network protocols to ensure the reliability and

correctness of data transmission.

WSNs are often used with a mesh topology. In a mesh topology, each device can operate not only as a node but also as a relay; packets are forwarded to and from an Internet-connected gateway in a multi-hop fashion [36]. Such networks have the ability of self-arranging and self-configuration, with mesh connectivity established among nodes in the network automatically. Compared with the way that all nodes in a star topology directly connect to the gateway [45], the use of a mesh topology provides a better flexibility and convenience, and enables the network expansion regardless of the signal limits. Mesh topologies also involve some other protocols, such as routing and transport protocols. All in all, these various protocols and features enrich the application scenarios of WSN-based IoT systems. When designing a WSN, different wireless technologies and protocols can be adopted in different scenarios according to actual needs and environmental factors, in order to maximize the overall performance.

1.3 Difficulty and Challenge

While practical WSN applications are becoming more versatile, there are still challenges in using WSN for sensing/monitoring across large geographic areas. The main reason is that conventional low-cost wireless technologies are only suitable for use across small geographic areas, and their advantages are no longer evident when facing a large area. The wireless transmission standards mentioned above, such as Wi-Fi, Bluetooth, Zigbee [43] [64] [15], and 6LoWPAN [73], are frequently used in buildings, parking lots and other small areas, because their average signal range is only about 10-100 metres [9, 29, 41, 74]. Unfortunately, in applications such as sensing/monitoring for large farms, such limited signal coverage is clearly not wide enough.

Using a mesh network may improve the network coverage to a certain extent; increasing the size of the mesh network, also increases, however, the equipment cost and network overhead at the same time. In Saskatchewan, for instance, the average farm area is 1449 acres in 2016 [55]. In order to entirely cover an average size farm in Saskatchewan by a WSN using a radio technology with a range of about 100 metres (assuming continuous land location), approximately a total number of 187 devices must be used, and the maximum number of communication hops is about 28. Networks with a large number of devices not only have excessive equipment and maintenance costs, but also have higher network overhead and error probability. Note also that farmlands are often divided by roads and rivers and not in one piece and the actual areas that a WSN needs to cover will be correspondingly larger. Therefore, the previously mentioned wireless technologies cannot adequately address the requirements of sensing/monitoring for Saskatchewan farms.

1.4 The Feasibility of Cellular Networks

Cellular networks are considered to be one of the alternatives for data acquisition systems. However, the coverage of cellular networks is usually restricted and unstable in rural areas. Figure 1.3 provides some

interesting data from OpenSignal Inc.[2] regarding the cellular network converge in Saskatchewan, Canada, in 2018. The map shows that the cellular network mainly serves the cities and major highways. However, outside these areas, coverage is either non-existent, or poor quality as indicated by the large number of bad signal points (red dots). Because of the spotty signal coverage, the cellular network cannot be considered as a guaranteed solution to remote sensing in western Canada. Furthermore, since the cellular network operates on licensed bands, charges are required for network connection, and the price is usually proportional to the number of devices and the amount of data [61]. Therefore, the cellular network is not preferred in this case.

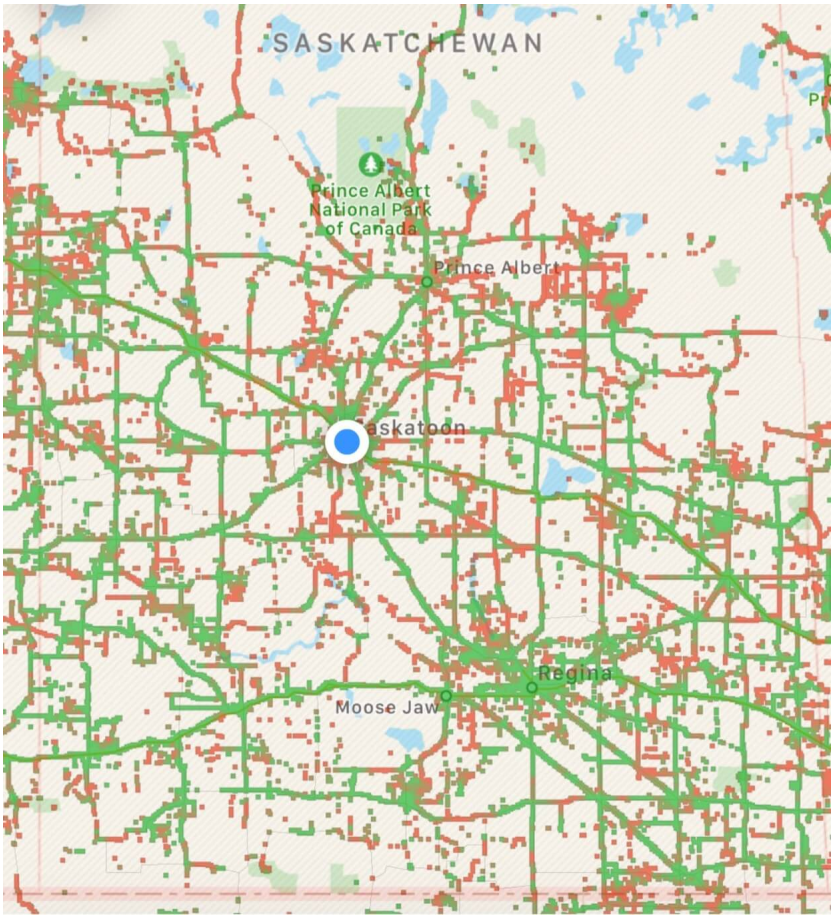


Figure 1.3: Cellular Converge Map in Saskatchewan From OpenSignal IOS App.[3]

1.5 LPWAN and LoRa

The emergence of LoRa fullfills the need for long-range communication in WSNs. LoRa is one of the standards for Low-Power Wide-Area Network (LPWAN). Chipsets implementing LoRa are developed and manufactured by Semtech.[4] The advantage of LoRa is the use of chirp spread spectrum (CSS) modulation, which allows a

[2]https://opensignal.com/networks
[3]from OpenSignal IOS app. Screenshot by author. Link: https://itunes.apple.com/app/opensignal/id598298030
[4]https://www.semtech.com/lora

LoRa signal to be transmitted long distances with relatively low device cost. Centenaro *et al.* describe the coverage range of LoRa as being 10-15 km in rural deployments and 3-5 km in urban deployments [11]. Such a long transmission distance makes LoRa a practical solution for long-distance IoT. At the same time, the power consumption of LoRa is also very low. Figure 1.4 provides some interesting information regarding the comparison between LoRa and other wireless technologies. From the figure, it is clear that LoRa has a longer signal range than any of the considered non-LoRa technologies, while still offering a relatively low power consumption. Because of its long-range and low-power nature, LoRa technology has been applied to many different fields. LoRa based applications are widely deployed now, for example in smart irrigation [83], soil moisture monitoring [53], rice field management [25], and provision of intelligent agricultural services [47].

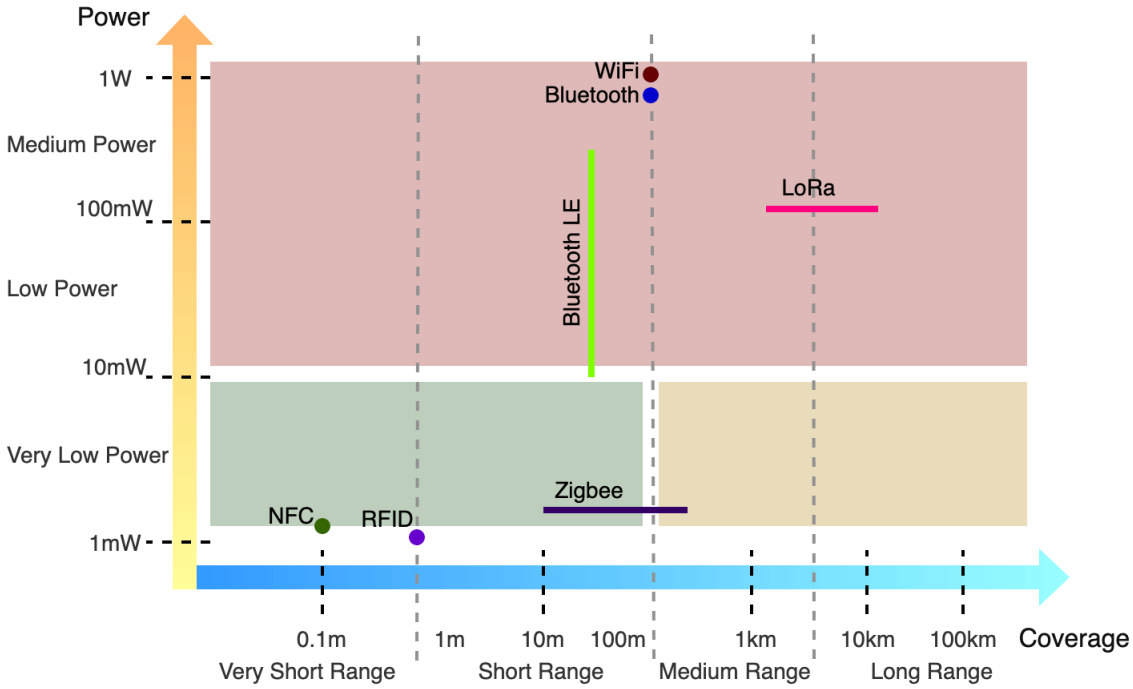


Figure 1.4: Power Consumption Versus Coverage for Different Wireless Technologies[5]

Using a LoRa-based WSN can significantly reduce the number of devices and bring additional benefits. For example, the costs of construction and maintenance are often proportional to the number of devices. If the number of devices drops, the construction and maintenance costs are reduced as well. Second, a multi-hop network with a small number of devices will also result in a lower maximum or average hop number. As the number of hops decreases, so does the power consumption, because fewer devices are involved on each path. The reduction in power consumption not only extends the life of the network, but also reduces the human and material resources for battery replacement. The error rate and inter-flow interference in the network will be diminished as the number of devices decreases.

[5]Recreated From Shirvanimoghaddam *et al.* [71].

1.6 Thesis Motivation

Application of IoT systems to the monitoring of crop growth has the potential to increase the efficiency of both crop breeding and production. However, this requires improved capabilities for collection of image data. For example in the P^2IRC project,[6] researchers are developing crop monitoring and data analysis tools to assist crop breeders. A current focus in this project concerns collection/analysis of images from field test plots. This has motivated the investigation of the possible use of a multi-hop LoRa network for diagnostic image collection.

Unfortunately, LoRaWAN, the current standard protocol for LoRa, supports only single-packet-at-a-time and single-hop transmission [1]. LoRaWAN is a media access control protocol and does not provide any special support for transmission of large messages. With the LoRaWAN architecture a star network topology is used. A star topology is simple and easy to manage, however, it limits the scalability of a LoRa wireless network. Although multiple gateways can be deployed to form multiple star topologies and extend the physical network range, a gateway in a LoRa network requires an Internet connection, and has substantially higher cost and power consumption requirements than an end device. Thus, use of multiple gateways may be undesirable or infeasible in an agricultural IoT system.

Multi-hop image transmission using LoRa is challenging. First, the performance of LoRa has rarely been studied in a wide range of applications, especially large message transmission. Without actual measurements, the feasibility of LoRa being used for large message transmission cannot be confirmed. Second, due to the small maximum transmission unit (MTU) of LoRa, a large message, such as an image, must be transmitted using many packets. Such continuous traffic increases not only the network utilization, but also increases the chance of packet collisions. Third, LoRa's low physical layer data rate makes it difficult to broadcast additional data frequently enough to maintain the network topology while performing image transmission tasks, which may lead to serious network congestion or even paralysis. Furthermore, in some jurisdictions, such as in Europe, there are frequency band duty cycle limits that must be considered [14]. (In other jurisdictions, such as in Canada and the US, other restrictions apply to the frequency band used for LoRa but these do not prevent long-duration or frequent transmissions [14] [32].) Finally, any protocol design needs to be as lean and lightweight as possible.

Pham proposed a carrier sense medium access (CSMA) protocol adapted to LoRa networks to avoid packet collisions for image data and transmitted small images through LoRa [56] [57]. Jebril *et al.* demonstrated the concept of point-to-point image transmission using LoRa at a variety of locations and showed that image transmission can be done with delivery times varying between 1 and 14 minutes depending on the spreading factor [33]. Both studies examine LoRa's possibilities in single-hop image transmission and the former reduces the packet collisions by applying CSMA. However, neither study investigated transport layer techniques for improving image transmission performance. As a result, the former can only transfer images less than 1

[6] https://p2irc.usask.ca/

KB, while the latter requires a fairly long transmission time. The first motivation, therefore, is to reduce point-to-point transmission time in LoRa networks from a transport layer perspective.

Liao *et al.* proposed a multi-hop LoRa wireless network using concurrent transmission to flood identical or different packets in parallel [44]. Sartori *et al.* described an approach using RPL, an IPv6 based routing protocol for low-power and lossy networks, to achieve deterministic multi-hop routing using the device addresses in LoRa networks [65]. The above methods are applicable to LoRa networks that transmit sensor data. However, when it comes to transmitting large messages, the flooding method will lead to serious network congestion or paralysis, while use of an IPv6 based protocol will require more packets to complete the transmission because the packet header is too large relative to the entire packet length, which will undoubtedly increase the overhead. Therefore, the second motivation is to design a lightweight multi-hop transport protocol for LoRa to achieve explicit packet routing on multiple hops while minimizing the overhead and network utilization.

An effective design needs to consider the specific physical characteristics of LoRa and minimize network overhead. Although, LoRa is designed initially to transmit single-packet data only, LoRa's data rate can still support transmission of larger messages, like small images. Such data transmission in LoRa can be achieved by designing a LoRa-specific transport protocol. If LoRa wireless networks can transmit larger messages, such networks can be applied to more scenarios. To the best of my knowledge, there are no prior studies that have addressed the multi-hop transmission for multi-packet data in LoRa.

1.7 Thesis Statement

This thesis intends to determine and prove the feasibility of LoRa in multi-packet transmission applications and multi-hop topologies. Two novel and lightweight protocols are proposed in this thesis, namely Multi-Packet LoRa (MPLR) and Multi-Hop LoRa (MHLR). MPLR introduces a reliable and fast multi-packet data transmission mechanism for LoRa, and MHLR utilizes the power of machine learning algorithms to achieve effective routing in LoRa multi-hop transmission. By minimizing the transmission time and reducing inter-flow interference, the proposed protocols can efficiently transmit multi-packet data, such as images, from the node to the gateway in a multi-hop manner. Both protocols are implemented in Micro-Python, and a set of performance experiments are carried out using a LoRa testbed network in laboratory, indoor and outdoor environments, measuring throughput, transmission time, packet collisions, inter-node fairness, prediction accuracy, and connection establishment delay.

1.8 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 discusses background and related works. Chapters 3 and 4 discuss the design of proposed solutions. Chapter 5 describes the experimental environment for the proposed work. Chapter 6 introduces the test components and tools used in this study and the results of the

performance evaluation experiments. Chapter 7 contains the conclusions and provides directions for future work.

Chapter 2

Background and Literature Review

2.1 LoRa Hardware and LoRaWAN Protocols

LoRa is one of the physical layer standards for LPWAN and was first developed and manufactured by Semtech.[1] It modulates signals in a sub-GHz ISM band using a chirp spread spectrum (CSS) technique and spreads a narrow band input signal over a wider channel bandwidth, in order to makes it resilient to radio frequency interference caused by other radio frequency devices and environmental noise [62]. LoRa can operate on several different frequency bands, due to the different regulatory requirements from region to region. For example, 868 Mhz and 915 Mhz frequency bands are used for LoRa communication in the EU and US/CA respectively [2]. Depending on the selected spreading factor and channel bandwidth, LoRa's physical layer data rate is from 300 bps to 37.5 kbps, while its signal range is from 10 km to 15 km line-of-sight in rural areas and 3 km to 5 km line-of-sight in urban areas [11]. For example, Seye *et al.* [68] evaluated LoRa's signal coverage in the Dakar peninsula and they observed good signal coverage and strength with a maximum communication range of 10 km. Although LoRa has such extensive signal coverage, its power consumption stays low. According to the measurement work done by Cheong *et al.* [12], the power consumption of a LoRa module is 117 mA in transmission (TX) mode and is 1.8 µA in sleep mode. In addition, a battery lifetime prediction shows that a battery with a capacity of 2000 mAh ensures a 10-year operational life of a typical LoRa device [12]. Due to this low power consumption characteristic and long signal range, LoRa has gained a lot of commercial and academic attention in recent years.

2.1.1 LoRa Physical Layer

Chipset

According to the product list[2] on the Semtech official website, there are two major types of chipsets designed for LoRa communications. One is a gateway dedicated chip, the main representatives being the SX1301 and SX1308, for outdoor and indoor communications, respectively. Both of them support up to 8 programmable parallel demodulation paths and dual digital Tx & Rx radio front-end interfaces. The only difference between

[1] https://www.semtech.com/lora
[2] https://www.semtech.com/products/wireless-rf

them is the sensitivity, which is the minimum magnitude of input signal required to produce a specified output signal having a specified signal-to-noise ratio [49]. The other chipset type is designed for node devices. On the product list, only the SX1272/73 and SX1276/78 are designed for US unlicensed frequency bands. These chips can only receive or transmit on one channel at a time, and so for most applications that do not require an acknowledgement would not be suitable for use for a gateway.

Spreading Factor and Bandwidth

LoRa supports a range of spreading factors (SF) to decide the trade-off between range and data rate [62]. From SF 7 to SF 12, a higher spreading factor can significantly extend the communication range, but also lowers the data rate and can cause more severe data collision at the receiver due to the longer on-air time [19]. Figure 2.1 demonstrates the change in length of a chirp symbol with a bandwidth of 250 kHz at SF 7, SF 8, SF 9, SF 10, SF 11, and SF 12 settings, respectively. As seen in the figure, an increase in SF by 1 doubles the transmission time of a symbol. The number of bits encoded by each symbol equals SF and so an increase in SF by 1 increases the number of bits transmitted per symbol by 1, but the data rate in bits per second still decreases because of the increased symbol transmission time. In Figure 2.1, linear "up-chirps" are shown. SF bits can be encoded per symbol by choosing one of 2^{SF} possible cyclical shifts of each chirp. LoRa also operates on three different bandwidths (BW), 125 kHz, 250 kHz, and 500 kHz respectively [19]. A wider bandwidth enables a higher data rate, but is less resilient to noise.

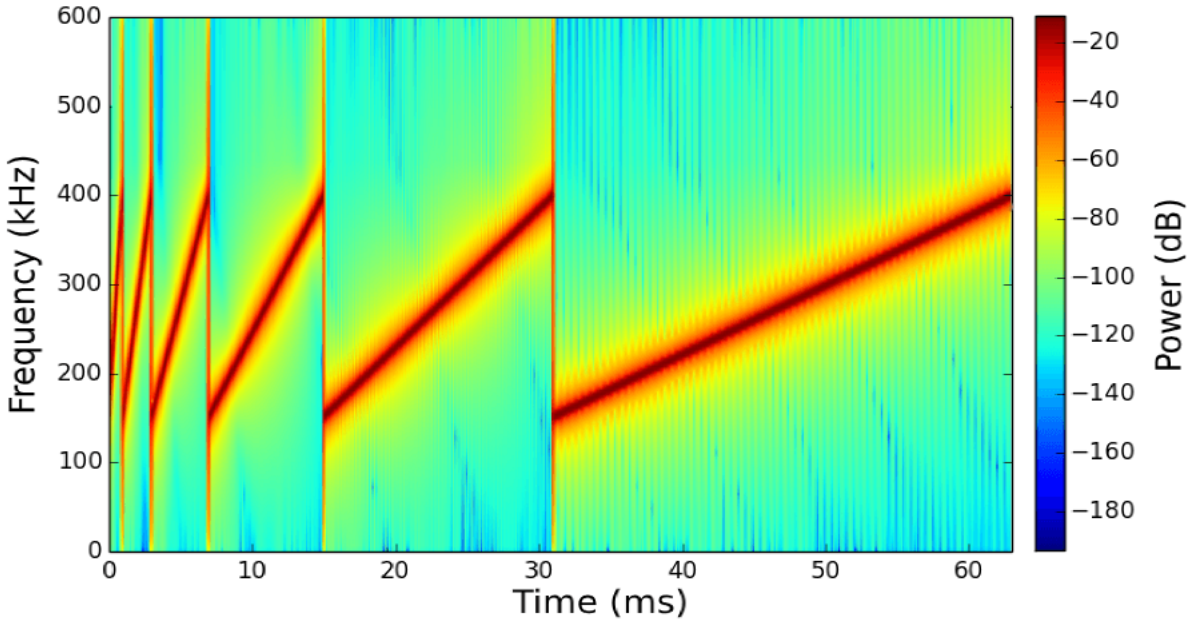


Figure 2.1: Comparison of LoRa Spreading Factors: SF 7 to SF 12[3]

SF and BW are two important factors in LoRa and different combinations of SF and BW lead to a variety

[3]Modified from http://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html

of data rates. The physical layer data rate of LoRa is given by the following expression [6]:

$$SF \times \frac{BW}{2^{SF}} \times CR, \tag{2.1}$$

where SF is the spreading factor, BW is the bandwidth, and CR forward error correction (FEC) code rate. With FEC, redundancy is added to the data stream to facilitate correction in case some bits are received incorrectly due to noise on the transmission channel. The code rate is defined as the proportion of the data stream that is non-redundant. Using this equation, the physical layer data rate of LoRa can be calculated for given values of SF, BW, and CR. Table 2.1 lists the physical layer data rate of LoRa under all possible combinations of spreading factor and bandwidth when the code rate is 4/5.

Table 2.1: LoRa Physical Layer Data Rate (in kbps, CR=4/5)

	500 kHz	250 kHz	125 kHz
SF 7	21.88	10.94	5.47
SF 8	12.5	6.25	3.12
SF 9	7.03	3.52	1.76
SF 10	3.91	1.95	0.98
SF 11	2.15	1.07	0.54
SF 12	1.17	0.59	0.29

The selection of SF and BW is mainly based on the received signal strength indicator (RSSI) value of the transport request packet. The RSSI is a measure of the power of the received signal relative to a reference value, in decibels. The minimum RSSI values required for different SF and BW combinations can be found on the applicable Semtech data sheet.

RSSI vs. Spreading Factor and Bandwidth

Due to different communication distances and obstacles between devices, RSSI values depend on the actual deployment. In LoRa, RSSI values determine the availability of the spreading factor and bandwidth. In other words, the RSSI value received by the receiver determines the SF and BW values that can feasibly be used for transmission. Semtech has given the sensitivity for each SF and BW combination in SX1276 datasheet, as shown in Table 2.2. This table records the minimum RSSI value required for each spreading factor and bandwidth. This table allows one to quickly and dynamically select the available and appropriate spreading factor and bandwidth.

Packet Structure

The packet structure of LoRa is presented in Figure 2.2. As shown in the packet structure diagram, the section between *Preamble* and *Payload* is encoded at a fixed code rate of 4/8, while the code rate of the *Payload*

Table 2.2: SX1276 LoRa Modulation Receiver Sensitivity (in dBm) [66]

	500 kHz	250 kHz	125 kHz
SF 7	-118	-122	-125
SF 8	-121	-125	-128
SF 9	-124	-128	-131
SF 10	-127	-131	-134
SF 11	-129	-133	-136
SF 12	-130	-134	-137

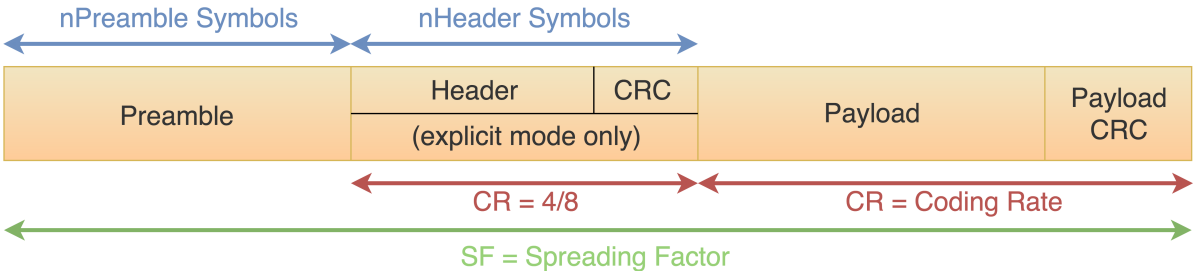


Figure 2.2: LoRa Packet Structure[4]

and the *Payload CRC* (cyclic redundancy check, an error detecting code) is optional within a certain range (details are described below). Therefore, the length of the encoded *Payload* and *Payload CRC* is variable and depends on the selected code rate.

Preamble

At the beginning of each packet, 12 up-chirp preamble symbols are added by default. It is used to synchronize the receiver with the incoming data flow. The size of the preamble can be reduced to minimize the duty-cycle at the receiver, but 8 bytes is the minimum preamble length.

Header

The header provides 3 major pieces of information about the current packet, the payload length in bytes, the payload FEC code rate, and whether or not an optional 16-bit CRC for the payload is used. The payload FEC code rate can be one of four possible values, namely 4/5, 4/6, 4/7, or 4/8. Smaller code rates have better error detection and correction capabilities, but incur more overhead. The header also has its own CRC for the receiver to validate. The header is explicit by default. However, in case the payload length, code rate, and CRC presence are fixed or known in advance, implicit header mode can be enabled, which removes the header from the packet and reduces the bytes that need to be transmitted.

[4]Recreated From the SX1276 Datasheet.

Data Payload

The data payload is a variable-length field that contains the actual application data. Depending on the selected spreading factor, the maximum length of the data payload in one LoRa packet is between 51 bytes and 255 bytes [1]. Compared to other wireless standards, LoRa's data payload is relatively small. This makes it more difficult to utilize complex protocol stacks with LoRa and to transfer large data units.

Payload CRC

The payload CRC is a 16 bit cyclic redundancy check value placed at the end of the packet, computed over the payload bits. It is optional, but allows the receiver to discard packets with invalid payload.

On-Air-Time Calculation

The On-Air-Time calculation is necessary and useful to predict the system duty cycle and channel occupancy. For LoRa, the estimated on-air-time can be computed by using the formulas offered in the Semtech datasheet [66]. Based on the choice of SF and BW, the symbol rate, R_{sym}, can be computed as

$$R_{sym} = \frac{BW}{2^{SF}}. \tag{2.2}$$

Once the symbol rate is computed, on-air time can then be computed, based on the symbol rate and packet length. The preamble duration can be expressed as

$$T_{preamble} = (n_{preamble} + 4.25) \times \frac{1}{R_{sym}}, \tag{2.3}$$

where $n_{preamble}$ is the programmed preamble length. The following formula gives the number of payload symbols:

$$n_{payload} = 8 + max(ceil(\frac{8PL - 4SF + 28 + 16 - 20H}{4(SF - 2DE)})(CR + 4), 0), \tag{2.4}$$

where PL is the number of payload bytes, H is 1 if an explicit header is used and is 0 otherwise, DE is 1 if a Semtech "low data rate" optimization is configured at both the sender and receiver and is 0 otherwise, and CR is the code rate. The payload duration can then be calculated as

$$T_{payload} = n_{payload} \times \frac{1}{R_{sym}}. \tag{2.5}$$

The total on-air-time of the entire packet becomes

$$T_{packet} = T_{preamble} + T_{payload}. \tag{2.6}$$

2.1.2 LoRaWAN Protocol

LoRaWAN is a well-known LPWAN media access control protocol that is designed for networks using LoRa or frequency-shift keying communication specifically. LoRaWAN networks use star topologies, where each node device only connects to one or many gateways through a single-hop communication [11]. A gateway is defined as a sink point in a LoRaWAN network where messages are forwarded from the LoRaWAN network to a central server through a TCP/IP connection. Roughly, up to 10,000 end devices can connect to a single gateway simultaneously [77]. Even so, due to the limited number of available channels, the overall throughput will decrease as the number of connected node devices increases [1]. According to LoRaWAN Specification v1.1,[5] node devices in LoRaWAN are classified into the following classes based on the different requirements of applications:

- Class A. Only node devices can initiate communication, and each communication is asynchronous. Each uplink transmission can be sent at any time, based on an ALOHA type of protocol. Each communication is followed by two short downlink windows for downlink transmission from the gateway. Such devices are considered the most power-efficient and can often run for years without replacing a battery.

- Class B. Based on class A, class B devices also utilize periodic beacons to synchronize with the network and open downlink at scheduled times. This allows the gateway to proactively initiate downlink communication at a scheduled time. Devices in class B consume more power than devices in class A due to the scheduled opening of the reception window. In the protocols proposed in this thesis, all non-gateway devices fall into this class.

- Class C. Unlike class A and class B devices, all node devices in class C always keep the downlink window open. In other words, the gateway can initiate downlink transmission at any time. Devices in this class are considered the most power-hungry, and typically only gateway devices fall into this class.

Limitations in LoRa/LoRaWAN

As LoRa and LoRaWAN are still in an early stage of development, their application scenarios have not been explored very much. Therefore, the LoRaWAN protocol also has the following shortcomings:

- **Limited MTU and No Transport Protocol.** According to the Semtech data sheet [66], the maximum payload size of a LoRa packet is between 51 bytes and 255 bytes, depending on the SF setting. Such payload size is only sufficient for small sensor data values, such as temperature and humidity, motion detection, and ambient light level, etc. For large data units, many packets have to be used for transmission. However, there is currently no transport protocol designed for LoRa wireless networks that efficiently supports transmission of messages requiring many packets.

[5] https://lora-alliance.org/resource-hub/lorawantm-specification-v11

- **Random Channel Access.** In LoRaWAN, pure ALOHA is utilized for media access control [1]. Pure ALOHA works very simply: it transmits whenever there is data to send, without sensing whether another transmission occurs on the same channel and time [76]. When a collision happens, the packets will not be received correctly and retransmissions will be required. Typically, as the number of end devices grows, the maximum throughput decreases as the chance of packet collisions increases. Thus, the random nature of pure ALOHA is not efficient and optimal in IoT systems [1]. In random channel access, each node device can send a data packet without sensing the status of the channel. In other words, there is no busy period detection before transmitting, and this can lead to a high probability of message collision during the busy period. When a collision occurs, a packet that is being transmitted has to be discarded and needs to be retransmitted again later. As a result, electricity is wasted. Also, the throughput cannot be guaranteed and latency is unbounded when random channel access is used, as node devices may need a very long time to send data successfully when the network is busy.

- **Single Hop Communication.** In reality, the actual maximum transmission distance is usually less than the theoretical distance. This is often due to the uneven terrain and obstacles, which weaken the signal. In such cases, single hop communication is no longer sufficient, no matter what the theoretical signal range is, and multi-hop communication is needed for bypassing the obstacles. Multi-hop communication is also helpful for extending the network to an even broader range. In other wireless communication protocols, such as Bluetooth, ZigBee, and Wi-Fi, multi-hop communication is feasible and helpful in many aspects. In contrast, LoRa lacks support for it.

2.2 LoRa Applications and Research Studies

Several studies have investigated the applicability and capacity of LoRa to facilitate agricultural management tasks. Zhao *et al.* proposed a smart irrigation system based on LoRa that connects to the Internet at the gateway [83]. Their system provides a convenient method for remote users to send commands to irrigation nodes and receive status information from irrigation nodes. With the aid of LoRa, their system has lower power consumption than a system using GPRS and allows a communication distance between the irrigation node and gateway of up to 8 km.

