Bruno Carneiro da Cunha

LoRaNet: Implementation of a LoRa Mesh Network

São Paulo

2022

Bruno Carneiro da Cunha

LoRaNet: Implementation of a LoRa Mesh Network

Capstone Project presented to the Computer Science Department of the Institute of Mathematics and Statistics of the University of São Paulo, as a partial requirement to obtain the degree of Bachelor of Science in Computer Science.

University of São Paulo

Institute of Mathematics and Statistics

Computer Science Department

Supervisor: Batista, D. M., Ph.D.

São Paulo

2022

Acknowledgements

Firstly, I would like to thank Daniel Batista, my advisor at USP, for leading our research work these last two years, setting an excellent example as a teacher and a researcher.

I would also like to thank my wife, my son, and family for supporting me during my undergraduate years, with their love, patience and understanding.

Abstract

LoRa radios are very useful for Internet of Things (IoT) devices, transmitting messages across several kilometers while using very low power. LoRaWAN is the industry standard for the upper layers of LoRa communications, but it is limited by the range of gateways, which can be costly to deploy in advance, and doesn't feature peer-to-peer communication between end-devices. This work suggests a protocol stack, called LoRaNet, for building a mesh network between LoRa devices using the AODV routing protocol and IPv4 addresses. A prototype implementation was developed in C/C++ for the ESP32 board using the Arduino IDE, although important features such as route repairing are still missing. Initial tests have shown that route latency and overhead increase linearly with the number of hops, and that can be a limiting factor for applications, along with concerns regarding duty cycle regional restrictions and battery usage. This project, however, has shown that multi-hop peer-to-peer transmissions using LoRa end-devices is a possible option for expanding the coverage of LoRa LPWANs for up to tens of kilometers.

List of Figures

List of Tables

List of abbreviations and acronyms

LoRa	Long Radio
IoT	Internet of Things
LPWAN	Low-power wide-area network
MANET	Mobile ad hoc network
TTN	The Things Network
AODV	Ad hoc On Demand Distance Vector Routing
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
AP	Access point
STA	Station
WLAN	Wireless Local Area Network
WANET	Wireless ad hoc network
LS	Link-State
DV	Distance-Vector
RREQ	Route Request
RREP	Route Reply
RERR	Route Error
RAM	Random-Access memory
ACK	Acknowledgment
SF	Spreading Factor
RAM	Data Random-Access memory
RFC	Request for Comments

Contents

1 Introduction

LoRa (long range) radios are very useful for Internet of Things (IoT) devices, transmitting messages across several kilometers of distance, while using very low power [1]. This proprietary spread-spectrum modulation technique, owned by Semtech [1], is used on smart cities devices, smart homes and buildings, agriculture and many other areas, enabling the creation of low-power wide-area networks (LPWAN) [2].

Since LoRa defines only the physical layer of the protocol stack, other protocols were developed to define the upper layers of the communication. LoRaWAN is the best known protocol created for such purpose, and is the de facto standard for message exchanging using LoRa-enabled devices. While LoRaWAN is very useful for most applications, secure and well maintained, it has one major limitation which is the reliance on **gateways**. End-node devices on a LoRaWAN network need to be within reach of a gateway, and the deployment of these devices can become costly and limit the range of a LPWAN [3].

This work introduces a new stack of protocol implementations called **LoRaNet**, which creates a mobile ad hoc network (MANET) using LoRa radios, allowing end-node devices to communicate directly with one another without the use of gateways, and to relay messages for other nodes, extending the range and versatility of current LoRa networks.

The first goal of this study is to develop a software library for Arduino [4] devices which can be used to transmit and receive messages on LoRaNet. The Arduino environment was chosen because it is one of the most common IoT platforms on which prototype applications are built [5], and is widely supported by manufacturers of development boards and by its active community.

The next goal is to assess the viability of this type of network using the ESP32 [6] board. After ensuring that route establishment works as intended, the performance of a prototype route will be evaluated to generate simple measurements such as latency and memory usage. Application developers can interpret this data to determine if an application could benefit from the use of this library.

1.1 Background

The problem that LoRaNet addresses is mainly the problem of network coverage for LoRa LPWANs. Commonly, when LoRaWAN is used, nodes can transmit sensor data to applications and receive commands, as long as they are within reach of a gateway. The Things Network (TTN) gateways can be found on several urban areas, but coverage can

[1] https://www.semtech.com

be sparse, even in such places [7]. This limits the range of applications that can be built around this stack of protocols. Additionally, gateways are not battery-powered and need a permanent Internet connection at all times, requirements which can be hard to meet in remote locations.

