Degree Project in Embedded Systems

Second cycle, 30 credits

Comparative Analysis of Distance Vector Routing and Flooding for LoRa

JOSHUA SADIQ

Comparative Analysis of Distance Vector Routing and Flooding for LoRa

JOSHUA SADIQ

Master's Programme, Embedded Systems, 120 credits
Date: May 16, 2026

Supervisors: Saul Rodriguez Duenas, Frida Hellström, Anton Tiberg
Examiner: Carl-Mikael Zetterling
School of Electrical Engineering and Computer Science
Host company: Prevas AB
Swedish title: Komparativ Undersökning av Distance Vector Routing och Flooding för LoRa

© 2026 Joshua Sadiq

Abstract

LoRa, short for Long-Range, is a technique for transmitting radio signals over long distances. It does so thanks to Chirp Spread Spectrum, a modulation technique which stores information in frequency chirps. The physical embedded hardware on which it can be found is also cost-effective, opening up for a multitude of use cases. It lacks, however, innate mesh functionality.

Applications of LoRa have implemented mesh functionality, most famously through the use of flooding. This thesis conducts a comparative analysis between such a method and Distance Vector routing, a routing algorithm and protocol which sends packets only through required nodes ensuring reduced Time on Air. The thesis also presents a Proof-of-Concept for Distance Vector routing by developing and implementing it in an existing framework, all in a simulated environment.

The findings indicate that Distance Vector routing performs better than Managed Flooding when dealing with a small number of nodes. In particular, the transmission air utilization is lower in said cases, which indicates that the energy usage in a physical hardware would also be lower for the algorithm. For other cases, the results are mixed and require further investigation.

Keywords

LoRa, Mesh Network, Routing Algorithms, Simulation, Energy Efficiency

Sammanfattning

LoRa, som står för Long-Range, är en metod för att överföra radio signaler på långa avstånd. Det är via Chirp Spread Spectrum som den åstadkommer detta, vilket är en modulationsteknik som lagrar information i frekvenssvep. Den fysiska inbyggda hårdvaran för LoRa är dessutom kostnadseffektiv, vilket öppnar upp för en mängd olika användningsområden. Den saknar emellertid meshfunktionalitet.

Tillämpningar har implementerat meshfunktionalitet, främst genom användandet av flooding. Detta arbete genomför en komparativ analys mellan en sådan metod och Distance Vector routing, ett dirigeringsalgoritm och nätverksprotokoll som skickar datapaket endast via nödvändiga noder, vilket säkerställer minskad sändningstid. Vidare presenterar arbetet också en Proof-of-Concept för Distance Vector routing genom att utveckla och implementera den i ett befintligt ramverk, helt i en simulerad miljö.

Resultaten indikerar att Distance Vector routing presterar bättre än Managed Flooding när noderna till antalet är få. Särskilt gäller detta sändningstiden, vilket tyder på att energiförbrukningen i fysisk hårdvara också skulle lägre för algoritmen. För övriga fall är resultaten blandade och kräver vidare undersökning.

Nyckelord

LoRa, Meshnätverk, Routing Algoritmer, Simulering, Energieffektivitet

Acknowledgments

There are many individuals to whom I owe gratitude for the support I have received throughout writing this thesis.

My examiner and supervisor at KTH, Carl-Mikael Zetterling and Saul Rodriguez Duenas, for their support and patience. Likewise, my supervisor at Prevas, Anton Tiberg, for always listening to my concerns and Frida Hellström for her patience and gentle encouragement. Axel Lindberg, for the laughter we've shared throughout our thesis', and Alexander Malmberg for his thorough correction of my work. Anneli Blomqvist and Jakob Ruhe, for setting me on the correct path early on. My sisters, for always motivating me to stay sharp. The list of people who made an impact on me and my thesis is extensive; I extend a sincere thank you to those who are not named.

Finally, I am immensely grateful to my greatest supporters and parents, Joseph and Hanna. This thesis would not be without their sage guidance and endless love. Thank you.

Stockholm, May 2026
Joshua Sadiq

Contents

1 Introduction **1**
1.1 Background . . . 2
1.2 Problem . . . 2
1.3 Purpose . . . 3
1.4 Goals . . . 3
1.5 Ethics and Sustainability . . . 3
1.6 Research Methodology . . . 4
1.7 Stakeholders . . . 5
1.8 Delimitations . . . 5
1.9 Structure of the thesis . . . 5

2 Background **7**
2.1 LoRa . . . 7
2.1.1 LoRa PHY . . . 7
2.1.1.1 Chirp Spread Spectrum . . . 7
2.1.1.2 Code Rate . . . 9
2.1.1.3 Spreading Factor . . . 9
2.1.1.4 Time on Air . . . 10
2.1.1.5 ISM Band . . . 10
2.1.2 LoRaWAN . . . 12
2.1.2.1 Star Topology . . . 12
2.1.2.2 MAC Protocol . . . 13
2.1.2.3 End-Device Classes . . . 14
2.2 Mesh Network . . . 15
2.2.1 Routing Algorithms . . . 15
2.2.1.1 Bellman-Ford algorithm and Distance Vector routing . . . 16
2.2.2 Meshtastic . . . 17
2.2.2.1 Meshtasticator . . . 17

2.3 Related work area 18
2.4 Summary 19

3 Methodology **21**
3.1 Research Process 21
3.2 Simulator 22
3.2.1 Test Design 22
3.3 Planned Data Analysis 23
3.4 Method Validity and Reliability 23
3.4.1 Validity 23
3.4.2 Reliability 24

4 Implementation **25**
4.1 Implementing the Routing Algorithm 25
4.1.1 Distance Vector Routing Protocol 26
4.1.2 Improving Transparency in Packet Flow and Increasing Metric Visibility 27
4.1.3 Virtual Environment 27
4.2 Experimental Test Setup 28

5 Results and Analysis **31**
5.1 Test metrics 31
5.2 Results 32
5.2.1 Case 1 32
5.2.2 Case 2 34
5.2.3 Case 3 36
5.2.4 Case 4 38
5.2.5 Case 5 40
5.2.6 Case 6 42
5.3 Analysis 44
5.4 Reliability and Validity Analysis 45

6 Discussion **47**
6.1 Experiment and Results 47
6.2 Real-World Applicability 48
6.3 Reflection 48

7 Conclusions and Future Work **51**
7.1 Conclusions 51
7.2 Limitations 51

7.3 Future work 52
7.3.1 Unfinished Work 52
7.3.2 Other Work 52

References **55**

A Python libraries **59**

List of Figures

2.1 Structure of the Long-Range (LoRa) frame. 8
2.2 Spectrogram of LoRa symbols encoded as frequency chirps, with permission granted by creator. 9
2.3 LoRaWAN star-of-stars topology with main server, gateways and end-devices. 12
2.4 Class A end-device receive slots. 14

4.1 Depiction of the discrete-event simulator's flow. 25
4.2 Comprehensive figure of all six test cases. 29

5.1 Case 1 results, with Distance Vector (DV) and Managed Flooding (MF) routing protocols. 33
5.2 Case 2 results. 35
5.3 Case 3 results. 37
5.4 Case 4 results. 39
5.5 Case 5 results. 41
5.6 Case 6 results. 43

List of Tables

2.1 Semtech SX1276 LoRa receiver sensitivity for varying bandwidth and Spreading Factors (in dBm). 10
2.2 Spreading Factor, payload size, Time on Air, and throughput rate relation. 11
2.3 Table over LoRa-usable frequencies in Europe. 11

5.1 Relevant test metrics, unit displayed, and description. 31

A.1 Python libraries and their respective versions used in the test setup's virtual environment. 60

List of acronyms and abbreviations

CR	Code Rate
CRC	Cyclic Redundancy Check
CSS	Chirp Spread Spectrum
DL	Downlink
EIRP	Effective Isotropic Radiated Power
FEC	Forward Error Correction
FIFO	First-In-First-Out
IoT	Internet-of-Things
ISM	Industrial, Scientific, and Medical
LoRa	Long-Range
LoRaWAN	Long-Range Wide Area Network
MAC	Media Access Control
PHY	Physical Layer
PoC	Proof-of-Concept
SF	Spreading Factor
SNR	Signal-to-Noise Ratio
ToA	Time on Air
UL	Uplink
WMN	Wireless Mesh Network

Chapter 1

Introduction

With the growing availability of embedded devices, new options for wireless long-distance communication are emerging. In rural areas where access to cellular towers or internet connection is sparse, satellite link is one viable choice. This, however, is costly and not scalable. An alternative from the domain of embedded systems is Long-Range (LoRa).

LoRa is a proprietary physical radio hardware and allows communication between LoRa devices over several tens of kilometers in distance, in some cases several hundred. An open-source network protocol, Long-Range Wide Area Network (LoRaWAN), exists that allows point-to-point communication between LoRa devices in a star-topology network. In situations where greater redundancy is required - such as when connection to a node can be lost - a mesh-network topology could offer the guarantee of continued linkage. LoRaWAN, however, does not inherently support a mesh-network topology.

To date, some research has been conducted which allows implementation of meshing for LoRa and LoRaWAN respectively. Open-source projects such as Meshtastic also exist, providing a mesh network protocol and firmware for LoRa. These options, however, lack certain key functionality which is required for a robust and reliable industrial-grade mesh-implementation.

This thesis aims to investigate and provide a Proof-of-Concept (PoC) implementation of said key functionality - namely transmission acknowledgement, transmission scheduling, and packet collision handling - while proposing a routing algorithm that is more energy-efficient than flooding.

1.1 Background

LoRa is a radio hardware developed by Semtech. It uses a proprietary Chirp Spread Spectrum (CSS) modulation technique for its signals that offers good resilience to interference and is Doppler-resistant [1]. When a LoRa device sends a signal, the chosen Spreading Factor (SF) determines the transmission intensity. The Spreading Factor, bandwidth of the data, and energy consumption of the device have a deeply-knitted relation. With a higher SF, the signal can reach further at the cost of reduced bandwidth and higher energy consumption. With a lower SF, the opposite is true. Finding the most efficient Spreading Factor for a given use case is vital in ensuring an energy-efficient device.

Since end-devices often run on battery power, energy efficiency is a key component to keep track of. Studies have shown that with the introduction of a mesh-network topology, energy consumption can be reduced compared to the LoRaWAN network protocol [2]. While LoRaWAN is readily available and offers functionality such as encryption, its lack of multi-hop functionality (i.e. utilizing other nodes in the network to forward a packet to its destination) reduces redundancy and network-reach distance, unlike what a mesh network could offer.

Mesh networks offer many benefits such as self-healing capabilities, reduced energy consumption, and smart selectivity [2], to mention a few. However there are challenges with implementing such a topology on LoRa. A mesh network inherently introduces the need to keep track of what devices are available in the network, so that a path for packet transmission can be found. If the decision-making is left to each individual node, more overhead is required for each device which consumes limited bandwidth from the information data. On the other hand, if a centralized unit is responsible for making such decisions then additional hardware and communication protocol between the unit(s) and end-devices is required. If, like Meshtastic, a flooding routing algorithm is used then concerns of network congestion and packet collision are raised, not to mention an ineffective battery usage.

1.2 Problem

The main goal of this thesis is to find a routing algorithm that can support a mesh network in LoRa with a higher energy-efficiency than flooding. Furthermore, investigating how key functionality like transmission

acknowledgement, transmission scheduling, and packet collision handling can be implemented in a mesh-network topology for the LoRa hardware is also of interest. Finally, developing a PoC that demonstrates these findings is relevant. All of this is presented in the following research question:

Can acknowledgement, scheduling, and collision-handling be implemented with a significantly more energy-efficient routing than flooding in a LoRa mesh-network?

1.3 Purpose

The purpose of this degree project and Master's thesis is to investigate energy-efficient routing algorithms for a LoRa mesh network and whether implementation of key functionality as described above is possible. Prevas, the host company, is interested in these findings. Likewise, the scientific community has a growing interest in LoRa as it offers wireless communication for a low price, and would therefore benefit from the findings of this thesis.

1.4 Goals

The goals of this project is to answer the research question and can be summarized in the following statements:

- Conduct a literature study and find a routing algorithm for the LoRa hardware which is more energy-efficient than a flooding algorithm
- Provide a PoC of the more energy-efficient routing algorithm in a simulated environment
- Implement transmission acknowledgement, transmission scheduling, and packet collision handling with the routing algorithm

1.5 Ethics and Sustainability

The results of this research could be a stepping stone in the contribution of economic, sustainable, social, and ethical solutions. Large hardware like cellular towers, centralized internet servers, and satellites in space can cost a lot economically and environmentally. With more specialized and cheap embedded hardware which utilize just as many resources as the task

requires, companies could reduce their costs considerably and at the same time reduce the toll on our world's resources. For areas where natural disasters are common, decentralized communication through a mesh network could offer robust connectivity when cellular connectivity or internet access is no longer guaranteed. Similarly, for societies and dictatorships where citizens are hampered from connecting to one another and the rest of the world wirelessly, decentralized mesh network with off-the-shelf components can offer the individuals their right to connectivity.