Payero *et al.* developed and tested a LoRa-based WSN to monitor soil moisture [53]. They constructed several sensor nodes using RFM95 LoRa radio and Decagon EC-5 sensors to collect soil moisture and send data to the coordinating gateway for further aggregation through the Internet. Their system was evaluated in a wheat field and accurately read the value of soil moisture from remote locations.

Ma and Chen used a LoRa-based wireless sensor network to develop a service platform for intelligent agriculture [47]. A multi-sensor component was built and deployed to sense the carbon dioxide concentration, temperature, air humidity, light intensity, soil temperature, soil moisture, wind direction and wind speed from the environment. Collected data was transmitted to a base station through LoRa communication.

Seye *et al.* performed extensive experiments to evaluate the signal coverage and strength of LoRa in Senegal [68] [67]. When using a spreading factor of 12, they observed good signal coverage and strength with a maximum communication range of 10 km in urban areas (Dakar, Senegal) and 15 km in rural areas (Namarel, Senegal). Based on these measurements, a proof-of-concept architecture called the COWShED was deployed to enable LoRa communication between livestock herders following seasonal patterns of migration in the Ferlo region, many areas of which lack cellular coverage [67].

2.2.1 Image Transmission Through LoRa

Due to the advantages of LoRa in communication range and power consumption, more possible use cases of LoRa are being explored, including image transmission. Pham realizes small image transmission in LoRa network through a series of methods and improvements [56] [57]. In that scenario, each LoRa device in the network sends a black-and-white image of 900 to 1,200 bytes per hour to the LoRa gateway. To do that, each image is split into 4 or 5 separate packets and transmitted one by one. This is done to break the LoRa load limit of 222 bytes. Pham also proposes a CSMA mechanism for LoRa networks to mitigate the collision of packets in the network. The proposed CSMA mechanism is inspired by the IEEE 802.11 CSMA mechanism, in which a node can transmit only after the channel has been sensed idle for a minimum time period. The performance evaluation results show that the proposed CSMA mechanism can effectively reduce packet collisions in LoRa networks for both short and long messages. Moreover, the proposed CSMA mechanism is more energy efficient than the CSMA mechanism from IEEE 802.11.

Jebril *et al.* used LoRa as the main infrastructure for mangrove monitoring through image transmission and conducted a comprehensive outdoor experiment in Sabak Bernam, Malaysia [33]. Similar to what Pham did, Jebril's approach is to split each image into separate packets and send each packet in turn. The difference is that a transmission command packet is sent before the data transmission and the sender needs to wait for the receiver to reply with the acknowledgement packet. The transmission task will not start until the sender successfully receives the acknowledgement packet, otherwise the task will be delayed. The benefit of it is to prevent unplanned data transmission before the receiver is ready to receive, thereby avoiding packet dropping and power wasting. They used the self-built LoRa devices to evaluate the proposed mechanism with point-to-point transmission from 1 kilometre to 7 kilometres at various locations in Sabak Bernam, Malaysia. The results show that the image transmission of 300 packets through LoRa is feasible at a distance of 1 kilometres to 7 kilometres. The transmission time varies from 1 minute to 14 minutes, depending on the value of the spreading factor used. When the distance is less than 5 kilometres, there is no significant difference in the packet loss rate among different spreading factors. However, when the transmission distance is greater than 5 kilometres, the packet loss rate of SF 7, SF 8 and SF 9 increases significantly, while the use of other spreading factors has seen only a modest increase.

Ji *et al.* implemented image transmission in a LoRa-based visual monitoring scheme for agriculture IoT system by segmenting the collected image and sending only the changed image area [35]. It is difficult

for LoRa to transmit full images frequently, since LoRa's data rate is very small. They noticed that in their agricultural scenario, the image changes very little over the course of the day. Therefore, instead of transferring the full image, only the changed image area needs to be transmitted. In order to reduce the computational complexity and energy consumption, their method is to divide the image into 256 grid patches. By comparing the corresponding grid patches in the two images, the change areas are calculated and sent. The experimental results show that only 24% of the 2820 experimental cycles actually performed transmission. During these transmission cycles, an average of 14.47 patches were sent per cycle, with an average transmission time of 5.22 seconds and a throughput of about 2.212 kbps.

2.3 Reliable Transport Mechanism

When multiple packets need to be sent continuously, there is no guarantee of the successful arrival of data, the order of arrival and the correctness of packets. Therefore, reliable transport protocols are needed to solve these problems. This section describes the reliable transport methods and protocols commonly used in Internet. Although they cannot be directly applied to WSNs, they bring useful insights into the proposed protocols.

2.3.1 Automatic Repeat Request

A protocol where the sender waits for confirmation before moving on to the next data item is often called automatic repeat request (ARQ) [76]. In other words, ARQ is used as an error-control technique for data transmission by using acknowledgements. The simplest ARQ scheme is the *stop-and-wait* algorithm, which waits for an acknowledgement for every packet (window size = 1) [76].

2.3.2 Block Acknowledgement

Block acknowledgement was first invented and applied in IEEE 802.11 wireless LANs (WLANs) to improve network communication quality by reducing the amount of overhead caused by acknowledgement packet [31] [59]. In simple terms, the block acknowledgement mechanism aggregates multiple acknowledgements into a single packet and therefore acknowledges a block of data packets by one single acknowledgement [31]. Such a mechanism reduces the network utilization and waiting time for acknowledgements, and improves channel efficiency. Based on network conditions, the block acknowledgement mechanism is also divided into two categories, namely immediate acknowledgement and delayed acknowledgement. The difference between the two is that the former sends the block acknowledgement packet immediately after the data transmission. In the latter case, after the end of data transmission, the receiver needs to confirm the completion of data reception with the sender before sending the block acknowledgement packet. In a network with high bandwidth and low latency, immediate block acknowledgement is more suitable. In moderate latency applications, the delay block acknowledgement is preferred [10] [31].

2.4 Network Topologies in WSNs

A network topology is a mapping of a communications network that defines how network devices are arranged and how information flows through the network [21]. The medium of wireless sensor networks sends data in the form of broadcast, thus devices in WSNs are connected via a software-level topology. This section introduces and analyzes five common network topologies in WSNs.

2.4.1 Bus Topology

In a bus topology, all nodes are connected to one single frequency channel and considered as peers. During communication, packets are broadcast over the network and are visible to other nodes in the same network, while only the intended receiver accepts and processes packets [21] [69]. Thus, the bus topology is realized by pure wireless broadcasting. Each node in the bus network can reach any node in the same network by specifying the receiver's unique address in the packet. Figure 2.3 shows an example of a bus topology that communicates on channel #1. Note that the dotted links between nodes are wireless links.

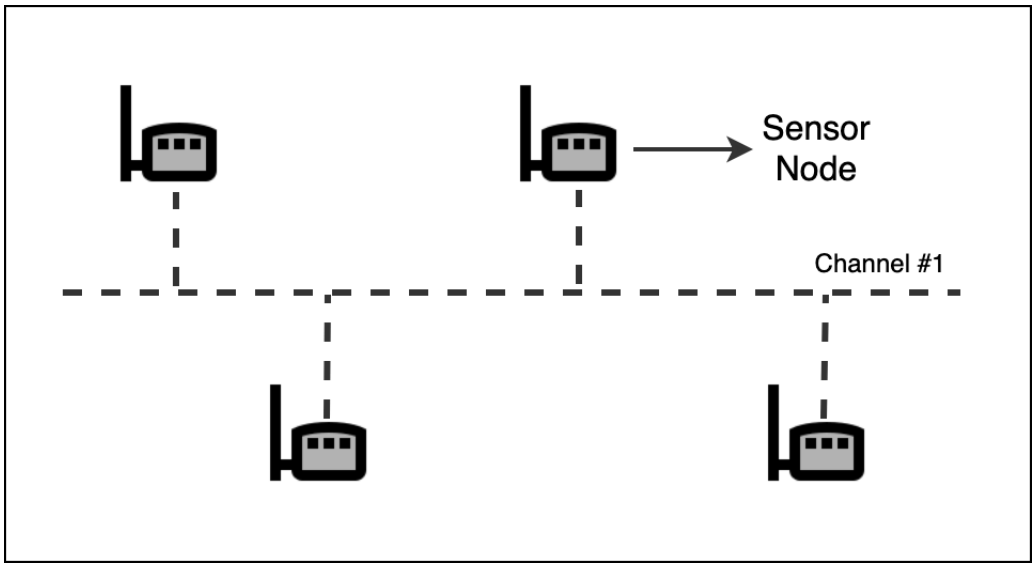


Figure 2.3: An Example of Bus Topology[6]

The bus topology is relatively easy to deploy because of the broadcast nature of the wireless spectrum. However, in the case of a large number of nodes or high communication duty, the bus network could experience serious network congestion. Therefore, with fewer nodes, the bus network has less congestion and each node can experience good performance [69]. Due to the single-hop communication between nodes in the bus network, the physical scope of the network is limited by the physical layer wireless media and cannot be further extended by software. Therefore, the bus topology is not suitable for wide area networks. Also, in

[6]Recreated From Sharma *et al.* [69].

the scenario of this thesis, the bus topology is not an appropriate method due to the deterministic data flow and high duty-cycle brought by image transmission.

2.4.2 Star Topology

The star topology is the standard topology used by the LoRaWAN protocol [11]. In a star topology, each node is connected to a central point called a gateway or sink point, and all packets are aggregated directly from each node to the gateway without any coordination between the nodes [21] [69]. Figure 2.4 shows an example of a star topology.

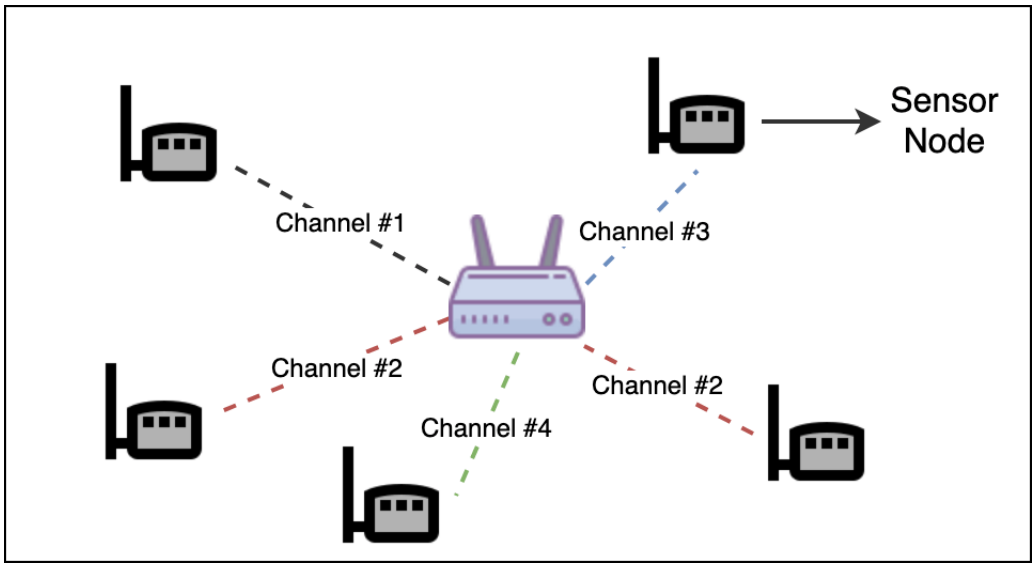


Figure 2.4: An Example of Star Topology[7]

Like a bus network, nodes and gateways must be on the same frequency channel to communicate. In a star topology, however the gateway can control the time and duration of node access through the appropriate MAC protocol. A gateway with multi-channel functionality, such as the LoRa-based SX1301 chip, can listen on 8 different channels at the same time, and each node in the network will be assigned a random frequency channel to communicate with the gateway, so as to alleviate traffic congestion.

Although star topologies can alleviate traffic congestion by using appropriate MAC protocols, star topologies are still limited by single-hop communication and cannot further extend the network range. Therefore, the star topology does not directly apply to the scenario in this thesis.

2.4.3 Chain Topology

The chain topology, also known as the line topology, is used to transmit packets from one end to the other through a series of nodes. The nodes in the chain topology communicate in a multi-hop fashion. Nodes in

[7]Recreated From Sharma *et al.* [69].

the chain topology are connected to two adjacent nodes, except for the first and last one [13]. Figure 2.5 shows an example of a chain network. As shown in the figure, a node on one side of the network needs to relay through three other nodes to send the packet to the gateway. Communication channels between any two nodes can be fixed or dynamically assigned. Under normal circumstances, different channels will be used as much as possible to reduce interference between flows.

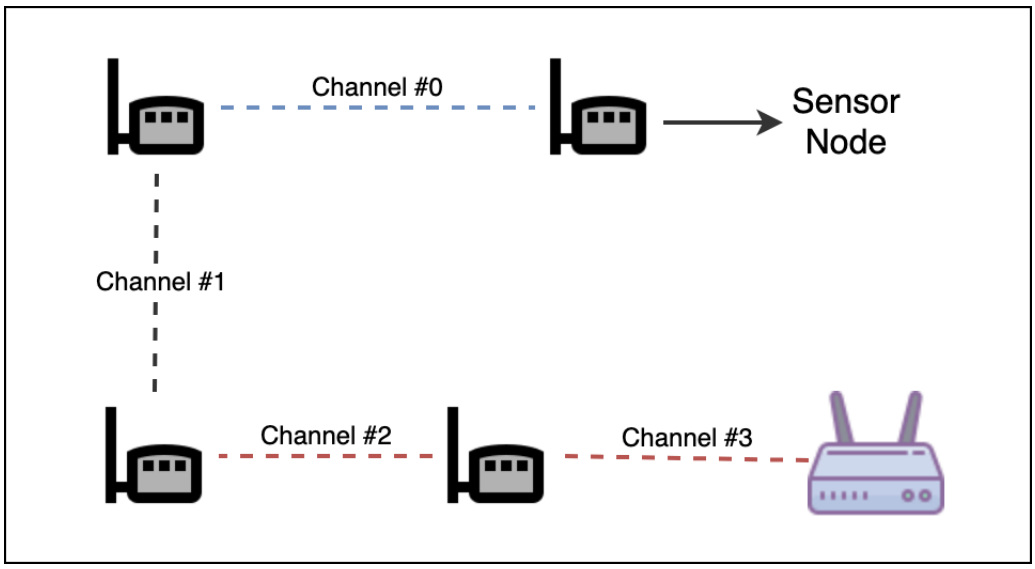


Figure 2.5: An Example of Chain Topology

The primary use of the chain topology is to extend the coverage of networks and to compensate for the limited signal range of individual physical medium. For example, when a wireless physical medium with a signal range of only about 100 metres is needed to send a packet to a receiving point 500 metres apart, four additional devices can be placed every 100 metres between these two devices to implement the chain topology and relay the packet. However, relaying itself brings additional overhead, resulting in decreased network throughput.

2.4.4 Mesh Topology

The mesh topology is another common method to solve the single-hop communication problem mentioned in bus and star topology. In a star topology, similarly, a central hub is selected as the data aggregation point of the network, with each node directly or indirectly connected to other nodes [21] [69]. Packets are transmitted directly between nodes that have direct contact with the gateway, while for nodes that have indirect contact with the gateway, packets are transmitted through relay devices and eventually routed to their destination.

As shown in Figure 2.6, packets have multiple paths to choose from to send a packet from source to destination. According to the stability and congestion of each route at run-time, the packet is sent to the destination through the optimal path by routing protocol. This approach provides robust network connections and allows topology changes. Especially in the WSN where the nodes frequently leave and rejoin the network,

the use of mesh topology can dynamically solve the optimal routing problem from the node to the gateway.

Mesh topology is suitable for self-managed networks with frequent topology changes. In particular, dynamic routing queries can cause additional latency and power consumption. The scenario in this thesis is a relatively stable hybrid network topology, which in most cases does not change except when the battery is depleted and the nodes are disconnected. Therefore, an appropriate network topology should provide multi-hop transport just like a mesh topology, but without the need for dynamic routing queries.

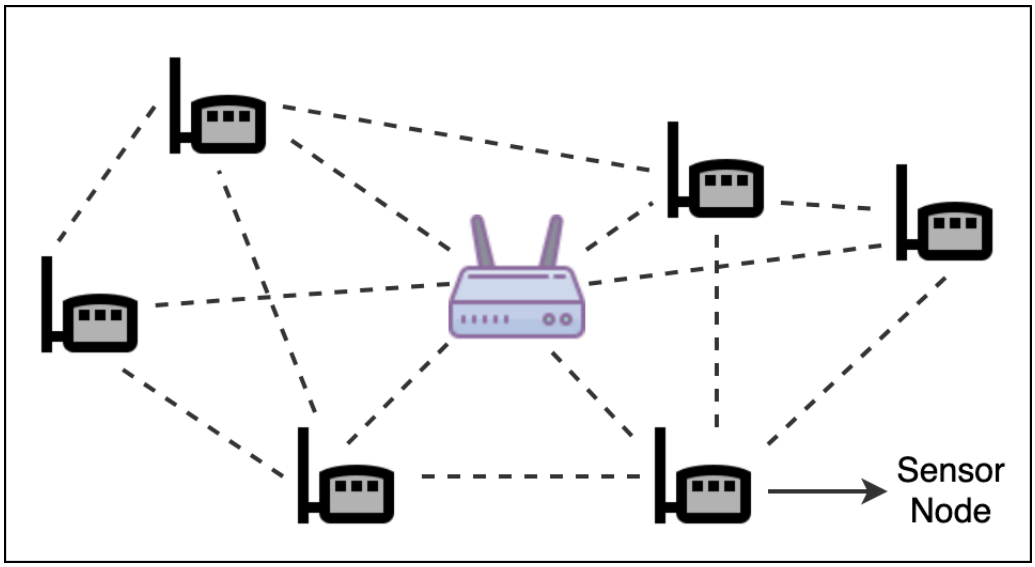


Figure 2.6: An Example of Mesh Topology[8]

2.4.5 Tree/Star-to-star Topology

A tree topology is a subset routing structure of a mesh topology and is a combination of bus and star networks [51] [69], as shown in Figure 2.7. This is because a tree topology supports both the use of multiple hops to transmit data across the network and the ability for multiple nodes to connect to a parent node. At the same time, a tree topology does not need to connect all adjacent nodes as a mesh topology does. A tree topology can also be viewed as a collection of star networks arranged in a hierarchy [79]. Therefore, it is also known as star-to-star (expanded star) topology. The construction of the tree topology occurs during the configuration step and the topology is adjusted periodically.

In a tree topology, periodic topology adjustment is typically used instead of proactive routing query, so that the nodes in the network can save electricity as much as possible to extend the lifetime of the whole network. There is only one sink point in the tree topology, which is the root node. All other nodes eventually need to send collected data to the root node. Combined with the above concepts and advantages, the tree topology is very consistent with the scenarios in this thesis.

[8]Recreated From Sharma *et al.* [69].

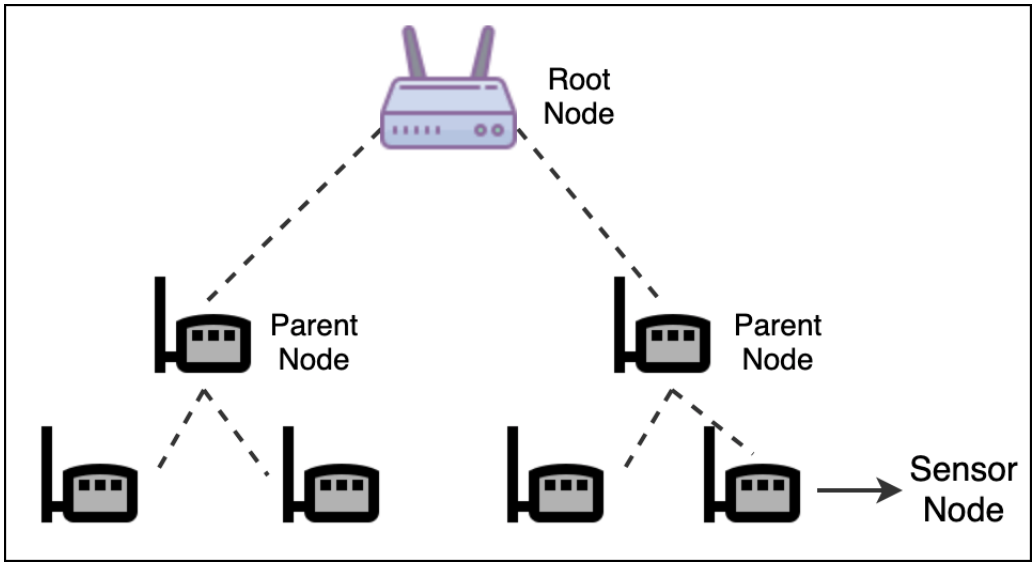


Figure 2.7: An Example of Tree Topology[9]

2.5 Multi-hop Routing

The routing protocol is the core of multi-hop communication, which defines how each packet in the network can be sent from source to destination in an efficient way. The main consideration is the next hop to which the packet should be forwarded on the path towards the destination. A well-designed routing protocol should not only consider the transmission rate of packets, but also consider the power consumption and network congestion. Thus, routing protocol plays an important role in the network with multi-hop communication. The following sections describe several routing protocols for WSNs and LoRa.

2.5.1 Routing Protocols for WSNs

There are three types of routing protocols in WSNs, namely proactive routing, reactive routing, and hybrid routing [40]. According to the requirements and goals of wireless sensor networks, various routing protocols are explicitly used. It is worth mentioning that there is no routing protocol that can fulfill all the requirements of all WSNs, since each routing protocol is optimized for specific requirements. The important characteristics of each routing protocol type are described below.

Proactive Routing Protocols

Table-driven routing takes the proactive approach. These routing protocols maintain a list of up-to-date routing information, and periodically distribute the routing information throughout the network [40]. Any packet relay is done immediately without delay, unless there is no route to the destination in the maintained

[9]Recreated From Sharma *et al.* [69].

table. However, the most significant disadvantages of table-driven routing are considerable storage for routes and slow reaction to topology/network changes [38]. Therefore, this approach is not recommended when the storage of nodes is limited or the network size is relatively large.

Reactive Routing Protocols

Reactive routing, on the other hand, finds routing information on demand by broadcasting and flooding route request packets [40]. In other words, there are no routes in a node's storage and the most up-to-date route/next hop is determined before the packet is sent. Because routing requests are real-time, reactive routing cannot avoid the high latency on route finding and request flooding can also cause network congestion.

Hybrid Routing Protocols

Hybrid routing combines the advantages of both proactive and reactive routing. It typically initiates the network with a proactive strategy and then reactively serves newly activated nodes [40]. The advantage of this is that the network construction of proactive strategy can form the network topology faster, and then the reactive topology adjustment can respond to the change of topology on the premise of saving electricity as much as possible. However, the performance of this routing protocol is affected by the number of newly activated nodes and the traffic volume.

2.5.2 Routing Protocol for LoRa

Several routing protocols for LoRa have been proposed in previous research work. Liao *et al.* proposed a multi-hop LoRa wireless network using concurrent transmission [44]. Concurrent transmission is a flooding based routing protocol, that allows multiple nodes to transmit the same packet simultaneously and considers synchronized packet collisions when multiple relays perform retransmissions at the same time, in order to enable simple and fast back-to-back packet relaying. Such quick flooding is sufficient for transmitting a small amount data efficiently. However, when the size of packet and duty-cycle is increased, its advantages are less obvious, and the extra overhead of flooding would often congest the network. In fact, flooding-based routing protocol is not suitable in the scenarios in the thesis.

Sartori *et al.* suggested RPL, a IPv6 based routing protocol for low-power and lossy networks [65]. Due to its IPv6-based connectivity, RPL has a remarkable performance of bi-directional communication and compatibility with existing IP based protocols. Regarding LoRa's limited packet payload length, adding the IP header will take 40 bytes from packet payload length (a 17.16% reduction in payload). Even if packet segmentation is employed, more packets need to be transmitted in each reception window. In order to send relatively large data objects via LoRa, the protocol design must allocate as much payload as possible to reduce the number of packet segments and thus the overhead of transmission.

Lundell *et al.* proposed a new routing protocol based on hybrid wireless mesh protocol and ad-hoc on-demand distance vector (AODV) routing [46]. Similar to AODV, a Route Request packet will be broadcast

every time when a device has a data packet to send, and the device will receive a reply packet that contains an explicit route. Such run-time route inquiry is protected against topology changes in ad-hoc mobile network. On the contrary, dynamic route inquiry creates a lot of overhead when the network topology is constant in most cases. In stationary sensor networks, routes from each source device to the respective destinations need to be calculated during network initialization. These routes are often optimal and do not need to be re-calculated if no devices enter or leave the network. When a device enters or leaves the network, routes can be re-calculated on demand. Such passive routing update is acceptable for a wireless sensor network where slight delay is tolerable. Thus, proactive routing protocols are not preferred in this scenario.

Moreover, the operational spreading factor and the bandwidth in all these proposed approaches are either pre-scheduled, or randomly matched during the run-time. From the literature search conducted, there is no such study of routing protocol for LoRa wireless network that has utilized spectrum opportunities to enhance the routing. This makes it difficult for the routing protocol to estimate the transmission time and cost of the route accurately by detecting the quality of the link, and give appropriate transmission settings and route selection.

Efficient Tree-Based Self-Organizing Protocol for IoT

Qiu *et al.* propose an efficient tree-based self-organizing protocol for sensor networks in IoT systems [60]. Such a protocol can construct a reliable tree-based network quickly by broadcasting packets to allow newly added nodes to join the network selectively. Each newly added node chooses how to join the tree topology by measuring the number of child nodes, hop, communication distance, and residual energy. The simulation results show that the proposed protocol has a shorter construction time, and the performance of success rate of the packet is much higher than both AODV and destination-sequenced distance-vector (DSDV) routing, and the network lifetime is much longer than the network using DSDV routing.

2.6 Machine Learning for Routing Selection

Routing selection is critical in routing protocols. The optimal path to deliver packets is selected by analyzing and judging each feasible path. In the past, through the analysis and comparison of the routes, complex metrics are developed to allow each node to calculate either a faster or more stable path to a destination. However, such an approach not only requires much effort to analyze and compare each possible routing situation, but also needs to examine the impact coefficient of each selected factor. With the rapid development of machine learning in recent years, similar problems can be effectively solved by machine learning algorithms.

2.6.1 Types of Machine Learning Systems

Machine learning consists of several different types of systems for different methods of learning. Depending on the type, size, and integrity of the data set, various machine learning systems can be used independently or

collaboratively. Described below are four common learning systems typically used for classification problems, in which the problem is to determine which category a new instance belongs to based on training data for a set of instances that contain known category members.

Supervised Learning

In supervised learning, desired solutions are included in the training data that one provided to the algorithm [23]. In other words, both features and solutions are learned in supervised learning to predict the new data instance. Classification is a typical supervised learning task which learning the features from a known instance and its category and predict the classes to which the unknown data instance belongs [23].

Unsupervised Learning

Unlike supervised learning, solutions are not provided to the algorithm via the training data, and so the system needs to learn without a teacher [23]. Unsupervised learning is commonly used in clustering task which clusters the instance based on their features without known solutions.

Batch Learning

In batch learning, all training is done with the entire training data set before inference, and without the ability of incremental learning [23]. Such training is usually done before deployment of the system to infer category membership since it will take a fairly long time and use a large amount computing resources. Sometimes days are needed to complete training on cluster of computers [70]. The trained system has to be re-trained using all training data when introducing new training data to the system.

Online Learning

Online learning is a general way to support incremental learning, where the model is updated by learning mini-batches of data sequentially [23]. In many cases, after the training data is learned, the machine learning model needs to be updated incrementally based on the real-time data. At this point, online learning is needed to quickly or autonomously adapt to changes in the system.

2.6.2 Common Supervised Learning Methods for Classification

Learning methods and algorithms for supervised classification have been widely studied in recent years. Here are five classification learning methods and algorithms that will be considered in this thesis. All of these methods and algorithms support incremental learning and are implemented in the library used.

- Multi-Layer Perceptron (MLP) is a feed-forward artificial neural network model which maps the input data set to the appropriate output data set [4]. It consists of a number of units organised into multiple layers. It consists of many cells organized into multiple layers. The first is the data input layer, the last

is the data output layer, and the remaining are the hidden layer(s). Three different algorithms can be used to implement the MLP model, namely the back propagation, the delta rule, and the perceptron [4]. The MLP model adopted in this thesis is implemented by the method of back propagation.

- Stochastic Gradient Descent (SGD) is an iterative method for optimizing objective functions with appropriate smoothness [50]. This method uses stochastic selection of samples to evaluate the gradient, so it can be regarded as a stochastic approximation of gradient descent optimization. This method can be used to train classifier based on linear support vector machine. Therefore, the SGD method used in this thesis adopts linear support vector machine by default.

- Decision Tree is one of the most popular approaches for classification and prediction [63] [75]. It classifies data in a tree structure, where each node is either a leaf (result) node or a decision node. The former is a class of instances, while the latter is some test performed on a single value, with a branch and sub-tree for each possible result of the test. Decision tree structures typically work from the top down, best separating a set of projects by selecting a criterion at each step [63]. The criteria used in this thesis is *Gini*, which is an impurity-based criteria that measured the divergence between the probability distributions of the target attribute's values [63]. In other words, it is the probability of incorrectly classifying a randomly chosen element in the data set if it was randomly labeled according to the distribution of labels in the subset [84].

- Random forest is an ensemble learning method of classification, which constructs a large number of decision trees during training, and the output class is determined by the mode of the output class of individual trees [26] [27]. It combines the bootstrap aggregating meta-algorithm to improve the stability and accuracy, and avoided over-fitting by reducing the variance of results. Since it is based on a set of decision trees, the criteria for random forest used in this thesis is also *Gini*.

- Ada Boost, short for Adaptive Boosting, is also an ensemble learning method of classification. Unlike in a random forest, where each decision tree has only one vote for the final result, Ada Boost combines the output of a series of weak learners (decision tree, for instance) into a weighted sum to represent the final output of the enhanced classifier [17]. It is adaptive because subsequent weak learners are adjusted to support instances that have been misclassified by previous classifiers.

2.6.3 Relevant Studies and Applications

Several studies that have been conducted to use machine learning algorithms to solve routing problems. Zhao *et al.* used support vector machine, a supervised learning model that used for classification, to process the vehicle data and generate routing metrics to enhance the effect of the features and the performance of the greedy perimeter stateless routing (GPSR) algorithm in vehicular ad-hoc networks [82]. Through the training of a large number of classified data and the simulation tests, it is proved that the proposed method has less packet loss and network delay compared to GPSR.

Hooda *et al.* developed a multi-criteria decision analysis method based ensemble framework using a hybridization of machine learning classifiers for the prediction of an optimal routing protocol for wireless sensor networks [28]. The performance of the proposed framework was validated with six different data sets from the machine learning data repository provided by University of California Irvine. The experimental results prove that the accuracy of the proposed ensemble classifier is better than other selected classifiers in all test data sets.