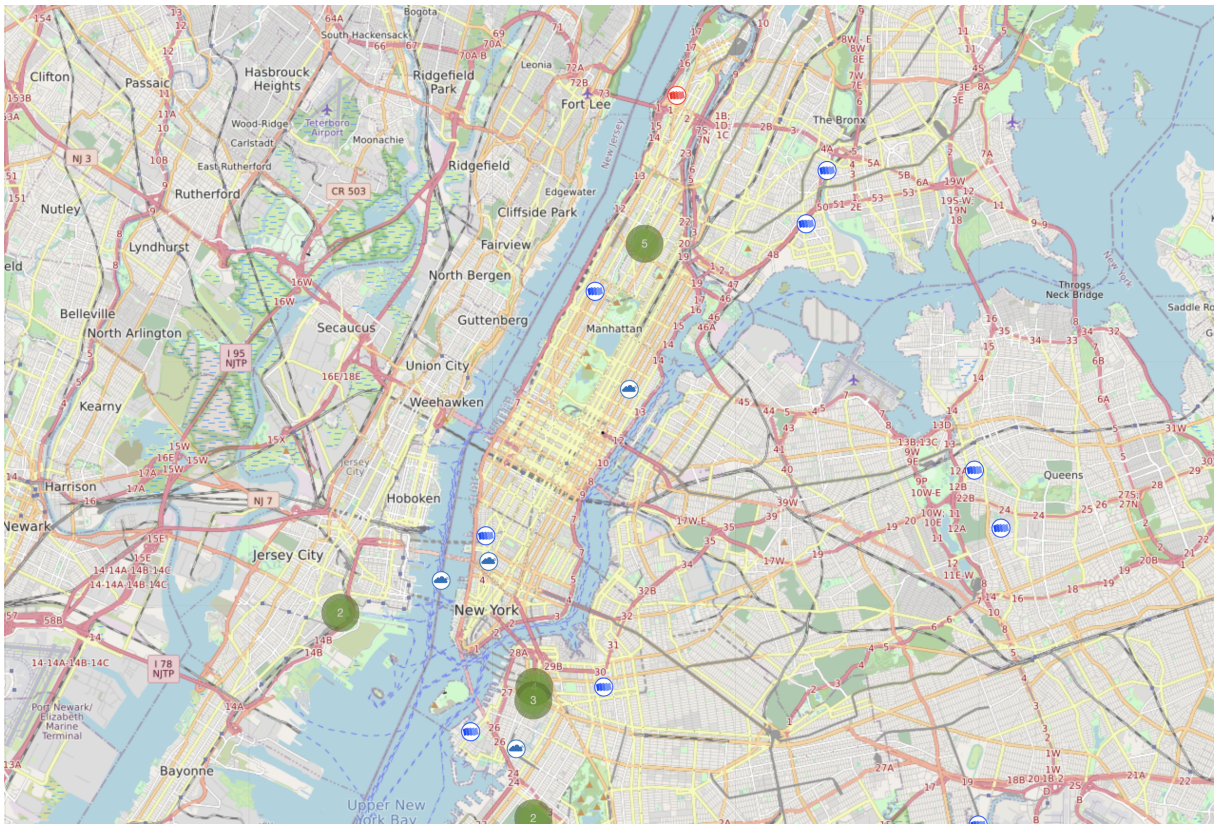


Figure 1 – TTN gateways around the area of Manhattan Island.
Blue and red icons are gateways, and green circles are a group of gateways located on the same area. **Screenshot captured from:** https://ttnmapper.org/

Another issue with current LoRa LPWANs is the lack of protocols for applications that don't necessarily communicate through the Internet. LoRaNet uses IPv4 for node addressing, features acknowledgment for link layer transmissions, and multi-hop routing of messages, all features which are valuable for applications that may exchange messages between end-node devices.

Addressing these topics, the present work aims to increase the versatility of LoRa devices, making it easier to create long range networks which could reach tens of kilometers without the need for infrastructure, while also enabling true mobile applications, where nodes can move freely and routes can be altered as needed.

1.2 Related Work

Lundell et al. [8] proposes a protocol for extending the coverage of LoRaWAN networks. It does so by drawing features from both Hybrid Wireless Mesh Protocol (HWMP) and Ad hoc On Demand Distance Vector Routing (AODV). However, their protocol routes messages between gateways, and not between end-devices, aiming to maintain compatibility with currently deployed devices.

ZigBee is based on the IEEE 802.15.4 standard [9] that defines low-rate wireless personal area networks (WPAN). It usually operates on the 2.4 GHz band, and has a common range of about 10-100m, featuring mesh networking for longer distance communications. ZigBee devices are often used in home automation, wireless sensor networks and industrial control systems.

1.3 Overview

- Chapter 2 introduces the basic concepts that will be used throughout the text, including networking models, IP addressing, routing and LoRa.
- Chapter 3 explains the implementation of LoRaNet and the architecture of its stack of protocols.
- Chapter 4 describes the experiments performed, along with the data obtained from these experiments.
- Chapter 5 analyzes the most important points from the previous chapters.
- Chapter 6 concludes this work, and explores future additional features for LoRaNet.

2 Foundations

This chapter outlines most of the technical knowledge used on the project. The concepts referenced here must be familiar to the reader, so that they can work through the next chapter without any issues.

2.1 Network Protocol

Most of the present work describes the **protocols** implemented in order for communication to be possible between LoRa devices. But first, what is a protocol?

> A network protocol is similar to a human protocol, except that the entities exchanging messages and taking actions are hardware or software components of some device (for example, computer, smartphone, tablet, router, or other network-capable device). [10, p. 8]

That is to say, a network protocol consists of *rules* that describe expected behavior from the hosts when communicating. If a person arrives in a room full of unknown people and greets everyone "Hello!", the typical response would involve a "Hi" or "Hello" from one or various of the recipients of that first greeting. In networking, there are various protocols that implement exactly the HELLO packet, see Open Shortest Path First (OSPF) [11], a routing protocol, for an example. Other human behaviors, such as not interrupting when someone else is speaking, are also mimicked by network protocols.

A protocol can implement a set of **services**. Internet Protocol (IP) [12], for example, is mainly remembered for its *addresses* that are used to discriminate one host from the other. Ethernet [13] implements *collision detecting* and recovery from this error. Wireless protocols may implement *acknowledgment*, when a receiver warns the sender it has successfully received a message. Different protocols will implement different services, and sometimes the same service can be implemented by more than one protocol at the same time.

Protocols are the core of computer networking, and careful protocol design is key to efficient communication.

2.2 Layering

In order to make very complex computer networks more modular and maintainable, it is common for protocols to be implemented in **layers**. Whenever using more than one

protocol, it is common to call the various layers a **protocol stack**.

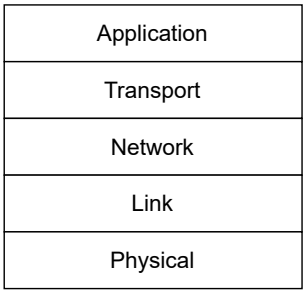


Figure 2 – The Internet protocol stack.

As seen above, in the abstraction the multiple layers are *stacked* on top of each other, but on a real transmission each header and the payload are sent and read sequentially. The Internet protocol stack is made of the five layers represented in figure 2. The following sections will provide a short introduction to each one.

2.2.1 Application Layer

This layer is used by applications to transmit their headers and payload. Some of the most relevant Internet protocols are application protocols, such as HTTP (Hypertext Transfer Protocol) [14], DNS (Domain Name System) [15][16], SMTP (Simple Mail Transfer Protocol) [17], and NTP (Network Time Protocol) [18]. Each packet conveyed by this layer is called a *message*.