KTH Royal Institute of Technology, the research institute and university where this project is being written at, wishes to contribute to the UN Global Sustainability Goals [3]. *The 17 Goals* is an initiative by the United Nations and adopted by all UN member states in an attempt to end poverty, improving health and education, reduce inequality, incite economic growth, and managing climate change for all countries across the globe [4]. With such a background, this thesis should contribute to the aforementioned goals. This research hopes to augment goals 9, 12, and 13 i.e. building resilient infrastructure, ensuring sustainable production, and taking urgent actions against climate change through the use of specialized equipment instead of generalized. Furthermore, this research also aims to contribute to goals 6 and 11 indirectly, namely LoRa's usage for ensuring clean water and more sustainable cities, e.g. through monitoring quality of water and air pollution, respectively.

1.6 Research Methodology

This thesis employs a mixed-methods approach, with both qualitative and quantitative methods. The reasoning for this choice is such that the qualitative methods can provide a comprehensive exploration of the problem field, leading to the best possible solution. This entails the literature study of LoRa, LoRaWAN, and routing algorithms. The quantitative methods, on the other hand, were chosen to measure the effectiveness of the proposed solution and its implementation. By recording test metrics such as energy consumption, network congestion, packet loss rate, and so on of the implemented routing algorithm, an extensive understanding of the improvement can be attained. For the PoC, an open-source Meshtastic-simulator will be extended upon by implementing key functionality and a new routing algorithm.

1.7 Stakeholders

Prevas, the company where this thesis is being done at, is interested in finding alternative robust and reliable communication methods to traditional cellular, internet, and satellite connectivity and benefits from the results of this research. While they offer resources in form of hardware, software, and a large knowledge pool there is an expectation to deliver new information on LoRa that adds to said pool and may be of use for future projects.

1.8 Delimitations

This thesis is limited to reading literature, implementing software, and conducting tests in a simulated environment. No hardware implementation will be done during this project, even though it is of utmost interest, as that itself is an entire thesis work. Furthermore, security and privacy (such as encryption) falls outside the scope of this project as well. The domain of security and privacy is large and would require much more time than what the thesis project can offer. This is left as future work.

Instead, a literature study will be conducted to build a pillar of knowledge this thesis can stand on as well as find missing gaps of research. Next, implementation of software will provide with a PoC for the routing algorithm and its key functionalities. Finally, tests will provide metrics on whether the PoC results in a better energy efficiency or not, along with certain other traits.

1.9 Structure of the thesis

This thesis is divided into seven chapters. Chapter 1 introduces the thesis and its purpose. Chapter 2 gives a background of the subject area and primes the reader for the coming chapters. Chapter 3 presents the methodology and gives a detailed insight into the relevant methods used for the project. Chapter 4 displays the final implementation of the proposed solution. Chapter 5 details the results and presents an analysis of said results. Chapter 6 entails a discussion about said results and overall methodology. Finally, Chapter 7 presents future work and concludes this thesis.

Chapter 2

Background

This chapter will provide the reader with theoretical background surrounding LoRa and mesh networking. Following that, related work to this project will be introduced that supplies further reading. Finally, a summary will bring the background to a conclusion.

2.1 LoRa

As mentioned in chapter 1, LoRa is a proprietary radio hardware and modulation technique developed by Semtech [5]. Through the use of CSS radio modulation LoRa can communicate over very long distances and is robust to interference [6]. In some literature LoRa and LoRaWAN is used interchangeably [7][1], but in this paper a clear distinction between the Physical Layer (PHY) and the network protocol (LoRaWAN) is made.

2.1.1 LoRa PHY

There are many parameters pertaining to the physical hardware of LoRa. These parameters are closely related and contribute to the way LoRa modulation works. The modulation technique allows scalability of frequency and bandwidth while also consuming less power [8]. Aside from CSS, LoRa also combines Forward Error Correction (FEC) to compensate errors which spawn from interference [9][1].

2.1.1.1 Chirp Spread Spectrum

CSS is a modulation technique that stores information in frequency chirps [1]. An advantage of it is its resistance to multi-path fading and being suitable

Figure 2.1: Structure of the LoRa frame.

for noisy environments. Furthermore, it attempts at providing low throughput communication over distances longer than ten km in the outdoors. All of this, without consuming too much power [10].

As mentioned, information is encoded into frequency chirps. These have a linear variation of frequency over time. Due to this variation, the frequency offset between a receiving LoRa unit and the transmitting LoRa unit are equal to the timing offset which leaves it as a trivial task for the decoder to eliminate. As the Doppler effect arises due to frequency offset, this increases LoRa's resistance to the phenomenon [1]. Additionally, some hardware can measure the frequency offset from the receiver center frequency and tolerate up to $\pm 25\%$ deviation from the bandwidth. This results in cheaper oscillator crystals being viable for the LoRa system, reducing the cost of the radio [11].

For transmitters and receivers developed by Semtech, a physical frame format is specified for the LoRa modulation - even though any arbitrary frame would work. A preamble appears at the start of the frame, with several upchirps. Upchirps go from a lower frequency and end at a higher. The final two upchirps contain a sync word - a single-byte that allows LoRa networks on the same frequency band to distinguish from one another. Following this, two and one-fourth downchirps complete the preamble.

After the preamble, an optional header with a Code Rate (CR) of $\frac{4}{8}$ can be fit into the frame. This header indicates the size of the payload in bytes. Since the optional header is only one byte, it effectively limits the payload size to 255 bytes*. In situations where the payload length, CR, and presence of Cyclic Redundancy Check (CRC) is known, the optional feature can be turned off, leaving an additional byte free.

Last, the payload and an optional payload CRC can be found at the end of the frame [1]. A figure summarizing this can be found in 2.1. Fig. 2.2 depicts a spectrogram of LoRa symbols, including the preamble and sync symbols as well as five additional payload symbols.

*One byte can assume binary values from 0 to 255.

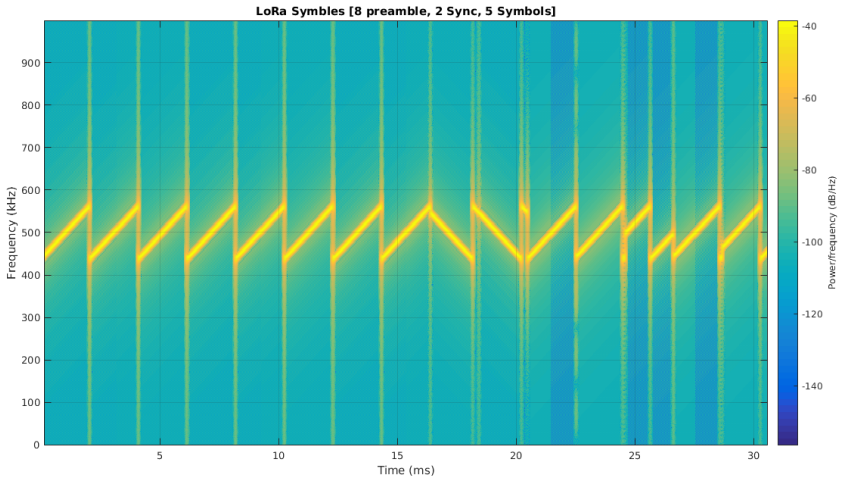


Figure 2.2: Spectrogram of LoRa symbols encoded as frequency chirps, with permission granted by creator.

2.1.1.2 Code Rate

As mentioned in 2.1.1 LoRa comes with FEC, partly responsible for the improved resistance to interference. The CR of the FEC is represented as $\frac{4}{n}$, where $n \in 5, 6, 7, 8$. The optional header, which follows the preamble, has a set CR of $\frac{4}{8}$, whereas the payload and the optional payload CRC can cover the entire range. A smaller CR has shown to reduce the Packet Error Rate [1].

2.1.1.3 Spreading Factor

The SF is a key component of the LoRa modulation as it determines the bandwidth. The symbols LoRa encodes information within consists of 2^{SF} chirps. In other words, a symbol encodes SF bits of information. It is trivial to deduce that increasing the SF by one will lead to a) the frequency span of a chirp divided in two and b) doubling the duration of a symbol. The bit rate also increases by one for a symbol. Since the symbol rate and the bit rates are proportional to the frequency bandwidth, an increase in the bandwidth leads to a proportional increase in the transmission rate [1]. This is presented below in equation 2.1, where T_S is the duration of a symbol and BW is the bandwidth:

$$T_S = \frac{2^{SF}}{BW} \tag{2.1}$$

The Spreading Factor takes a value from 7 to 12. As the SF increases,

Table 2.1: Semtech SX1276 LoRa receiver sensitivity for varying bandwidth and Spreading Factors (in dBm).

BW	SF	7	8	9	10	11	12
125 kHz		-123	-126	-129	-132	-133	-136
250 kHz		-120	-123	-125	-128	-130	-133
500 kHz		-116	-119	-122	-125	-128	-130

the signal's reception grows more reliable [7] since the Signal-to-Noise Ratio (SNR) increases [9]. An advantage of the different SFs is that orthogonal values do not interfere with other LoRa signals on the same channel [5]. There is a clear relation between the SF and the bandwidth of a LoRa device, as the receiver sensitivity decreases for an increase in SF [11], see table 2.1. There is also a clear correlation between the Time on Air (ToA) and SF [12], as explained in 2.1.1.4.

There are drawbacks of a higher SF. The data throughput rate is decreased and can lead to more severe data collisions as the transmission time increases [13], which leads to increased energy consumption [9].

2.1.1.4 Time on Air

As the name indicates, Time on Air is the delta time from when a signal is transmitted from one device to when it is received by another. As stated previously, a higher SF can increase the ToA and thereby the energy consumption [14]. The size of the packet being transmitted also affects the ToA [15]. See table 2.2 for a reference of relation between the SF, throughput, and the ToA as found in [15]. When the energy consumption lacks a measurable method, the ToA is a reliable substitute due to its direct relation to energy usage.

2.1.1.5 ISM Band

One benefit of LoRa is that it works in the Industrial, Scientific, and Medical (ISM) band [16]. The ISM band is an unlicensed band, meaning it requires no license to operate the radio hardware within this band. The ISM band is also globally recognized [14].

LoRa, in Europe, transmits its data within 863-870 MHz with a duty cycle restriction of 1% as regulated by the European Telecommunications Standards Institute [5]. In the US, 902-928 MHz is designated as the ISM frequency [17].

Table 2.2: Spreading Factor, payload size, Time on Air, and throughput rate relation.

Max Payload Size			
SF	**Payload Size** [Bytes]	**Max ToA** [s]	**Throughput Rate** [bit/s]
7	242	0.394	4913.7
8	242	0.697	2777.6
9	115	0.677	1358.9
10	51	0.698	584.5
11	51	1.479	275.9
12	51	2.793	146.1

Table 2.3: Table over LoRa-usable frequencies in Europe.

Channel Plan	**Channel Frequency** [MHz]	**Bandwidth** [kHz]	**Duty Cycle**	**Max EIRP** [dBm]
EU433	433.175 433.375 433.575	125	< 10%	+12
EU863	868.10 868.30 868.50	125	< 1%	+16

Two disadvantages of using LoRa in the ISM band pertain to the unlicensed nature of the band. First, an entity with malicious intentions could, legally, emit random LoRa symbols on the air. This could lead to transmissions being jammed, according to [1]. Second, interference from non-LoRa radio devices running on the same unlicensed channel could degrade the signal quality and decrease the effective range of the wanted signal [14].

From [17] and [18], relevant frequency ranges and duty cycle requirements can be extracted and is presented as table 2.3. The major difference between EU433 and EU863 are the channel frequencies, the duty cycle restrictions, and the maximum Effective Isotropic Radiated Power (EIRP). The latter is a measure of power radiated by an antenna.