Hu and Fei proposed an adaptive and energy-efficient routing protocol for underwater delay tolerant networks (DTNs) using a reinforcement learning technique, called Q-learning [30]. By considering communication distance, neighbour density, and residual energy, they constructed a new reward function to evaluate the gain or cost of the system for the machine learning model to learn. The simulation results show that the proposed protocol has significant advantages compared to other existing routing protocols in terms of energy efficiency, end-to-end delay, delivery rate, and storage overhead.

Khan *et al.* presented an efficient and energy-aware clustering protocol using support vector machine [39]. Compared to the LEACH protocol [24], their proposed protocol has a more extended network lifetime in NS2 simulation, and is more power efficient in large-scale WSNs.

Vinutha *et al.* designed a non-uniform sampling and reliable routing for efficient energy saving and high-speed data communication using back propagation neural network [78]. Such routing can significantly reduce the control signal overhead charges and enhances the reliability in packet transmission by predicting energy robust and near-by nodes. The simulation results show that battery energy consumption is improved without the trade-off of the speed of data communication.

2.6.4 Result Scoring

After the machine learning model training is completed, the learning/training results should also be evaluated to see if the training is successful and effective. There are three standard evaluation methods to quickly understand the results of learning. Here are brief descriptions of the three approaches:

- Classification Accuracy measures the quality of the predictions given a test set and the corresponding labels in the case of supervised learning algorithms [23]. To be precise, this score is the mean accuracy that the model predicts for a given set of tests and labels.

- Cross-Validation splits the training set into complementary subsets and each model is trained against a different combination of these subsets and validated against the remaining parts [23]. This verification method is helpful to understand the accuracy of prediction of unknown data instances after learning the known data instances.

- Classification Report displays the precision, recall, f1, and support scores for the model. Precision is the ability of a classifier not to label an instance positive that is actually negative, while recall is the ability of a classifier to find all positive instance. F1 score is a weighted harmonic mean of precision

and recall, such that 1.0 is the best and 0.0 is the worst. Support is the number of actual occurrences of the class in the specified data set. The detailed calculation of precision, recall, and f1 will be given in Chapter 5.

- Receiver operating characteristic curve is a common tool used with binary classifiers, which plots the true positive rate against the false positive rate [23]. Through the analysis of two rates, we can further understand the correct prediction rate in samples with positive prediction and in samples with positive labels, respectively.
- Learning Curve is an important visualization tool to show the relationship of the training score vs. the cross validated test score [54]. It can show how much the model benefits from more data, and if the model is more sensitive to error due to variance or bias.

2.7 Summary

This chapter discusses some of the basics of LoRa, and some of the relative protocols and work involved. Firstly, the feasibility of the image transmission in LoRa network is preliminarily proved. Secondly, some common network topologies and multi-hop network protocols in WSNs are introduced, along with some relative studies on applying machine learning in routing problem. At last, five kinds of most commonly used classification learning methods and algorithms is explained, as well as the methods of judging and evaluating learning results are discussed.

Chapter 3

Multi-Packet LoRa Reliable Protocol (MPLR)

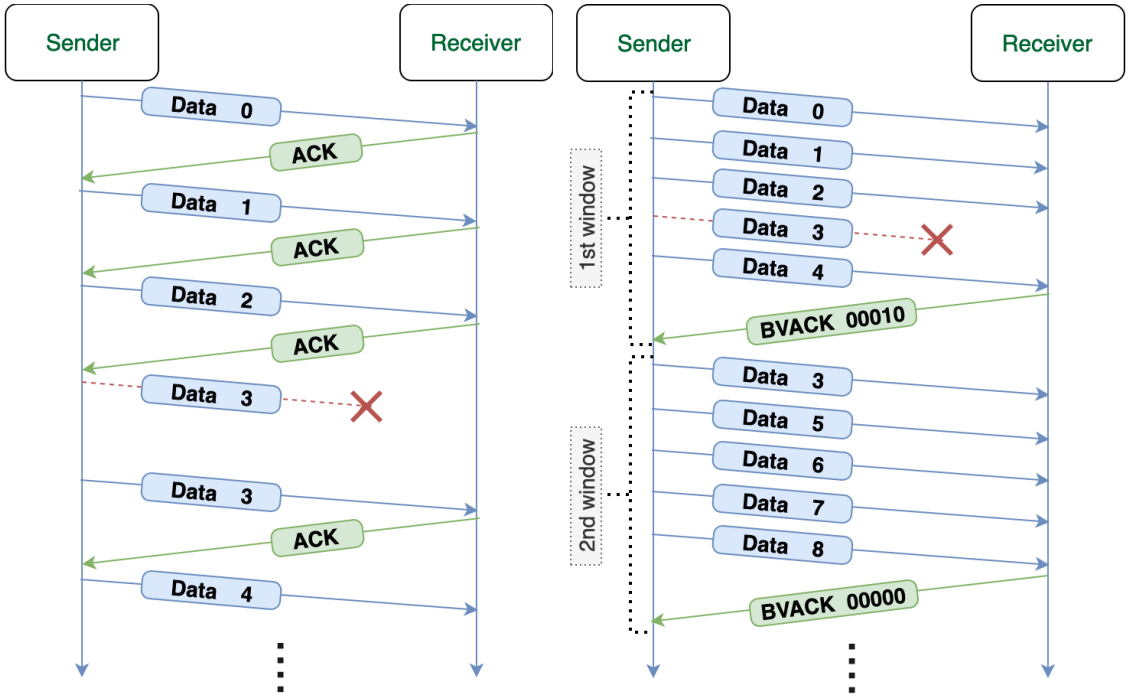
A new reliable and efficient LoRa transmission protocol, named MPLR, is designed and developed to enrich the image data transmission in LoRa network. This chapter discusses the details of the design.

3.1 Design of MPLR

Even a highly compressed image is too big to fit in a single LoRa packet, given its maximum transmission unit (MTU) of 255 bytes. Therefore, an image must be transmitted using multiple packets. A 9 KB image, for example, needs to be encoded and segmented into at least 48 MTU packets before being transmitted through LoRa. If the stop-and-wait method is used in transmission, as shown in Figure 3.1a, the sender waits for an acknowledgement per packet to ensure that the data arrives correctly and thus the packet delivery rate is severely limited. Also, this increases the power consumption of both sender and receiver, and the acknowledgement traffic increases the network load and the required transmission rate of the receiver.

The Internet transport protocols, transmission control protocol (TCP) and user datagram protocol (UDP), have been widely used in all kinds of networks, including WSN. UDP does not provide reliable transmission; its performance in terms of reliability is dependent on the network infrastructure or application. However, due to the unstable connectivity in WSN, a reliable transport protocol is required to ensure successful transmissions. Thus, UDP alone is not suitable in this scenario. TCP is the transport protocol which provides guaranteed reliability [58] [42]. TCP is also frequently used in IP-based WSNs, such as Zigbee and 6LoWPAN network. However, TCP is too heavyweight for a LoRa wireless network to use. First, the TCP packet header occupies at least 20 bytes. Due to the small payload of LoRa, the 20-byte reduction will have a noticeable impact on LoRa wireless network throughput. Second, TCP relies on the IP protocol. In other words, the use of the TCP also requires the IP protocol to be enabled. As a result, the available payload will be further reduced. Therefore, TCP is also not suitable for LoRa wireless network.

One objective in designing a new reliable delivery protocol is to reduce the number of acknowledgements that need to be sent and the cumulative time spent waiting for them. For this purpose, a new protocol based on batched packet transmissions and bit vector acknowledgements is designed and proposed. As shown in Figure 3.1b, according to the size of the transport window, the sender sends a batch of data packets to the receiver consecutively. The correctness of each packet is verified through the forward error correction in the



(a) Stop-and-Wait **(b)** MPLR

Figure 3.1: Comparison of Packet Transmission Protocols

LoRa physical layer and the checksum in the protocol header. The status of each packet delivery is returned to the source via a bit vector acknowledgement packet (BVACK) that contains a bit for every packet in the batch, which is inspired by the block acknowledgement from IEEE 802.11 [31]. If the bit in a particular index position is 0, the corresponding packet was received, and 1 if not received. Based on this, the sender can determine which data packets are lost/corrupted. Note that since LoRa employs half-duplex communication, ACKs or NACKs cannot be returned concurrently with data packet transmissions by the sender.

Due to the limited packet payload size in LoRa, the implementation of MPLR uses a lightweight packet format. The packet format is shown in Figure 3.2. The packet format consists of the header and data payload. The header has a total of 16 bytes, namely *Destination EUI* (4 bytes), *Source EUI* (4 bytes), *Service number* (1 byte), *Sequence number* (2 bytes), *Flag* (1 byte), *Payload Size* (1 byte), *Batch Size* (1 byte), and *Checksum* (2 bytes). *Destination EUI* and *Source EUI* are the unique identifiers of destination and source device, respectively. The *Service number* identifies the service to which the data packet belongs. The *Sequence number* is used to order the packets. The *Flag* field indicates the packet type. The possible values are *SYN, SYN-ACK, DATA, BVACK, FIN*, and *ACK*. *Payload Size* and *Batch Size* describe the length of the data payload and the size of the current batch, respectively. Finally, *Checksum* is used to check the correctness of the header. The data payload portion still has a space of up to 239 bytes. This header covers the functionalities of both data link and transport layers. The overhead of this 16-byte header is only 6.3%.

Figure 3.3 illustrates the MPLR protocol for single-hop data transmissions. Similar to many other reliable

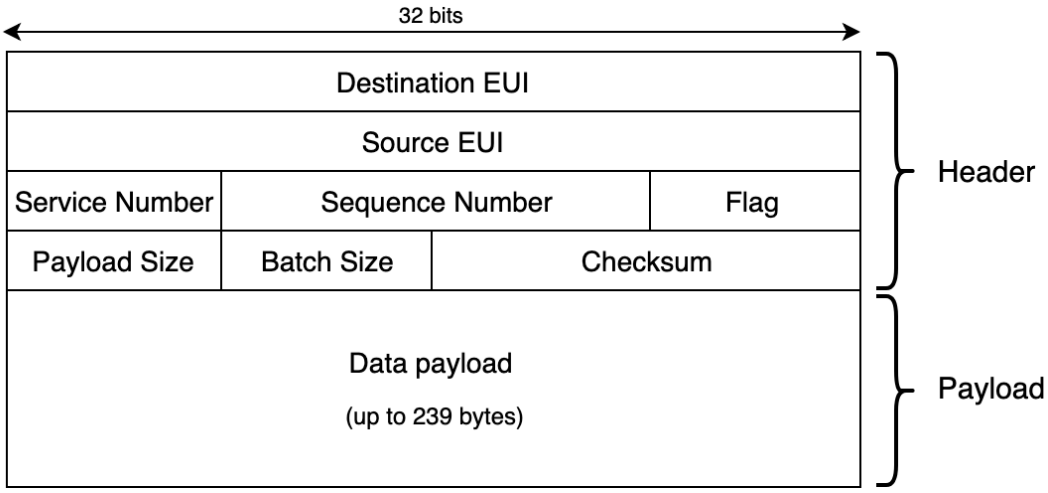


Figure 3.2: MPLR Message Format

communication protocols, a connection is first established between the sender and receiver. Then, the sender sends all the packets in the first batch, and waits for a BVACK from the receiver. By default, the batch size is 40, which is enough to send a 9 KB image in one batch if we use a payload of 239 bytes, and can be increased or decreased as needed. The sender then transmits the next batch of packets, including, if necessary, retransmissions of packets not correctly received from the previous batch. When all data packets have been sent, the nodes will perform a 4-way handshake connection termination through the FIN and ACK packets to ensure that they have dropped the connection and complete the current transmission task. This connection termination mechanism is inspired by the connection release mechanism in TCP, and is considered to be effective in preventing inconsistency of coordination [76]. The receiver will then return to *LISTEN* mode and wait for the next connection request. The proposed connection and termination method is similar to that used in TCP, but establishment is simplified as the image data is up-link only.

3.1.1 Data Channel Reservation

Although using MPLR can efficiently reduce the image transmission time, it still takes about 9.6 seconds to transmit a 12 KB image (with SF 7/BW 250), according to LoRa's physical layer data rate in Table 2.1. Compared to sensor data transmission of only a few bytes per message, LoRa-based image transmission has a higher duty cycle and potential for congestion when there are multiple nodes using the same gateway.

When sending data packets using the stop-and-wait protocol, collisions due to congestion from data packets from other devices will affect the goodput of the transmission (rate of correctly received packets). With MPLR, only request packets from other nodes could possibly interfere with the data transmission of a node that has established a connection with the gateway, assuming that the gateway does not accept multiple concurrent connections on the same channel. It is possible, however, to eliminate even these collisions by using a data (event) channel distinct from the control channel.

Therefore, it is beneficial to design and implement a data channel reservation protocol. The current

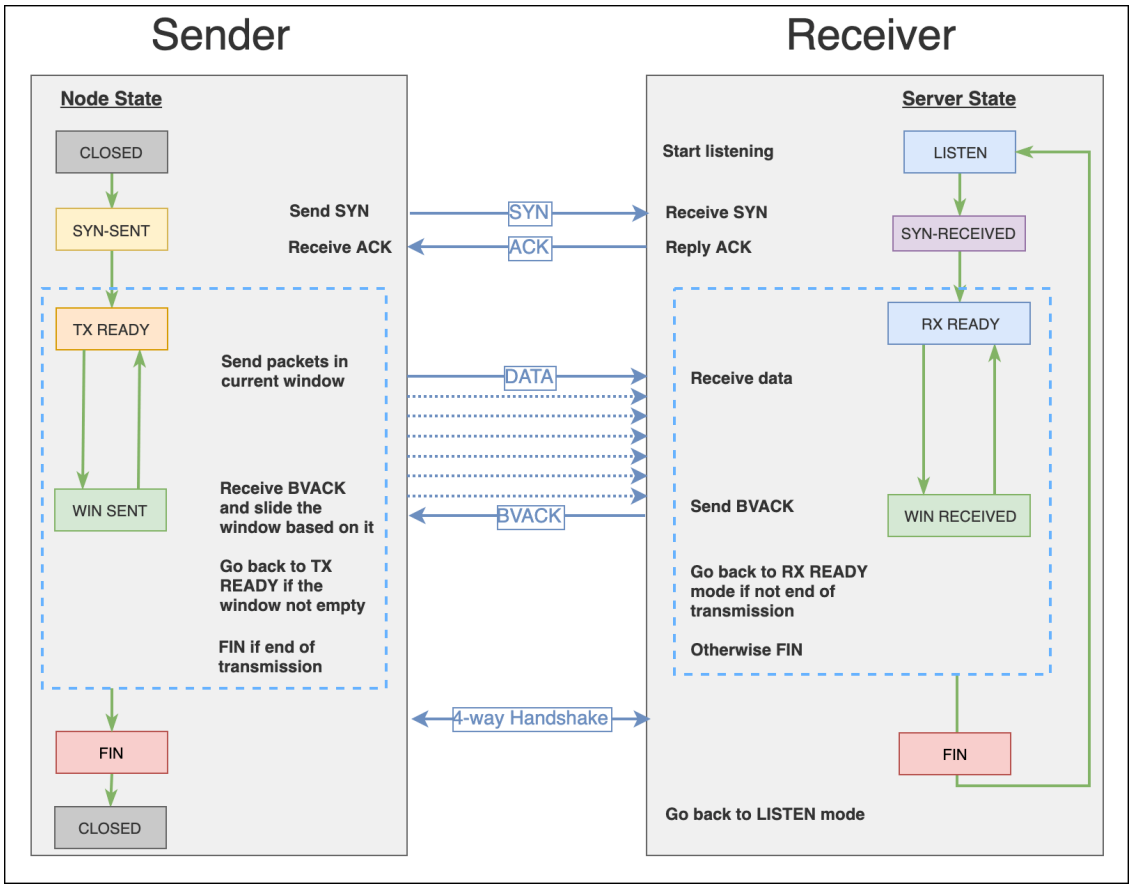


Figure 3.3: Connection Management: Sender/Receiver

gateway only supports listening on a single channel at once, and so the protocol need not support multiple active data channels. However, the approach is easily generalized for scenarios in which the gateway hardware supports concurrent use of multiple channels. Such a generalization is potential future work. As shown in Figure 3.4, an idle gateway continuously listens for request packets on the control channel. When the gateway receives a request packet, it pseudo-randomly selects a channel to use as the data channel and informs the requesting node of its choice.[1] Then both the gateway and the node will switch from the control channel to the target data channel and perform the transmission described in the previous section. When the transmission is complete, the gateway will return to the control channel and wait for a new request. Note that any new requests that are made on the control channel while the gateway is listening to the data channel will not be received because the control channel and the data channel are on two different frequencies and do not interfere with each other. Other devices will not send any packets on any channel other than the control channel until the gateway answers their request.

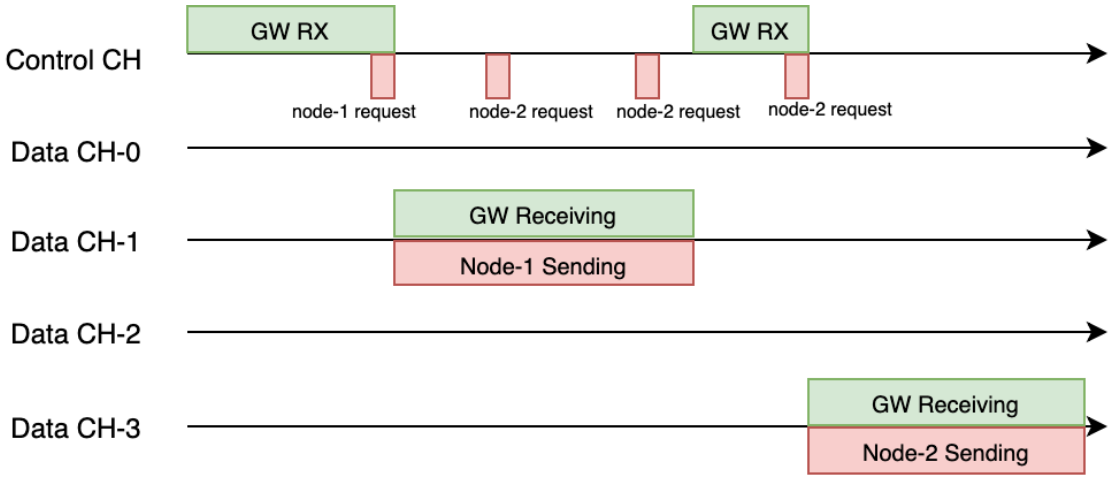


Figure 3.4: Data Channel Reservation Timing

The data channel selection is not made in an entirely random fashion, since the gateway will avoid choosing channels on which a high loss rate has been experienced. When a particular channel is in use as the data channel, the gateway monitors the packet loss rate on that channel. Such an approach can avoid the use of channels that may be used by other devices in the receiver range or have other reliability issues.

Figure 3.5 shows an example of data channel reservation between the LoRa node and the gateway. The node first sends a *SYN* packet to request connection establishment with the gateway on the control channel. If the request packet is not received by the gateway, the node will sleep for a random time before its next connection attempt. When the gateway receives the request packet, it assigns a data channel and indicates it in the acknowledgement packet. After the gateway replies, it will immediately switch to the assigned data channel and wait for the confirmation from the node. The node will also switch to the assigned data channel

[1] In scenarios where frequency hopping is employed, the "data channel" in our description here would actually correspond to a sequence of channels.

after receiving the *SYN-ACK* correctly and send an *ACK* as well.

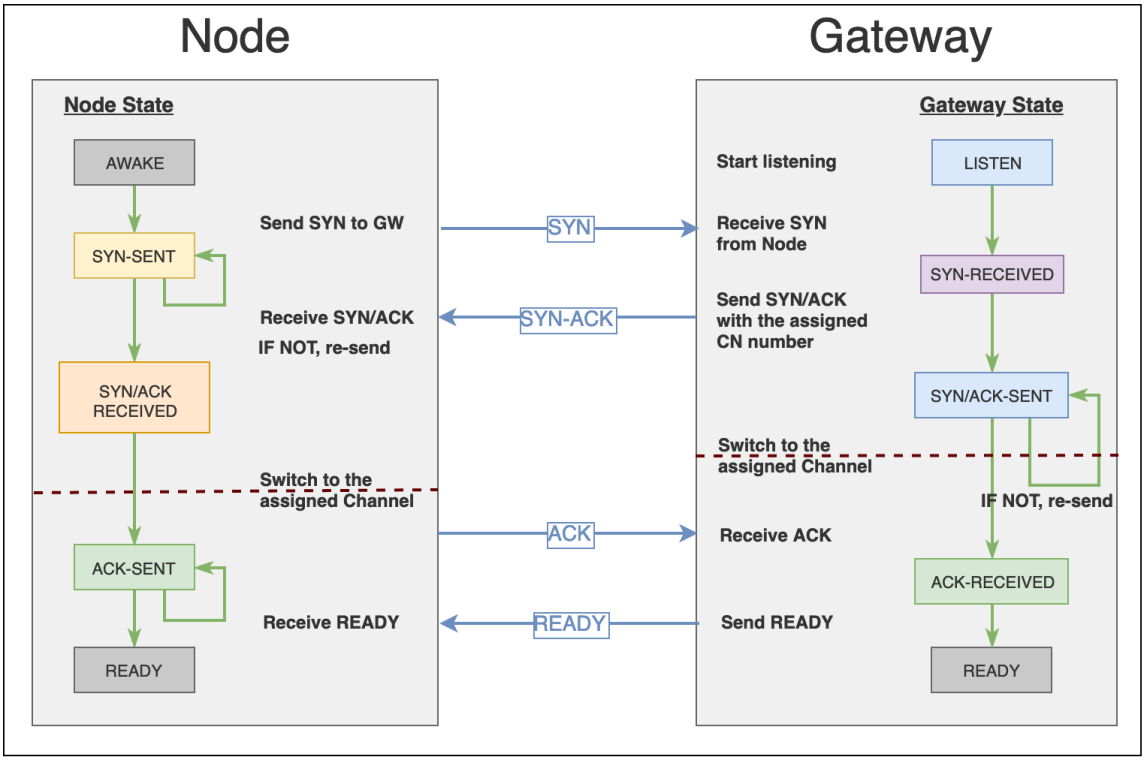


Figure 3.5: Data Channel Reservation: Node and Gateway

To ensure that the node does not begin transmission before the gateway is ready to begin receiving on the data channel, the gateway will send a *READY* packet to indicate that it is prepared to start receiving data. After the node receives the *READY* packet, it can begin the data transmission. The whole process is a 4-way handshake connection, but using two different channels.

3.1.2 Frequency Plan

In order to be compatible with sensor data transmission and image data transmission in the network and to avoid packet interference, channels in the frequency band should be planned in advance according to different bandwidths. All frequencies are divided into three categories in the proposed protocol: 125 KHz, 250 KHz, and 500 KHz bandwidth. Each category contains two groups of 16 available channels. The advantage of this is that when large amounts of data, such as images, are being transferred, the large bandwidth channels can be used. At the same time, it does not affect the transmission of small data over the same network, because small data items will be transmitted using channels with smaller bandwidth.

The specific frequency planning method is as follows: the reserved frequency of the control channel is 902 MHz to 903.5 MHz. These control channels will be used to respond to requests for three different bandwidth data transfer tasks. After 903.5 MHz, the guard band of 400 KHz is added to avoid interference of signals on data transmission channels. The frequency range from 903.9 MHz to 906.9 MHz is then divided into 16

channels with a width of 125 KHz, with a 75 KHz guard band between two channels. Similarly, frequencies from 910.3 MHZ to 916.3 MHZ and from 918.5 MHZ to 927.5 MHZ will also be divided into 16 channels with a width of 250 KHZ and 16 channels with a width of 500 KHZ, respectively. The 250 KHZ and 500 KHZ channels have 150 KHz and 100 KHz guard bands respectively. The specific plan of the channel is shown in Table 3.1.

Table 3.1: Frequency Plan in MPLR

BW 125		BW 250		BW 500	
903.9	905.5	910.3	913.5	918.5	923.3
904.1	905.7	910.7	913.9	919.1	923.9
904.3	905.9	911.1	914.3	919.7	924.5
904.5	906.1	911.5	914.7	920.3	925.1
904.7	906.3	911.9	915.1	920.9	925.7
904.9	906.5	912.3	915.5	921.5	926.3
905.1	906.7	912.7	915.9	922.1	926.9
905.3	906.9	913.1	916.3	922.7	927.5

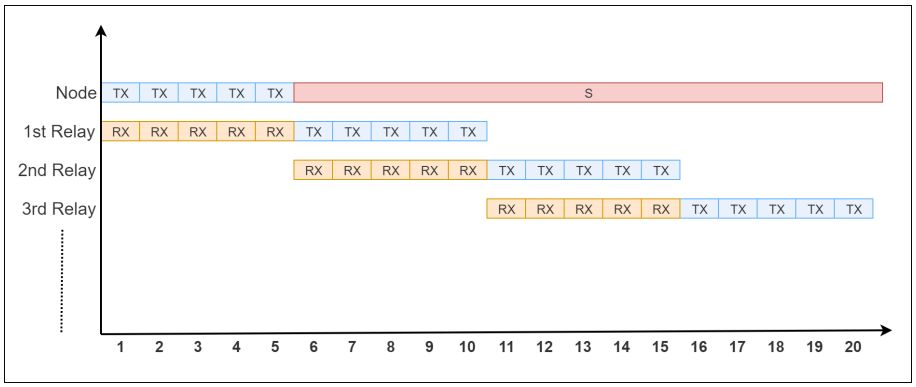
3.2 Design of Pipeline Transmission For Chain Network

There is more than one way to transfer multiple packets from a source node to a destination node. One possibility is to send all packets to the first relay before any retransmission. When all packets reach the first relay, it begins to retransmit. Another possibility is to retransmit each batch as soon as it reaches each relay point. The latter transmission mechanism can realize pipelining transmission in the network, so it is theoretically faster than the former one. The comparison between the two transport methods is shown in Figure 3.6.

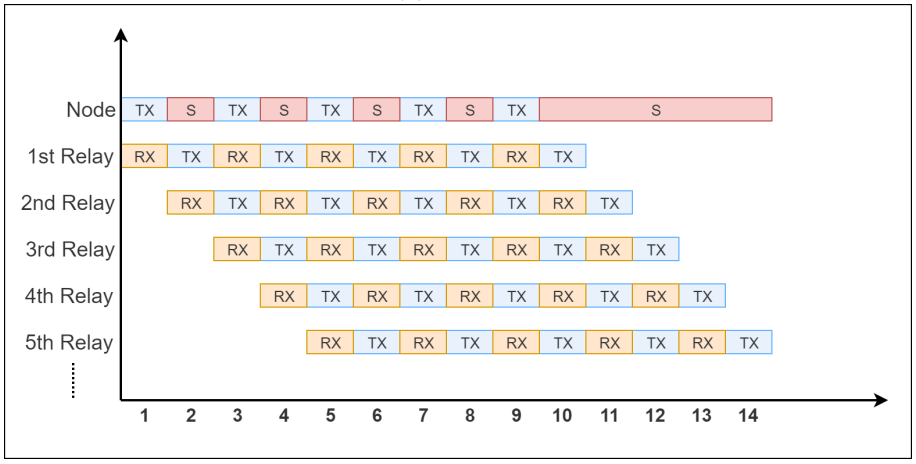
It can be clearly seen from Figure 3.6(a) that the third relay can receive all the batches at the earliest at the 15th time position and forward the first batch at the 16th time position without using pipelining transmission. In Figure 3.6(b), a third relay can receive all the batches at the 11th time position and forward the last batch at the 12th time position when the pipelining transmission is used.

Figure 3.7 presents the workflow model for data transmission from a node to a gateway, through the relay. It is very similar to the model in Figure 3.3, although the relay was added to the transmission. The only difference is that when the relay establishes a connection with the node, the relay will reply with an *ACK&WAIT* packet to the node. An *ACK&WAIT* packet tells the node that it is communicating with a relay, instead of a gateway.

After the node receives an *ACK&WAIT*, it will enter sleep mode for a while and wait for the next instruction from the relay. At the same time, the relay will establish a connection with the next relay or



(a) Non-Pipe



(b) Half-Duplex Pipe

Figure 3.6: Comparison Between Transport With and Without Using Pipeline

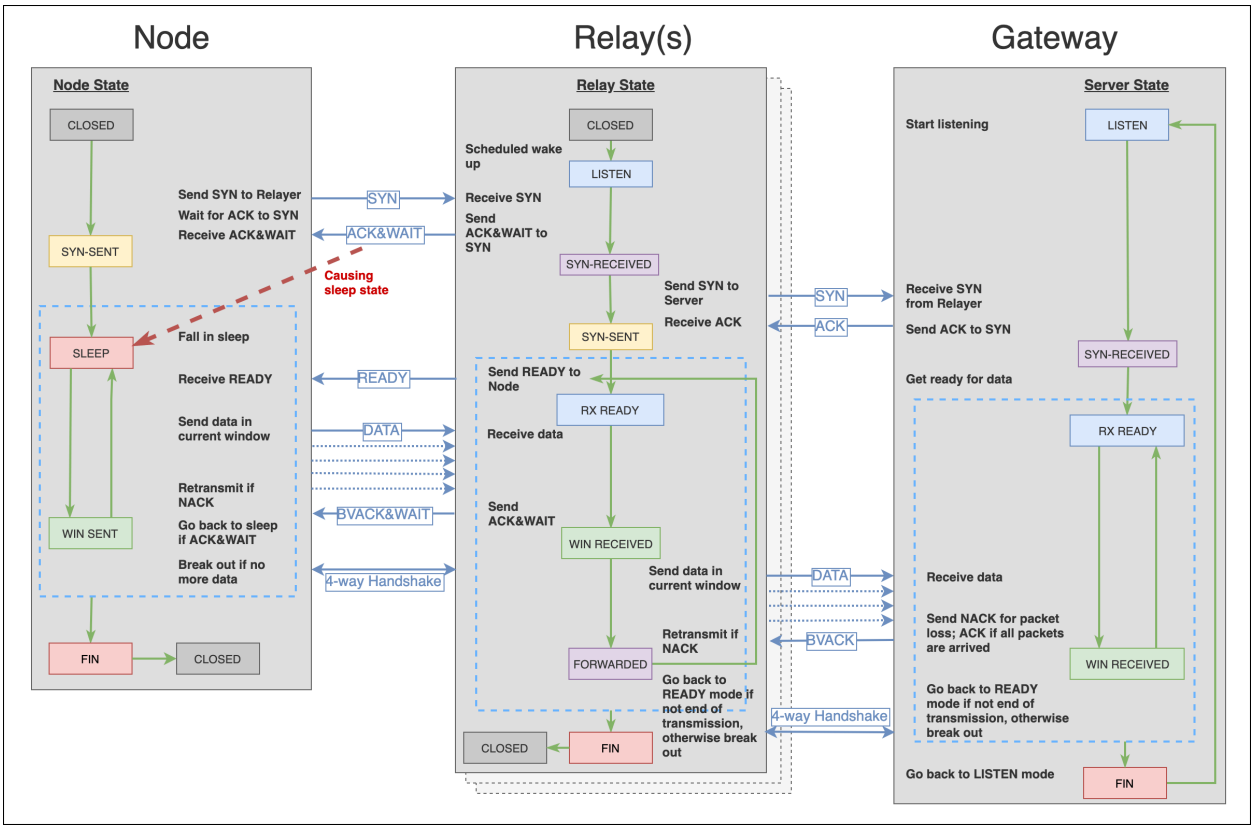


Figure 3.7: A Workflow Model for Data Transmission from Node to Gateway

gateway. The reason for this is to prepare for the pipeline transmission which invokes all available devices on the route to perform pipelining. After the relay established a connection with the next relay or gateway, it will send a *READY* packet to the node and starts the transmission. When the first transmission window is complete, the relay will put the node to wait again and forward all the received packets to the next relay or gateway first. All devices repeat the above steps until the end of the transmission.