LI	VN	Mode	Strat	Poll	Prec
Root delay					
Root dispersion					
Reference ID					
Reference timestamp (64)					
Origin timestamp (64)					
Receive timestamp (64)					
Transmit timestamp (64)					
Extension field (optional)					
Key identifier					
Message digest (128)					

Figure 3 – An example of an application layer protocol header, the NTP header.

2.2.2 Transport Layer

There are two main transport protocols: TCP[19] and UDP[20]. Transmission Control Protocol (TCP) is a connection-oriented protocol, providing services like delivery guarantee, segmentation of messages, flow control and others. User Datagram Protocol (UDP) is message-oriented, meaning it is connectionless and provides no guarantee of delivery.

Both of these protocols also write **port numbers** on their headers. On a single host, many applications may want to communicate at the same time with remote hosts, these applications are distinguished by addressing each one with a port number.

It is conventional to call a transport-layer packet a *segment*.

Figure 4 – UDP header format.

2.2.3 Network Layer

This is arguably the most important layer, and it is home to the protocol which bears the name of the most famous network in the world, the Internet Protocol (IP). Every data transmission on the Internet must use IP, and the packets on this layer are called *datagrams*. Ideally, each host on the Internet is known by its address, commonly a 32-bit integer when using IPv4, split into four *bytes* and formatted as follows.

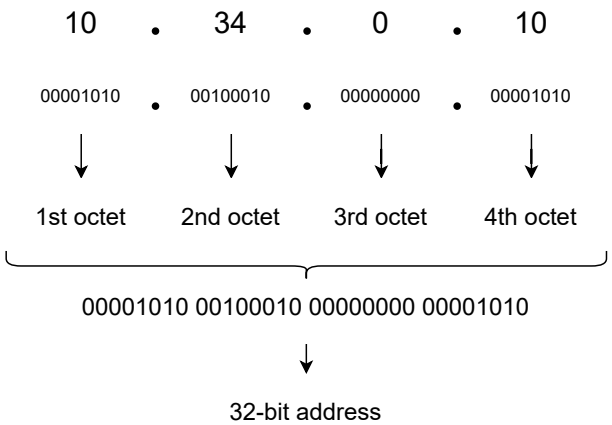


Figure 5 – IPv4 address format.

A newer version of IP, known as IPv6[21], identifies each node with 128-bit addresses, and is slowly replacing IPv4 as the standard protocol for data packets on the Internet.

The network layer also contains routing protocols, which Internet routers use to configure the best paths for a transmission.

2.2.4 Link Layer

The link layer handles only single-hop transmissions. It is only used by a host or router to move a packet from itself to the next hop, or to receive a packet from a neighbor host or router. It may hold no information about the end destination or source of a message, and the packets at this level are called *frames*.

This layer may provide reliable delivery, collision handling or other services. The most common link layer protocols are Ethernet and WiFi protocols.

2.2.5 Physical Layer

Finally, on the physical layer, we have protocols that move *bits* from one end of the link to the other end. LoRa is a physical layer protocol.

2.3 Mobile ad hoc networks

The most common home or enterprise Wi-Fi networks use what is called infrastructure mode in the IEEE 802.11[22] protocol definition. In infrastructure mode, all stations (STAs) transmit and receive frames only from the access point (AP), discarding all other intercepted frames. This model works very well because there is no need to establish a route for each frame. However if there is any issue with the AP, or if the STA moves too far from the AP, the STA will not be able to transmit and receive from the WLAN anymore.

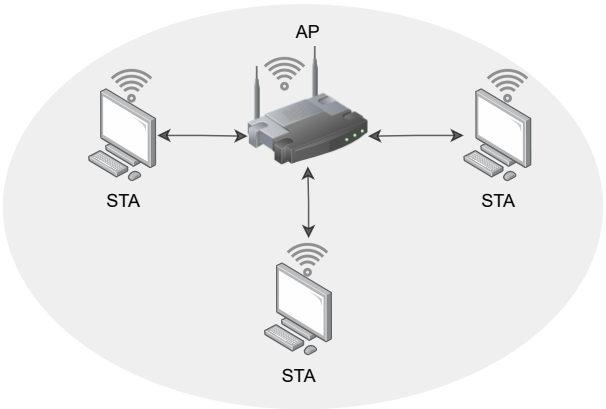


Figure 6 – IEEE 802.11 infrastructure mode.

There is another 802.11 operation mode, though, and that is called the **ad hoc** mode. In ad hoc mode, each device communicates directly with each other, without any reliance on existing infrastructure. The topology of the network is decentralized and dynamic, and nodes may forward data to form multi-hop wireless routes.

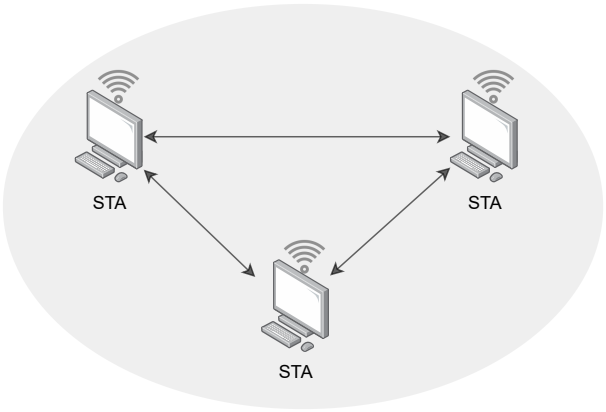


Figure 7 – IEEE 802.11 ad hoc mode.

This type of network is known as a wireless ad hoc network (WANET), or **mobile ad hoc network** (MANET). In a MANET, nodes are mobile and the topology of the network may change anytime due to changes in connectivity between nodes.

2.4 Low Power Wide Area Network

A Low Power Wide Area Network (LPWAN) is a relatively new term, which was coined to describe network technologies that transmit small data packets over long ranges, with low energy consumption, and low cost of deployment. [23].