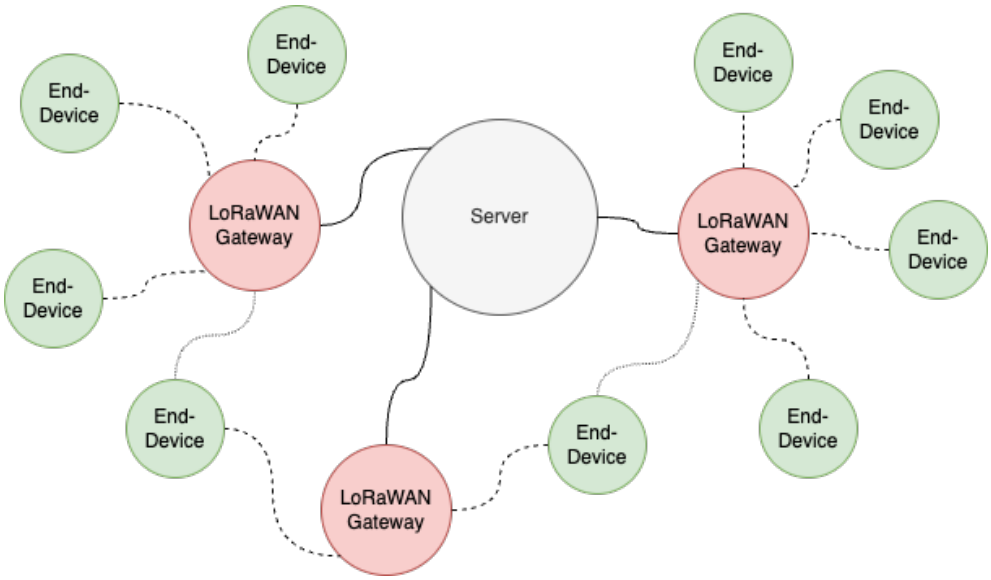


Figure 2.3: LoRaWAN star-of-stars topology with main server, gateways and end-devices.

2.1.2 LoRaWAN

LoRaWAN is a communication protocol and architecture maintained by the LoRa Alliance [16]. In the LoRaWAN network model, the data transmitted by nodes can be received by several gateways. These gateways transfer the data to the network server through some form of backhaul, typically through Wi-Fi, satellite, Ethernet, or cellular. There are three classes of end-devices in this network model [6].

2.1.2.1 Star Topology

LoRaWAN uses a *star-of-stars topology* [19]. A regular star network typically consists of a central hub where all end-devices connect to. Benefits of such a network topology is its reliability, as the failure of one connection would not affect the rest of the network. Furthermore, the direct connection between each end-device and the server ensures no transmission collision (particularly when direct cable link is used) and security as accessing another end-device requires going through the server. However, there is one particularly prominent disadvantage with this setup that affects a LoRaWAN network considerably: the single point of failure. If the central server stops working (for which there could be a variety reasons for why), none of the end-devices can send or receive information. This effectively neutralizes the entire network [20].

An extended topology of the star network one is the star-of-stars topology, which offers a solution to the disadvantage. As depicted in fig. 2.3, the star-of-stars network expands the star topology by adding additional gateways which create their own *subnetworks*. These gateways then connect to a central server hub. The gateways for each subnetwork allows storing information if the central server would go down, mitigating the previous issue. Moreover, implementing subnetworks increases the scalability of the entire network. This does introduce further hardware overhead, which may not be desirable in some circumstances [19][20].

2.1.2.2 MAC Protocol

A Media Access Control (MAC) protocol ensures that data transmission between two devices is carried out in an orderly fashion. Without a MAC layer, the movement of data would be unpredictable and collision rate much higher [21]. By arranging the transmissions flow of packets, the MAC layer ensures predictability [22].

LoRaWAN's MAC protocol is based on the ALOHA protocol, according to the LoRaWAN specifications [23]. The IEEE 802.15.4 specification, originally defined in 2003, sets a standard for low-rate Wireless Personal Area Networks [24]. LoRaWAN's higher-layer MAC protocol imitates the IEEE-specification, such as the authentication method which utilizes the same 4-octet message integrity code as the IEEE 802.15.4 standard, amongst other traits [19].

The MAC layer is responsible for two aspects: the Downlink (DL) and the Uplink (UL). The DL is when a device receives a signal from the gateway or base station. The opposite direction of signal transmission is defined as UL. For the DL, LoRaWAN's MAC protocol balances network communication efficiency and battery usage. This structure is dependent of the *class* of the end-device, which are detailed in 2.1.2.3 [22].

As LoRaWAN end-devices channel connection is based on the ALOHA protocol, there is no inherent collision management. Since the channel is not confirmed to be idle prior to transmitting a packet, collision can occur between packets which are sent on the same frequency and with the same SF. As there is a proportional correlation between amount of end-devices and packet collision rate, duty cycle is implemented in LoRaWAN as a measure to reduce the collision rate. The duty cycle determines how regularly the end-device may transmit signal [22]. Using Europe as an example, LoRa devices are assigned a 1% duty cycle - for every one second the device transmits, it

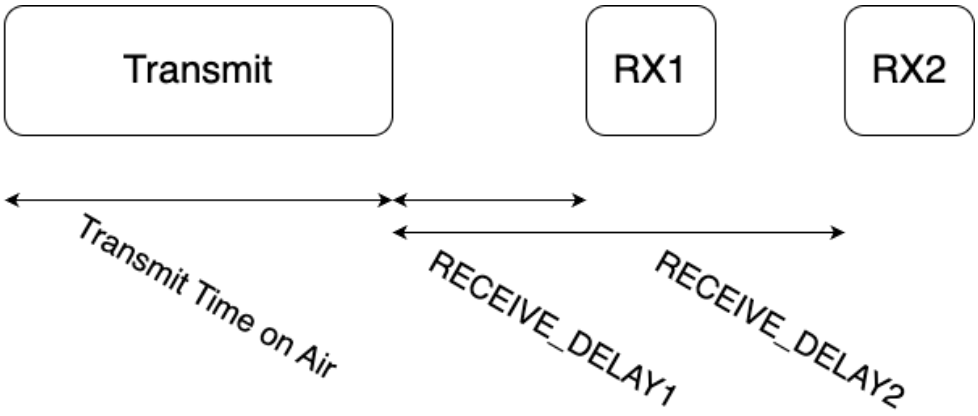


Figure 2.4: Class A end-device receive slots.

may not transmit for 99 seconds [23].

2.1.2.3 End-Device Classes

LoRaWAN has three end-device classes: *Class A*, *Class B*, and *Class C* [19]. These stand for *All end-devices*, *Beacon*, and *Continuously listening*, respectively. All end-devices being used in a LoRaWAN setup must implement the features of *Class A* [23].

Class A opens two short receive windows following an UL transmission, see fig. 2.4, where it receives DL transmissions. It cannot transmit another UL unless it either has received a DL in either of the two windows, or the last window has expired. *Class B* is appropriate when reception of DL should not be dependent on the end-device's DL, or when there is a need for predictable time slots. Furthermore, the *Class B* end-device is optimized for running on battery power, whether it's mobile or stationary. It operates by implementing an additional time slot when it listens for DL transmission, on top of the previous windows inherited from *Class A*. This is accomplished by the gateway as it regularly sends a beacon which synchronizes all the end-devices to open a reception window at regular intervals [23].

Finally, *Class C* end-devices - *Continuously listening* - extend the functionality of *Class A* devices but implement a continuously listening window *RX2*. This window is activated between the transmit time and the first DL window (RX1). During RX1, *RX2* stops listening and reopens its DL once the RX1 finishes. This class of end-devices is best suited when there is enough power available to support this energy-inefficient mode [23]. From this, it is apparent that the energy requirement of the various classes increases

from *A* to *C* [22].

2.2 Mesh Network

It is common for Internet-of-Things (IoT) network topologies to adapt a star-based (depicted for LoRaWAN) or a tree-based hierarchy. However, many applications of IoT networks would benefit from a more dynamic topology where the constituents of the network can adapt themselves according to changing conditions. Wireless Mesh Networks (WMNs) offer such a solution [25].

Since the nodes in a WMN are connected to each other in a non-hierarchical fashion, it is possible to route information from a source to a destination more efficiently than in a typical hierarchical network topology. This is possible as the WMN doesn't require network gateways to pass the information through. Another benefit of this topology is multi-hop information relaying. If information needs to go from node A to node C, but the latter is out of range for node A, the source can utilize node B (which reaches both the former nodes) to pass the information to its destination [25].

Other benefits of mesh networks are a) the reduced cost of deployment, installation, and maintenance overhead, b) the workload distribution is more dynamically spread out over the nodes and c) in case that a node fails, the rest of the network is not necessarily impacted. However, many of these qualities depend on which routing algorithm is implemented in the specific WMN. Unlike traditional network topologies, there are not mutually agreed common standards for WMNs and thus there are many different protocols available for varying forms of applications [25].

As mentioned in chapter 2.1.2, LoRaWAN transmits information in a star-of-stars topology, which is a hierarchical system. With modifications it would be possible to implement a mesh network instead [25]. At this time, there are many varying implementations proposed by the scientific community, as presented further below in chapter 2.3.

2.2.1 Routing Algorithms

Routing algorithms are responsible for ensuring the flow of packets in a network. With optimal algorithms, an efficient flow can ensure that information goes from source to destination. There are several different categories of routing algorithms that offer different benefits and disadvantages [26].

The Adaptive Routing Algorithm consists of three different subcategories: Isolated-, Centralized-, and Distributed Adaptive Routing Algorithms. Common property for all three is that the system is adaptive and can redirect nodes based on new information. Isolated algorithms lets each node determine the best path without seeing a map of the entire system or "discussing" with neighboring nodes. In a centralized system, a controller determines the path for all packets. The controller is a separate unit that is aware of all nodes in the system. In distributed systems, the nodes collaborate to determine the best path. Routing tables are used to create a map of the system [26].

In Non-Adaptive Routing Algorithms, the packets are passed without path optimization through the system. Flooding, one such form of algorithm, makes a node flood the entire network with the packet. Each consecutive node that picks up the packet will flood the network with it again, passing it from its source to its destination. Random Walk, another form of the non-adaptive algorithm, has a node send the packet to a random neighboring node. This can be highly inefficient, but is simple and decentralized [26].

A third category, Hybrid Routing Algorithms, introduces subcategories such as Link-State Routing and Distance Vector routing. The former has nodes exchange link-state advertisements which creates a detailed map of the network and can thereby find the shortest path. Nodes configured with Distance Vector routing, on the other hand, exchange information of routing tables in several iterations with neighbors [26].

2.2.1.1 Bellman-Ford algorithm and Distance Vector routing

In 1956 and 1958, Lester Ford Jr. and Richard Bellman respectively published what is known as the Bellman-Ford algorithm [27][28]. This algorithm is the basis for the Distance Vector routing [29].

The routing algorithm uses the following equations:

$$\begin{aligned} D(i,i) &= 0, \ for\ all\ i \\ D(i,k) &= min[d(i,j) + D(j,k)], \ else\ j \end{aligned}$$

where $D(i,j)$ represents the total distance for a packet to move from source i to destination j. The term *distance* refers to some metric for distance; for more simple solutions, this could be the amount of hops a packet takes from its destination to its source. In more complex systems, latency could be the metric or part of the overall distance analysis. $d(i,j)$ is the individual distance between directly neighboring nodes *i* and *j*. The second line in the equation therefore shows a recursive algorithm to find the total distance when hopping

between two distant nodes *i* and *k*, where the total distance $D(i, k)$ is calculated as the minimum distance between neighboring nodes *i* and *j* (where *j* can be a system of nodes that neighbor both *i* and *k*). The first line in the equation indicates that the distance for moving a packet to the same node as it originates from is 0 [29].

Distance Vector routing protocol is built on this algorithm, as mentioned earlier. In the protocol, each node keeps entries in a table to every other node in the system. The entries contain the distance to these nodes. Each node will send a routing update containing the contents of its table to other neighboring nodes in a regular fashion. Upon receiving a table from a neighbor, the node adds the distance to non-neighboring nodes based on the its neighbor's table. Over several iterations of sharing updated routing tables with neighboring nodes, each node in a network will eventually have a full routing map [29].

2.2.2 Meshtastic

Meshtastic is an "open-source, off-grid, decentralized mesh network built to run on affordable, low-power devices." It uses LoRa and implements a custom firmware and network protocol to allow messaging between user devices. It utilizes a form of flooding routing algorithm to propagate information. A user can deploy LoRa gateways to allow communication between two devices that use the Meshtastic software. The messages can utilize gateways deployed by other users to forward the packet to its destination, while ensuring that only intended receiver can read the message [30]. In test cases done by the community of Meshtastic, messages achieved a travel range of 331km [31].

2.2.2.1 Meshtasticator

Meshtasticator consists of two simulators for Meshtastic. An event-driven simulator which allows placement of nodes and simulates how packets between various nodes flow in the network, and an interactive simulator which allows for debugging communicating nodes without access to physical hardware. Both simulators emulate the firmware of Meshtastic [32].

The source is written in Python and, since it is open-source, can be modified to allow new configurations and implementation of different routing algorithms. The event-driven simulator provides a node placement graph, packet flow graph, node and message statistics as well as relevant metrics at the end of the simulation [32].