This design is flexible because it can be used in a variety of different situations: single-hop transmission between node and gateway, multi-hop transmission through a relay, and multi-hop transmission through multiple relays. Figure 3.8 illustrates this principle when multiple relay devices are deployed between the node and the gateway; this design can still satisfy the multi-hop transmission task among multiple relay devices.

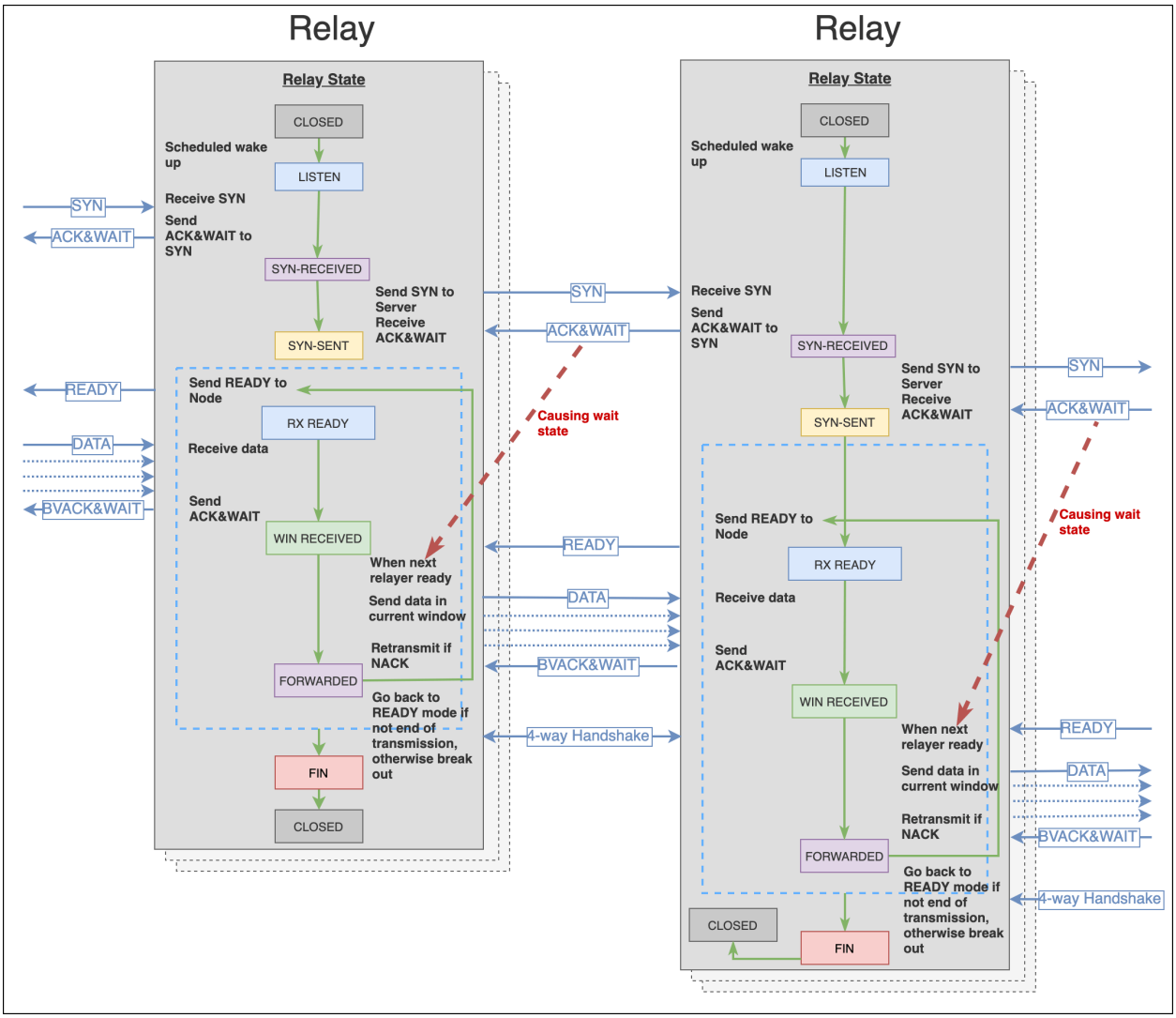


Figure 3.8: A Workflow Model for Data Transmission Between Relays

3.2.1 Storage Requirement for Pipeline Transmission

When using the pipelining-based MPLR protocol in chain network, the relay devices need extra storage space to buffer all packets in one window until all packets in the window have been successfully received and forwarded to the next node. Devices in IoT systems are often restricted and don't have much storage space. Adding extra storage to relay devices not only will increase the build cost, but also will shorten the battery life. This is a shortcoming of the proposed protocol.

3.3 Implementation of MPLR

The implementation of MPLR consists of three classes: *Packet*, *State Machine* and *MPLR* logical set. The *Packet* class is mainly responsible for constructing and parsing the packets in the proposed LoRa network in Figure 3.2. It converts each field into relative bytes and appends to the packet. It is worth mentioning that the order of bytes used in the proposed LoRa network is big-endian. The actual implementation is shown in listing 3.1.

```
def bin(self):
    ''' encode all field to bytes
        Return: byte string of all fields
    '''
    rst = unsigned8(self.pVersion).to_bytes(1,'big')      # 8bits
    rst += unsigned32(self.destEUI).to_bytes(4,'big')     # 32bits
    rst += unsigned32(self.sourceEUI).to_bytes(4,'big')   # 32bits
    rst += unsigned8(self.service).to_bytes(1,'big')      # 8bits
    rst += unsigned16(self.Seq).to_bytes(2,'big')         # 16bits
    rst += unsigned8(self.Flag).to_bytes(1,'big')         # 8bits
    rst += unsigned8(self.payloadSize).to_bytes(1,'big')  # 8bits
    rst += unsigned8(self.batchSize).to_bytes(1,'big')    # 8bits
    rst += unsigned16(self.checksum).to_bytes(2,'big')    # 16bits
    rst += self.BodyBytes

    return rst
```

Listing 3.1: MPLR Packet Formation

The implementation of MPLR protocol adopts the mode of state design pattern [18]. A simple *State Machine* class is implemented in the protocol package to manage all states and execute corresponding actions. As shown in listing 3.2, the implementation of this *State Machine* class is simple and meets the most basic requirements of a state machine. It first records a) the current state, b) all the actions/handlers that need to be performed corresponding to that state, and c) the parameters required by the handler function. It provides functions to receive one or more new states and operations, allowing classes to execute based on added states and handlers. Finally, the *run()* function executes only the handler corresponding to the current

state and updates its state and the parameters needed for the next function based on the execution result.

```
class StateMachine(object):
    """ A class contains basic ops for state pattern
    """
    def __init__(self):
        """Init attributes
        """
        self.state = None
        self.handlers = dict()
        self.handlerParam = None

    def addEvent(self, state, handler):
        """add state event to class
        """
        if len(self.handlers.keys()) == 0:
            self.state = state
        self.handlers[state] = handler

    def bulkEventAdd(self, evenDict):
        """add events from a dictionary
        """
        for k, v in evenDict.items():
            self.add_event(k,v)

    def run(self,event):
        """run a event and update its state
        """
        self.state, self.handlerParam = self.handlers[event](self.handlerParam)
```

Listing 3.2: Simple State Machine

The *MPLR* class contains all the required states and corresponding handlers. First, all states are defined in the *MPLRState* class for better reference. Each handler function is then implemented in turn in the MPLR class, and each function returns a tuple value containing the new state and the parameters required for the corresponding handler function. If no arguments are required, the second value in the tuple will be *None*. Finally, the relationship between state and handler is mapped in the dictionary in the MPLR constructor. Listing 3.3 shows an abstraction of this MPLR class.

```
class MPLRState:
    GW_CMU       = 0x01
    Node_CMU     = 0x02
    SendWin      = 0x03
    ListenWin    = 0x04
    WinACK       = 0x05
    SendReSend   = 0x06
    Fin          = 0x07
```

```
    Finished     = 0x08

class MPLR(object):
    """Reliable Transport class
    """
    def __init__(self, _freq=None, _lora=None):
        self.nodeRegist = {
            MPLRState.Node_CMU: self.Node_CMUInit,
            MPLRState.SendWin: self.Node_SendWin,
            MPLRState.WinACK: self.Node_WinACK,
            MPLRState.Fin: self.Node_Fin,
            MPLRState.Finished: self.Node_Finished
        }

        self.gatewayRegist = {
            MPLRState.GW_CMU: self.GW_CMUInit,
            MPLRState.ListenWin: self.GW_ListenWin,
            MPLRState.SendReSend: self.GW_SendReSend,
            MPLRState.Fin: self.GW_Fin,
            MPLRState.Finished: self.GW_Finished
        }
        ...
    def Node_start(self, _service, _path):
        ...
    def GW_start(self, _service):
        ...
    def Node_CMUInit(self, _):
        ...
    def GW_CMUInit(self, _):
        ...
    def Node_SendWin(self, obj):
        ...
    def Node_WinACK(self, obj):
        ...
    def Node_Fin(self, _):
        ...
    def Node_Finished(self, state):
        ...
    def GW_ListenWin(self, cache):
        ...
    def GW_SendReSend(self, obj):
        ...
    def GW_Fin(self, _):
        ...
    def GW_Finished(self, state):
```

```
54          ...
```

Listing 3.3: MPLR Logic Set

3.4 Summary

This chapter introduces the design principle and implementation process of MPLR protocol. Firstly, a method to reduce the waiting time of acknowledgements is proposed, which improves the transmission efficiency through the methods of batched packet transmission and bit vector acknowledgement. Second, the channel reservation mechanism is used in the MPLR protocol to avoid inter-flow interference caused by the use of MPLR in star-topology by separating the control channel from the data channel. In addition, the MPLR protocol in the chain network is optimized, and pipeline transmission is adopted to greatly shorten the time of packet reaching its final destination.

Chapter 4

Multi-Hop LoRa Protocol (MHLR)

Single-hop image transmission is limited by maximum transmission range as well as by uneven terrain and unexpected obstacles that weaken or block the signal. In order to facilitate the application of LoRa-based image transmission technology in practical deployments, the limitations of single-hop transmission must be considered. Multi-hop communication is introduced in the protocol stack, which solves the problem of signal blocking/loss and further expands the network range of the system. This chapter contains the design of the Multi-Hop LoRa (MHLR) protocol, which is a multi-hop communication protocol based on LoRa.

4.1 Topology and Routing Type

As an example scenario, consider an agricultural image acquisition application in which one LoRa node is used as the gateway (destination) for all packets and placed at the edge of the field. All other sensor devices are randomly deployed and communicate with the gateway in single or multiple hops. A tree topology is used to connect all the devices on the field. As shown in Figure 4.1, the gateway acts as the root node of the whole network, and all other devices act as the relays or leaf/end nodes of the whole tree network. In addition to transmitting its own sensed data, the relay is responsible for relaying data from its children. However, an end node simply sends its sensed data to its parent node. Each new sensor device trying to join the network must select a node as its parent to forward the packet to the gateway. If the new node is close enough to the gateway, it can simply choose the gateway as its parent. The method of parent node selection is discussed in routing.

The topological structure of an agricultural image acquisition system tends to be stable without drastic changes. Occasionally, small changes may occur due to transient blocking, equipment running out of power, or the addition of new nodes. For rare topology changes, a proactive routing protocol is redundant and can quickly run out of power. Also, the reactive routing protocol is not suitable because the flooding of routing requests will lead to network clogging. On the contrary, a hybrid routing protocol is more suitable for the design scenario.

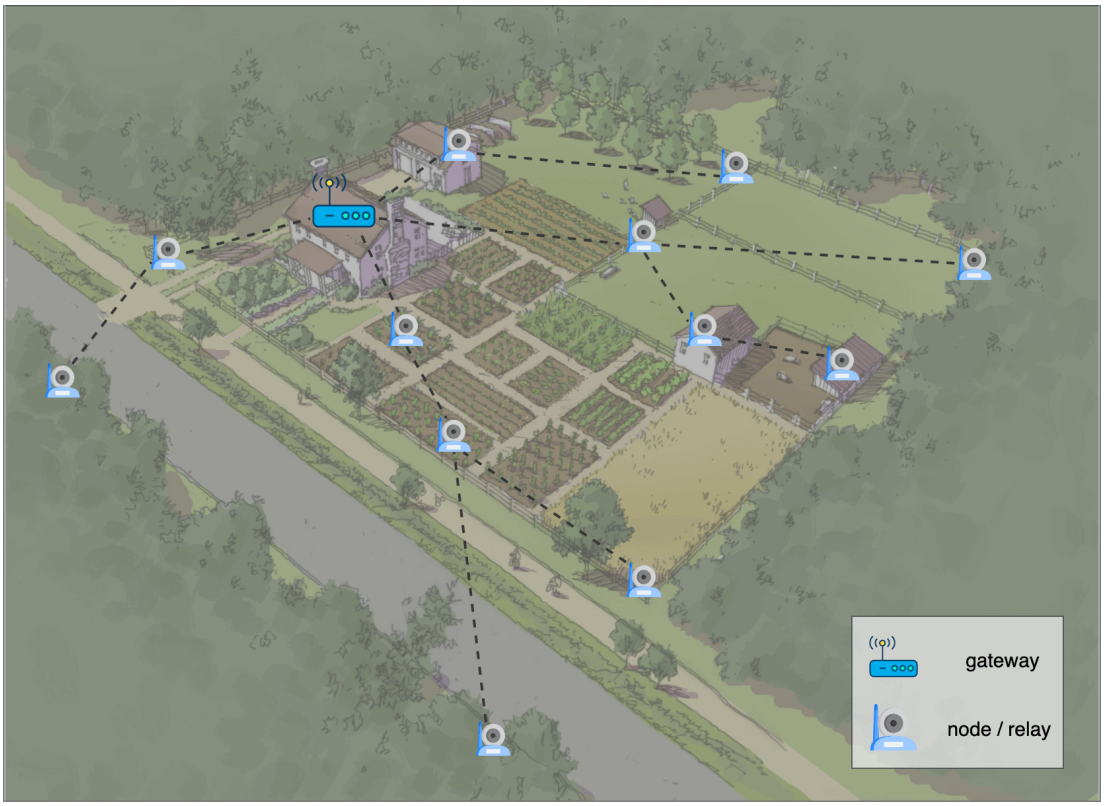


Figure 4.1: An Example of Tree Topology in the Scenario[1]

4.2 System Semantic Model

Similarly as described for the MPLR protocol, all devices fall into three categories: gateways, nodes, and relays. This section details the workflow model for each device. It is worth noting that the clock synchronization problem is beyond the scope of this study, and many robust solutions can be implemented by hardware or software methods.

As shown in Figure 4.2, since the power of the gateway device is unlimited, when the device is turned on, it will be in the listening state until a packet arrives. When the gateway receives a packet, it first looks for its protocol identifier. If the gateway receives an MPLR request packet and it matches the address in the packet, it then starts using the methods in Chapter 3 to receive the data. When data is received, the gateway returns to listening, and another process forwards the received data over wired connection to the final data center. If the packet received is a routing inquiry or heartbeat packet, the gateway replies with the corresponding inquiry result or heartbeat echo, and then returns to listening mode.

As described in previous chapters, node devices are usually battery-powered, and they cannot be in listening or operating mode for long periods of time. Therefore, in this design, the node devices are asleep most of the time and only wake up at planned times. If a node does not have any parent nodes, or if a

[1]Background image modified from https://www.pinterest.ca/TPUDC/.

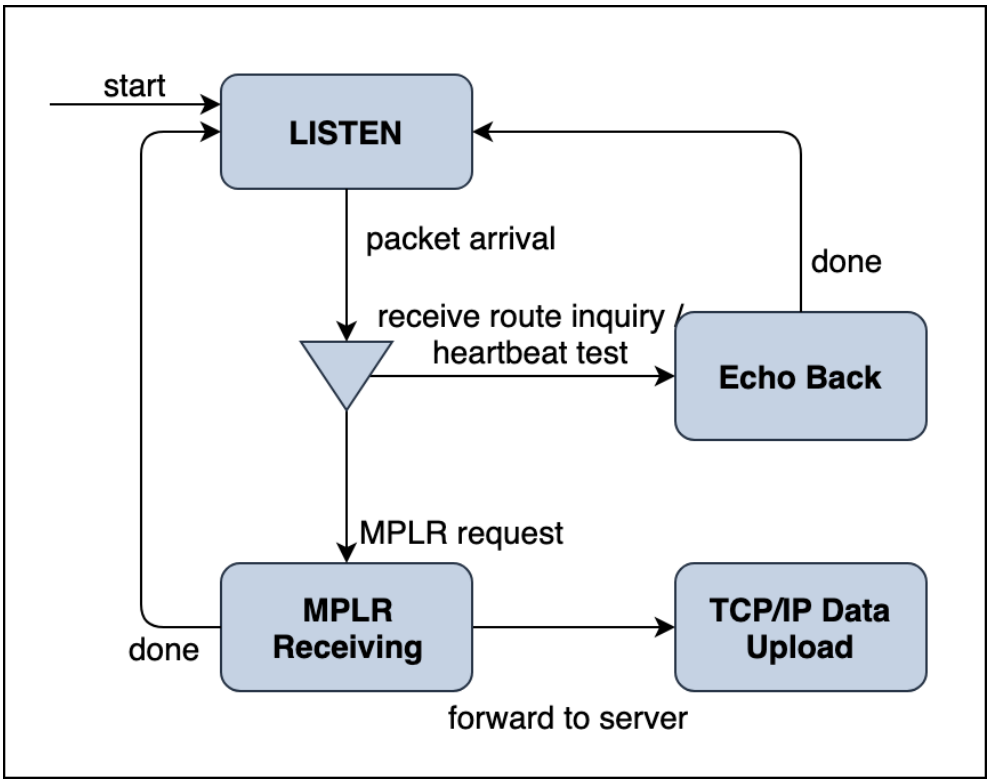


Figure 4.2: A Workflow Model of Gateway

connection to the parent node needs to be tested, the routing inquiry or heartbeat packets need to be sent on wake-up and routing selection is made, if necessary. If the node is already in the tree topology network and needs to send the collected data, the MPLR protocol is used to send the data to the parent node. When all tasks are completed, the node returns to sleep until the next wake up. The details are shown in Figure 4.3.

Relay devices are similar, except that relay devices are also responsible for routing inquiries, heartbeat tests, and data relay of their children. The specific working model of relay devices is shown in Figure 4.4. When a relay device receives a routing inquiry or heartbeat test of the child node, it needs to make a corresponding response. If the child node needs to send the collected data, the relay needs to first receive the data through the MPLR protocol, and then send the data to its parent node through the MPLR protocol as well.

4.3 Packet Formation

In the workflow models of MHLR protocol, different routing inquiry and heartbeat packets need to be sent between parent nodes and child nodes in order to realize network construction and maintenance. This section discusses the format of these messages in detail.

All packets in the MHLR protocol are much shorter than those in the MPLR protocol, since the routing control messages have no payload and no additional fields to control continuous traffic. The format of routing

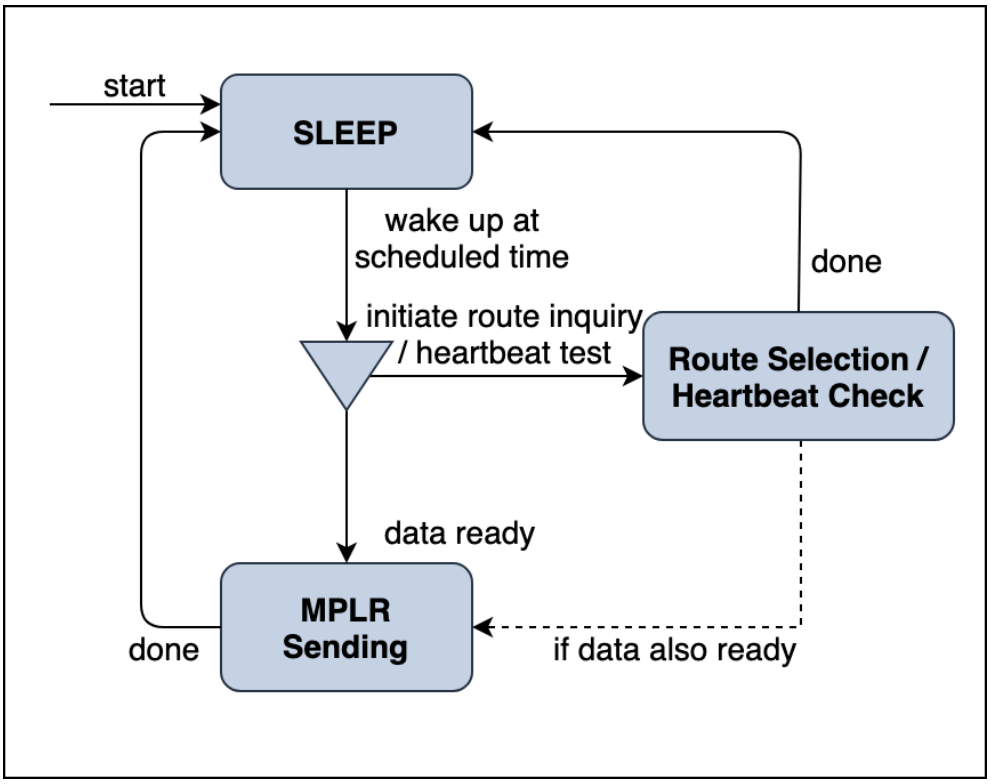


Figure 4.3: A Workflow Model of Node

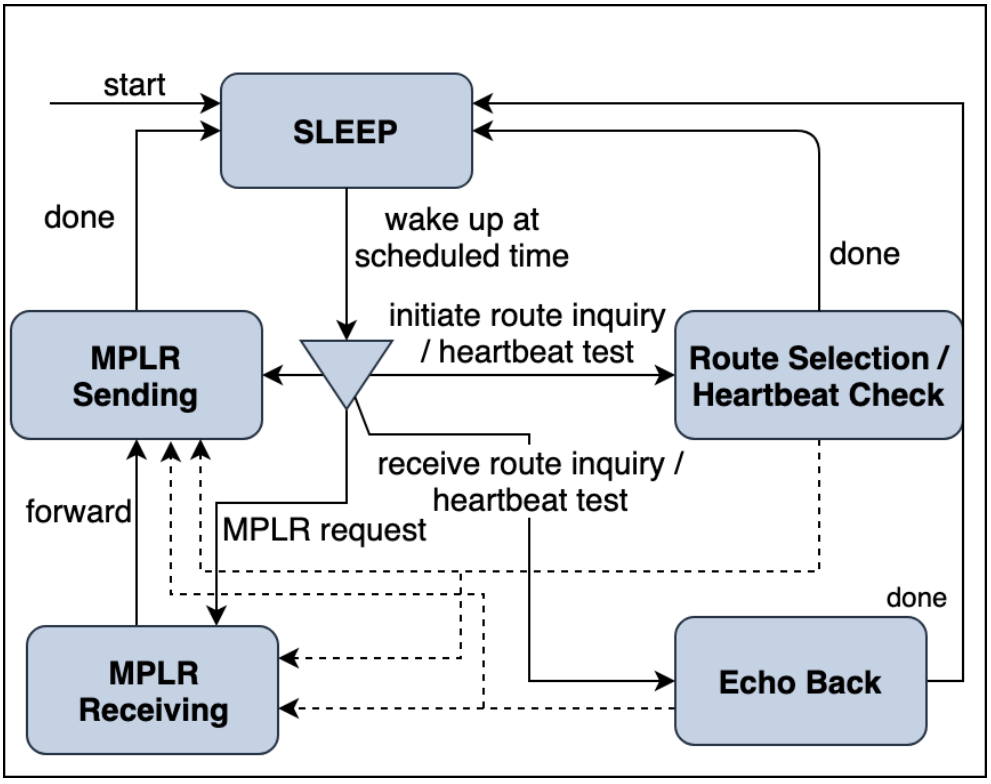


Figure 4.4: A Workflow Model of Relay

inquiry packet, shown in Figure 4.5, contains only three fields, for a total of 8 bytes. The *ProtoVersion* identifies the type of packet separated from the MPLR protocol. Here, the value of the *ProtoVersion* is a unique identifier number used to identify the MHLR protocol. Note that the MPLR protocol also needs to add this field to help distinguish it from the MHLR packets. There are two types of routing control messages: inquiry/heartbeat and reply. The *Inquiry Type* field is used to distinguish between these routing control messages to avoid confusion. The last field is *Source Equipment Unique Identifier (EUI)*, which represents the unique identity of the sender. Since the sender does not know who the recipient is when sending the routing query message, destination EUI and other information are not required. This packet format can also be used for heartbeat packets to test connection to existing parent node.

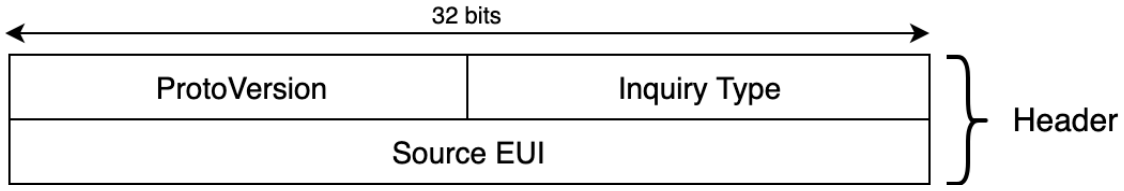


Figure 4.5: Routing Inquiry/Heartbeat Message Format

In the reply packets for routing inquiry and heartbeat, in addition to the EUI used to confirm the sender and receiver, there are five fields that help the device choose a route. As shown in Figure 4.6, the reply packet contains five additional fields, namely, *Hop Number*, *Child Number*, *Relay Time*, *SF*, and *BW*. The following sections explain how these fields are collected and used.

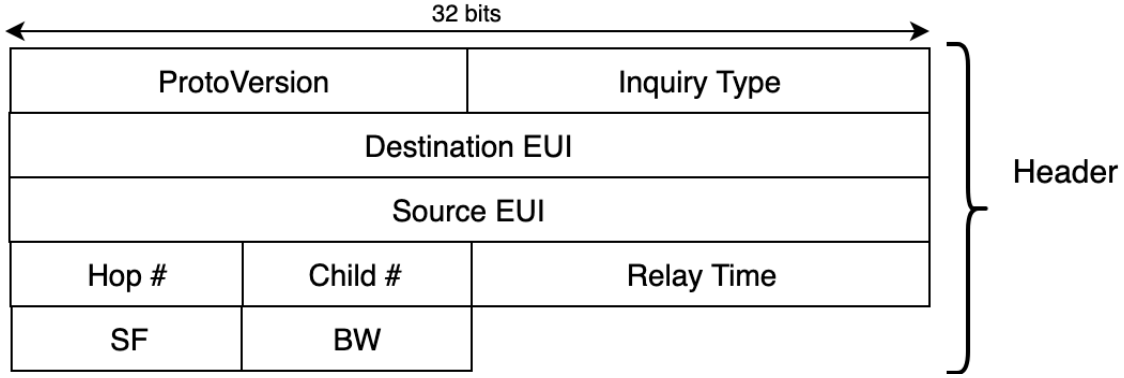


Figure 4.6: Route/Heartbeat Reply Message Format

4.4 Network Formation and Maintenance

During network construction or when a newly added node wants to join an existing network, the joining node needs to broadcast routing inquiry packets to find nearby nodes. After sending, the joining node keeps its receive window open for α seconds to receive all replies. The value of α is a programmable time in the proposed protocol, and the default is 3 seconds. All nodes that receive the inquiry and have joined the

network will reply to the inquiry with the information requested. The receiver will select the best SF and BW according to the RSSI value of the inquiry packet and Table 2.2 (provided by Semtech), and send it back with the reply packet. The joining node then stores these responses sequentially and performs routing selection. The next section describes the details of routing selection.

With periodic heartbeat packets sent, a node that has joined the network can test whether its connection to the parent node has been maintained. If the topology changes and the original parent node can no longer be connected, the node needs to select a new parent node based on the reply packets of other nodes. If the node finds a better relay than the existing parent node, it will also select a new parent. The original parent removes the child from the list if it does not detect it at the next task sending point.

4.5 Supervised Learning for Routing Selection

The introduction of LoRa in Section 2.1.1 shows that several configuration parameters such as spreading factor and bandwidth will directly affect the transmission time and rate. In general, the use of smaller spreading factors and greater bandwidth results in higher physical layer data rates. Other factors need to be considered when choosing a path in a multi-hop network, such as the distance from the gateway and the number of children of the parent node. Several studies have embedded machine learning algorithms into wireless sensor networks to optimize device clustering and data aggregation [3] [37]. This section discusses how the proposed protocol selects features of routing information and uses this data to train the machine learning model and build a software-defined network topology with the shortest transmission time.

4.5.1 Feature Selection

Table 4.1: Features Selected for Machine Learning Method

Spreading Factor	the spreading factor that will be used
Bandwidth	the bandwidth that will be used
Parent Hop Number	number of hops from parent node to the gateway
Family Size	number of nodes participating in parent's network
Parent Relay Time	number of calculated seconds it takes an image to reach the gateway from the candidate parent node

Five different features are selected for machine learning methods, which are summarized in Table 4.1. These are features similar to those often used in conventional routing protocols and are an appropriate number for machine learning models. As mentioned above, *spreading factor* and *bandwidth* are the two basic characteristics that affect the data rate of LoRa physical layer, so that the *spreading factor* and *bandwidth* used by the parent node are the characteristics that need to be used in machine learning. The *parent hop number* is the known optimal distance from the parent node to the gateway. The number of times a packet

needs to relayed. Hop count is a common feature often used in routing protocols because it provides an abstract distance vector for routing computation. Therefore, the *parent hop number* is also selected as one of the properties. The *family size* is the number of nodes connected to the parent node. Since LoRa can communicate with only one node at a time, the number of children affects the average wait time for each child. If the parent node has only one child, the child node can send data to the parent node without waiting. If the parent node has five children, the last child must wait for all the other four children to complete the transmission before it can transmit. Therefore, the *parent child size* is also one of the key parameters affecting the transmission time. Finally, although the number of hops has been considered, this is not sufficient in the LoRa context because each relay can use a different spreading factor and bandwidth, depending on the environment. The *spreading factor* and *bandwidth* here only record the values used by the node and its parent, but it is not possible to know the spreading factor and bandwidth used by the parent and above. Therefore, the *parent relay time* needs to be considered as one of the characteristics. The value of *parent relay time* is calculated locally at each candidate parent node, and is the sum of the time it takes for the node to transmit an image to its parent node and the *parent relay time* of its parent node.