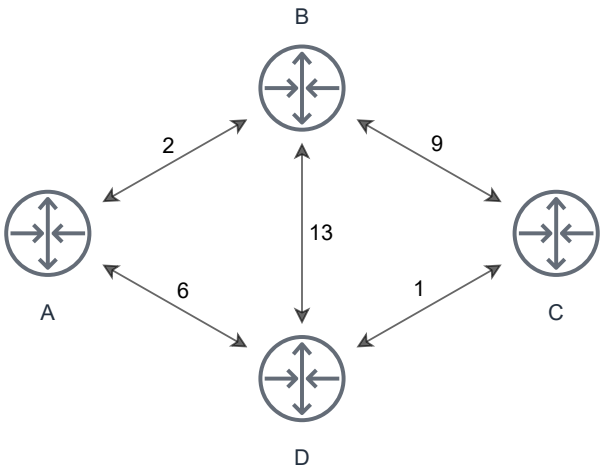
Sigfox, LoRaWAN, and NB-IoT are leading LPWAN technologies. They usually operate on lower frequencies than cellular or Wi-Fi networks, hence their low data rate.

2.5 Routing algorithms

Of major importance to this work is the issue of routing packets, specially in ad hoc networks. A good routing algorithm will create routes that have the least cost, while keeping communications overhead to a minimum.

2.5.1 Link-State Routing

In a link-state (LS) algorithm, nodes need to periodically or continuously broadcast to all the other nodes the information about their direct neighbors. Only when a node has knowledge of all the network link costs will it be able to compute the best routes to each node, usually applying Dijkstra's algorithm [24].



Link-State Packets (sources, destinations and costs)

A		B		C		D	
B	2	A	2	B	9	A	6
D	6	C	9	D	1	C	1
		D	13			B	13

Figure 8 – Sample subnet and link-state packets.

LS algorithms can sometimes generate a large amount of routing overhead, since all the information about the link costs needs to flood the network.

2.5.2 Distance-Vector Routing

Another class of routing algorithms is the Distance-Vector (DV) algorithm. In distance-vector routing, a node does not need to have awareness of the entire network, but only of its direct neighbors. To set up a route, a node will exchange messages with its neighbors, and they in turn will do the same iteratively until the algorithm stops.

Sometimes DV algorithms are called *decentralized*, because each node will compute the best routes individually using only the information about their direct neighbors, and the information given by their direct neighbors. And sometimes LS algorithms are called *centralized*, because computation of best routes will only begin when a node has the information about all the link costs on the network.

2.6 Proactive and Reactive protocols

Nodes that use *proactive routing protocols* are continuously flooding their neighbors with routing packets, without any consideration to the actual need for routes. They are more suitable to large bandwidth networks, where overhead is not a major concern.

Networks with low data rates usually benefit more from *reactive routing protocols*. The nodes on such networks will flood their neighbors with routing packets when the need for a route arises, only then will a node attempt to find the best route for a given destination.

2.7 AODV

Ad hoc On-Demand Distance Vector (AODV) is a reactive routing protocol for wireless ad hoc networks, intended for use by mobile nodes [25]. AODV features low network utilization, low processing and quick adaptation to dynamic link conditions. It ensures loop freedom[1] using destination sequence numbers.

The following are the main message types associated with this protocol.

2.7.1 Route Request (RREQ)

A node broadcasts a RREQ when it needs a route to a destination and does not have one available. Neighboring nodes will rebroadcast the RREQ packet as long as they do not have an active route to the destination, or are not the destination themselves.

An originating node will also attempt to reduce network-wide dissemination of RREQs by using an expanding ring search technique. In this technique, the node will start with a lower time to live (TTL)[2] value, and increase this value only if the first attempts at creating a route are not successful.

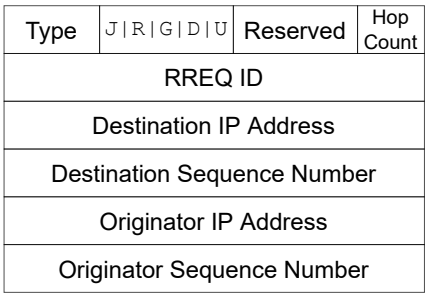


Figure 9 – RREQ header.

When a node receives a RREQ, it creates or updates a route to the previous hop as well as to the originator IP address. This way, when a RREQ is traversing the network

[1] "*A routing-table loop is a path specified in the nodes' routing tables at a particular point in time that visits the same node more than once before reaching the intended destination.*"[26]

[2] TTL is an unsigned integer added to the IP header, which is decremented every time that packet reaches a new host. Once the TTL value reaches zero the packet must be discarded, as this prevents packets from being transmitted endlessly between hosts.

towards the destination, a reverse route to the originator IP address is being set up by the intermediate nodes.

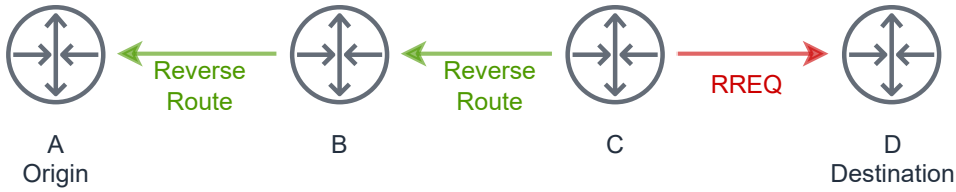


Figure 10 – Reverse Route configuration.

2.7.2 Route Reply (RREP)

The Route Reply message is generated by a node when it is the destination, or it has an active route to the destination. The RREP is then unicast back to the RREQ originator, along the reverse route established by the intermediate nodes.

Type	R | A	Reserved	Prefix Size	Hop Count
Destination IP Address				
Destination Sequence Number				
Originator IP Address				
Lifetime				

Figure 11 – RREP header.

Similarly to the RREQ processing, when a node receives a RREP it creates or updates a route to the previous hop as well as to the destination IP address.