2.3 Related work area

There is a growing body of scientific contributions for how a mesh solution can be implemented for LoRa. Some suggest utilizing the existing LoRaWAN protocol and making modifications. Others propose new network protocols that may utilize the underlying hardware.

Both [14] and [15] present overviews of recent advancements and classifications of multi-hop communication, respectively, for LoRa and LoRaWAN mesh networks. [26] also presents an overview and introduction, but for routing protocols.

[13] shows how a mesh network of LoRa devices can transmit information to a single gateway without the need of LoRaWAN, allowing for multi-hop communication using only the nodes. [10] also introduces mesh networks between LoRa end devices, allowing transmission of packets between said devices, but extends the existing LoRaWAN protocol in an attempt to increase fault tolerance should a network relay fail to work.

Another method of implementing mesh networking is through the MAC protocol. [2] does so by extending LoRaWAN and allowing end devices to communicate with each other by introducing repeater nodes, which form subgroups and connect to end devices which are (often) underground. The repeater nodes communicate with the gateway using the existing LoRaWAN protocol, but utilize a modified MAC protocol when communicating with the end devices. [33] takes a different approach and proposes an improved protocol that replaces LoRaWAN. Furthermore, it allows for time-division multiple-access, a shortcoming of LoRaWAN, that reduces the data collision rate. [34], on the other hand, allows each end device to communicate with one another using a custom MAC protocol. This effectively enables multi-hop in the LoRa network.

A third approach is through the implementation of a C library, which [35] has done. By applying the library into LoRa end devices, the nodes become part of a mesh network which allows communication to and from other nodes. A proactive routing protocol is used, which maintains a map of the network by sending routing messages to neighbors to update a routing table. Finally, [16] develop a new routing protocol that combine Hybrid Wireless Mesh Protocol and Ad hoc On-Demand Distance Vector routing protocol to enable multi-hop in a LoRaWAN system.

2.4 Summary

LoRa is a cheap and capable hardware that comes with many benefits. Its long range, low price, and resistance to noisy environments are some good properties. LoRaWAN is a communication protocol built particularly for LoRa and allows for setting up a network. However, it lacks multi-hop and mesh functionality. Mesh networks for IoT-devices lack mutually-agreed standards for WMNs and is dependent on the implemented routing algorithm, but can allow for more complex topologies with improved performance. There are a variety of different routing algorithms with their own sets of advantages and disadvantages. There is a growing collection of scientific studies that has proposed and implemented mesh functionality for LoRa both through the use of LoRaWAN and with different network protocols.

Chapter 3

Methodology

This chapter presents the planned methods and experimental test setup for the project. Section 3.1 defines the research process, 3.2 the workings of the simulator as well as the platform used for testing, 3.3 explains how the data was collected and analyzed, and 3.4 concludes the chapter with an examination of the used methods' validity and reliability.

3.1 Research Process

As mentioned in 1.6, this thesis uses a mixed-methods approach for its work. It is essential to gain a comprehensive understanding of the field prior to starting the project, which is done through a qualitative literature study of the problem field. Following that, the method and experimental test setup are chosen - these also belong to the qualitative field. Test metrics are then collected through a quantitative approach, which are used to understand the results and draw conclusions which answer the problem formulation. These steps are detailed below:

- Conduct a literature study and find a routing algorithm for LoRa which is more energy-efficient than a flooding algorithm
- Find a method and experimental test setup which can be used to collected data to compare the usage of flooding against more energy-efficient routing algorithms
- Collect data from experiments to compare energy usage and determine whether there is a more energy-efficient algorithm than flooding

3.2 Simulator

Meshtasticator was chosen as the framework for testing, which is detailed in 2.2.2.1. Its open-source nature and being written in Python allows for implementing new routing algorithm with ease. As Meshtastic is built with LoRa in mind, the simulator (which utilizes the same firmware) is most fitting for the task of simulating LoRa nodes. Finally, the simulator displays test metrics relevant to the experiments and thereby has a testing framework befitting the project [30].

The event-driven simulator in the Meshtasticator suite provides several metrics once the simulation finishes, such as successful packet delivery ratio, average delay, average transmission air utilization, packet collision ratio, and more. Unfortunately, there is no metric for each node's energy usage. As stated in 2.1.1.4, the Time on Air (or *Tx air utilization*, one of the reported metrics of the simulator) would be a suitable proxy for the energy usage which is sought after.

3.2.1 Test Design

As the main research question seeks to answer whether a more energy-efficient routing algorithm than flooding exists for LoRa, a test metric which indicates the energy usage of LoRa nodes from the event-driven simulation is crucial. The simulator, however, lacks metrics for energy efficiency. Implementing adequate measurement of energy usage would require complex equations, wherein the validity of said equations would grow in complexity the more accurate results are required. Instead the simulator offers a measurement of Time on Air for each packet, which can act as a proxy for energy usage. As mentioned before, this project aims to conduct a *relative* comparison; for such purpose, measuring the average air transmission utilization would suffice to answer the research question.

The test is designed to conduct a series of measurements of the average transmission utilization for the flooding routing algorithm that comes with the simulator. With the same parameters (i.e. amount of nodes, positions, messages sent, etc.) similar tests will be conducted for the new routing algorithm. Comparison of each tests' average Time on Air will be made, as well as comparison between the total average Time on Air for each algorithm.

Aside from Time on Air, storage space and computation time are important as well. The difference in storage space will be measured through the size of the program code and RAM usage of the simulator during runtime.

For computation time, the total runtime of the program will be compared before and after modifications. These results will offer more perspectives to understand the transmission utilization results from.

3.3 Planned Data Analysis

At the end of a simulation run, the simulator offers several metrics. The most notable one for this thesis' purposes is *Average Tx air utilization*. This percentage is calculated as the arithmetic mean for all the nodes' transmission air time in the network. By comparing this metric prior to and post modification, a conclusion can be drawn whether the chosen routing is more efficient than flooding.

3.4 Method Validity and Reliability

As the method for determining the research question is through a simulation, its validity and reliable reproduction of results is of high relevance. As stated in [36], a "Simulation is experimentation with a model" and that "Simulation involves creating a model which imitates the behaviors of interest". The purpose of a simulation is to tradeoff the complexity and cost of reality in contrast to a model which is manageable and observable.

3.4.1 Validity

The simulator mimics the radio section through software and runs on the Meshtastic firmware. The PHY layer software calculates the airtime of a packet, depending on the set of LoRa parameters used (Bandwidth, spreading factor, coding rate, and data rate). It also uses a configurable path-loss model which replicates the physical environment the nodes may be in (such as urban regions or open fields). Aside from that, the PHY software determines whether packets collide based on aforementioned LoRa parameters [32].

The MAC layer software of the simulator calculates the transmission delay of a packet. This delay is the time the MAC layer would need to convert a message into a transmissible packet. Based on Meshtastic, the MAC layer software also uses a listen-before-talking procedure, which allows it control whether there is an incoming transmission to the node. It also introduces either random or SNR-based delay before transmitting the packet. The communication between nodes is done through Simpy [32].

To determine the validity of the simulator quantitatively is a non-trivial task, but with the information above it is much more approachable from a qualitative perspective.

3.4.2 Reliability

The ability to reliably reproduce experimental results is an important trait of scientific studies. Given the same set of parameters (such as the hardware used in the project or software version) and the same test design, results from the simulation should be the same.

The only non-accountable parameter of the simulator would *seem* to be Python's *random* library. However, the seed can remain constant for each simulation run. Furthermore, as the *random* module generates pseudo-random numbers [37] the experimental test setup can consider even the random aspect of the simulation to be an accountable parameter.

For a scientific experiment conducting similar tests as this thesis, the results should reliably be the same as long as the same test design and parameters are utilized.

Chapter 4

Implementation

Chapter 4 explains the steps of implementing the routing algorithm; section 4.1 goes specifically into detail on this. It further explains how packet flow is projected and metrics presented, ending with a technical description of the environment used. The experimental test setup's design and usage is described in the following section 4.2, where all the test cases are detailed. This section gives a background to the reader for understanding the results in chapter 5.

4.1 Implementing the Routing Algorithm

The discrete-event simulator, based on the Simpy framework, goes through the flow as described in fig. 4.1.

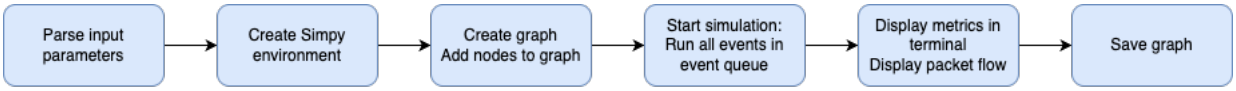


Figure 4.1: Depiction of the discrete-event simulator's flow.

Simpy works by taking processes (or events) from an event queue in a First-In-First-Out (FIFO) manner and running them. These processes can be functions that are placed in the queue through the `environment.process(function)` function. Once the simulation starts, the events run in a sequential manner. It is possible to mimic concurrency, which occurs in the simulator.

The simulator creates a graph and places the nodes requested by the user (the user may place the nodes on their own). Once the network is setup, the simulation runs for a determined time during which time the various nodes sends packets to and from each other. The generation, movement of and

listening to nodes is depicted in the terminal. At the end, key metrics are displayed.

To implement the Distance Vector routing protocol, certain changes and implementations were made.

4.1.1 Distance Vector Routing Protocol

As described in 2.2.1.1 a key component of the Distance Vector routing protocol is the *routing vector* containing distances to a node's neighbors. The existing `MeshNode` class needed to be extended, and most changes were implemented as additional methods or attributes to this class.

The `initRoutingTable` method prepares the node for incoming information about distance to neighbors by generating a list matrix, with the amount of lists equal to nodes in the simulated network. Following Bellman-Ford's algorithm, it sets the distance to itself as *0* and, as an initial value, the distance to all other nodes as "infinity".*

For the node to find its neighbors, the `findNeighbor` method was implemented. It broadcasts a packet with arbitrary contents and `maxHop = 1`, an attribute of packets determining how far the packet is allowed to hop between nodes. All neighboring nodes, upon receiving, broadcast the packet after decrementing its `maxHop`. With each packet, information such as source node, latest transmission node, destination node and more is attached. When the source node receives its packet from a neighbor, it can determine its neighbor using `isPacketFromNeighbor` and set the distance cost for that node to *1* using `setNeighborCost`.

Once all nodes have found their respective neighbors the nodes proceed with sharing their respective *routing vectors* to said neighbors. The receiving node confirms that the received routing vector comes from a neighbor, and proceeds to update its routing table using the Bellman-Ford algorithm. If the distance to a non-neighbor in the node's own routing table is larger than found in the received routing vector, the distance in the routing table is updated to be equal to the distance in the routing vector plus one unit of distance. This last calculation depends on the packet hopping to the neighboring node, and then utilizing the same distance as found in the routing vector.

This process is iterated several times until the distances for all nodes are no longer infinity, for each routing table. This finish condition is implemented specifically for the simulation - in hardware, a different approach would be required to determine the finish condition.

*In practice, this was set to a high number.

Finally, the implementation of sending packets using the Distance Vector routing protocol is done through `calculateHopCount` and `determineNextTransitNode`. The former is used to calculate max hop count required for a packet to reach a destination node. The latter utilizes this method to find the next transit node for the packet. This method is engaged each time a packet reaches a new node. By enumerating through the routing table, a neighboring node can be found with distance in its routing vector to destination node equal to the packet's current `maxHop-1`. The next transmit node, therefore, would be that neighboring node. This process is repeated until `maxHop` reaches 0, at which point the packet has reached its destination node.

4.1.2 Improving Transparency in Packet Flow and Increasing Metric Visibility

In order to observe the movement of packets to and from various nodes as well as collecting status messages during this process, meticulous logging was implemented. Each node's event (generating message, broadcasting packet, receiving packet, retransmitting a packet, and so on) were timestamped and logged. Packets were given an ID to track their movement between nodes. When a packet returned to its origin node during the neighbor-finding phase of the simulation, a status message was logged to indicate which neighbor node "bounced" the packet back to its origin. The delay, i.e. time from transmitting a packet to a node receiving it, was also noted. At the end of the simulation, various metrics such as messages created, packets sent, collisions, average delay (ms), average transmission air utilization, and more were displayed.

To understand how the routing tables took shape over the course of the simulation, all nodes' routing tables were printed each time any one of them were updated. This allowed a clear and visible timeline being created for the development of the tables. For each event that led to the tables being printed, a status message indicated what triggered the change.