When the network is newly established and the relay device has not transmitted any image data, the *parent relay time* is unknown. At this point, an initial value of -1 can be assigned to indicate the uncertainty, and the request initiator then assigns an average relay time to the node whose relay time is unknown. Although this may skew the selection at first, it can be corrected later through connection check and adjustment. Because of this, it is recommended to use a more frequent connection check during the topology construction and reduce this frequency when the topology is stable.

4.5.2 Vector Formation

By replying to the routing inquiry, the routing inquiry initiator is aware of these five features. When multiple replies are received, the requester selects the optimal routing route, the parent node, through the machine learning model. Before using the machine learning model, it is necessary to use the appropriate feature combination structure so that they can be correctly learned by the model. Therefore, the known features need to be converted into vectors before they can be fetched into the machine learning model.

Consider the five features contained in each response as a vector, denoted as

$$Vector_i = [SF_i, BW_i, Hop\#_i, FamilySize_i, ParentRelayTime_i]. \tag{4.1}$$

The idea is that by giving two vectors, the machine learning method can select a vector with a shorter expected transmission time. Based on this goal, the classification algorithm is the most suitable one in this scenario. Of course, regression algorithms can also be used to predict the transmission time and then compare the two predicted transmission times. However, predicting the actual numerical transmission time values is a harder problem than just predicting which of two routings will give a lower transmission time. As a result, classification makes routing decisions easier and more accurate.

When classification is used, two vectors need to be converted into one for the machine learning method to classify. Here, the subtraction method is used to combine two vectors, as follows:

$$\begin{aligned} Result_{i-j} = [&SF_i - SF_j, BW_i - BW_j, Hop\#_i - Hop\#_j, FamilySize_i - FamilySize_j, \\ &ParentRelayTime_i - ParentRelayTime_j]. \end{aligned} \tag{4.2}$$

By subtracting each value in the second vector from the corresponding value in the first vector, a new vector containing the vector difference between the two vectors is obtained. In each position of the new vector, a positive value means that the value of the element in the first vector is higher, a negative value means that the value of the corresponding element in the second vector is higher, and 0 means the same.

There are only two labels of $Result_{i-j}$, namely, *1* and *0*. If the label is 1, then the path to the gateway corresponding to vector i has a shorter transmission delay than that corresponding to vector j. If the label is 0, then the path to the gateway corresponding to vector i has a longer transmission delay than that corresponding to vector j. If the transmission times for the two paths are the same, the label is set according to which vector was received first. It is worth noting that in the training data set, the label of $Result_{i-j}$ is known by comparing actual transmission times. In practice, the label of $Result_{i-j}$ is predicted by the machine learning model based on the input vector.

4.5.3 Learning System

The training of classification models requires a lot of computing resources. Thus, training machine learning models on restricted devices can reduce the lifetime of the whole WSN system and is not recommended. Alternatively, model training can be performed on a PC or cluster and then the trained model can be deployed on each of the restricted devices. This thesis adopts a learning system combining both online training and real-time classification.

Training machine learning models requires a lot of data, computation and time; therefore, the training can be performed on high-performance devices, and the model can be saved and applied to each node after the training. In this way, each node can directly use the model after training without wasting extra energy on training. As shown in Figure 4.7, the whole learning model is divided into offline training and real-time classification. Details are given in the following paragraphs.

Incremental Learning

Features are selected from training data, and feature vectors are extracted by using the above methods for machine learning model training. This happens on a desktop or server cluster, not on the deployed IoT device. When the machine learning model is trained, it can be saved and distributed to each deployed IoT device. The size of the trained classification models varies from a few kilobytes to tens of kilobytes, so the distribution and storage of the models does not result in excessive overhead.

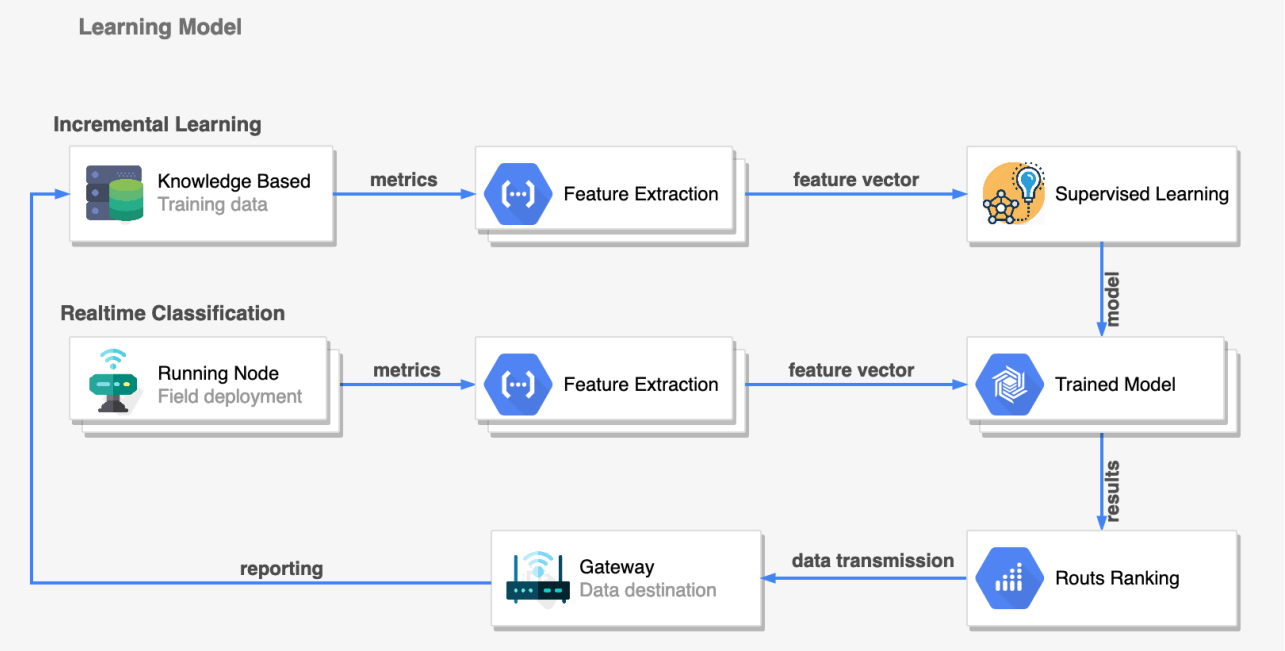


Figure 4.7: Overview of the Learning System

It is called incremental learning because each time the gateway receives an image task, it uploads the path information to the trainer for real-time learning. This is because the unpredictability of the actual deployment environment reduces the accuracy of model judgments. To understand the impact of the environment on routing, real-time training can improve accuracy by collecting real routing data to update the model. The gateway can know the time and route details of each transmission task by obtaining sub-tree formation information and adding creation time to the MPLR protocol header. The obtained routing information is then uploaded to the training machine from the gateway and used for incremental learning. The updated model can be distributed to each node periodically through the MPLR protocol. In this scenario, for example, because the deployment environment is relatively stable, the distribution of the updated model is not very frequent and can generally be done once a day. Incremental learning is a good tool for making machine learning models adaptive to the environment, but since it is beyond the scope of this thesis, it will not be evaluated in Chapter 6.

Real-time Classification

As soon as the deployed node receives the new model, it is immediately applied. The deployed node will extract each available path and generate an input vector. The new model determines the routing options provided and predicts the fastest path to send data. This is effective because the model has been trained and the expected steps do not require much time and effort. It is worth noting that when a large number of node devices join the network at the same time, the proposed protocol cannot avoid routing oscillation, that is, a large number of nodes rushing to one optimal parent node and switching to another optimal parent

node. Since there will not be a large number of nodes joining at the same time in the designed scenario, the proposed protocol does not handle the problem of routing oscillation. This is left for future work.

4.5.4 Implementation of MHLR

In the implementation of MHLR, three different routing control packets are implemented by one class. As Listing 4.1 shows, the packet generator dynamically generates the corresponding packets based on the indicated message type.

```
    def bin(self):
        """form inquiry message:
            cINQUIRY:   8  Bytes
            cHEARTBEAT: 8  Bytes
            cACK:       18 Bytes
        """
        rst = unsigned8(self.PVersion).to_bytes(1,'big')            # 16 bits
        rst += unsigned8(self.MessageType).to_bytes(1, 'big')       # 16 bits
        rst += unsigned48(self.Source).to_bytes(4, 'big')           # 32 bits
        if self.mType == MessageType.cACK
            rst += unsigned8(self.Hop).to_bytes(1, 'big')           # 8  bits
            rst += unsigned8(self.Child).to_bytes(1, 'big')         # 8  bits
            rst += unsigned8(self.relayTime).to_bytes(2, 'big')     # 16 bits
            rst += unsigned8(self.SF).to_bytes(1, 'big')            # 8  bits
            rst += unsigned8(self.BW).to_bytes(1, 'big')            # 8  bits

        return rst
```

Listing 4.1: Message Packing

Similar to the MPLR class, as shown in listing 4.2, MHLR also has a proprietary class that contains all the required states and corresponding handlers. All states are also defined in the MHLRState class, followed by each handler function in turn in the MHLR class. Each of handler returns a tuple value containing the new state and parameters required for the corresponding handler function. If no argument is required, the second value in the tuple will be None. Finally, mapping of all the states and corresponding handlers to the dictionary is performed in the MHLR constructor.

```
class MHLRState:
    INQUIRY = 0x01
    WAITING = 0x02
    ACK     = 0x03
    LISTEN  = 0x04
    EDGING  = 0x05
    Data    = 0x06

class MHLR(object):
```

```
    def __init__(self, role):
        ...

        # Registry Table
        self.Regist = OrderedDict()
        if role == NetRole.Gateway:
            self.Regist[State.LISTEN]   = self.GW_Listen
            self.Regist[State.ACK]      = self.GW_ACK
            self.Regist[State.Data]     = self.GW_MPLR
        else:
            self.Regist[State.INQUIRY]  =  self.Node_Inquiry
            self.Regist[State.WAITING]  =  self.Node_Waiting
            self.Regist[State.EDGING]   =  self.Node_Edging
            self.Regist[State.Data]     =  self.Node_MPLR

    def GW_Listen(self, _):
        ...
    def GW_ACK(self, obj):
        ...
    def GW_MPLR(self,obj):
        ...
    def Node_Inquiry(self, _):
        ...
    def Node_Waiting(self, _):
        ...
    def Node_Edging(self, _):
        ...
    def Node_MPLR(self, _):
        ...
```

Listing 4.2: MHLR Model

The code in Listing 4.3 is used to train the machine learning model on the PC side. This code will train five different machine learning methods and store the results of the trained model.

```
""" on PC or cluster """
mlDic = {
    'MLPClassifier': MLPClassifier(activation='relu',solver='lbfgs', alpha=1e 5,
    hidden_layer_sizes=(11,), random_state=45),
    'SGD': SGDClassifier(loss="log", max_iter=2000, tol=1e 4, n_jobs=4),
    'DecisionTreeClassifier': DecisionTreeClassifier(max_depth=5),
    'RandomForestClassifier': RandomForestClassifier(max_depth=5, n_estimators=10,
    max_features=1),
    'AdaBoostClassifier': AdaBoostClassifier(),
}

for name, model in mlDic.items():
```

```
    # partical data training
    model.partial_fit(X, Y)
    # save the result
    dump(model, name+'.joblib')
```

Listing 4.3: Machine Learning Prediction

The trained machine learning model is applied as soon as it is received by the deployed node. When inquiry responses are received, the model is used to determine the best parent node. The implementation process is shown in Listing 4.4.

```
""" Node side """
mlModel = load('MLPClassifier.joblib')
...
# receive inquiry reply
parent = reqPool[0]
if len(reqPool) == 1:
    return parent

for idx in range(1,len(reqPool)):
    rst = mlModel.predict([parent,reqPool[idx]])
    if rst == 0:
        parent = reqPool[idx]
return parent
```

Listing 4.4: Machine Learning Prediction

All the vectors in the response packets are sorted in the order they were received and compared using machine learning model in turn. After each comparison, the better vector is saved and compare it with the rest. Such comparisons can be made in a linear order of n times.

4.6 Summary

MHLR is a hybrid routing protocol applying machine learning to choose a routing path. Devices using the MHLR protocol will form a tree network topology after network construction, and periodically evaluate the connection of the network and the adaptation to network topology changes through heartbeat packets. The MHLR protocol can not only realize the multi-hop transmission of sensory data, but also can be used together with the MPLR protocol for the transmission of large messages. In the aspect of machine learning, 5 features are extracted by capturing the information of routing and used in the machine learning model. The trained models will be distributed to each node for application.

Chapter 5

Experimental Methodology

Experimental methods for both MPLR and MHLR are discussed in this chapter in detail. This chapter also describes the image compression method, hardware equipment and equipment settings used in each experiment. Besides, the specific experimental scheme and topology of each test are also introduced.

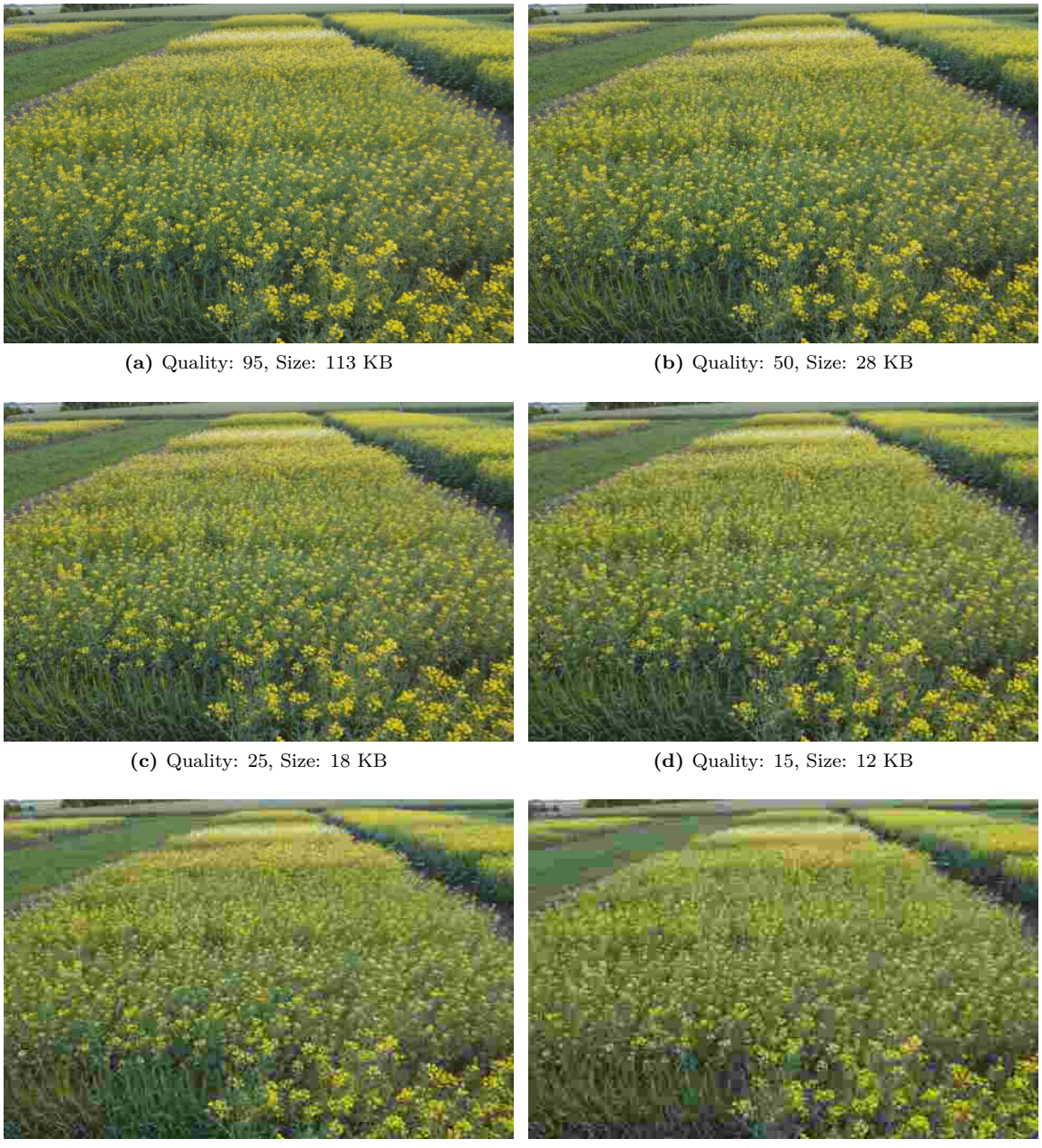
5.1 Experimental Image and Compression

Images can provide specific information for breeders and producers, such as flowering time and flower density. They can also provide the physical status of the remote IoT system, such as camera/sensor positioning, or physical impediments that affect image capture, like weather conditions or wildlife interference. This can be useful for directing system operation. For example, a rainstorm may change the desired data collection parameters for a period of minutes/hours; a human operator or automated adaptation system could change the frequency of image capture or soil moisture reading.

In many such use cases, images can be greatly compressed without impacting their usefulness. In our experiments we use images from a field-deployed camera that generates a 3280x2464 full-resolution image of 5.8 MB. The Pillow[1] Python imaging library was used to resize the image to 480x320 pixels, as might be sufficient for a monitoring application, and to apply JPEG compression. An image quality parameter can be passed to the compression algorithm to indicate the degree of compression, which is on a scale from 1 (most compression) to 95 (least compression). This parameter can be tuned depending on application requirements.

To illustrate the impact of different settings for the image quality parameter, Figure 5.1 shows results from applying compression to a resized 480x320 pixel image from a canola field. The caption for each subfigure gives the corresponding value for the image quality parameter as well as the resulting size of the image in kilobytes. With quality 50 and 25, there is little loss in quality, and the size is reduced to 28 KB and 18 KB, respectively. Significant image distortion is noticeable when the image quality is 10 and 7, but this quality may be acceptable in some applications. Thus, by using JPEG compression [34], a 480x320 resized image compressed to 113 KB (highest quality) can be converted into an image of only 28 KB, 18 KB, 12 KB, 9 KB or 7 KB, with quality parameter settings of 50, 25, 15, 10 and 7, respectively, with only small to moderate loss of quality.

[1] https://pillow.readthedocs.io/en/stable/



(a) Quality: 95, Size: 113 KB

(b) Quality: 50, Size: 28 KB

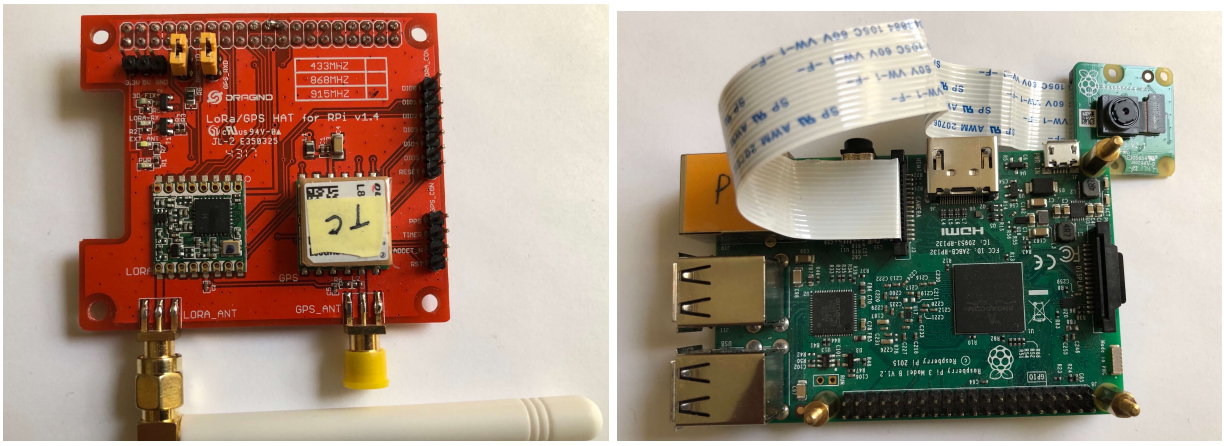
(c) Quality: 25, Size: 18 KB

(d) Quality: 15, Size: 12 KB

(e) Quality: 10, Size: 9 KB

(f) Quality: 7, Size: 7 KB

Figure 5.1: Image Quality and Size Comparison



(a) Dragino LoRa Hat **(b)** RPi 2 model B

Figure 5.2: Experimental Device

5.2 Experimental Device

In all experiments, Raspberry Pi (RPi), a single-board computer, was used as the host device in conjunction with LoRa modules[2] manufactured by Dragino. The LoRa module is a SX1276[3] chip-based transceiver that communicates data through a serial peripheral interface, and this transceiver can only listen on one channel at a time. In this scenario, the only difference between the gateway and the node is that the gateway has an Ethernet connection. Generic drivers for each physical component are easily found on the Internet. Depending on the specifications of the different modules, however, the pin settings need to be modified in the driver.

Figure 5.2 (a) shows the Dragino LoRa hat used in this study. This hat contains an SX1276 LoRa chip and a GPS module. Since the GPS module was not used in this design, only the LoRa chip is equipped with an omni-directional antenna. The upper two rows of pins are used to connect the host device. Figure 5.2 (b) shows an RPi 2 model B machine. This single-board computer provides rich physical connection interfaces, such as USB and Ethernet ports, which is convenient for quick development and debugging during outdoor experiments. Two rows of pins at the bottom of RPi are used to connect with LoRa hat and to communicate through serial peripheral interface. The specific wiring of all experimental devices are listed in Table 5.1.

5.2.1 LoRa Configuration

The LoRa parameter settings used in all experiments are listed in Table 5.2. The output power is set to 15 dBm in all our experiments and the performance through a range of power levels will be examined in future studies.

[2]http://www.dragino.com/products/lora/item/106-lora-gps-hat.html
[3]https://www.semtech.com/products/wireless-rf/lora-transceivers/sx1276

Table 5.1: Wiring Between RPi and LoRa

LoRa/GPS Hat	RPi 2/3 Pin#	RPi 2/3 GPIO#
DIO0	4	N/A (5V power pin)
RESET	11	17
MOSI	19	10
MISO	21	9
SCK	23	11
NSS	24	8

Table 5.2: LoRa Configuration

Name	Value
Spreading Factor	7, 8, 9, 10, 11, 12
Bandwidth (kHz)	500, 250, 125
Coding Rate	4/5
Implicit Header Mode	false
Preamble	8
Output Power (dBm)	15

5.2.2 Deployment and Experimental Parameters

Seven experiments evaluated LoRa's ability in image transmission and the performance of the proposed protocols. These experiments include full payload transmission, point-to-point experiment with just a single sender and receiver, experiment using a star topology, experiment using a chain topology, indoor routing experiment and outdoor routing test. In all experiments, except the first experiment, the maximum MPLR batch size was selected as 40 packets and the maximum MTU size was selected as 160 bytes.

Full Payload Transmission

When testing the transmission performance of LoRa, different packet loss rates were experienced with different spreading factor and bandwidth. The reason is that the larger the spreading factor and the bandwidth, the longer the chirp symbol will be and the more influenced by the environmental noise. To understand the influence of packet loss rate in LoRa transmission when different settings are used to transmit packets of different sizes, full payload transmission experiment is designed. The purpose of this experiment is to measure the stable delivery rate of two LoRa node under different transmission settings and data sizes. The experiment was carried out in a laboratory environment with two LoRa devices 5 metres apart, one for a transmitter and the other for a receiver. The experiment used all possible combinations of spreading factor and bandwidth to send plain text payload of 10 bytes, 50 bytes, 100 bytes, 150 bytes, and 189 bytes, which are encoded into

packet sizes of 17 bytes, 69 bytes, 137 bytes, 201 bytes, and 255 bytes, respectively. To avoid random errors, each packet was sent 100 times to calculate a stable average packet loss rate. Both devices recorded the time and success of sending and receiving packets, and the packet loss rate can then be calculated through simple calculation.

Point-to-point Transmission

In order to verify whether the proposed MPLR protocol is more advantageous than the state-of-the-art protocol in the transmission time of large messages, point-to-point transmission experiments are designed and carried out in the laboratory to evaluate the transmission performance of the MPLR protocol. Similarly, two LoRa devices were placed 5 metres apart. Using different spreading factor and bandwidth combinations, MPLR protocol and Stop-and-wait protocol are used to evaluate the image transmission of different sizes respectively, and their performances are thus compared. Packet loss was not introduced in the first set of tests, to understand the advantages of the MPLR protocol under ideal conditions. In the second set of experiments, 2%, 5%, and 10% artificial packet loss rates were introduced. Since noise from the environment and from other devices using the same frequency range are common in IoT systems, the introduction of a 2% to 10% artificial packet loss rate can test the proposed protocol's ability to resist noise. In each transmission, both devices record the spreading factor, bandwidth, file size, transmission time, and artificial packet loss rate has been introduced. Similarly, to avoid random errors, each transmission was repeated 5 times to get a more stable average transmission time. Any transmission that experienced non-artificial packet loss was discarded and redone.

Transmission in Chain Network

Section 3.2 introduced an optimization of the MPLR protocol in chain network that utilizes the pipelining transmission to reduce the time required for multi-hop transmission across a line. To verify this optimization, a chain network test experiment is designed in the laboratory environment. In this experiment, a 3-hop chain network consisting of 4 LoRa devices was constructed. The transmitter and receiver are at both ends of the chain network, and the two other LoRa devices in the middle are the relays. Both pipelining and non-pipelining MPLR protocols are used to transmit images at a size of 27 KB under SF 7 and BW 500 settings respectively. The experiment was carried out five times to calculate the average time and avoid random errors. Each device records the timestamps of packets arriving and sending to analyze protocol performance.

Transmission in a Star Network

In order to prevent packet collisions caused by inter-flow interference in star topology network, channel reservation is introduced in Section 3.1.1 to separate control channel and data channel. The purpose of this experiment is to verify that the MPLR+Channel reservation mechanism has more advantages than the state-of-the-art stop-and-wait+ALOHA mechanism. Both protocols are tested in the outdoor star topology

network. Four configurations of nodes are used in this experiment with a star topological network of 5, 10, 15, and 20 devices respectively. Each group of experiments used MPLR+Channel reservation mechanism and stop-and-wait+ALOHA mechanism to conduct 125-minute image transmission experiment respectively. The test area is a 200x300 metre park, where a LoRa gateway is placed at the corner of the test area to receive all images, and the rest of the devices are randomly placed in the test area as sender nodes. All devices used a spreading factor of 8 and a bandwidth of 250 kHz. In each 125-minute experiment, each node generated a 9 KB image for transmission every 5 minutes. If an image transmission had not completed before a new task at that node was generated, the new task was queued. The reception of packets at the gateway was stopped after 125 minutes, even if there were image transmission tasks queued at nodes. All experiments did not introduce any artificial packet loss, nor packet loss caused by random errors, so it is based on the actual deployment environment. All devices record data related to image transmission in the log for later analysis.

Training Data Collection and Learning

Machine learning models require a large amount of data for training. Therefore, the purpose of this experiment is to generate training data for machine learning models by deploying tree topology-based networks of different sizes. Also, the collected training data can be used to provide initial evidence that the model based on selected features can accurately select routes in LoRa multi-hop networks by analyzing the learning results of training data. Since the MPLR protocol and tree topology based LoRa multi-hop network is first proposed in this thesis, there are no existing deployments to refer to. As a result, this experiment constructs a rich number of tree topologies to generate and obtain routing data for training, and all topologies used are listed in Figure 5.3. These topologies varied from 1 hop to 3 hops, with each layer of the network having a number of child nodes ranging from 1 to 3. In each topology, the root node is the gateway, the leaf node is the transmitting node, and the non-leaf node is the relay. Each topology was tested individually and used all possible combinations of SF and BW to transmit 9 KB images. Each image transmission was repeated three times to obtain a stable average transmission time and to avoid random errors. During the experiment, all the devices recorded the number of child nodes, the time of image arrival and transmission, and the relay waiting time. By analyzing the logs, the total transmission time of each image and the parent relay time can be calculated. During the process of data collection and reduction, since it is not known which parent node the child node will choose to connect to in practical application, each possible combination of child node and parent node forms a routing record to enrich the data set. After obtaining this information, they can be combined in pairs to form the subtraction vector shown in Formula 4.2 for machine learning models to learn.

Indoor Routing Test

After training the machine learning model with training data, the prediction ability of these trained models must be further verified to see if they can correctly predict the optimal path in the real indoor environment. To this end, indoor routing tests were conducted in a medium-sized computer lab to further evaluate the

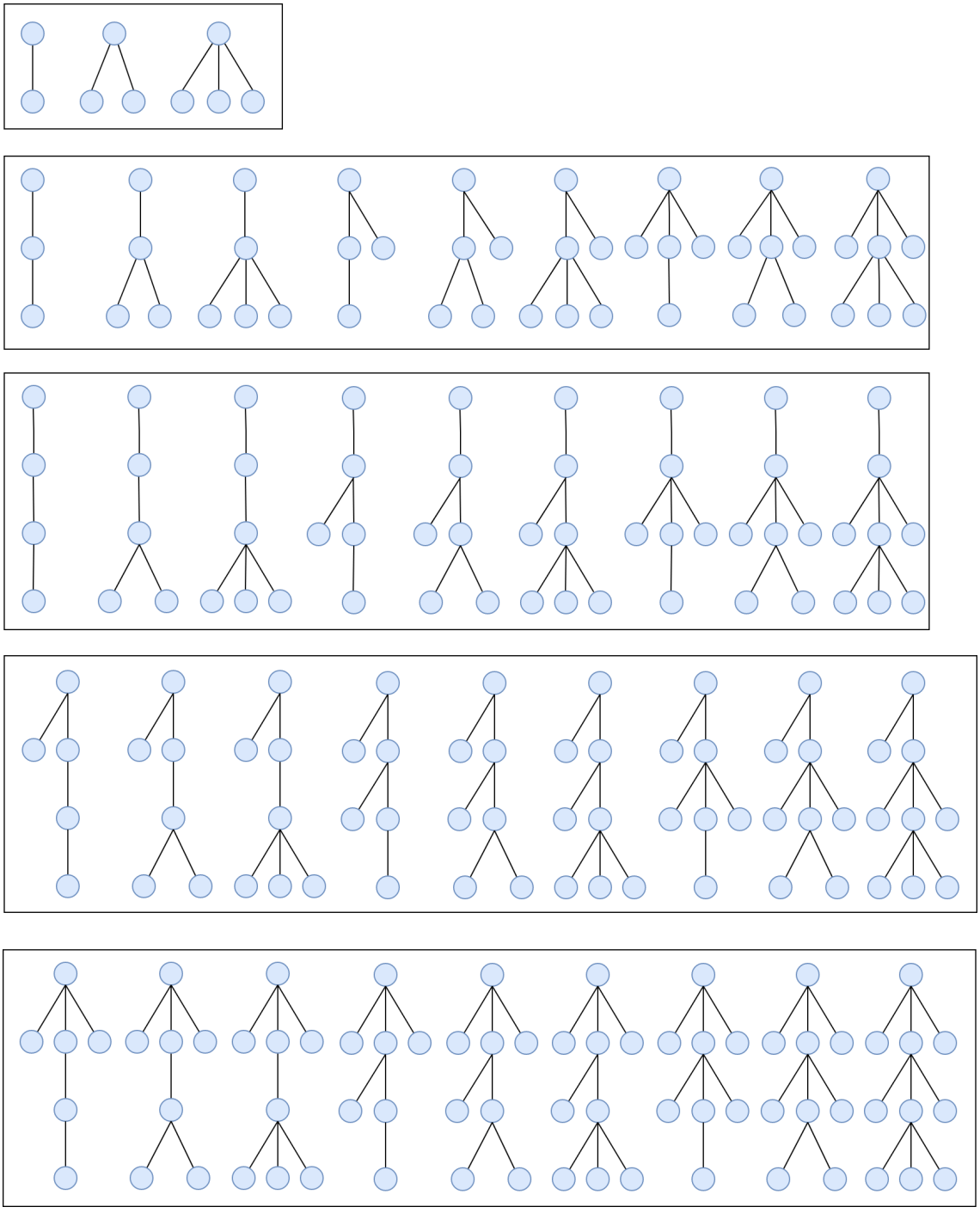


Figure 5.3: Topologies used in Training Data Collection Experiments

learning result. The MHLR protocol is not used in this experiment to automatically form network topologies. Instead, multiple hard-coded tree topologies were built to generate more routes in a real indoor environment. Using hard-coded tree topologies can increase the diversity of indoor routing data without changing the location of devices, and can record the total transmission time under various routing options. These data were used to evaluate the accuracy of classification of five machine learning models trained by the training data. These topologies used up to 10 LoRa devices, which contain both optimal and bad network routes. Each topology has at least one hop and at most three hops; each network layer has at least one child node and up to 3 child nodes. Similarly, all possible combinations of spreading factor and bandwidth are used to transmit 9 KB images one by one. During the experiment, each device recorded the parent node selected by the device and the number of child nodes, as well as the timestamp of image arrival and sending. The total time used for image transmission and the relay time of the parent node can be calculated by log analysis. In this experiment, packet loss caused by random errors is not ignored. In other words, the impact of packet loss is retained in the collected data set. All routing data will be obtained by parsing device logs, and each pair of routing information constituted into a subtraction vector for machine learning model to predict.