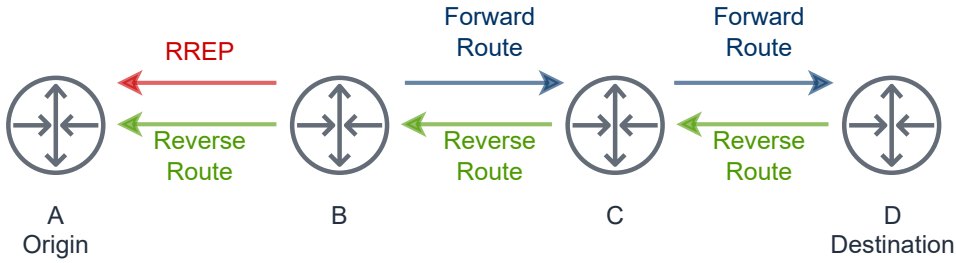


Figure 12 – Forward Route configuration.

2.7.3 Route Error (RERR)

Whenever a node detects a route break of any kind, for example if it receives a data packet destined to a node for which it does not have an active route, or if it detects

a link break for the next hop of an active route, then a node may choose to either try to repair the route or generate a RERR message.

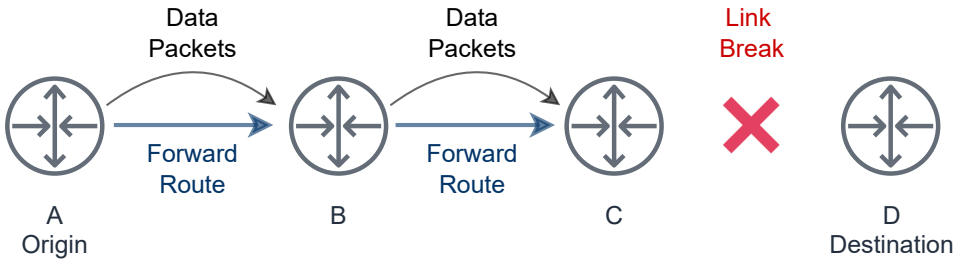


Figure 13 – Data packets cannot be forwarded from C to D due to a link breakage.

When repairing a link break, a node will broadcast a RREQ to the destination. If however, the repairing node does not receive a RREP at the end of the discovery period, then a RERR message may be unicast back to the previous hop or broadcast depending on the situation.

Type	N	Reserved	Dest Count
Unreachable Destination IP Address			
Unreachable Destination Sequence Number			
Unreachable Destination IP Address (2)			
Unreachable Destination Sequence Number (2)			
...			

Figure 14 – RERR header with a variable number of unreachable destinations.

A node that receives a RERR message will invalidate any existing routes to that destination, and may choose to unicast or broadcast the RERR depending on whether it is an intermediate node in an active route for that destination.

2.8 LoRa

LoRa is a modulation technique developed by Semtech, a company based in France, and a founding member of the LoRa Alliance. It uses license-free radio frequencies to enable long range communication for LPWANs.

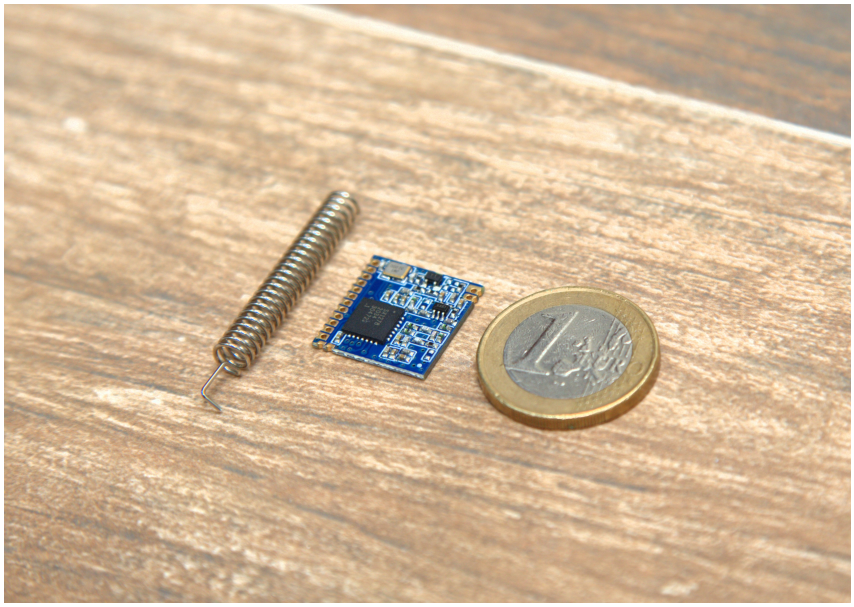


Figure 15 – LoRA SX1278 transceiver and antenna with 1 euro coin for scale.

Through the use of a proprietary spread spectrum modulation, LoRa devices can have ranges of up to 10km, and data rates stand between 0.3kbps and 27kbps.

The spreading factor (SF) is an important parameter for LoRa modulation. It controls the speed of data transmission, greatly affecting the range of radios. Lower spreading factors cause faster data rates and shorter range. Conversely, higher spreading factors mean slower data rates and longer transmission ranges.

2.9 LoRaWAN

Since LoRa defines only the physical layer of the protocol stack model referenced in figure 2, the LoRa Alliance develops and maintains LoRAWAN to manage the link and network layers.

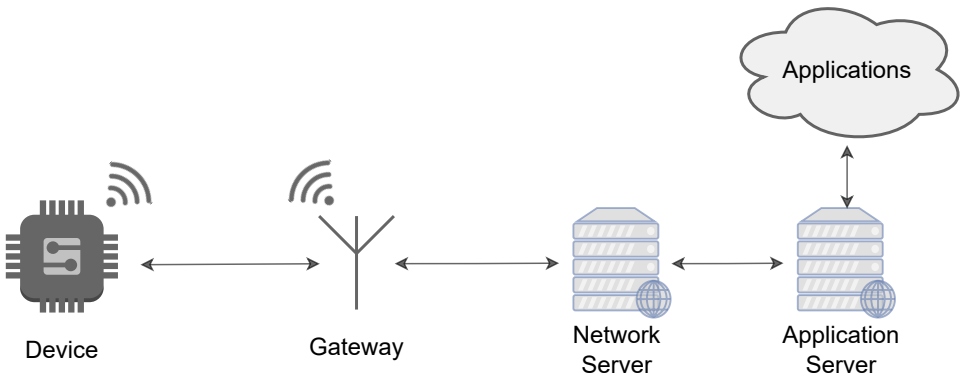


Figure 16 – LoRaWAN Network Architecture.