4.1.3 Virtual Environment

To ensure that the results are reproducible, the software used and specific versions are explained here. The most notable detail is the usage of Meshtastic 2.5.11, prior to the 2.6 release which introduced next-hop routing and several other key features of interest.*

*https://meshtastic.org/blog/meshtastic-2-6-preview/

During development, Python libraries required specific versions to work together. For this reason, a virtual environment with Python 3.10 was used. Protobof, in particular, required a newer version than used by Meshtasticator. All libraries and their respective versions are listed in appendix A.

4.2 Experimental Test Setup

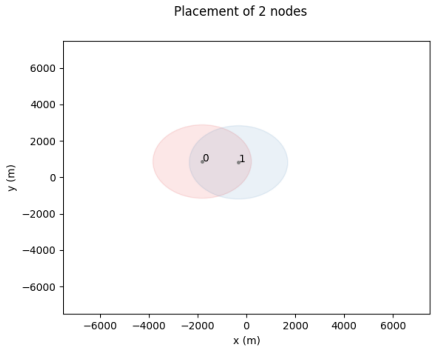
The order of steps the simulation progressed through were the same as the order detailed above. First, the nodes were instantiated. This includes generating their routing tables. After that each node would, at a random time, send a packet to find their neighbors using flooding. This method would update their routing tables. The amount of packets each node sent was equal to $n-1$, with n being the amount of nodes in the network. Subsequently each node would share their routing vectors with their neighbors. As each node received new potential paths to non-neighboring nodes, their respective routing tables were updated using this information. Over several iterations each node's table was updated and achieved convergence. Finally, the nodes sent a number of packets to random nodes.

A total of six test cases were simulated. Most mimic potential real-world scenarios, leaving a few as initial introductory test setups. Case 1, as found in fig. 4.2a, placed two nodes in close proximity, enabling two-way communication. This case was an initial test to ensure the nodes could communicate.

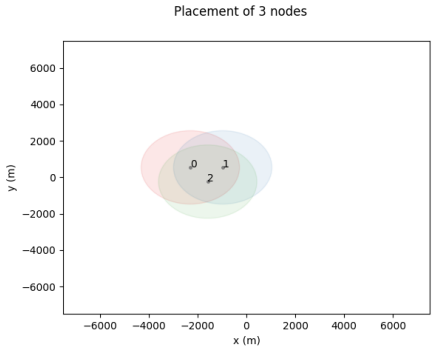
Case 2 extends the previous test setup by adding another node, Node 3. Node 3 is placed in a fashion that allows all nodes to communicate with one another without needing to forward their message through another node, see fig. 4.2b. This test was setup to create a constant with Case 3.

Case 3, as seen in fig. 4.2c, introduces a shift in node layout. Similar to Case 2, it consists of three nodes. In this design however, Node 0 and Node 2 can only reach Node 1. This leaves Node 1 to act as a intermediator for the other two nodes. This arrangement tests the core functionality of this project - hopping.

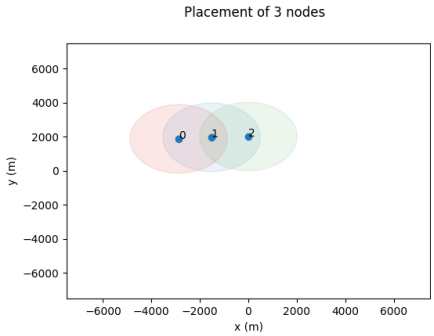
Case 4 through Case 6 leave the realm of initial setups and simulates real-world layouts. Case 4 (fig. 4.2d) in particular places eight nodes in a straight line, where a node may only communicate with two another nodes at most. Nodes at the very end of the line - such as Node 0 and Node 7 - rely on the intermediate nodes to allow packets to reach their destinations. This composition mimics a potential long-distance communication setup. In the simulation, this accounts to nearly 10 km from end-to-end.



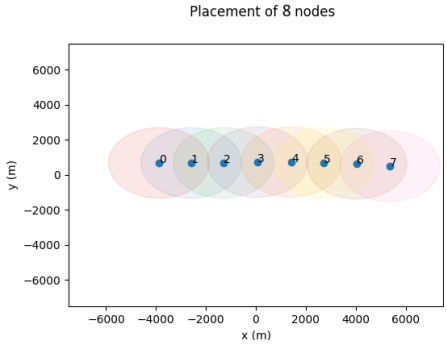
(a) Case 1 depicts two nodes in close proximity.



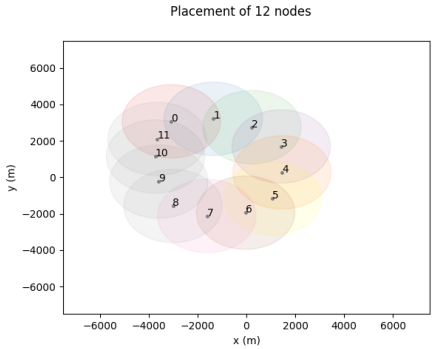
(b) Case 2 depicts three nodes in close proximity.



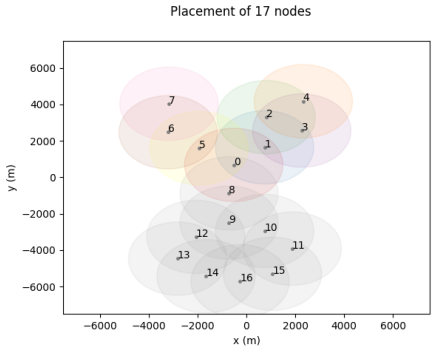
(c) Case 3 depicts three nodes, whereas two requires the third node to communicate.



(d) Case 4 depicts eight nodes in a line.



(e) Case 5 depicts twelve nodes in ring.



(f) Case 6 depicts 17 nodes spread out in a fashion that could fit within an urban environment.

Figure 4.2: Comprehensive figure of all six test cases.

Case 5 places twelve nodes in a ring pattern. Similar to the previous case, nodes require other nodes to pass their packets to the packets' destinations. The number of nodes is considerably more than before, increasing the risk for packet collision and losing information. The distance between nodes is smaller as well, the radius of the circle reaching around 3 km. See fig. 4.2e.

Finally, Case 6 presents a slightly different layout. Here, it is imagined that the nodes are placed throughout a city allowing connectivity throughout it. The setup could fit in the greater Stockholm area. This test utilizes 17 nodes, which increases the risk of packet loss even more than before but also allows packets destined to the lower half of the network to take different paths. This layout is presented in fig. 4.2f.

To ensure a consistent and controlled test setup, each test case was simulated twice; once in the unmodified, baseline Meshtasticator simulator and once in the simulator utilizing Distance Vector routing. The number of packets which were sent within the node were set to be nearly equal in both version. The simulated time remained constant for both versions, allowing easier comparison once the number of packets could be controlled for. Another controlled factor was node placement. If the nodes are not placed in precisely the same position for a specific case in both simulator versions, it could affect the test results. Fortunately, Meshtasticator offers an option to save the simulation setup, and then loading it for a separate simulation. Using this process allowed the position of the nodes for both versions to remain unchanged. Key test metrics which were taken into account when setting up the experiment were the average delay of packet transmission, average transmission air utilization, and percentage of packets collided. The average percentage of transmission utilization per packet was also calculated and taken into account to ensure the difference in packets when comparing the two versions would not affect the results as much.

Chapter 5

Results and Analysis

This chapter presents the results from the experiments described in 4.2. To prepare the reader, the presented test metrics are explained in section 5.1. In section 5.2, each case's results are displayed and explained. An analysis of the results is then given in section 5.3. The chapter wraps up by giving an analysis of the results' reliability and validity, to strengthen their feasibility, in section 5.4.

5.1 Test metrics

There are four test metrics that are of particular interest in comparing the baseline simulator to the modified, as exhibited in table 5.1 below:

Table 5.1: Relevant test metrics, unit displayed, and description.

Test metrics	**Unit**	**Description**
Average delay	Milliseconds	Time for packet to reach destination node
Average Tx air utilization/node	Percentage	How long a node transmits its packets
Average Tx air utilization/packet	Percentage	Weighted Tx air utilization
Packets collided	Percentage	Amount of packets colliding

The *average delay* is crucial to maintain or reduce. If the transmission air utilization is reduced but average delay increased considerably the trade-off is important to be aware of. The *average Tx air utilization per node* is the key metric for this experiment, as it is the proxy for energy consumption. The longer it takes a node to transmit a packet, the more energy it will require in a

real-world application. For most test cases, the number of packets transmitted in the unmodified simulator compared to the modified differ no more than 15%. In two cases, Case 4 and 5, the difference was over 20% and 43% due to difficulty aligning the amount of packets generated. To ensure this difference does not affect the results, the *average Tx air utilization per packet* was also measured. Weighing the transmission utilization gives a higher precision of the results. Finally, amount of *packets collided* is relevant to compare the two versions.

5.2 Results

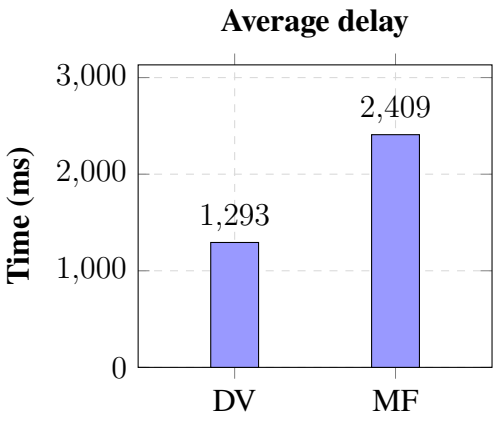
5.2.1 Case 1

Case 1 had a total of 68 and 66 packets transmitted for Distance Vector routing algorithm and Managed Flooding algorithm, respectively. For the average delay, Distance Vector routing algorithm takes only 53.7% of the delay in average to transmit a packet.

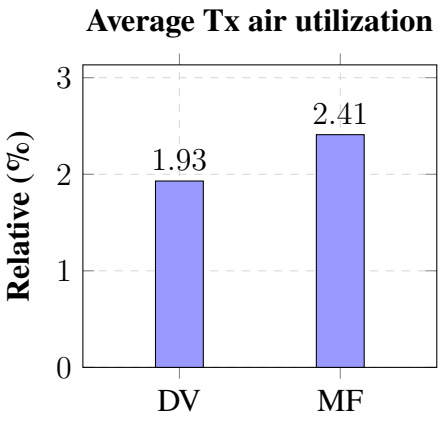
In regards to the average transmission air utilization, the new algorithm performs at 1.93% whereas Managed Flooding is at a higher 2.41%.

To ensure a fair comparison, the transmission air utilization is weighted and presented as an average transmission air utilization *per packet*. Note that the results are two orders of magnitude smaller than fig. 5.1b, which is indicated by the unit in the vertical label. Distance Vector routing requires on average 23% less air time per packet for this case.

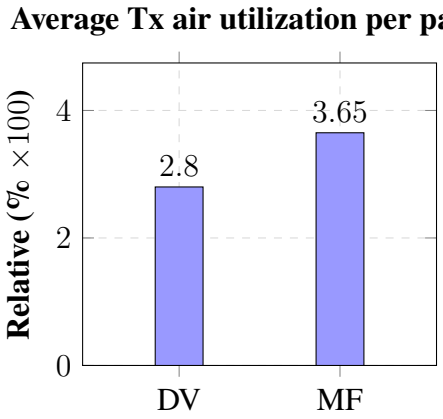
No packets collided during the simulated scenario for Case 1.



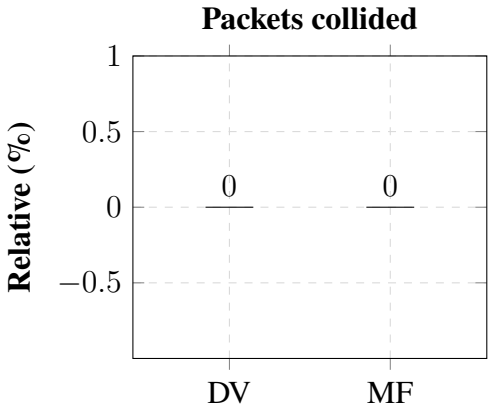
(a) Average delay for Case 1.



(b) Average transmission air utilization.



(c) Weighted average transmission air.



(d) Percentage of packets that collided.

Figure 5.1: Case 1 results, with Distance Vector (DV) and Managed Flooding (MF) routing protocols.