Outdoor Routing Test

In order to study the performance of MPLR and MHLR protocols in outdoor environment, an outdoor routing experiment was conducted in a low-density residential area. The purpose of this experiment is to verify the feasibility of MPLR and MHLP protocols in building LoRa networks in outdoor environments. Ten LoRa devices were deployed in the region, including one gateway, four relays, and five nodes. The experiment adopted a spreading factor of 10 and a bandwidth of 250 kHz to transmit an image of 9 KB every 5 minutes. In this experiment, since the parent relay time is unknown at the beginning, the first two connectivity checks were performed every 30 seconds and then every 5 minutes. The machine learning model used in the experiment is an MLP model trained by the training data mentioned earlier, and the trained model is distributed to each relay and node device prior to the start of the experiment, which is used for intelligently route selection and automatically network topology construction. During the experiment, each device recorded its parent node, the number of the child nodes, as well as the start and end time of each image transmission. The resulting topology, as well as the transmission time and route of each image data, can be obtained through log analysis. Similarly, since this is a field deployment, the experiment did not introduce any artificial packet loss, and there is no avoidance of any packet loss due to random errors.

5.2.3 Performance Metrics

The full payload transmission experiment considers the maximum payload that is stable under different combinations of bandwidth and bandwidth. By sending packets that grow in size, until find the largest and most stable transmission payload under each combination of settings.

The performance metric considered in the point-to-point experiment is the average image transmission

time with or without packet loss respectively. Compared to the stop-and-wait protocol, MPLR protocol shorts the waiting time for acknowledgment packets during the transmission process by using the patch transmission. Therefore, comparing the average transmission time required by the two protocols can illustrates the advantages of the proposed MPLR protocol. The performance metric of the transmission experiment in the chain network is the arrival time of each packet batch in each relay and gateway. Since the pipelining transmission is used to optimize the MPLR protocol for chain network, each packet batch can be delivered in a different way and at a different time from the normal MPLR protocol. Therefore, the way of relaying and performance of each protocol in the chain network can be analyzed by examining the arrival time of each packet batch in each relay and gateway. For the tests using a star topology, the distribution of the image transmission time, the number of packet collisions, the number of successfully transmitted images, and inter-node fairness were examined. These performance metrics can clearly show if the proposed protocol with channel reservation mechanism can greatly reduce inter-flow interference in image transmission tasks, and have a higher success rate. It is also necessary to know whether the mechanism has certain fairness in the network, so that all nodes can fairly complete image transmission within a certain time and transfer almost the same number of images.

The purpose of training data collection and learning is to generate sufficient training data for the machine learning models to learn, and to evaluate whether the selected features are good enough for the machine learning models to infer the route with the shortest transmission time. For the latter, its performance metric is the classification accuracy, precision, recall, f1, and the learning curve of machine learning models after learning the training data set. Classification accuracy is defined as the ratio of number of correct predictions to the total number of input samples. A good classification accuracy is that the trained model can identify the route with the shortest transmission time in most of the route pairs. Precision is the ratio of the number of label 1 in the correct prediction results to the number of all label 1 in the prediction results. It is given by the following expression:

$$\text{precision} = \frac{\#\text{ of true label 1}}{\#\text{ of true label 1 } + \#\text{ of false label 1}}. \tag{5.1}$$

Recall is the ratio of the number of label 1 in the correct prediction results to the number of label 1 in all data instances. It can be computed as

$$\text{recall} = \frac{\#\text{ of true label 1}}{\#\text{ of true label 1 } + \#\text{ of false label 0}}. \tag{5.2}$$

F1 is calculated as

$$\text{F1} = 2 * \frac{\text{precision } * \text{ recall}}{\text{precision } + \text{ recall}}. \tag{5.3}$$

Learning curve, as an effective visual tool, is used to represent the relationship between training scores and cross-validation scores with a varying number of training data.

For the indoor routing experiments, the performance metrics are also the accuracy, precision, recall and f1 of routing prediction. This experiment investigates whether the machine learning model trained by training data can accurately identify the route with shortest transmission time among all possible routes in indoor environment. Finally, the performance metrics of outdoor routing experiments are the success of automatic network topology construction, topology construction time, image transmission time, network throughput and number of the retransmissions. This experiment examines the ability of the whole protocol to construct the network topology in the outdoor environment by using the machine learning model trained with indoor training data, and analyzes the correctness of the resulting topology with respect to the objective of minimizing transmission times. Besides, the performance of the multi-hop transmission of images is also investigated.

5.3 Summary

This chapter first described the compression technique that used to reduce the size of experimental image data. The LoRa devices and device configuration that used in all experiments are presented. Seven different experiments are designed, the design scheme and experimental environment of each experiment are introduced precisely. Finally, the performance metrics of each experiment are discussed in detail.

Chapter 6

Evaluation

The results of experiments designed to test the performance of MPLR and MHLR protocols are discussed and analyzed in this chapter. These protocols are compared with the state-of-the-art methods to determine the quantitative performance improvement.

6.1 Full Payload Transmission

Theoretically, as the packet size increases, its signal wave will also become longer and more susceptible to interference. This can lead to the loss of the signal strength or even packet corruption during the transmission. The full payload transmission experiment was conducted in the laboratory to see how the packet loss rate for LoRa wireless transmission varies with packet size and transmission settings. Each experiment last 5 minutes and the transmitting node sent a packet every 5 seconds. Table 6.1 shows the average delivery rates of packets of different sizes and settings obtained through experiments, with a standard deviation of 0.26.

It is worth mentioning that packet corruption and the loss of the signal strength is inevitable in wireless communication, even for the most reliable connections. Therefore, although the delivery rates under many transmission settings in the experiment were as high as 100%, there is still a possibility of packet corruption in the actual deployment. Since the maximum buffer size of the SX1276 chip is 255 bytes, any packets larger than this size will be cropped, regardless of the transmission settings used, and resulting in a delivery rate of 0%. As can be seen from the table, from SF 7 to SF 11, except for the use of SF 11 and BW 125, any packet size that smaller than or equal to the buffer size has a high delivery rate in the laboratory environment. When using combination of SF 11 and BW 125, or SF 12, packet delivery rates decrease with the increase of data size, even in low-interference laboratory environments. This is especially noticeable when using SF 12 and BW 125 to transmit packets of 200 bytes, the packet delivery rate drops below 50%. Such a delivery rate is considered a volatile or unstable transmission.

Through the observation and analysis of Table 6.1, the maximum packet length is computed for different combinations of transmission settings under stable transmission. All maximum packet lengths are listed in Table 6.2. All settings can be used to send packets of up to 255 bytes except for SF 11-BW 125, SF 12-BW 250, and SF 12-BW 125, which can only send packets of up to 149 bytes, 137 bytes, and 41 bytes, respectively. These three settings should be avoided in image transmission, not only because their maximum packet size

Table 6.1: Packet Delivery Rates for Different SF, BW, and Data Size

Bytes		17	69	137	201	255
SF	BW					
7	500	100%	100%	100%	100%	100%
	250	100%	100%	100%	100%	100%
	125	100%	100%	100%	100%	100%
8	500	100%	100%	100%	100%	100%
	250	100%	100%	100%	100%	100%
	125	100%	100%	100%	100%	100%
9	500	100%	100%	100%	100%	100%
	250	100%	100%	100%	100%	100%
	125	100%	100%	100%	100%	100%
10	500	100%	100%	100%	100%	100%
	250	100%	100%	100%	100%	100%
	125	100%	100%	100%	100%	100%
11	500	100%	100%	100%	100%	100%
	250	100%	100%	100%	100%	99%
	125	99%	94%	92%	85%	78%
12	500	99%	99%	99%	97%	92%
	250	99%	91%	91%	87%	81%
	125	98%	75%	61%	46%	27%

is limited, but also because the data rate resulted by these three settings is very small.

In summary, when using LoRa to transmit image data, stable settings at full payload should be used. The unstable settings should be avoided and can only be used to transmit sensor data.

Table 6.2: Maximum Data Size(in bytes) for Stable Transmission under Different SF and BW

SF/BW	500	250	125
7	255	255	255
8	255	255	255
9	255	255	255
10	255	255	255
11	255	255	149
12	255	137	41

6.2 MPLR

This section discusses the results of three experiments for evaluating the performance of MPLR protocol, namely, point-to-point transmission, chain network transmission, and star network transmission.

6.2.1 Point-to-point Transmission

The point-to-point experiments were performed with no introduced packet loss first, using all combinations of spreading factor and bandwidth, except for the three lowest data rate combinations. Each measurement was repeated 5 times and the average transmission time is provided in Table 6.3. Due to the stable test environment and short communication distance, the standard deviation of multiple experiments is around 0.006. To allow easier visual comparison of these results in Table 6.3, table entries with average transmission times between 40 and 60 seconds are shown with light grey shading, and those with times exceeding 60 seconds are shown with dark grey shading.

From the table, the average image transmission times are substantially lower with MPLR than with stop-and-wait. The average reduction in transmission time when using MPLR, over all settings and image sizes, was 24%, with a maximum of 56% and minimum of 8.8%. This reduction in transmission time allows for more spreading factor and bandwidth options when using MPLR, while maintaining an economical operation time and power consumption. Unfortunately, due to LoRa's small physical layer data rate, the use of most transmission combinations will take more than 40 seconds when transmitting 28 KB or more of image data, whether using MPLR or stop-and-wait.

When packet loss is introduced, both protocols degrade, but in different manners. The point-to-point transmission experiments were repeated by adding different artificial packet loss rates to evaluate the change in the transmission time of the 9 KB image for both protocols. Experiments were run with packet loss rates

Table 6.3: Transmit Times (secs) with No Packet Loss

(a) MPLR: 9 KB

	BW 500	BW 250	BW 125
SF 7	4.92	9.65	19.11
SF 8	8.54	16.90	33.62
SF 9	15.19	30.20	60.22
SF 10	27.25	54.32	
SF 11	49.42		

(b) Stop-and-wait: 9 KB

	BW 500	BW 250	BW 125
SF 7	7.85	13.14	23.72
SF 8	11.99	21.28	39.64
SF 9	19.43	36.45	73.47
SF 10	33.03	63.53	
SF 11	58.60		

(c) MPLR: 12 KB

	BW 500	BW 250	BW 125
SF 7	6.83	13.45	26.66
SF 8	11.90	22.07	46.91
SF 9	21.18	42.13	84.03
SF 10	38.80	75.79	
SF 11	68.95		

(d) Stop-and-wait: 12 KB

	BW 500	BW 250	BW 125
SF 7	11.03	18.34	33.15
SF 8	17.08	29.64	74.11
SF 9	48.34	60.51	97.81
SF 10	62.07	88.61	
SF 11	81.90		

(e) MPLR: 18 KB

	BW 500	BW 250	BW 125
SF 7	10.02	21.98	44.06
SF 8	19.97	36.95	68.54
SF 9	30.97	64.41	128.35
SF 10	55.54	110.69	
SF 11	100.72		

(f) Stop-and-wait: 18 KB

	BW 500	BW 250	BW 125
SF 7	19.33	26.77	48.29
SF 8	24.34	43.18	81.05
SF 9	59.45	73.11	152.93
SF 10	67.06	129.16	
SF 11	130.54		

(g) MPLR: 28 KB

	BW 500	BW 250	BW 125
SF 7	15.77	31.00	64.57
SF 8	29.01	56.35	109.13
SF 9	49.29	102.72	195.41
SF 10	88.42	181.72	
SF 11	164.32		

(h) Stop-and-wait: 28 KB

	BW 500	BW 250	BW 125
SF 7	25.37	42.30	90.77
SF 8	53.57	68.44	128.37
SF 9	88.51	146.40	226.21
SF 10	106.17	215.53	
SF 11	207.87		

of 2%, 5%, and 10%. Table 6.4 gives the transmission times averaged over 5 measurements for each protocol and parameter setting, with a standard deviation of 0.048. Using MPLR, transmission time increases by a percentage only a little greater than the packet loss percentage, since packet losses indicated in each BVACK do not require a timeout for detection, and since packet loss rarely increases the number of batches used to transmit an image (just the size of the last batch). However, a significant increment in transmission time is observed when using stop-and-wait, as each packet loss leads to a timeout and retransmission. The reductions in transmission time when using MPLR, averaged over all bandwidth and spreading factor settings, were 30%, 42%, and 49%, for packet loss rates of 2%, 5%, and 10%, respectively.

Table 6.4: Transmit Times (secs) with Packet Loss (9 KB image)

(a) MPLR: 2% Loss

	BW 500	BW 250	BW 125
SF 7	5.09	9.95	19.66
SF 8	8.82	17.39	34.53
SF 9	16.09	31.04	61.84
SF 10	28.01	55.78	
SF 11	50.78		

(b) Stop-and-wait: 2% loss

	BW 500	BW 250	BW 125
SF 7	11.00	16.35	28.13
SF 8	16.14	24.49	44.50
SF 9	23.85	39.98	79.73
SF 10	36.55	67.56	
SF 11	63.54		

(c) MPLR: 5% loss

	BW 500	BW 250	BW 125
SF 7	5.30	10.36	20.47
SF 8	9.18	18.11	35.97
SF 9	16.28	32.32	64.39
SF 10	29.18	58.10	
SF 11	52.88		

(d) Stop-wait: 5% loss

	BW 500	BW 250	BW 125
SF 7	19.18	22.72	35.89
SF 8	23.15	33.26	56.67
SF 9	30.92	48.25	83.75
SF 10	43.70	76.89	
SF 11	73.71		

(e) MPLR: 10% loss

	BW 500	BW 250	BW 125
SF 7	5.51	10.77	21.28
SF 8	9.54	18.83	37.40
SF 9	16.93	33.61	66.96
SF 10	30.33	60.40	
SF 11	54.98		

(f) Stop-and-wait: 10% loss

	BW 500	BW 250	BW 125
SF 7	25.45	25.45	45.25
SF 8	29.23	39.97	60.33
SF 9	35.74	56.52	93.68
SF 10	51.90	86.12	
SF 11	81.78		

The results of experiments with and without artificial packet loss show that the proposed MPLR protocol has an obvious advantage over the stop-and-wait protocol in transmission time, and the higher the packet loss rate, the more obvious the advantage is. The proposed protocol has significant advantages over Jebril's mechanism, which requires at least 1 minutes and 7 seconds to transmit an image of about 26 KB.

6.2.2 Transmission in Chain Network

In this experiment, four LoRa devices are placed on a straight line and 5 metres apart from each other to form a chain network topology. The two ends of the topology are the transmitter and receiver nodes, while the remaining two devices in the middle are the relays. The experimental plan was to relay a image of 27 KB from one end of the chain network to the other, using both the MPLR and the optimized MPLR protocol. The experiment was repeated three times to get an average. The average results of the experiments are depicted in Figure 6.1, with a standard deviation of 1.41. The dotted lines are the result of using the MPLR protocol, while the dash lines are the result of using the optimized (pipelineing) MPLR protocol.

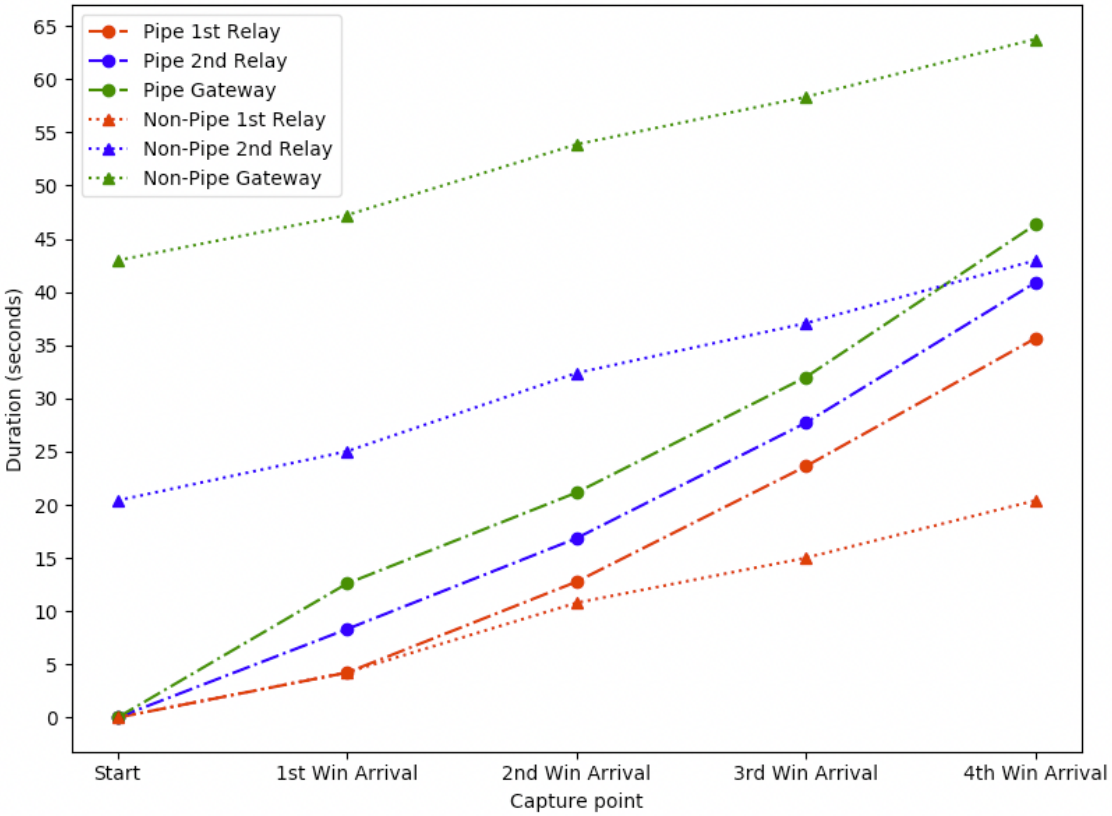


Figure 6.1: MPLR v.s. Pipelineing MPLR in 3-Hop Transmission [data size: 27 KB]

Although the first relay that in the pipelining transmission sent out every transmission window later than in the non-pipelining transmission, all the devices after the first relay benefit from the pipelining transmission. By using pipelining, all transmission windows arrived at second and third relays were earlier, and the entire transmission task was terminated earlier as well. When using pipelining transmission, the closer the device is to the gateway, the sooner the transmission task will be completed. Therefore, in this case, the use of the pipelining transmission can be used to transmit data windows in parallel in the network and accelerate the entire transmission process. By comparing the time added for each new hop, the total transmission time of the network using pipelining transmission only increased the transmission time of a single data window,

while the non-pipelining network increases the transmission time of an whole image transmission. These experimental results show that the use of pipelining transmission can effectively improve the transmission efficiency in the chain network in the MPLR protocol.

6.2.3 Transmission in Star Network

In this experiment, the main performance measurement is the transmission time for each node sending a single image to the gateway in a network with different numbers of LoRa nodes. Due to the length of the test, only a single replication was conducted, but the transmission of 25 images per node was considered sufficient to ensure measurement reliability. Figure 6.2 shows the distribution of the transmission times for each 9 KB image (using a spreading factor of 8 and bandwidth of 250 kHz) with 5, 10, 15, and 20 nodes using MPLR in conjunction with the data channel reservation protocol, and stop-and-wait in conjunction with ALOHA. The box plots only include successful transmissions, as Section 5.2.2 indicates the effect of backlog on sending.

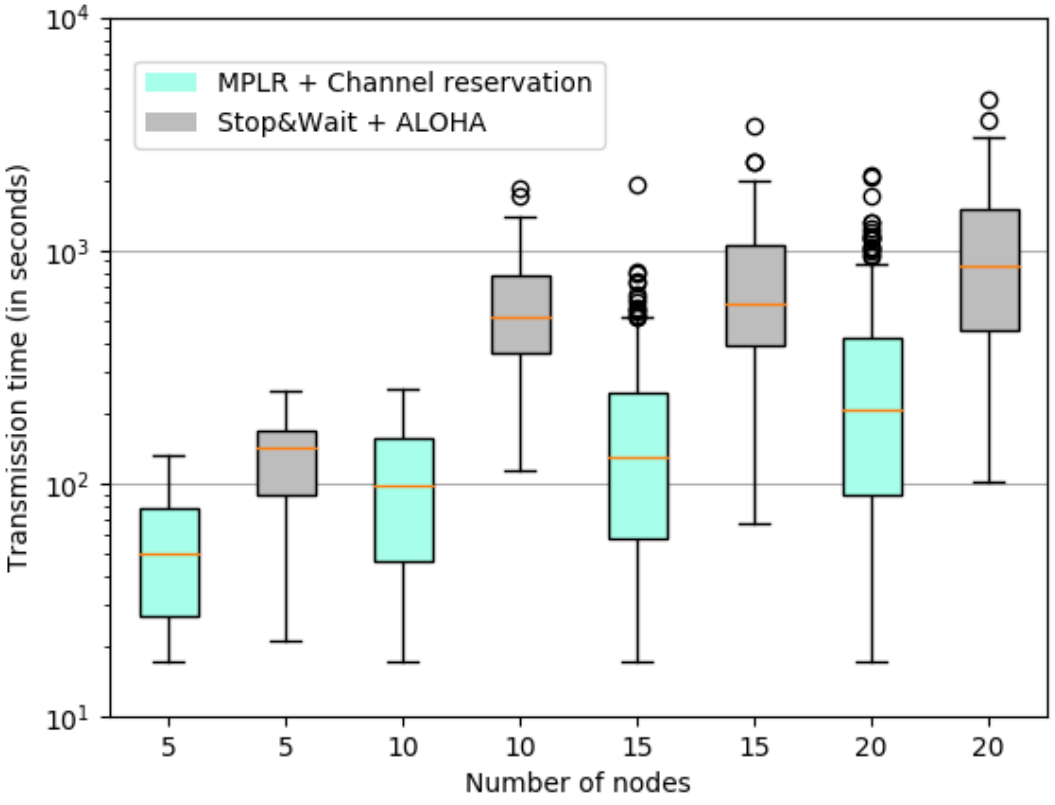


Figure 6.2: Image Transmission Time Distribution

When using MPLR+channel reservation, the average image transmission time (time from when an image transmission task is generated at a node until the transmission is successfully completed) was 55.6, 104.5, 177.8, and 304.8 seconds, with 5, 10, 15, and 20 nodes, respectively. The image from the first node to successfully reserve the data channel is always delivered after about 17 seconds, with transmissions from

other competing nodes then being delayed. When using stop-and-wait+ALOHA, the average transmission time increases greatly when the number of nodes in the network grows, due to collisions among data packets. Even if only 5 nodes are in the network, using stop-and-wait+ALOHA requires an average of 132 seconds to send an image, about 2 times longer than MPLR+channel reservation. The arrival time of the first image in each scenario is also longer with stop-and-wait+ALOHA, particularly as the number of nodes grows (17.2 to 114 seconds with 10 nodes), but dependent on the actual pattern of collisions. For 15 nodes, the stop-and-wait+ALOHA protocol has more variation than for 10 nodes, but only a 35% increase in median transmission time for a 50% increase in the number of nodes.

In the case of 5 nodes, the transmission time for the last image to be successfully received at the gateway was about 133 seconds when using MPLR+channel reservation, but was about 250 seconds with stop-and-wait+ALOHA. Scaling up to the 20 node scenario, the last image to be successfully received when using MPLR+channel reservation, within the 125 minute experiment duration, had a transmission time of about 35 minutes (since in this scenario there is substantial queuing of the transmission tasks), while the last image to be successfully received when using stop-and-wait+ALOHA had a transmission time of about 74 minutes. With 10 nodes, stop-and-wait+ALOHA was almost over 7 times slower than MPLR+channel reservation (250 vs. 1850 seconds).

Figure 6.3 shows a comparison of the average number of packet collisions during a single image transmission. As the number of nodes in the network increases, the number of packet collisions increases with both methods. However, MPLR+channel reservation produced fewer packet collisions under the same number of LoRa nodes.

Figure 6.4 depicts the number of images received by the LoRa gateway for each network size. Since an image transmission task is generated at each node every 5 minutes in these experiments, the gateway should receive in total 125, 250, 375 and 500 images, respectively. When there are only 5 nodes in the network, the gateway always receives 125 images. When the number of LoRa nodes in the network is increased to 10, only 114 images are successfully received using stop-and-wait+ALOHA, prior to the experiment termination after 125 minutes, due to congestion and corresponding packet loss; the gateway receives all images in the network using MPLR+channel reservation.

Similarly, when the number of nodes in the network is increased to 15 and 20, when using MPLR+channel reservation the gateway receives 334 and 353 images, respectively, while with stop-and-wait+ALOHA the number of delivered images remains at approximately 120. With MPLR+channel reservation, the network reaches a plateau more gradually than with stop-and-wait+ALOHA. Although this transmission frequency is much higher than the anticipated frequency of demand in actual projects, it allows us to determine the capacity when using MPLR+channel reservation.

The image delivery fairness performance difference between MPLR+channel reservation and stop-and-wait+ALOHA is presented in Figure 6.5. As the number of nodes increases beyond 5, they are differentially affected by the added traffic and potential congestion. With stop-and-wait+ALOHA and 20 nodes, one node

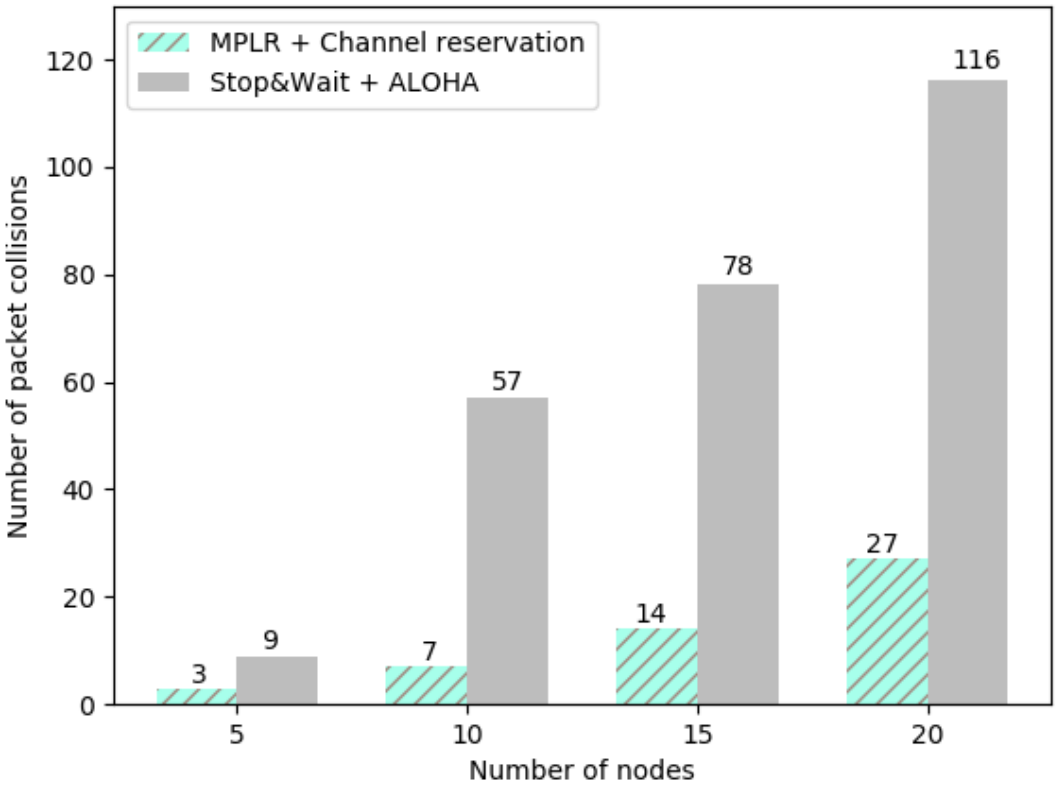


Figure 6.3: Packet Collisions

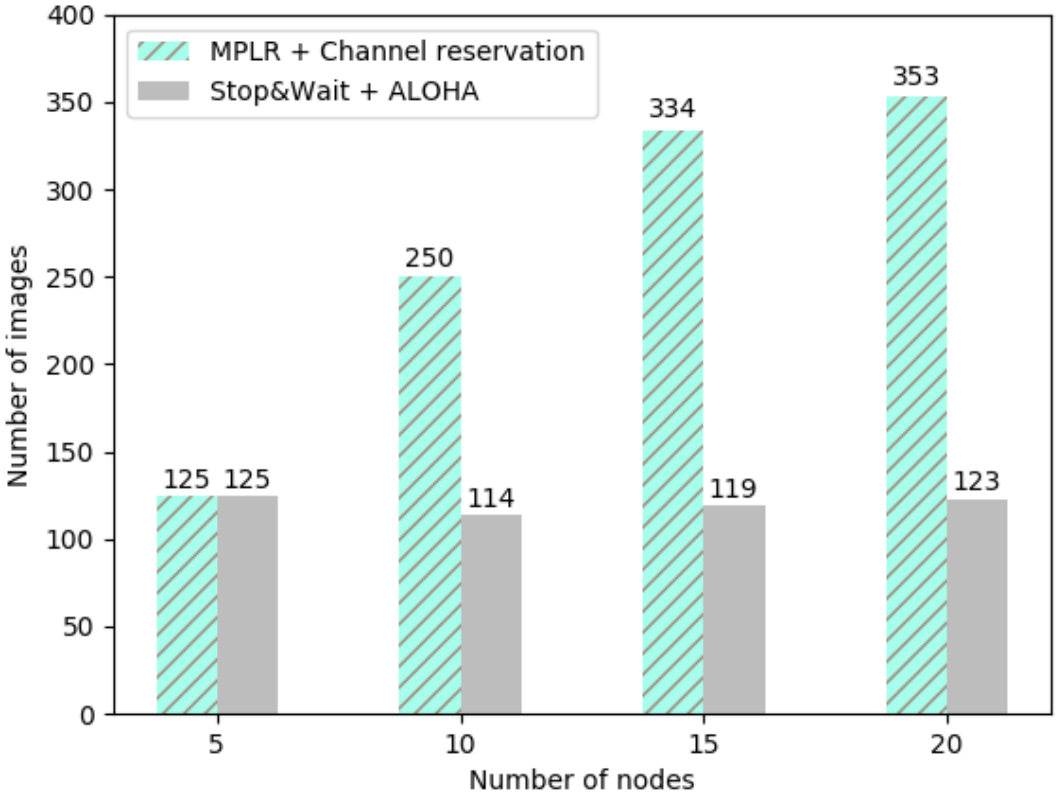


Figure 6.4: Successful Image Transmissions

is limited to one successful image transmission. This is 25% of the median for this configuration, indicating there are other factors that influence which nodes suffer most from collision.