As seen above, LoRaWAN end-devices transmit their data to *gateways* that forward those data packets to a centralized network server, which in turn will forward them to application servers. User applications can receive the data stream by subscribing using the Message Queuing Telemetry Transport (MQTT)[3] protocol [27].

End-devices must authenticate to an existing network in order to be able to send and receive data. Authentication is achieved by the use of Over-The-Air-Activation (OTTA) or Activation By Personalization (ABP). LoRaWAN ensures secure end-to-end communication by the use of standard cryptographic algorithms and 128-bit keys.

The protocol also provides Medium Access Control (MAC) services, controlling the frequencies, data rate, and power for all devices.

2.10 Arduino

Arduino is an open-source hardware and software project, maintained by the Arduino LLC company and by a global community of software developers, hardware manufacturers, and hobbyists.

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Figure 17 – Arduino sample *sketch*.

Programs for Arduino hardware (called *sketches*) are written in C and C++ and compiled using the Arduino IDE. The Arduino environment provides well maintained libraries for interaction with various sensors, motors and transceivers, including the LoRa transceiver.

2.10.1 ESP32 board

The ESP32 board, created by Espressif [4], is a system on a chip (SoC) microcontroller with integrated Wi-Fi and Bluetooth. It can feature a dual-core or single-core CPU, 320 KB RAM, several programmable GPIOs, and peripheral interfaces. This board can be programmed, among other options, with the Arduino IDE.

[3] Publish/subscribe protocol for exchange of messages using a broker.
[4] https://www.espressif.com

3 LoRaNet

In this chapter a model for the LoRaNet network will be presented and explored in detail. A prototype implementation of the network was written, and features all the minimum functionality needed to perform initial testing on the ESP32 board, programmed with the Arduino IDE.

3.1 Overview

LoRaNet is a network with a mesh topology, there is no hierarchy between the nodes. The devices in this network may be the source or destination of data, or may also serve as intermediate nodes, forwarding data to other destinations.

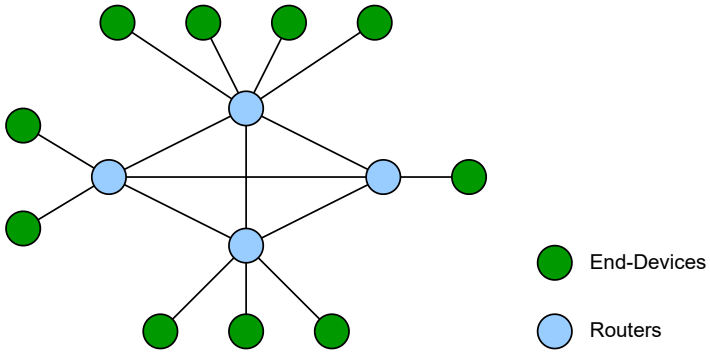


Figure 18 – Star-of-stars topology.

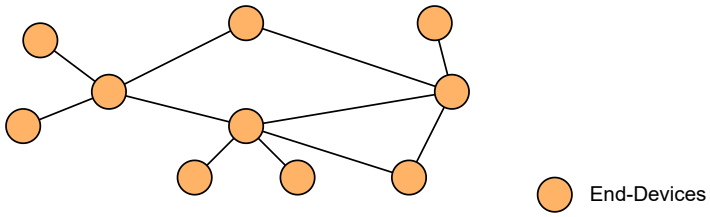


Figure 19 – LoRaNet mesh topology.

Single-hop or multi-hop routes will be built on demand for every packet that needs to be sent across the network, using the AODV protocol explained in the previous chapter.

3.2 Operation

Upon boot, a LoRaNet device will read its persistent EEPROM [28] to check if it has already been configured before. In a negative case, the device will generate new credentials for joining the network, choosing a random IP address that will identify that device in communication with its peers. Next time this device boots up, it will recover this address and any other necessary information, like the current AODV sequence number, from the EEPROM.

Whenever a device has data to send, it will queue that data into a *First In, First Out* (FIFO) data structure, where it will be processed and sent as soon as an active route for that destination is available.

A LoRaNet device may process received packets at any time using the hardware interrupt feature of LoRa radios. Devices must support interrupt callbacks, or else they will not be able to forward or receive data from their neighbors.

The sketches written for LoRaNet devices must also periodically call a `run()` function, so that various procedures may be executed, such as checking if there are any new available routes for queued data packets, or removing inactive routes from the routing table.

3.3 Limitations

The LoRa physical layer imposes some limitations for the operation of LoRaNet devices. Firstly, the LoRa PHY header has a one-byte *length* field [29], meaning the maximum payload size for a LoRa packet is 255 bytes.

Secondly, duty cycle limitations for open ISM (Industrial, Scientific, and Medical) bands worldwide impose an even greater limit on transmission by LoRa devices. As recommended on Adelantado et al. [3], LoRa devices must operate on the lowest possible SF, so as to reduce both the time on air and the off period of the radios.

3.4 Addressing

As mentioned in section 3.2, each device in the network will choose a random address to identify itself. Addresses will be IPv4 addresses, chosen from the `10.0.0.0` address range, reserved for private networks by the Internet Assigned Numbers Authority (IANA) [30]. IPv4 addresses were chosen, instead of IPv6, to reduce the size of the headers and try to stay within the limitations of section 3.3.

The chosen 24-bit block provides a range of 16,777,214 addresses to choose from, making address collisions improbable, depending on the size of the network. Address

collisions, when detected, must be handled by the nodes.

Through the use of private network addresses for LoRa nodes, LoRaNet makes it possible for nodes to exchange packets with the Internet. Nodes that wish to run an Internet gateway should generate route replies whenever they receive a route request to a public Internet address, as well as keep a Network address translation (NAT) table to allow for bi-directional communication.