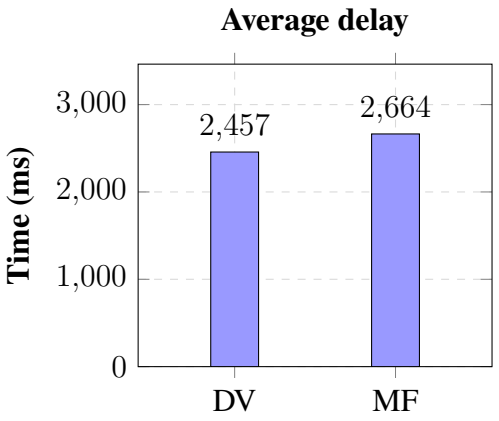
5.2.2 Case 2

In this scenario, for three nodes with full reachability in the network, a total of 98 and 100 packets were sent, respectively. For the average delay, it is notable that Distance Vector routing reaches similar levels as Managed Flooding does, despite some distance between the two.

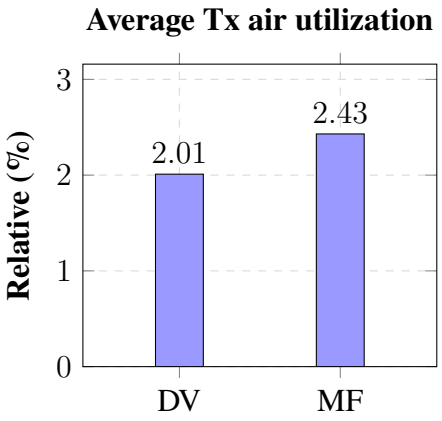
The transmission air utilization in this case indicates that the modified algorithm is somewhat higher than for the previous case. Despite this there is a considerable difference between it and the initial algorithm.

The weighted average transmission air utilization per packet is once again lower for Distance Vector routing by 15.6%.

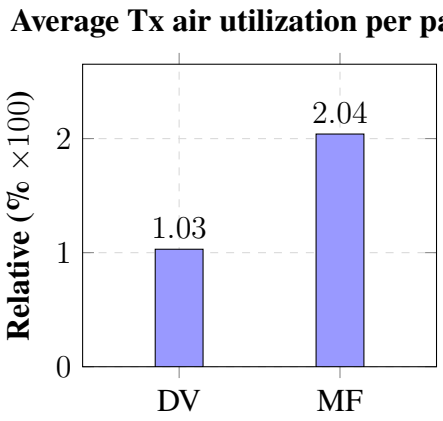
In this case, packets did collide. 1.03% of the packets in the new algorithm collided and did not reach their destination node, whereas 2.04% for base version collided.



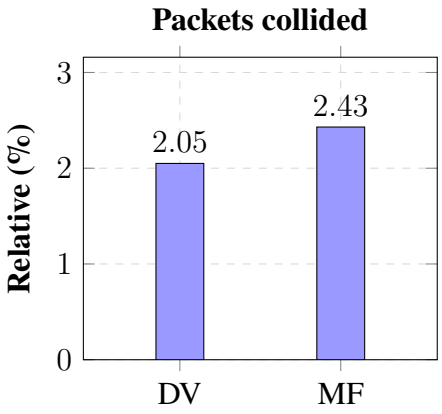
(a) Average delay for Case 2.



(b) Average transmission air utilization.



(c) Weighted average transmission air.



(d) Percentage of packets that collided.

Figure 5.2: Case 2 results.

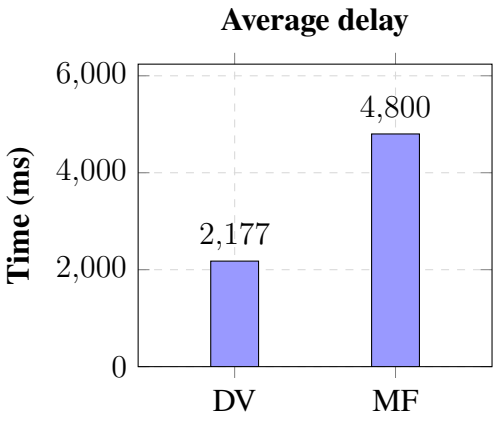
5.2.3 Case 3

Case 3, where three nodes are placed in a straight line requiring one of the nodes to act as a middleman for the other two, transmitted a total of 138 and 161 unique packets, respectively. For this constellation of nodes, Managed Flooding exhibits significantly reduced performance than Distance Vector routing does as indicated by fig. 5.3a.

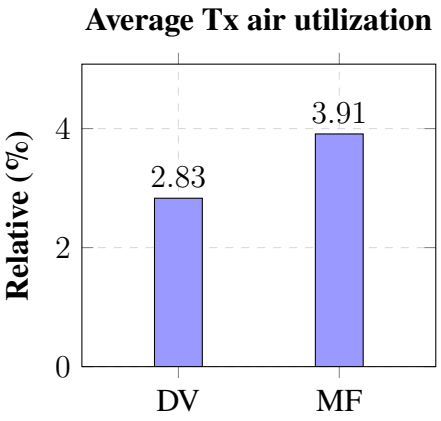
The average transmission time for both algorithms increase compared to the previous case. The difference in increase is higher for the base algorithm than the modified one as apparent in fig. 5.3b.

The trend for weighted average transmission time continues with a lower value for Distance Vector routing.

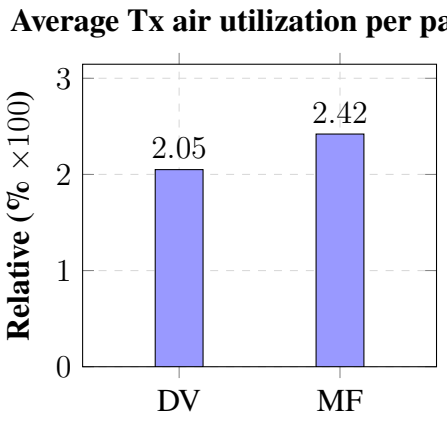
In Case 3, the percentage of packets that collide is much higher than before. For Managed Flooding in particular the increase is considerable.



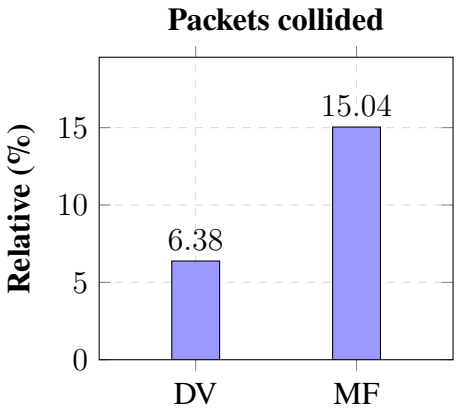
(a) Average delay for Case 3.



(b) Average transmission air utilization.



(c) Weighted average transmission air.



(d) Percentage of packets that collided.

Figure 5.3: Case 3 results.

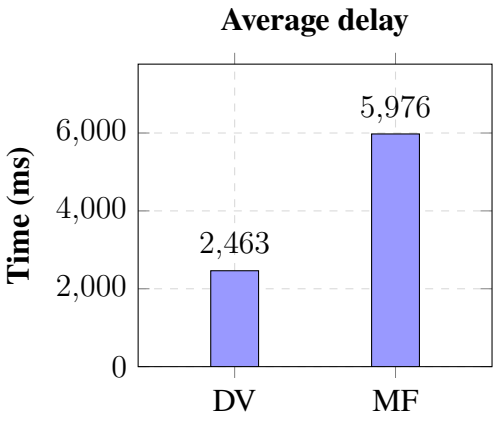
5.2.4 Case 4

Case 4, the first to introduce a real-world scenario in terms of node placement, sent a total of 824 and 644 packets for Distance Vector routing and Managed Flooding, respectively. The difference in average delay was larger for this case than the previous. Case 3 saw an increase of 220% when using Managed Flooding instead of the other algorithm; this case finds an increase of 242%.

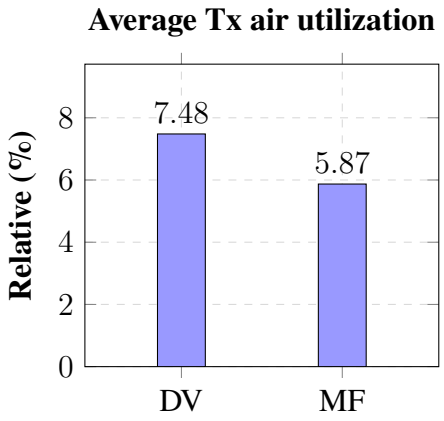
In this setup, preliminary analysis suggests that the trend shifts in terms of the transmission air utilization. Distance Vector routing achieves a higher percentage for the average compared to Managed Flooding. In this scenario, the weighted average plays a considerable role.

The weighted average transmission air utilization per packet indicates that the two algorithms are closer in results than initially shown by fig. 5.4b. Interestingly, the value in this case is much lower than all other cases.

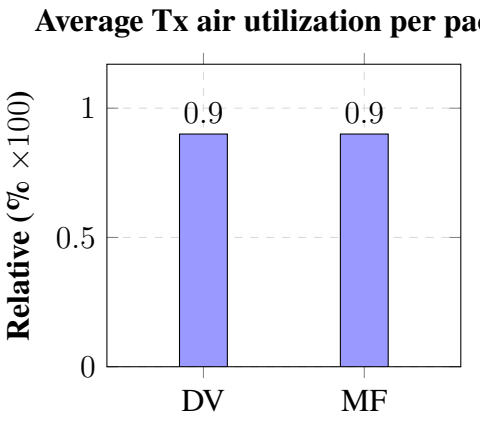
The percentage of packets that collide is higher for Managed Flooding in this case.



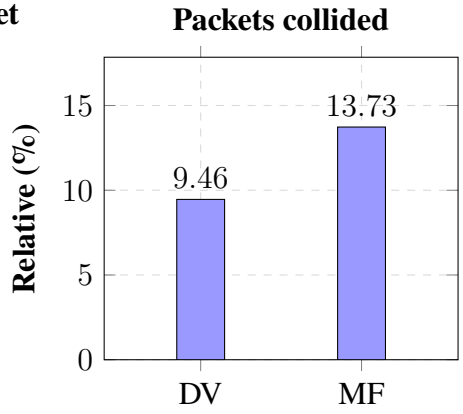
(a) Average delay for Case 4.



(b) Average transmission air utilization.



(c) Weighted average transmission air.



(d) Percentage of packets that collided.

Figure 5.4: Case 4 results.

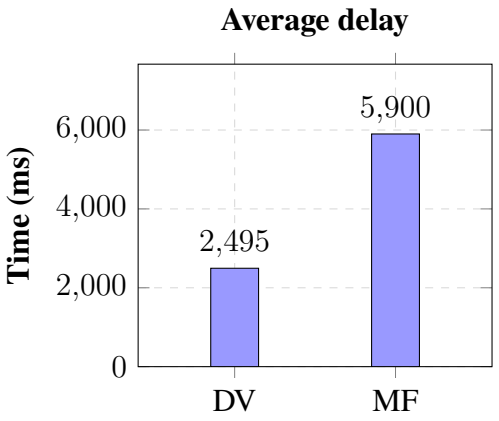
5.2.5 Case 5

This case, Case 5, places twelve nodes in a ring. Following the same order of presentation as in prior cases - Distance Vector routing first, followed by Managed Flooding - a total of 669 and 1161 packets were sent respectively. As for the average delay, Managed Flooding consumes nearly twice the time that Distance Vector routing does.

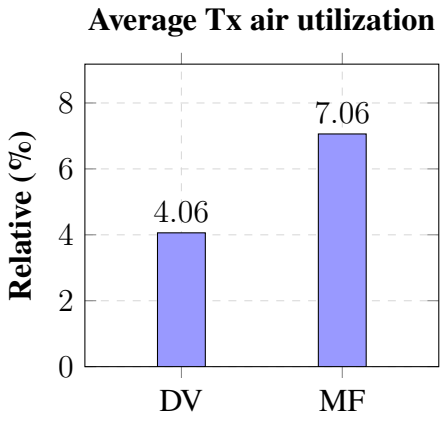
The average transmission air utilization appears higher for the modified algorithm in this case.

Finally, for the weighted transmission utilization, each packet requires even less than in the previous case. The difference between the two algorithms is nearly negligible.

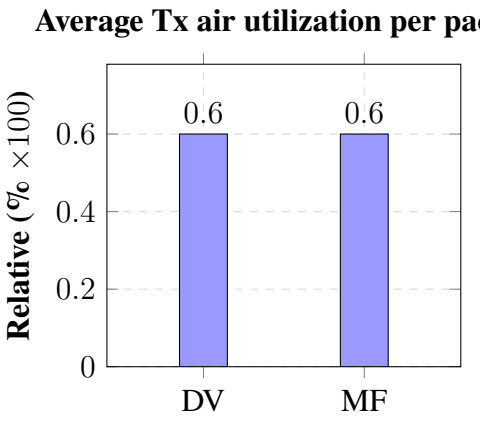
In this scenario, both algorithms have a higher level of packets colliding. Distance Vector routing compares slightly worse.