Under MPLR+channel reservation, the distribution of the number of images successfully delivered for each node is considerably more condensed. 50% of the nodes are no more than 2 images off from the median with 20 nodes and the range is even tighter for 15 nodes. The most extreme outlier for the 20-node scenario can still deliver 13 images (72% of the median).

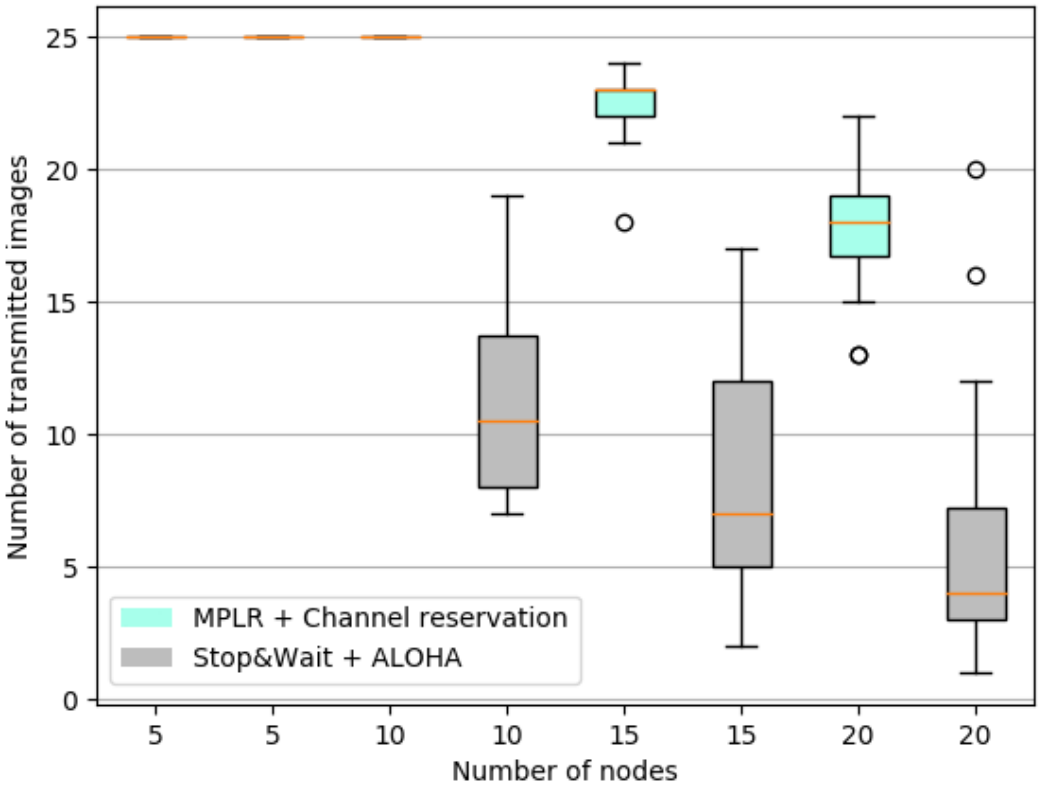


Figure 6.5: Inter-Node Fairness

6.3 MHLR

Three experiments were conducted to provide training and testing data for MHLR protocol and to evaluate the performance of MHLR protocol. In this section, the experimental results are discussed and analyzed in detail to verify the possibility of using machine learning to form network topology and select route.

6.3.1 Training Data Collection and Learning

Through extensive laboratory experiments, a large number of routing information in different topologies was collected. This information can not only be used to train the machine learning model, but also as preliminary verification of the feasibility of using machine learning to select routes and the correctness of the selected

features.

Firstly, each routing information is paired with all other routing information one by one to form the subtraction vectors by using Equation 4.2, and the resulted vectors are fed into five different machine learning models for learning. It is worth mentioning that in the actual routing, when the transmission time difference between two routes is very close, then either route can be selected for transmission. Therefore, when using the machine learning model for prediction, 4 thresholds were used, namely 0%, 5%, 10% and 15%. When the transmission time difference is less than the threshold value, the both routes are considered to be the same and excluded from the data set. This is called **fuzzy classification**, which can be used to test whether learning efficiency can be improved when different degrees of similar data are ignored. Table 6.5 summarizes the accuracy of machine learning methods for the collected routing information.

Table 6.5: Training Data Learning Score

Algorithm	Accuracy (>= 0%)	Accuracy (>= 5%)	Accuracy (>= 10%)	Accuracy (>= 15%)
MLP	90.2%	92.2%	93.9%	95.3%
SGD	90.1%	91.9%	93.6%	95.2%
Decision Tree	85.2%	86.8%	88.9%	90.7%
Random Forest	86.6%	89.0%	90.5%	91.2%
Ada Boost	89.9%	91.8%	93.4%	95.0%

As can be seen from the table, even without fuzzy classification, the accuracy of all machine learning methods is more than 85%, among which MLP has the highest accuracy of 90.18%. When the degree of fuzzy classification increases by 5%, the accuracy increases by 1% to 2%. Therefore, it can be proved that the routing information represented by the selected features can be learned by machine learning methods and has a high accuracy.

Figure 6.6 shows a classification report of training data set in five models. In the figure, *1st Route* represents the training data labeled 1 and *2nd Route* represents the training data labeled 0. Almost all classifiers have higher precision and recall when predicting label 1 than predicting label 0, which means more label 1 data is correctly predicated. Random forest, however, has a higher precision when predicting label 0. This is also evident in the f1 results. The reason for this is that in the training data set, the occurrence (support) of label 1 is almost 2.5 times higher than label 0, which makes the models learn more and better about label 1. Thus, a lower precision and recall are presented when predicting label 0 due to insufficient training data set. In any case, the accuracy of precision and recall of most models for both classes exceeds 85%, which is considered to be a high accuracy rate.

The evaluation of accuracy on the training data set shows the ability to grasp the known data, but cannot test the ability to predict the unknown data. In order to understand the predictive power of those machine learning models, the method of cross-validation should be used to verify it. During cross-validation, the training data set is divided into five equal parts, one part as testing data set and the other part as training

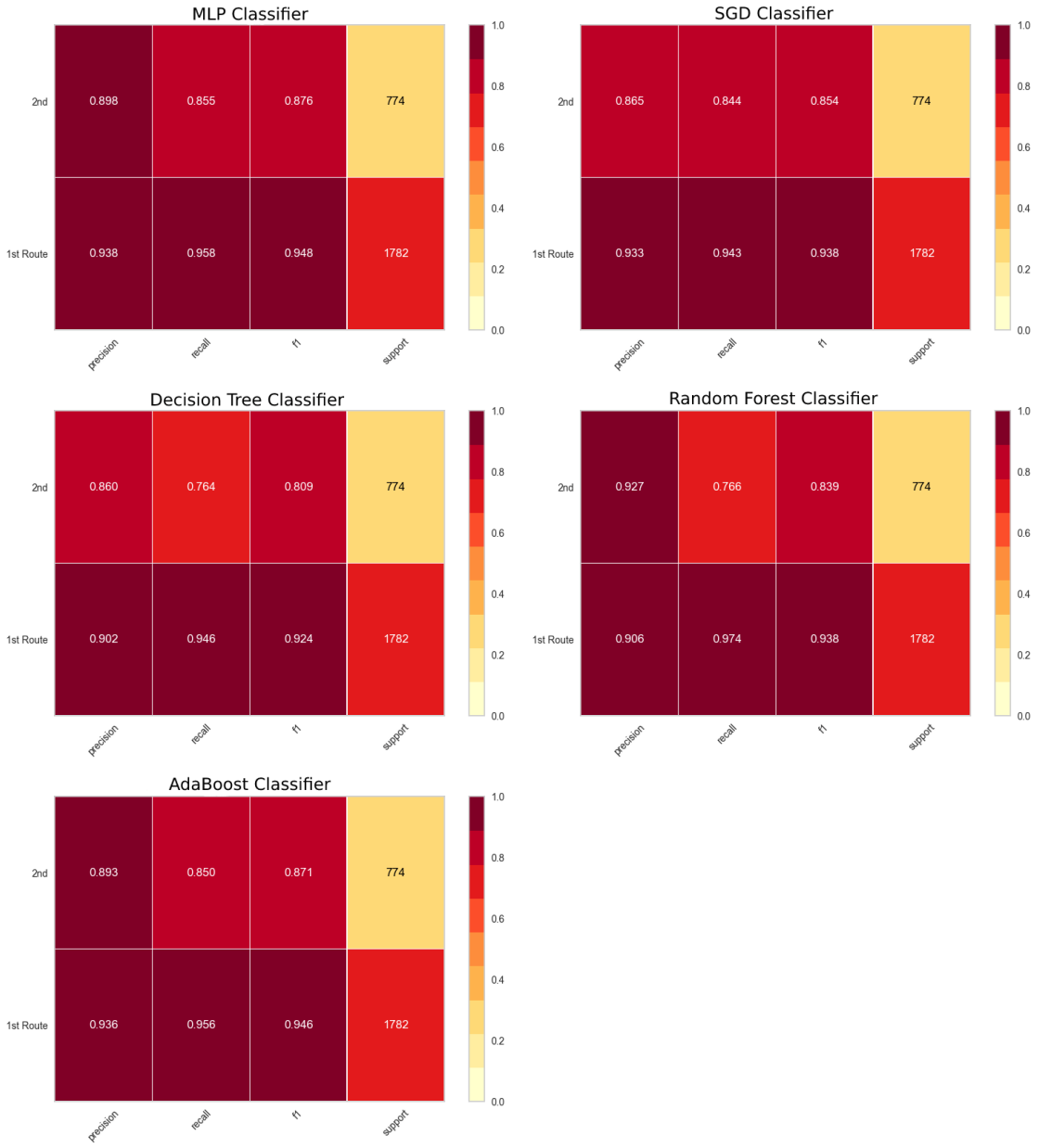


Figure 6.6: Classification Report For Training Data Set (Fuzzy Classification = 0%)

data set. The results are shown in Table 6.6.

Table 6.6: Training Data Set Cross-Validation Accuracy

Algorithm	Accuracy (>= 0%)	Accuracy (>= 5%)	Accuracy (>= 10%)	Accuracy (>= 15%)
MLP	88.9% (+/- 3%)	90.6% (+/- 3%)	92.6% (+/- 3%)	94.1% (+/- 3%)
SGD	89.4% (+/- 3%)	90.6% (+/- 3%)	92.6% (+/- 3%)	94.3% (+/- 2%)
Decision Tree	84.0% (+/- 3%)	86.2% (+/- 3%)	87.6% (+/- 2%)	89.6% (+/- 2%)
Random Forest	85.1% (+/- 3%)	87.3% (+/- 3%)	89.5% (+/- 3%)	90.6% (+/- 4%)
Ada Boost	88.6% (+/- 2%)	90.5% (+/- 2%)	92.5% (+/- 2%)	94.2% (+/- 2%)

Although the accuracy on cross-validation is not as high as on the training data set, the accuracy still reaches more than 85% without using fuzzy classification, except decision tree. Similarly, when the degree of fuzzy classification is increased by 5%, the prediction accuracy is improved by 1-2%. The standard margin of error for most forecasts is around 3%. In general, most machine learning methods are good predictors of routing performance.

Figure 6.7 shows the classification report of the cross-validation results. Several phenomena presented in Figure 6.6 are also reflected in this result. Although in this case it was to predict unknown data instances, the precision and recall results of cross-validation were not significantly degraded. This also indicates that the model has good generics.

In order to better understand the learning ability of each machine learning model, a receiver operating characteristic curve is drawn to facilitate us to know the true positive rate and false positive rate of the trained model. Figure 6.8 shows the ROC curves of five machine learning models. The blue dotted line is the neutral (reference) line for random guesses with an accuracy rate of 50%. The left side of the neutral line predicts better results, and the right side predicts worse results. All machine learning models are on the left side of the neutral line, near the top left vertex, indicating that all these methods conclude good prediction results. Ada Boost, MLP and SGD were the best, while Random Forest and Decision Tree were slightly worse.

Figure 6.9 shows the training curves of the five models for the training data set. Among them, SGD, Decision Tree, and AdaBoost classifiers achieved good convergence after learning the training data set. MLP and Random Forest classifiers have the accuracy gaps of 4-5% between training data and validation data. This indicates that MLP and Random Forest classifiers fit the known data well, but the ability of the generalization is not at its best. The solution to this problem is to provide more training data so that the model can learn better. In addition, the cross-validation scores of Decision Tree and Random Forest classifiers increased with the increase of the number of training instances, while all other classifiers were in a relatively flat learning trend. This indicates that the cross-validation scores of both Decision Tree and Random Forest classifiers may improve when more training instances are added. On the contrary, these additional training instances will not help the cross-validation scores of other classifiers. As the accuracy rate has been higher than expected

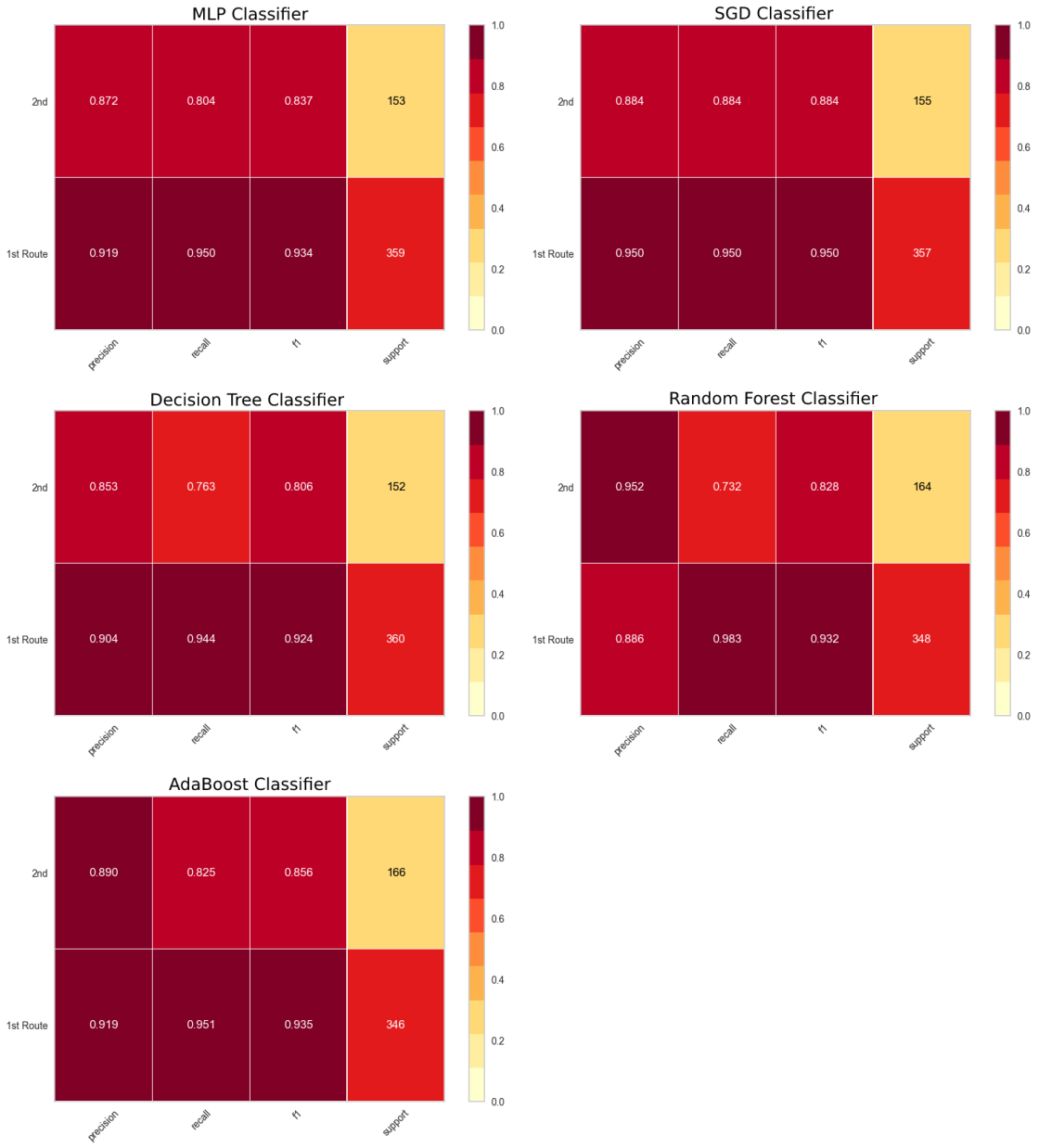


Figure 6.7: Classification Report For Cross-Validation (Fuzzy Classification = 0%)

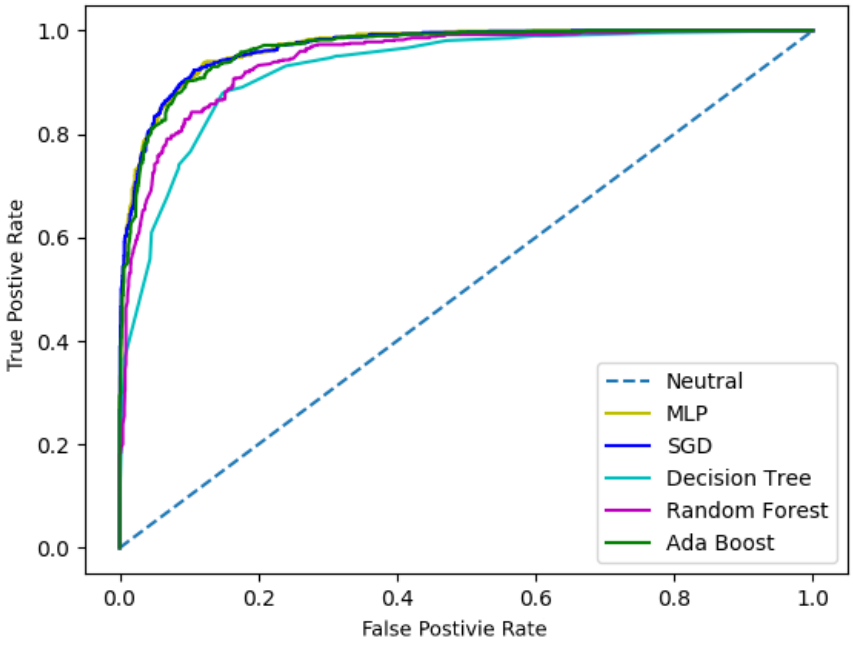


Figure 6.8: The ROC for Training Data Set (Fuzzy Classification = 0%)

(85%), there were no plans to collect more training data. That could be part of future work.

The sizes of the models after training are varied, and the size of each model after learning the training data set is listed in Table 6.7. The size of SGD classifier is the smallest, only 1 KB, and the size of MLP and Decision Tree classifier are 5 KB each. Due to their small size, these three classification models will not cause too much overhead when distributed to each node device. However, the sizes of Random Forest and AdaBoost classifiers are 44 KB and 31 KB, respectively. Passing such a large size through the LoRa network incurs additional overhead and affects the throughput of the network. Therefore, these two classifiers are not recommended in actual deployment if the classification model needs to be updated.

Table 6.7: The File Size of the Trained Models

Name	Size (KB)
MLP	5
SGD	1
Decision Tree	5
Random Forest	44
AdaBoost	31

Figure 6.9: Learning Curve of Training Data (Fuzzy Classification = 0%)

6.3.2 Indoor Testing

The indoor experiments show that the machine learning model trained with laboratory training data set can accurately predict the better route for the deployed LoRa multi-hop network in indoor environment. Similarly, the multi-hop transmission data of different topologies in indoor environment was collected, and Formula 4.2 was used to convert it into the input subtraction vector for machine learning methods to predict. These machine learning models have been trained with laboratory training data set prior to prediction. The prediction results are shown in Table 6.8.

Table 6.8: Indoor Routing Prediction

Algorithm	Accuracy (>= 0%)	Accuracy (>= 5%)	Accuracy (>= 10%)	Accuracy (>= 15%)
MLP	91.1%	92.6%	93.8%	95.7%
SGD	90.8%	93.3%	94.3%	95.8%
Decision Tree	86.2%	87.0%	90.1%	91.9%
Random Forest	87.9%	88.5%	90.1%	91.6%
Ada Boost	91.0%	92.7%	93.4%	95.1%

All machine learning models trained with laboratory data can accurately predict better routes in indoor environment. Even without fuzzy classification, the lowest performing model, Decision Tree, was 86.2% accurate, while the best-performing model, MLP, was 91.1% accurate. Similarly, as the degree of fuzzy classification increases, the accuracy of all algorithms also improves. It is noteworthy that the average accuracy in Table 6.8 was slightly higher than that in Table 6.6. One reason is that in this indoor deployment, some of the actual topologies formed are not as complex as those designed in the laboratory, so predictions are easier and results more accurate. The experimental results show that the machine learning model trained with laboratory data can predict the path in indoor environment accurately.

Figure 6.10 shows a classified report of indoor routing data. Although the data volume of label 1 and label 0 has been increased, the precision and recall of label 1 are still higher because the data volume of label 1 is twice as large. From the performance of all models, although the deployment changes in indoor environment have an impact on the prediction results of models, the precision and recall of most models remain above 80%. This score can be further increased if incremental learning is used to dynamically learn changes in routing conditions.

Figure 6.11 shows an ROC curve for prediction of indoor deployment data by five machine learning models. The results are not surprising and all the models have higher prediction accuracy, so all the curves fall to the left of the neutral line, very close to the top left. This further illustrates the high learning rate and selected features of the machine learning methods.

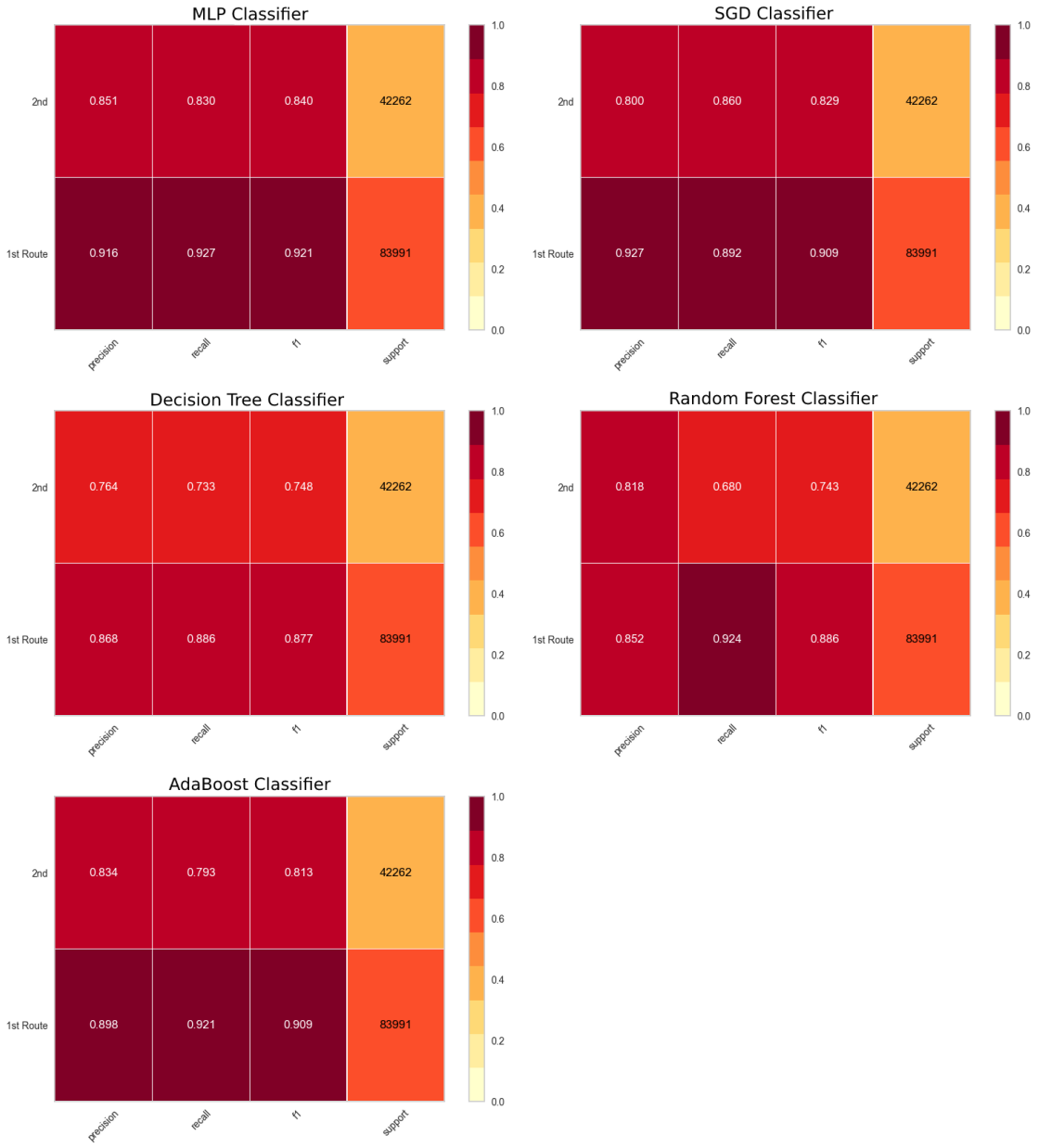


Figure 6.10: Classification Report For Indoor Routing (Fuzzy Classification = 0%)

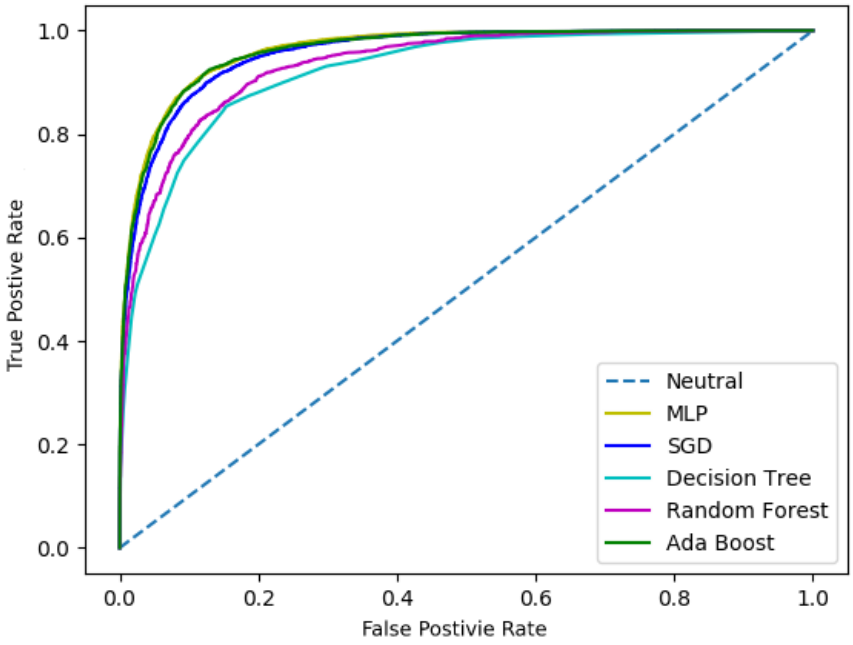


Figure 6.11: The ROC for Indoor Routing (Fuzzy Classification = 0%)

6.3.3 Outdoor Testing

So far, all experimental results show that the proposed protocol can accurately choose the routing in an indoor environment. Next, its performance in topology formation and routing selection in outdoor environment is analyzed. In the outdoor experiment, 10 devices were deployed in a low-density residential area to transmit images to the LoRa gateway through multi-hop transmission. Figure 6.12 shows the location of the device deployment and the results of network topology are automatically formed by the trained MPL model.