3.5 Implementation

The prototype implementation was tested on the ESP32, specifically the **Heltec LoRa 32** [31] board, and developed using the Arduino IDE.

3.5.1 Proposed Architecture

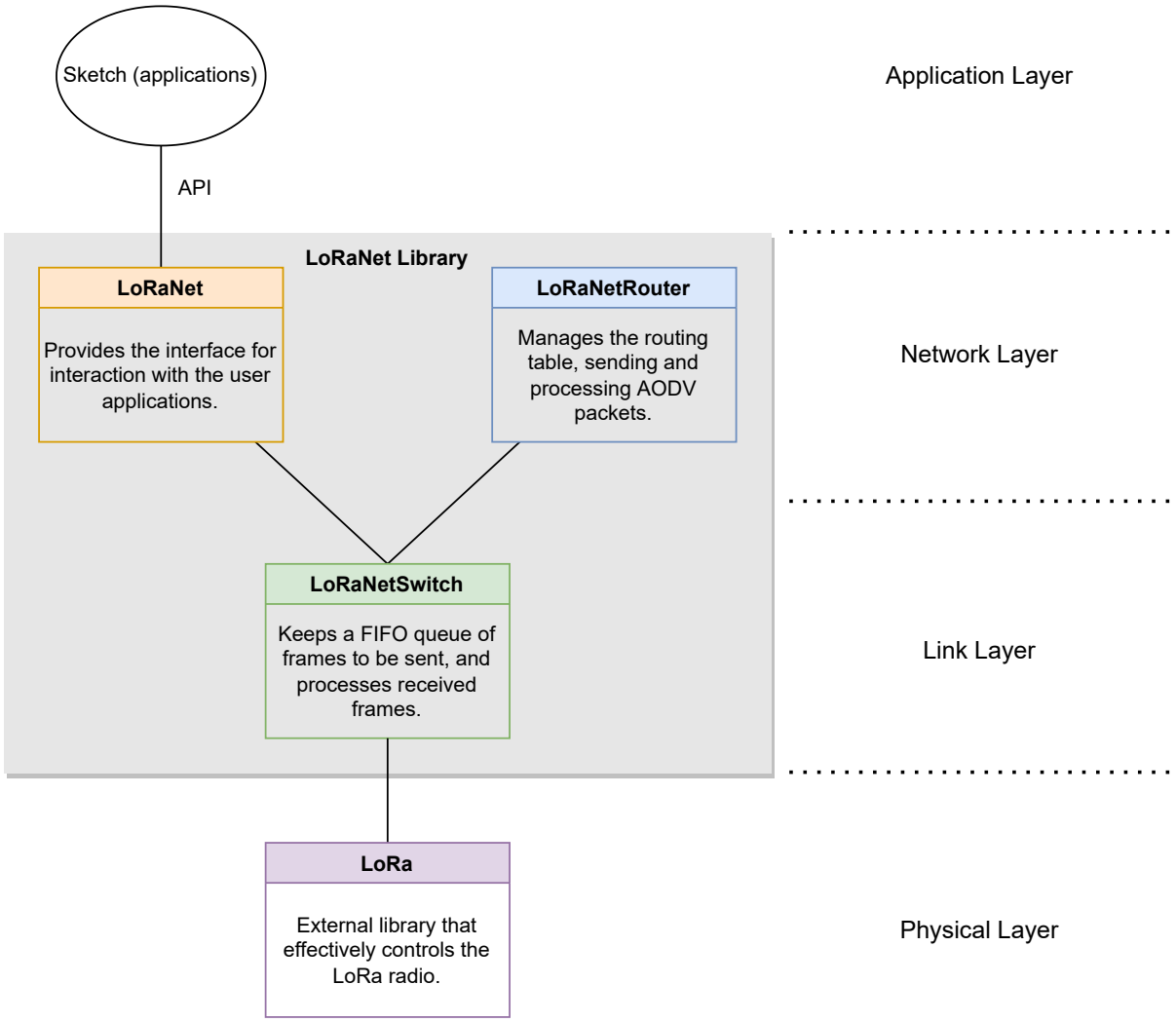


Figure 20 – LoRaNet modules.

3.5.2 LoRaNet

This module is responsible for the interaction with the user applications, and the initialization of the network layer header and payload fields. The IPv4 header was modified to include only the necessary fields for proper operation on LoRaNet.

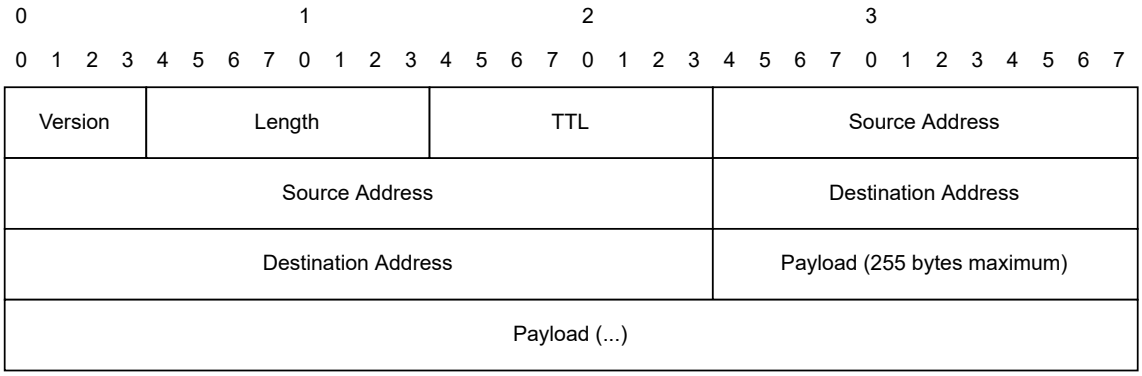


Figure 21 – LoRaNet modified IPv4 Header.

All incoming or outgoing data packets are handled by this class, and a callback function must be registered by the user application in order to receive messages.

3.5.3 LoRaNetRouter

The LoRaNetRouter module handles the routing table, processing incoming AODV messages. The AODV headers were implemented following RFC3561 [25] instructions, as seen on figures 9, 11, and 14.