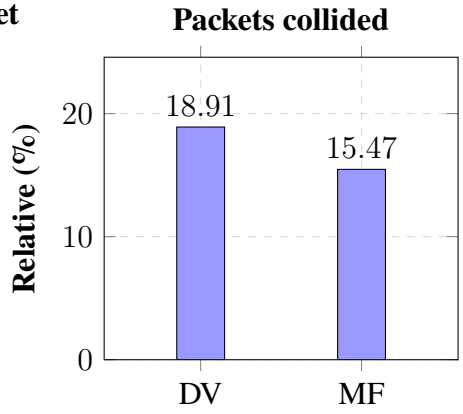
(a) Average delay for Case 5.



(b) Average transmission air utilization.



(c) Weighted average transmission air.



(d) Percentage of packets that collided.

Figure 5.5: Case 5 results.

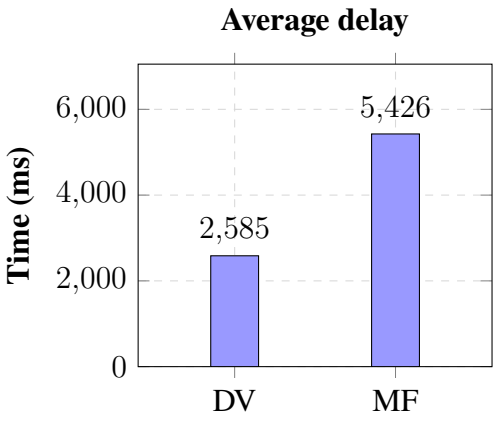
5.2.6 Case 6

This case, with its nodes spread out in a seemingly random pattern, shows some interesting results. Unique packets sent were 1395 and 1428, respectively. The trend for average delay continues, with the new algorithm achieving a considerably lower value than the old one.

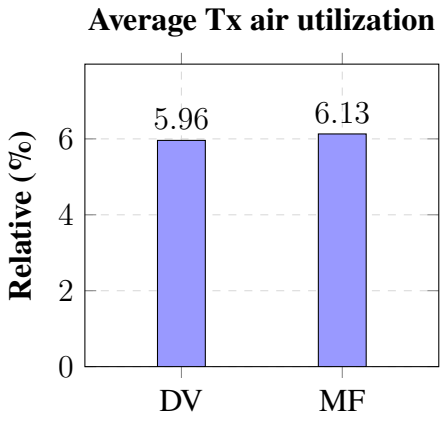
The difference for this metric is not as large as in previous cases. Distance Vector performs slightly better than Managed Flooding, when looking at the unweighted average transmission air utilization.

When comparing the weighted average transmission utilization, as with prior cases the difference becomes negligible.

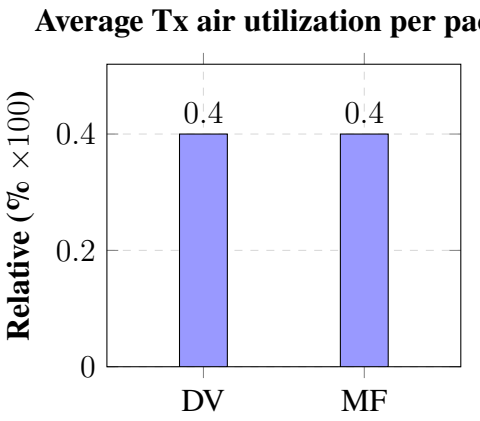
The percentage of packets that collide in this scenario is higher for Distance Vector routing, but high in general for both algorithms.



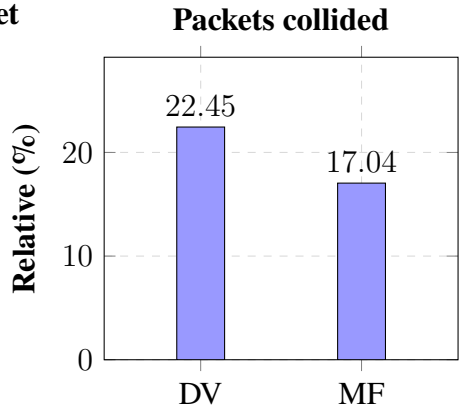
(a) Average delay for Case 6.



(b) Average transmission air utilization.



(c) Weighted average transmission air.



(d) Percentage of packets that collided.

Figure 5.6: Case 6 results.

5.3 Analysis

There are some interesting insights to be made from the results in 5.2. The analysis begins with the **average delay**, where Distance Vector routing outperformed Managed Flooding in nearly all cases. For Case 2, although it gave better results than the base algorithm, the difference was negligible. This indicates that packet transmission using Distance Vector routing, a smarter routing protocol than Managed Flooding, reduces the average time it takes for packets to reach their destination node in a variety of network constellations.

For the **average transmission air utilization**, the results were mixed. for the first three cases as well as the fifth case, Distance Vector routing performed better than Managed Flooding. Managed Flooding did considerably better than Distance Vector routing in Case 4, and the results were similar for Case 6. As mentioned earlier in chapter 4.2, to ensure that this metric would show a fair distinction between the two algorithms, the average transmission air utilization per packet was also calculated. It shows a more accurate judgment of the situation. The reason for presenting the unweighted transmission value is to create a contrast. Due to this, it is hard to draw conclusive insights using this metric.

The **average transmission air utilization per packet** shows, as mentioned above, a more accurate outcome. For the first three cases, with up to three nodes, Distance Vector routing performs better than Managed Flooding. For the remaining cases, the difference is trivial. A conclusion that can be drawn from this is that the former protocol performs better for networks with less nodes.

Finally, when comparing **packet collisions**, no collisions occur for either algorithm in Case 1. For cases 2, 3, and 4 Distance Vector routing performs better, whereas Managed Flooding does so for Case 5 and 6. Although the collection of nodes in Case 2 are not in a strict straight line, it can be approximated to be so from a perspective further away. With that in mind, cases 2 through 4 indicate a lower rate of colliding packets when placed in a straight line, if applying Distance Vector routing. In contrast, with more complex constellations as in Case 6 or increasing area as in Case 5, it could be more beneficial to apply Managed Flooding for reducing the level of packet collisions.

5.4 Reliability and Validity Analysis

A reliability analysis displays that the results are reproducible and strengthens the reliability, credibility, and conclusion of the experiment. A validity analysis establishes that the metrics tested are chosen correctly to prove or disprove the initial hypothesis.

Certain factors were controlled to ensure reproducibility. This includes using the same node placements for the same case in both algorithms' simulations. The SF was unchanged for all experiments as well. Other non-determinate factors which can fluctuate require statistical methods to ensure a decreased risk of losing reproducibility. This would be the transmission delay for packets, the transmission air utilization, and packet collision. To decrease the risk, the average for delay and transmission air utilization was used as a result metric. As there are between several hundred to thousands of packets, the average value gives a more reproducible result. As an additional step in this process, and to ensure that the packet collision-metric was not a one-time occurrence, the same simulation for the same case was conducted several times. The results of those simulations were then weighted and presented.

To ensure that the measured test metrics are indeed valid and relevant to the research question, the purpose of the experiment has to be revisited in chapter 1 and chapter 3.1. The research question, as stated in 1.2, focuses on comparing energy-efficiency for Distance Vector routing to the control algorithm, Managed Flooding. In 3.2.1 it is argued that the ToA can be used as a proxy for energy usage. For that reason, the transmission air utilization is a key metric in this experiment. Although acknowledgment, scheduling, and collision-handling were not implemented and tested in the experiments, the average delay as well as packet collision rate are vital to provide contrast to the results. As mentioned, if the delay or collision rate were to increase considerably for the modified algorithm even though the transmission air utilization remained low, the results would not provide an objective account of the algorithm.

Chapter 6

Discussion

This chapter raises a discussion of the project. It begins with a deliberation of the project's strengths and shortcomings as well as potential improvements in section 6.1. In the following section 6.2, assessment of how applicable the results are in real-world scenario is made. Finally, a reflection of how well the research question was answered, along with other project specifics, can be found in 6.3.

6.1 Experiment and Results

From this experiment, the results paint a mixed picture in regards to using Distance Vector routing as a protocol for LoRa using Meshtastic, at the very least in a simulated environment. If the only property of interest is the average delay for a packet, the protocol is an obvious choice. Once the other metrics are taken into account, then the choice of protocol is not as obvious. For use cases where networks have a few nodes, Distance Vector routing remains a clear contender. With an increasing number of nodes the routing algorithm underperforms for certain characteristics, depending on the placement of nodes. In those cases, a flooding-based algorithm performs better in general.

The experiment was conducted using Meshtasticator, a simulation software based on the Meshtastic firmware. No modifications were made to the underlying firmware or the workings of the simulation software itself - the method in which the packets are sent, received, and handled by the nodes is what was altered. The reliance on this software suite can be seen as deficiency of the project. Furthermore, although the simulator is based on well-studied modeling of the real-world radio behavior, it is vital to remember that it is *only a simulation*. A simulation provides a lower-resolution image of the

natural world, allowing experiments to conduct faster at the cost of accuracy and precision.

There are minor improvements which could give a significant rise to result accuracy. One such improvement is planning node placement based on a geographic map. By considering an applicable scenario where a series of LoRa nodes could be used and planning node placement based on it, the relative distances could then be used when placing nodes in Meshtasticator. This would bring the simulation closer to a real scenario. Additionally, using the geographic terrain (forest, mountains, urban area) and changing the simulator's path-loss model would further increase the authenticity of the results.

6.2 Real-World Applicability

Another point of discussion is how applicable the results of this experiment is to a scenario in the physical world. As mentioned previously a simulation is a model of reality, although often a low resolution one. Due to this properties that would otherwise affect the behavior of the studied system may be lost unwillingly or for the sake of a tradeoff. It is therefore difficult to determine the applicability of the results.

A natural question that arises is *what adaptations are required to make the model satisfactorily fit?* One suggestion to answer the question is to investigate an existing model for determining energy usage based on one or several proxy metrics. While transmission air utilization is strongly linked to energy usage, it alone does not give a comprehensive account. Understanding what other metrics play a part in energy expenditure of a LoRa system could improve the accuracy of the model.

6.3 Reflection

This section deals with questions pertaining to how well the research question was answered, what could be further investigated, and what goals were met.

To reiterate the research question: *Can acknowledgement, scheduling, and collision-handling be implemented with a significantly more energy-efficient routing than flooding in a LoRa mesh-network?* The main focus of this research was finding a *significantly more energy-efficient routing than flooding*. The average transmission air utilization per packet showed a difference between results for both algorithms to be between 0.2% and

22.2%. In particular, the first three cases presented a 15% to 22% increase in transmission utilization when using Managed Flooding instead of Distance Vector routing. This difference can be considered significant and this indicates that, in the domain of this simulation and for low number of nodes, there is a routing algorithm more energy-efficient than flooding.

Due to time constraints, acknowledgement, scheduling, and collision-handling were not investigated. Collisions of packets can be studied from the results, but no analysis is conducted on this property. Meshtasticator also offers a display of how packets moved from node to node which can be used to study scheduling, but no investigation was conducted here either. During implementation, a contemporary solution to packet transmission had to be made. When a node is transmitting, it cannot receive. With growing number of nodes, the risk of complete packet collision, due to busy channels, during neighbor-finding phase was particularly high. To combat this, random delays were introduced to allow packets crucial for finding neighbors to reach their final destinations. This could be interpreted as a form of scheduling, but no analysis of its impact was put into focus and thus scheduling is not considered to have been investigated in this work.

There were three goals presented in 1.4. 1) Finding a routing algorithm more energy-efficient than flooding through a literature study, 2) providing a PoC in a simulated environment, and 3) implementing transmission acknowledgement, transmission scheduling, and packet collision handling. The first two goals were met, as presented in this paper. Focus on the first two goals ultimately diverted time and resources from accomplishing the final goal. The implementation of the functionality with this algorithm is, instead, left as future work.

Chapter 7

Conclusions and Future Work

Conclusions and Future Work discusses what conclusions and insights can be drawn about this project (7.1), the limitations of the work (7.2), and finally what was left undone as well as future work (7.3).

7.1 Conclusions

Restating 6.3 and the findings from 5, it can be concluded that Distance Vector routing is a more energy-efficient routing algorithm than Managed Flooding for a small number of nodes in the network (at the very least, in a simulated environment using Meshtastic firmware). Further investigation has to be conducted to understand what other scenarios this algorithm excels in, and where it falls short.

There are a number of steps in this process which could be done differently. Reducing the research questions further and adding additional research points if time grants it would allow a more focused inspection of the main focus. Discussing the problem with those experienced of the subject field would assist in narrowing down the thesis.