In Figure 6.12, the red dot is the LoRa gateway, the yellow dot is a relay device, and the green dot is a node device. Other gray dots are nodes that attempt and fail to connect to relay devices/gateways due to obstacles or power lines. Through the prediction of the MPL model trained by training data set, the whole network establishes a tree network topology with gateway as the root node. In this network, each node chooses to connect directly to the gateway or chooses a node near the gateway for retransmission. Nodes do not choose nodes that are further away from the gateway for propagation. For example, node 3 does not choose node 2 which is far away from the gateway for relaying, but chooses node 4. This is also because node 4 is only one hop away from the gateway and node 2 is two hops away from the gateway. Alternatively, nodes can achieve the fastest transmission by reducing the number of relays. For example, node 9 chooses node 7 over node 8 for relaying because node 8 requires one more relay than node 7, increasing the time it takes for data to reach the gateway. Finally, in the case of three hops, node 1 transmits the image back to the gateway through two relays of node 2 and node 4. The outdoor experiment proves that LoRa multi-hop

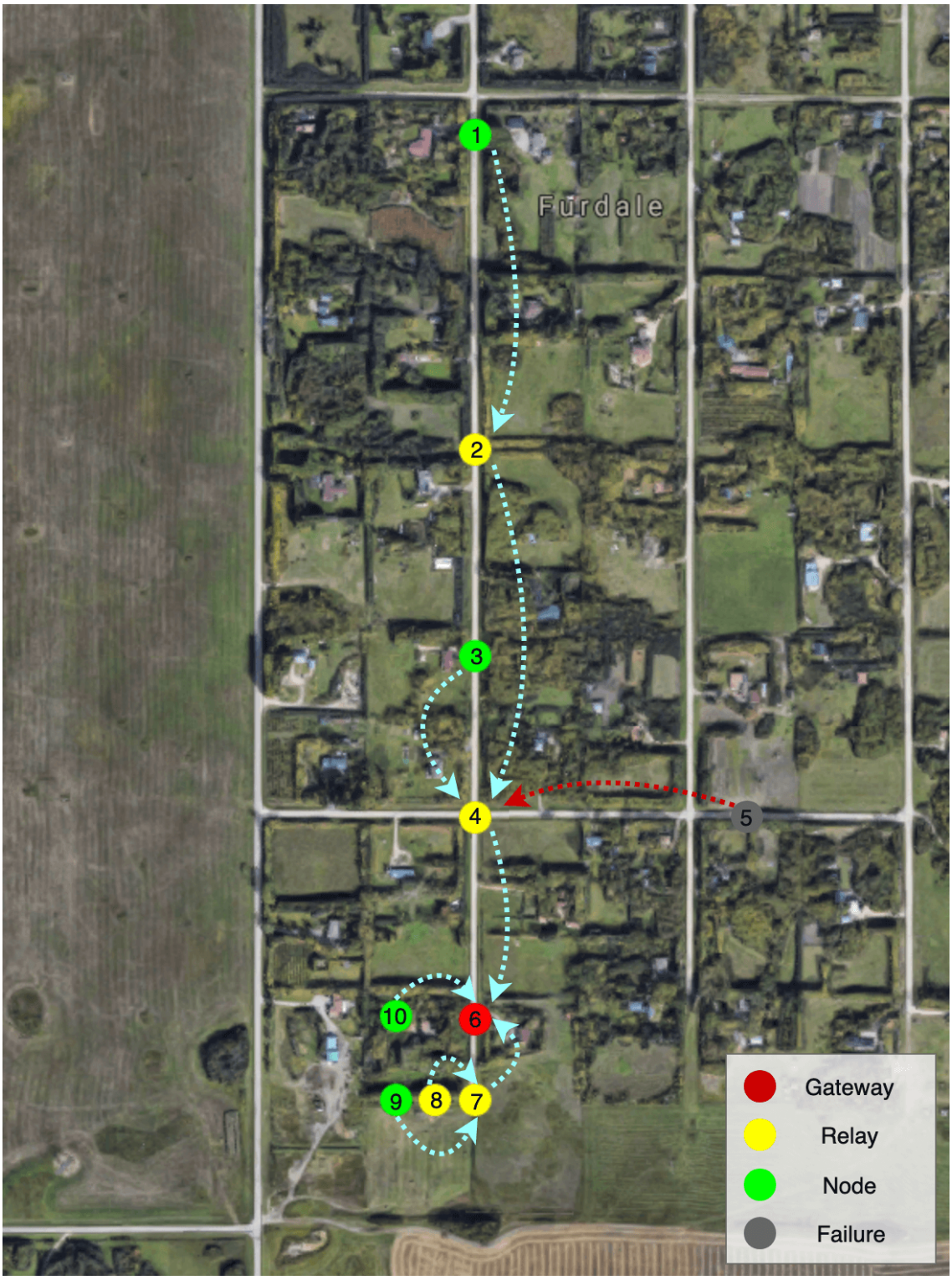


Figure 6.12: Outdoor Experiment Topology

network can transmit image information in an outdoor environment.

In this experiment, the transmission time, throughput, average retransmission times, and establishment delay are also recorded in the log file. Through the analysis of log files, it is found that each node in this experiment sends 3-5 images on average. Data is averaged and summarized in Table 6.9.

Table 6.9: Transmission Time, Throughput, Re-transmission, Establishment Delay Between Nodes

Path	Trans. Time (second)	Throughput (bps)	Avg. Re-trans.	Establishment Delay (second)
$1 -> 2$	60.69	151.85	3	4.17
$2 -> 4$	62.34	147.83	2	4.17
$3 -> 4$	59.25	155.54	1	4.17
$4 -> 6$	54.35	169.57	0	4.17
$7 -> 6$	54.50	169.10	0	4.17
$9 -> 7$	54.34	169.60	1	44.90
$10 -> 6$	54.34	169.60	0	4.17

Except for the connections with retransmission and long-distance, the outdoor transmission performance is very close to the optimal transmission time in the laboratory environment given in Table 6.3. This indicates that LoRa multi-hop network is feasible for image transmission under optimistic outdoor environment and experimental verification in the laboratory. Due to the longer transmission distance, the transmission signal becomes unstable and packet loss occurs. Thus, the transmission time of nodes 3 and 4, 2 and 4, as well as node 2 and 1 has been extended to varying degrees, and the throughput on the connection has been reduced. However, with the help of the MPLR protocol, the time required for retransmissions is reduced and does not increase as significantly as with the stop-and-wait. Moreover, most nodes only have a establishment delay of about 4.17 seconds when joining the network, except when node 9 wants to join the network through node 7, the delay increases to 44.9 seconds. After analyzing the device log, it was found that the reason for such a long establishment delay was that node 9 first established a connection with node 8, since the request packet from node 9 was not received by node 7. When node 9 sends heartbeat packets to adjust the network, it finds that node 7 is the more suitable parent node. Therefore, the whole process took 44.9 seconds.

By analyzing the results of the outdoor experiment, it was shown that it is feasible to realize automatic topology construction and routing selection in LoRa network by using machine learning. In addition, the image transmission in outdoor environment is roughly the same as that in indoor environment, except that there is more environmental noise in outdoor environment, which leads to packet loss and thus retransmission. Through the analysis of establishment delay, it can be found that the parent node selection based on machine learning can be completed in a short time, while any non-optimal routing can be automatically adjusted after a fixed amount of time. Unfortunately, comparisons with conventional methods of topology and route construction are not planned for this thesis, but is part of future work. However, experimental results from a number of relevant studies indicate that the use of machine learning performs better than the conventional

methods in terms of network topology and route construction [28] [82] [39].

6.4 Summary

The results of seven different experiments are discussed and analyzed in detail in this chapter. The maximum payload size under different combinations of spreading factor and bandwidth was first examined and found that the payload size of 255 bytes is fairly stable in most settings, except for SF 12-BW 250, SF 12-BW 125, and SF 11-BW 125. These exceptions require smaller payload sizes to maintain transmission reliability. The performance of the MPLR is then evaluated against the transmission time. The results show that the MPLR protocol has an obvious advantage in transmission time regardless of packet loss, which is 24% less than the transmission time of stop-and-wait protocol on average. Besides, the MPLR protocol has been shown to be able to optimize the use of chain networks. By cooperating with the channel reservation protocol, the proposed LoRa network is superior to the state-of-the-art LoRa network in terms of transmission time distribution, packet collision, delivery rate and fairness. The MHLR protocol is then tested in three different experiments to demonstrate the possibility of using machine learning to construct network topologies and select routes. By using appropriate machine learning method and degree of fuzzy classification, the learning results of training data set and indoor experiment data set reach more than 90% accuracy. In the outdoor low-density environment, the results show that MHLR can effectively establish a high-quality tree topology and accurately select better route with the help of machine learning.

Chapter 7

Conclusions

7.1 Thesis Summary

This thesis proposes two novel and lightweight protocols to prove the LoRa's possibilities in image transmission applications and multi-hop topologies. By reducing the transmission time, the proposed protocols can transmit image data from the node to the gateway in a multi-hop manner faster and more efficient. The two proposed protocols, namely Multi-Packet LoRa (MPLR) and Multi-Hop LoRa (MHLR), provide assistance in improving transport performance and precise routing paths, respectively.

A lightweight reliable delivery protocol called MPLR was first designed and implemented for image transmission in LoRa to enable remote image monitoring in our agricultural IoT system. MPLR batches data packet transmissions and uses bit-vector acknowledgements so as to greatly reduce the number of required acknowledgement packets and the time spent waiting for them. To avoid packet collision caused by congestion, data channel reservation is used so that the request packet and data packet can be transmitted on different channels to ensure the sustained successful data transmission rate. MHLR realizes efficient routing in LoRa multi-hop transmission by utilizing the power of machine learning algorithms, and proves that machine learning is sufficient for routing selection in wireless sensor networks. Compared with conventional routing algorithms, routing protocols using machine learning algorithms can accurately predict the disparity of two routes by learning historical routing data by themselves, without too much manual calculation and thinking.

Several laboratory tests were performed first to understand LoRa's ability to transmit large messages. It is found that the maximum reliable transmission payload size varies when using different spreading factor and bandwidth. Most SF and BW settings can transmit data up to 255 bytes in a single packet, while when using SF 11-BW 125, SF 12-BW 250, and SF 12-BW 125, only up to 149 bytes, 137 bytes, and 41 bytes can be transmitted stably, respectively. Both protocols are implemented and evaluated using a LoRa testbed network in both laboratory and outdoor environment. The results show that using MPLR for image transmission in point-to-point communication can reduce the time by an average of 24%. When packet loss was introduced, the average transmission time is at least 30% improved over stop-and-wait and as much as 49% improvement is experienced (for 10% packet loss). In the star network experiments, a shorter average transmission time was experienced with MPLR+channel reservation, and there is no increase in delivery

time of the first image with increased network density. Compared with stop-and-wait+ALOHA method, MPLR+channel reservation achieves a higher fairness among the participating nodes.

Results from both training data learning and indoor test show that the selected features can represent the key information of the routing paths well and provide valuable insights for the machine learning algorithms. Through the learning of this information, most of the machine learning algorithms can achieve more than 85% accuracy, whether using learning score or cross-validation verification. The best algorithm can achieve more than 90% accuracy. In the outdoor experiment, the tree topology established by the trained multi-layer perceptron model is evidenced, and each node chooses the best parent node to build the network, without additional hops. Even in the outdoor experiment, the throughput of the network remained similar to the results of laboratory tests. In addition, the average connection establishment delay remained at 4.17 seconds, and non-optimal connection can be automatically adjusted within one minute when the frequency of the heartbeat packet is 30 seconds.

7.2 Contribution

The study described in this thesis is an attempt to use LoRa for long-range image transmission and utilize machine learning for LoRa network construction. It has the following contributions to the community:

- **Design and implement a transport protocol for image transmission in LoRa.** This study is the first to improve the performance of LoRa in multi-packet transmission by designing and implementing a transport protocol. The proposed protocol effectively reduces the time required for multi-packet transmission over LoRa networks. This not only reduces the power consumption of the devices, extends the network lifetime, but also offers LoRa much more possibilities in application.

- **Provide channel reservation mechanism to avoid inter-flow interference in LoRa network.** The proposed protocol introduces channel reservation into the LoRa network to solve the data packet collision problem caused by the use of pure-ALOHA protocol. Such method can significantly reduce the data packet collision rate in LoRa network, make the transmission of multi-packet message more stable, and avoid wasteful transmission.

- **Prove the possibility of automatic LoRa multi-hop network construction and routing through machine learning.** This study also attempts to automatically construct LoRa-based multi-hop network and select optimal routing path by learning routing characteristics using machine learning models. The results of both indoor and outdoor experiments show that the proposed mechanism is feasible and acquires a high degree of accuracy in routing.

7.3 Future Work

The proposed work can be improved in several different aspects. Several key future works are illustrated in the following sections.

7.3.1 Power Consumption

Due to equipment limitations, the LoRa device used in the experiment was unable to record power consumption. As a result, the power consumption of devices cannot be referred to in routing. However, in many routing protocols, power consumption is considered as one of the important factors that determine the performance of wireless sensor networks, and is also considered as one of the important routing characteristics. If a routing protocol does not take into account power consumption and the estimated residual electricity of the device, devices on a popular route may be used so frequently that the battery will be exhausted early. On the contrary, if the power consumption and residual electricity of the device are considered and observed during the routing selection, the routing protocol can choose the node with low historical power consumption for relay, so as to avoid frequent use of the device with low residual electricity and extend the network life. Therefore, in order to maintain the nature of the low power consumption of LoRa, it needs to be further studied.

7.3.2 Extensive Deployment

In Chapter 6, several experiments were conducted in laboratory, Spinks Lab, and small outdoor areas, respectively. The results of the experiments, while proving that the proposed work was effective, are insufficient compared to real deployment environments, such as large farms. Therefore, it is necessary to further verify the reliability and scalability of the proposed work by deploying it to a real and large-scale farm environment.

7.3.3 Advanced Compression

In addition to JPEG, there are many other suitable image compression algorithms that can further reduce the amount of bits needs to be transmitted and improve the transmission efficiency. For example, JPEG 2000, the next generation of JPEG, has a better compression performance with improved image quality. Moreover, in addition to image compression algorithm, video compression algorithm can also be applied in proposed scenario. In fact, video is composed of multiple consecutive images. When the consecutive images do not change, there is no image data to be transmitted. When changes occur, it only need to transfer the locally changed new image data to the gateway, rather than sending the entire new image. This is better than sending the whole picture every time.

7.3.4 Security

In the proposed work, security is not taken into account, although this is very important in actual deployment. A wireless sensor network without security mechanism will be vulnerable to malicious attacks, data leakage, data tampering, and other threats. Therefore, appropriate security measures, such as information encryption and access restrictions, need to be incorporated into the protocol to ensure that the WSN is safe from threats and attacks.

7.3.5 Multi-Channel Gateway

In the proposed work, the LoRa gateway is assumed to only send or listen on one channel at a time. Most of the LoRa gateways on the market are based on SX1301 chips and can send or listen to eight different channels at the same time. The next step could be to adjust the proposed protocol appropriately and apply it to a multi-channel gateway. At the same time, check whether the proposed protocol will also improve the transmission efficiency with the improvement of gateway capacity.

References

[1] Ferran Adelantado, Xavier Vilajosana, Pere Tuset-Peiro, Borja Martinez, Joan Melia-Segui, and Thomas Watteyne. Understanding the Limits of LoRaWAN. *IEEE Communications Magazine*, 55(9):34–40, 2017.

[2] Godfrey Anuga Akpakwu, Bruno J. Silva, Gerhard P. Hancke, and Adnan M. Abu-Mahfouz. A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges. *IEEE Access*, 6:3619–3647, 2018.

[3] Mohammad Abu Alsheikh, Shaowei Lin, Dusit Niyato, and Hwee-Pink Tan. Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communications Surveys Tutorials*, 16(4):1996–2018, 2014.

[4] Mutasem Khalil Alsmadi, Khairuddin Bin Omar, Shahrul Azman Noah, and Ibrahim Almarashdah. Performance Comparison of Multi-layer Perceptron (Back Propagation, Delta Rule and Perceptron) algorithms in Neural Networks. In *IEEE International Advance Computing Conference*, pages 296–299, Patiala, India, March 2009.

[5] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, October 2010.

[6] Aloÿs Augustin, Jiazi Yi, Thomas Clausen, and William Mark Townsley. A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. *Sensors (Basel)*, 16(9, 1466):1–22, September 2016.

[7] Eyuel D. Ayele, Kallol Das, Nirvana Meratnia, and Paul J.M. Havinga. Leveraging BLE and LoRa in IoT network for wildlife monitoring system (WMS). In *IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 342–348, Singapore, Singapore, February 2018.

[8] Milija Bajceta, Petar Sekulić, Bozo Krstajic, Slobodan Đukanović, and Tomo Popovic. A Private IoT Cloud Platform for Precision Agriculture and Ecological Monitoring. In *13th International Conference on Electrical, Electronic and Computing Engineering*, page 2, Zlatibor, Serbia, June 2016.

[9] Nick Baker. Zigbee and Bluetooth. *Computing & Control Engineering*, 16(2):20 – 25, 2005.

[10] Orlando Cabral, Fernando Velez, Albena Mihovska, Alberto Segarra, and N. Kalyana Rama Prasad. Optimization of multi-service IEEE802.11e block acknowledgement. In *IEEE Radio and Wireless Symposium*, pages 380–383, San Diego, CA, January 2009.

[11] Marco Centenaro, Lorenzo Vangelista, Andrea Zanella, and Michele Zorzi. Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios. *IEEE Wireless Communications*, 23(5):60–67, October 2016.

[12] Phui San Cheong, Johan Bergs, Chris Hawinkel, and Jeroen Famaey. Comparison of LoRaWAN classes and their power consumption. In *IEEE Symposium on Communications and Vehicular Technology (SCVT)*, pages 1–6, Leuven, Belgium, November 2017.

[13] Per Christensson. Network topology definition. https://techterms.com/definition/networktopology, October 2007. Accessed: 2019-06-30.

[14] Jordi Carrabina David Castells-Rufas, Adrià Galin-Pons. The Regulation of Unlicensed Sub-GHz bands: Are Stronger Restrictions Required for LPWAN-based IoT Success? *CoRR*, abs/1812.00031, 2018.

[15] Kauê Vinicius de Oliveira, Henri M. Esgalha Castelli, Sidney José Montebeller, and Thais G. Prado Avancini. Wireless Sensor Network for Smart Agriculture using ZigBee Protocol. In *IEEE First Summer School on Smart Cities (S3C)*, pages 61–66, Natal, Brazil, August 2017.

[16] Chibuzor Edordu and Lionel Sacks. Self Organising Wireless Sensor Networks as A Land Management Tool in Developing Countries: A Preliminary Survey. In *Proceedings of the London Communications Symposium*, pages 1–4, London, UK, January 2006.

[17] Yoav Freund and Robert E. Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, August 1997.

[18] Erich Gamma, John Vlissides, Ralph Johnson Dr, and Richard Helm. *Design Patterns CD: Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[19] Sakshama Ghoslya. All About LoRa and LoRaWAN, 2017. http://www.sghoslya.com/p/lora-is-chirp-spread-spectrum.html.

[20] K. Gopavanitha and Sholipuram Nagaraju. A low cost system for real time water quality monitoring and controlling using IoT. In *International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pages 3227–3229, Chennai, India, August 2017.

[21] David Groth and Toby Skandier. *Network+ Study Guide, 4th Edition*. SYBEX Inc., Alameda, CA, 2005.

[22] Yu Gu and Tiaobin Jing. The IOT Research in Supply Chain Management of Fresh Agricultural Products. In *2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC)*, pages 7382–7385, Dengleng, China, August 2011.

[23] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition, 2017.

[24] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, pages 10 pp. vol.2–, Maui, HI, January 2000.

[25] Tadaaki Hirata, Keitaro Terada, Masaharu Toyota, Yuya Takada, Keiko Matsumoto, and Mikiko Sode Tanaka. Proposal of a Power Saving Network for Rice Fields Using LoRa. In *IEEE 6th Global Conference on Consumer Electronics (GCCE)*, pages 1–4, Nagoya, Japan, October 2017.

[26] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, Montreal, Quebec, Canada, August 1995.

[27] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, August 1998.

[28] Nishtha Hooda, Seema Bawa, and Prashant Singh Rana. MCTOPE Ensemble Machine Learning Framework: A Case Study of Routing Protocol Prediction. In *IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, pages 92–99, Kathmandu, Nepal, October 2018.

[29] Ivan Howitt and Jose A. Gutierrez. IEEE 802.15.4 Low Rate - Wireless Personal Area Network Coexistence Issues. In *IEEE Wireless Communications and Networking*, volume 3, pages 1481–1486, New Orleans, LA, March 2003.

[30] Tiansi Hu and Yunsi Fei. An Adaptive and Energy-efficient Routing Protocol Based on Machine Learning for Underwater Delay Tolerant Networks. In *IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 381–384, Miami Beach, FL, August 2010.

[31] IEEE. IEEE Standard for Information technology–Local and metropolitan area networks–Specific requirements–Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements. *IEEE Std 802.11e-2005 (Amendment to IEEE Std 802.11, 1999 Edition (Reaff 2003)*, pages 1–212, November 2005.

[32] Innovation, Science and Economic Development Canada. Digital Transmission Systems (DTSs), Frequency Hopping Systems (FHSs) and Licence-Exempt Local Area Network (LE-LAN) Devices, 2017.

[33] Akram H. Jebril, Aduwati Sali, Alyani Ismail, and Mohd Fadlee A. Rasid. Overcoming Limitations of LoRa Physical Layer in Image Transmission. *Sensors (Basel)*, 18(10, 3257):1–22, September 2018.

[34] Mukund Jha, Amarjeet Singh Yumnam, and Dilip Raju. Comparison between image codecs: JPEG and JPEG2000. In *International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 532–535, New Delhi, India, March 2014.

[35] Mookeun Ji, Juyeon Yoon, Jeongwoo Choo, Minki Jang, and Anthony Smith. LoRa-based Visual Monitoring Scheme for Agriculture IoT. In *IEEE Sensors Applications Symposium (SAS)*, pages 1–6, Sophia Antipolis, France, France, March 2019.

[36] Jangeun Jun and Mihail L. Sichitiu. The nominal capacity of wireless mesh networks. *IEEE Wireless Communications*, 10(5):8–14, October 2003.

[37] Shahina K and V. Vaidehi. Clustering and data aggregation in wireless sensor networks using machine learning algorithms. In *2018 International Conference on Recent Trends in Advance Computing (ICRTAC)*, pages 109–115, September 2018.

[38] Mandeep Kaur, Navdeep Kaur, and Raju Sharma. Comparison and Analysis between ReactiveRouting Protocols in MANET using Opnet17.5v. *International Journal of Innovations in Engineering and Technology (IJIET)*, 6(1), October 2015.

[39] Feeza Khan, Saira Memon, and Sana Hoor Jokhio. Support Vector Machine based Energy Aware Routing in Wireless Sensor Networks. In *2nd International Conference on Robotics and Artificial Intelligence (ICRAI)*, pages 1–4, Rawalpindi, Pakistan, November 2016.

[40] Avni Khatkar and Yudhvir Singh. Performance Evaluation of Hybrid Routing Protocols in Mobile Ad Hoc Networks. In *Second International Conference on Advanced Computing Communication Technologies*, pages 542–545, Rohtak, Haryana, India, January 2012.

[41] Hlabishi I. Kobo, Adnan M. Abu-Mahfouz, and Gerhard P. Hancke. A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements. *IEEE Access*, 5:1872–1899, 2017.

[42] Ka-Cheong Leung and Victor O. K. Li. Transmission control protocol (TCP) in wireless networks: issues, approaches, and challenges. *IEEE Communications Surveys Tutorials*, 8(4):64–79, 2006.

[43] Jianting Li and Yingpeng Zhang. Design and Accomplishment of the Real-Time Tracking System of Agricultural Products Logistics Process. In *International Conference on E-Product E-Service and E-Entertainment*, pages 1–4, Henan, China, November 2010.

[44] Chun-Hao Liao, Guibing Zhu, Daiki Kuwabara, Makoto Suzuki, and Hiroyuki Morikawa. Multi-Hop LoRa Networks Enabled by Concurrent Transmission. *IEEE Access*, 5:21430–21446, 2017.

[45] Yu Liu, Kin-Fai Tong, Xiangdong Qiu, Ying Liu, and Xuyang Ding. Wireless Mesh Networks in IoT networks. In *International Workshop on Electromagnetics: Applications and Student Innovation Competition*, pages 183–185, London, UK, May 2017.

[46] Daniel Lundell, Anders Hedberg, Christian Nyberg, and Emma Fitzgerald. A Routing Protocol for LoRA Mesh Networks. In *IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, pages 14–19, Chania, Greece, June 2018.

[47] Yi-Wei Ma and Jiann-Liang Chen. Toward intelligent agriculture service platform with lora-based wireless sensor network. In *IEEE International Conference on Applied System Invention (ICASI)*, pages 204–207, Chiba, Japan, April 2018.

[48] Mahmoud Shuker Mahmoud and Auday Mohamad. A Study of Efficient Power Consumption Wireless Communication Techniques/ Modules for Internet of Things (IoT) Applications. *Advances in Internet of Things*, 06(02):19–29, 2016.

[49] Subhas Chandra Mukhopadhyay. *Intelligent Sensing, Instrumentation and Measurements*, volume 5. Springer, 2013.

[50] Deanna Needell, Rachel Ward, and Nati Srebro. Stochastic gradient descent, weighted sampling, and the randomized kaczmarz algorithm. In *Advances in Neural Information Processing Systems 27*, pages 1017–1025. Curran Associates, Inc., 2014.

[51] Dragos Niculescu, Sudeept Bhatnagar, and Samrat Ganguly. Channel assignment for wireless meshes with tree topology. In *8th International Conference on Communications*, pages 383–386, Bucharest, Romania, June 2010.

[52] Liwu Pan, Mingzhe Xu, Lei Xi, and Yudong Hao. Research of livestock farming IoT system based on RESTful web services. In *5th International Conference on Computer Science and Network Technology (ICCSNT)*, pages 113–116, Changchun, China, December 2016.

[53] José O. Payero, Ali Mirzakhani Nafchi, Rebecca Davis, and Ahmad Khalilian. An Arduino-Based Wireless Sensor Network for Soil Moisture Monitoring Using Decagon EC-5 Sensors. *Open Journal of Soil Science*, 07(10):288–300, 2017.

[54] Claudia Perlich. *Learning Curves in Machine Learning*, pages 577–580. Springer US, Boston, MA, 2010.

[55] Heather Persson, Bryan Schlosser, Johannes Eisele, and Greg Pender. Saskatchewan Agriculture By the Numbers, May 2018. https://thestarphoenix.com/news/saskatchewan/saskatchewan-agriculture-by-the-numbers.

[56] Congduc Pham. Enabling and deploying long-range IoT image sensors with LoRa technology. In *IEEE Middle East and North Africa Communications Conference (MENACOMM)*, pages 1–6, Jounieh, Lebanon, Apr 2018.

[57] Congduc Pham. Robust CSMA for Long-Range LoRa Transmissions with Image Sensing Devices. In *Wireless Days (WD)*, pages 116–122, Dubai, United Arab Emirates, April 2018.

[58] Jon Postel. Transmission control protocol. *Rfc*, 2(4):595–599, 1981.

[59] Neeli Prasad and Anand Prasad. *802.11 WLANs and IP Networking: Security, QoS, and Mobility*. Artech House Universal Persona. Artech House, 2005.

[60] Tie Qiu, Xize Liu, Lin Feng, Yu Zhou, and Kaiyu Zheng. An Efficient Tree-Based Self-Organizing Protocol for Internet of Things. *IEEE Access*, 4:3535–3546, 2016.

[61] Brian Ray. Examining Cellular IoT: Cost, Battery, & Data, August 2015. https://www.link-labs.com/blog/cellular-iot.

[62] Usman Raza, Parag Kulkarni, and Mahesh Sooriyabandara. Low Power Wide Area Networks: An Overview. *IEEE Communications Surveys Tutorials*, 19(2):855–873, 2017.

[63] Lior Rokach and Oded Maimon. Top-down induction of decision trees classifiers - a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 35(4):476–487, November 2005.

[64] Gadikota Sahitya, Narayanam Balaji, Challa Dhanuanjaya Naidu, and Sindu P. Abinaya. Designing a Wireless Sensor Network for Precision Agriculture Using ZigBee. In *IEEE 7th International Advance Computing Conference (IACC)*, pages 287–291, Hyderabad, India, January 2017.

[65] Benjamin Sartori, Steffen Thielemans, Maite Bezunartea, An Braeken, and Kris Steenhaut. Enabling RPL multihop communications based on LoRa. In *IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–8, Rome, Italy, October 2017.

[66] SEMTECH. SX1276/77/78/79 DATASHEET. Technical report, SEMTECH, 2016.

[67] Madoune Robert Seye, Moussa Diallo, Bamba Gueye, and Christophe Cambier. COWShED: Communication Within White Spots for Breeders. In *22nd Conference on Innovation in Clouds, Internet and Networks (ICIN 2019)*, pages 1–4, Paris, France, February 2019.

[68] Madoune Robert Seye, Bamba Gueye, and Moussa Diallo. An evaluation of LoRa coverage in Dakar Peninsula. In *8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 478–482, Vancouver, Canada, October 2017.

[69] Divya Sharma, Sandeep Verma, and Kanika Sharma. Network topologies in wireless sensor networks: A review. *International Journal of Electronics & Communication Technology*, 4(Spl - 3):93–97, April - June 2013.

[70] Bhuvan M. Shashidhara, Siddharth Jain, Vinay D. Rao, Nagamma Patil, and G. S. Raghavendra. Evaluation of machine learning frameworks on bank marketing and higgs datasets. In *2015 Second International Conference on Advances in Computing and Communication Engineering*, pages 551–555, Dehradun, India, May 2015.

[71] Mahyar Shirvanimoghaddam, Kamyar Shirvanimoghaddam, Mohammad Mahdi Abolhasani, Majid Farhangi, Vahid Zahiri Barsari, Hangyue Liu, Mischa Dohler, and Minoo Naebe. Paving the Path to a Green and Self-Powered Internet of Things, August 2018. https://arxiv.org/abs/1712.02277v2.

[72] Himanshu Singh, Vishal Pallagani, Vedant Khandelwal, and U. Venkanna. IoT based Smart Home Automation System using Sensor Node. In *4th International Conference on Recent Advances in Information Technology (RAIT)*, pages 1–5, Dhanbad, India, March 2018.

[73] Zeldi Suryady, Mohd Hafiz Md Shaharil, Khairina Abu Bakar, Reza Khoshdelniat, Gopinath Rao Sinniah, and Usman Sarwar. Performance Evaluation of 6LoWPAN-based Precision Agriculture. In *25th International Conference on Information Networking (ICOIN)*, pages 171–176, Barcelona, Spain, January 2011.

[74] Rohan Tabish, Adel Ben Mnaouer, Farid Touati, and Abdulaziz M. Ghaleb. A comparative analysis of BLE and 6LoWPAN for U-HealthCare applications. In *7th IEEE GCC Conference and Exhibition (GCC)*, pages 286–291, Doha, Qatar, November 2013.

[75] Nooritawati Md Tahir, Aini Hussain, Salina Abdul Samad, Khairul Anuar Ishak, and Rosmawati Abdul Halim. Feature Selection for Classification Using Decision Tree. In *4th Student Conference on Research and Development*, pages 99–102, Selangor, Malaysia, June 2006.

[76] Andrew S. Tanenbaum and David J. Wetherall. *Computer Networks*. Pearson Education, 2013.

[77] Floris Van den Abeele, Jetmir Haxhibeqiri, Ingrid Moerman, and Jeroen Hoebeke. Scalability Analysis of Large-Scale LoRaWAN Networks in ns-3. *IEEE Internet of Things Journal*, 4(6):2186–2198, December 2017.

[78] Ch. B. Vinutya, Niranjana Nalini, and B.S. Veeresh. Energy Efficient Wireless Sensor Network using Neural Network based Smart Sampling and Reliable Routing Protocol. In *International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pages 2081–2085, Chennai, India, March 2017.

[79] WAGmob. Telecom Engineering. In *Telecom Engineering*. WAGmob, 1.5 edition, 2013.

[80] Leif R. Wilhelmsson, Miguel M. Lopez, and Dennis Sundman. NB-WiFi: IEEE 802.11 and Bluetooth Low Energy Combined for Efficient Support of IoT. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, San Francisco, CA, March 2017.

[81] Jichun Zhao, Junfeng Zhang, Yu Feng, and Jianxin Guo. The study and application of the IoT technology in agriculture. In *3rd International Conference on Computer Science and Information Technology*, pages 462–465, Chengdu, China, July 2010.

[82] Liang Zhao, Yujie Li, Chao Meng, Changqing Gong, and Xiaochun Tang. A SVM based routing scheme in VANETs. In *16th International Symposium on Communications and Information Technologies (ISCIT)*, pages 380–383, Qingdao, China, September 2016.

[83] Wenju Zhao, Shengwei Lin, Jiwen Han, Rongtao Xu, and Lu Hou. Design and Implementation of Smart Irrigation System Based on LoRa. In *IEEE Globecom Workshops (GC Wkshps)*, pages 1–6, Singapore, Singapore, December 2017.

[84] Ting Zhi, Hongbin Luo, and Ying Liu. A Gini Impurity-Based Interest Flooding Attack Defence Mechanism in NDN. *IEEE Communications Letters*, 22(3):538–541, March 2018.