7.2 Limitations

The most considerable limitation for application of the results is the usage of a simulation in the experiment. Another hindering factor for the results is founding the experiment on the Meshtastic firmware. It is possible that developing a firmware and software for the LoRa hardware from scratch would present clearer results. A better understanding of the underlying system would

also be attainable. The scope of such a project would be much larger than what this work entailed. A third limitation arising from only using software to investigate the research question is the lack of hardware. Had the experiment been applied on hardware and radio communication been tried in the physical realm, the reliance on transmission air utilization as a proxy for energy usage would not be necessary. Due to the nature of this thesis, the metric was used as a substitute for energy usage, attaining *comparative* results rather than *quantitative*.

7.3 Future work

Understanding of LoRa and its capabilities is growing increasingly fast in academia. The potential and practical use cases it offers are high, but has yet to be realized. Below are suggestions on future work based on and relating to this thesis, respectively.

7.3.1 Unfinished Work

As discussed in section 6.3, transmission acknowledgement, transmission scheduling, and packet collision handling have yet to be properly investigated. If these functions can be implemented for a routing algorithm like Distance Vector, and examining their effect on the test metrics, a more robust protocol can be argued for.

Transferring the knowledge gained from the simulation to physical hardware would be a pivotal transition in understanding how the new algorithm performs in a real-world scenario. It would also indicate whether it is a feasible, practical choice and for which network constellations.

7.3.2 Other Work

There are many tangential projects that explore the realm of LoRa with this thesis as basis. Although it was mentioned in 4.1.3 the release of Meshtastic version 2.6 introduced Next-Hop Routing. This addition to the official firmware, which the simulator is built on, is essentially following a similar process as Distance Vector routing. After an initial broadcast using Managed Flooding and receiving acknowledgement, the following packets from the same origin node can be transmitted using Next-Hop Routing since a path to the destination node now has been established. Conducting an experiment where Next-Hop Routing is used as control could yield interesting results.

For this thesis, Distance Vector routing was chosen as the algorithm to compare to the traditional flooding method. There are many different routing algorithms with varying benefits and drawbacks. Another routing algorithm close to Distance Vector routing is Link-State routing. Examining how well it performs compared to the base algorithm and Distance Vector routing could be of interest.

Applying the method of this thesis to physical hardware would also be an interesting experiment to conduct. The results would indicate more accurately whether the algorithm is fit for the intended use case, compared to simulating it.

References

[1] A. Augustin, J. Yi, T. Clausen, and W. Townsley, "A study of lora: Long range & low power networks for the internet of things," *Sensors*, vol. 16, no. 9, p. 1466, Sep. 2016. doi: 10.3390/s16091466 [Pages 2, 7, 8, 9, and 11.]

[2] C. Ebi, F. Schaltegger, A. Rüst, and F. Blumensaat, "Synchronous lora mesh network to monitor processes in underground infrastructure," *IEEE Access*, vol. 7, pp. 57 663–57 677, 2019. doi: 10.1109/ACCESS.2019.2913985 [Pages 2 and 18.]

[3] "Kth research and the un global sustainability goals," Mar. 2021, accessed on June 9, 2025 (Through web.archive.org). [Online]. Available: https://web.archive.org/web/20250213030134/https://www.kth.se/en/om/miljo-hallbar-utveckling/forskning-miljo-hallbar-utveckling/kth-s-forskning-och-fn-s-globala-hallbarhetsmal-1.863900 [Page 4.]

[4] "The 17 goals," Jun. 2025, accessed on June 9, 2025 (Through web.archive.org). [Online]. Available: https://web.archive.org/web/20250608143326/https://sdgs.un.org/goals [Page 4.]

[5] J. S. A. Joakim Eriksson, "Investigating the practical performance of the lorawan technology," Master's thesis, 2017. [Pages 7 and 10.]

[6] "Lorawan - what is it?" LoRa Alliance, Tech. Rep. [Pages 7 and 12.]

[7] L. V. Davide Magrin, Marco Centenaro, "Performance ealuation of lora ntworks in a smart city scenario," in *Autho9r7iz8e-d1 l-ic4e6n7s3e-d8 u9s9e9 li-m0i/t1e7d*, no. reception. IEEE, 2017. [Pages 7 and 10.]

[8] M. S. Riccardo Berto, Paolo Napoletano, "A lora-based mesh network for peer-to-peer long-range communication," *Sensors*, vol. 21, no. 13, p. 4314, Jun. 2021. doi: 10.3390/s21134314 [Page 7.]

[9] U. R. Martin Bor, John Vidler, "Lora for the internet of things," Feb. 2016. [Pages 7 and 10.]

[10] R. P. Centelles, R. Meseguer, F. Freitag, L. Navarro, S. F. Ochoa, and R. M. Santos, "Loramoto: A communication system to provide safety awareness among civilians after an earthquake," *Future Gener. Comput. Syst.*, vol. 115, pp. 150–170, 2021. doi: 10.1016/J.FUTURE.2020.07.040 [Pages 8 and 18.]

[11] *SX1276/77/78/79 - 137 MHz to 1020 MHz Low Power Long Range Transceiver*, Semtech, Aug. 2016. [Online]. Available: https://www.mouser.com/datasheet/2/761/sx1276-1278113.pdf [Pages 8 and 10.]

[12] E. D. Ayele, C. Hakkenberg, J. P. Meijers, K. Zhang, N. Meratnia, and P. J. Havinga, "Performance analysis of lora radio for an indoor iot applications." IEEE, 2017. doi: ng [Page 10.]

[13] H.-C. Lee and K.-H. Ke, "Monitoring of large-area iot sensors using a lora wireless mesh network system: Design and evaluation," *IEEE Transactions on Instrumentation and Measurement*, vol. 67, pp. 2177–2187, 2018. doi: 10.1109/TIM.2018.2814082 [Pages 10 and 18.]

[14] A. W.-L. Wong, S. L. Goh, M. K. Hasan, and S. Fattah, "Multi-hop and mesh for lora networks: Recent advancements, issues, and recommended applications," *ACM Computing Surveys*, vol. 56, no. 6, pp. 1–43, Jan. 2024. doi: 10.1145/3638241 [Pages 10, 11, and 18.]

[15] J. R. Cotrim and J. H. Kleinschmidt, "Lorawan mesh networks: A review and classification of multihop communication," *Sensors*, vol. 20, no. 15, p. 4273, 2020. doi: 10.3390/S20154273 [Pages 10 and 18.]

[16] D. Lundell, A. Hedberg, C. Nyberg, and E. Fitzgerald, "A routing protocol for lora mesh networks." Chania, Greece: IEEE, 2018. doi: 10.1109/WoWMoM.2018.8449743. ISBN 978-1-5386-4726-4 pp. 14–19. [Pages 10, 12, and 18.]

[17] I. T. Union, "Radio regulations," 2020. [Pages 10 and 11.]

[18] L. A. T. C. R. P. Workgroup, *LoRaWAN Regional Parameters*, LoRa Alliance, May 2021. [Page 11.]

[19] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, "Long-range communications in unlicensed bands: the rising stars in the iot and smart

city scenarios," *IEEE Wireless Communications*, vol. 23, pp. 60–67, 2016. doi: 10.1109/MWC.2016.7721743 [Pages 12, 13, and 14.]

[20] "Star network topology," Mar. 2016, accessed on April 6, 2025 (Through web.archive.org). [Online]. Available: https://web.archive.org/web/20151222163638/http://teach-ict.com/gcse_computing/ocr/215_communications_networking/network_topologies/miniweb/pg4.htm [Pages 12 and 13.]

[21] I. M. J.C. Mackin, *MCSA/MCSE Self-Paced Training Kit (Exam 70-291)*. Microsoft Press, 2003, ch. Chapter 1. ISBN 0735614393 [Page 13.]

[22] L. Jiang, "Comparision of lora and nb-iot in terms of connectivity," Master's thesis, 2020. [Pages 13 and 15.]

[23] "Lorawan™ specification," LoRa Alliance, Tech. Rep., Jan. 2015. [Pages 13 and 14.]

[24] "Ieee standard for telecommunications and information exchange between systems - lan/man specific requirements - part 15: Wireless medium access control (mac) and physical layer (phy) specifications for low rate wireless personal area networks (wpan)," *IEEE Std 802.15.4-2003*, pp. 1–680, 2003. doi: 10.1109/IEEESTD.2003.94389 [Page 13.]

[25] A. Cilfone, L. Davoli, L. Belli, and G. Ferrari, "Wireless mesh networking: An iot-oriented perspective survey on relevant technologies," *Future Internet*, vol. 11, no. 4, p. 99, Apr. 2019. doi: 10.3390/fi11040099 [Page 15.]

[26] S. K. S. Ujjwal Sinha, Vikas Kumar, "Routing algorithms : A review." [Online]. Available: https://www.mouser.com/datasheet/2/761/sx1276-1278113.pdf [Pages 15, 16, and 18.]

[27] L. R. F. Jr., "Network flow theory," RAND Corporation, Tech. Rep. P-328, 1956. [Page 16.]

[28] R. E. Bellman, "On a routing problem," *Quarterly of Applied Mathematics*, vol. 16, no. 1, pp. 87–90, 1958. [Page 16.]

[29] C. Hedrick, "Routing information protocol," Jun. 1988, accessed on June 12, 2025 (Through web.archive.org). [Online]. Available: https://web.archive.org/web/20241001000000*/https://www.rfc-editor.org/rfc/pdfrfc/rfc1058.txt.pdf [Pages 16 and 17.]

[30] "Meshtastic," Jun. 2025, accessed on June 7, 2025 (Through web.archive.org). [Online]. Available: https://web.archive.org/web/20250605081203/https://meshtastic.org/ [Pages 17 and 22.]

[31] "Meshtastic ground record," Jun. 2025, accessed on June 7 2025 (Through web.archive.org). [Online]. Available: https://web.archive.org/web/20250604071720/https://meshtastic.org/docs/overview/range-tests/#current-ground-record-331km [Page 17.]

[32] "Meshtasticator," Jun. 2025, accessed on June 7, 2025 (Through web.archive.org). [Online]. Available: https://web.archive.org/web/20250527172452/https://github.com/meshtastic/meshtasticator [Pages 17 and 23.]

[33] H. Huh and J. Y. Kim, "Lora-based mesh network for iot applications." Limerick, Ireland: IEEE, 2019. doi: 10.1109/WF-IoT.2019.8767242. ISBN 978-1-5386-4981-7 pp. 524–527. [Page 18.]

[34] B. Sartori, S. Thielemans, M. Bezunartea, A. Braeken, and K. Steenhaut, "Enabling rpl multihop communications based on lora." Rome, Italy: IEEE, 2017. doi: 10.1109/WiMOB.2017.8115756. ISBN 978-1-5386-3840-8 pp. 1–8. [Page 18.]

[35] J. M. Solé, R. P. Centelles, F. Freitag, and R. Meseguer, "Implementation of a lora mesh library," *IEEE Access*, vol. 10, pp. 113 158–113 171, 2022. doi: 10.1109/ACCESS.2022.3217215 [Page 18.]

[36] K. P. White and R. G. Ingalls, "Introduction to simulation," in *2015 Winter Simulation Conference (WSC)*. IEEE, Dec. 2015. doi: 10.1109/wsc.2015.7408292 pp. 1741–1755. [Page 23.]

[37] "Python docs: Random," Jun. 2025, accessed on jun 24, 2025 (Through web.archive.org). [Online]. Available: https://web.archive.org/web/20250618105538/https://docs.python.org/3/library/random.html [Page 24.]

Appendix A

Python libraries

Table A.1: Python libraries and their respective versions used in the test setup's virtual environment.

Library	Version
async-timeout	4.0.3
bleak	0.22.3
certifi	2025.1.31
charset-normalizer	3.4.1
contourpy	1.3.1
cycler	0.12.1
fonttools	4.55.8
idna	3.10
kiwisolver	1.4.8
matplotlib	3.8.4
meshtastic	2.5.11
numpy	1.26.4
packaging	24.2
pandas	1.5.3
pillow	11.1.0
protobuf	5.29.3
pubsub	0.1.2
pyobjc-core	10.3.2
pyobjc-framework-Cocoa	10.3.2
pyobjc-framework-CoreBluetooth	10.3.2
pyobjc-framework-libdispatch	10.3.2
pyparsing	3.2.1
Pypubsub	4.0.3
pyserial	3.5
python-dateutil	2.9.0.post0
pytz	2025.1
PyYAML	6.0.2
requests	2.32.3
scipy	1.14.0
simpy	4.1.1
six	1.17.0
tabulate	0.9.0
typing_extensions	4.12.2
urllib3	2.3.0

TRITA-EECS-EX-2026:429
Stockholm, Sweden 2026

www.kth.se