3.5.4 LoRaNetSwitch

This class operates the link layer, performing functions such as managing the frame queue, sending and processing acknowledgments (ACKs), listening after a transmission, and retrying unacknowledged frames. This header was loosely based on the IEEE 802.11 header, but modified to reduce overhead.

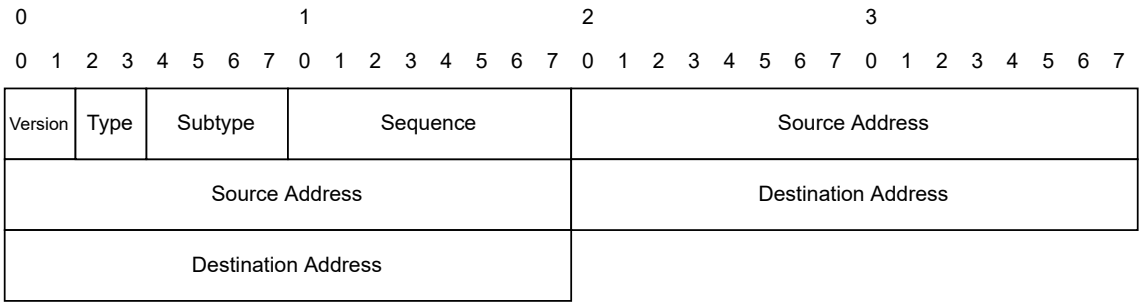


Figure 22 – LoRaNet link layer Header.

4 Experiments

These experiments were planned to give developers a general idea of what kind of performance to expect from devices using LoRaNet.

4.1 Setup

The following tests were completed using four *Heltec WiFi LoRa 32(V2)* boards. They are by no means real-world tests, but laboratory benchmarks. All the devices were inside the same room and within range of each other, therefore manual routing table manipulation was used to build 2-hop and 3-hop routes. The LoRa radios were operating on the lowest spreading factor (SF7), which has the fastest data rate, and on the 433MHz frequency band.

Table 1 – Nodes with their respective addresses

Node	**IP Address**
Origin	10.7.97.46
Intermediate 1 (if present)	10.151.36.241
Intermediate 2 (if present)	10.248.208.24
Destination	10.105.154.79

4.2 Test 1: Route construction latency

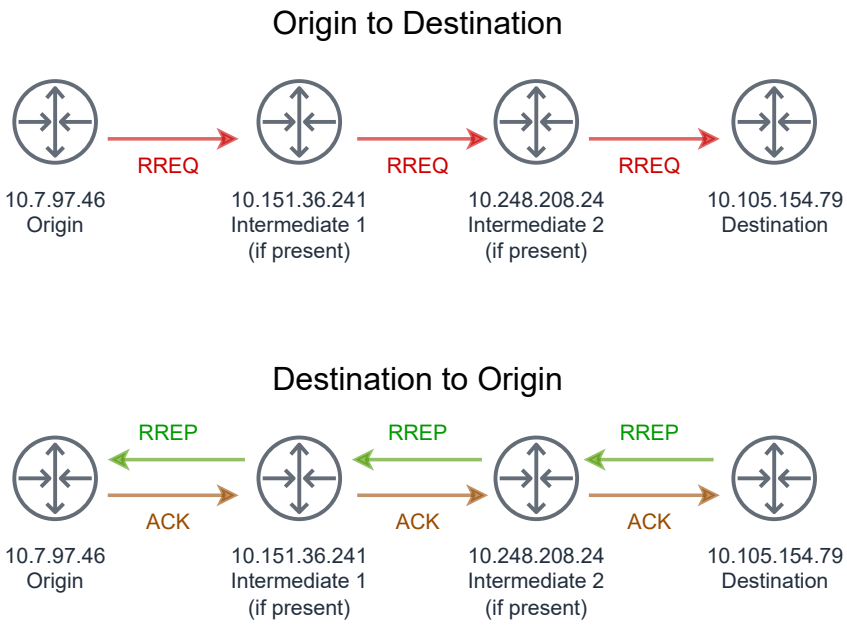


Figure 23 – Test 1 diagram.

This test measured the amount of time it took for a node to discover a valid route for a given destination. The timer started when the RREQ was sent, and stopped when the RREP was received by the originating node. The tests were done with 1, 2, and 3 hops between the originating node and the destination node, and repeated 400 times. In figure 24, each bar represents the mean value of the measured latencies, with the error bar showing the standard deviation.

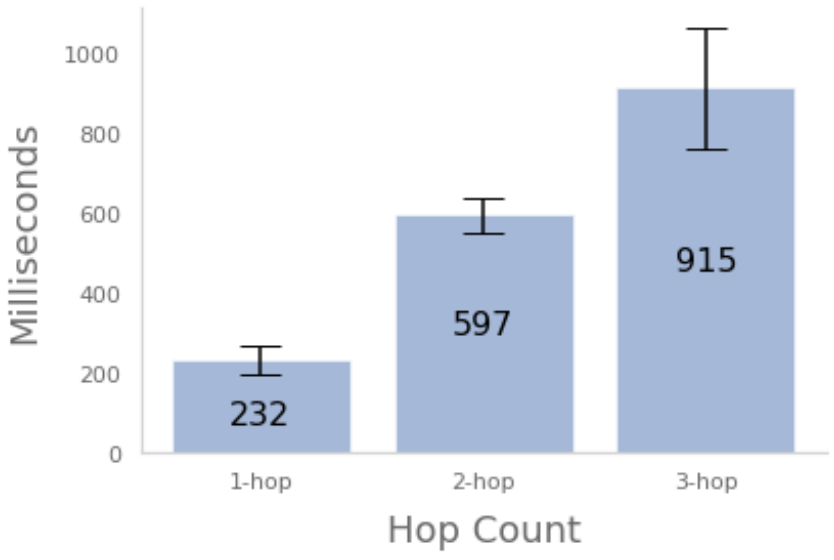


Figure 24 – Test 1 results.

4.3 Test 2: Data packet round-trip delay

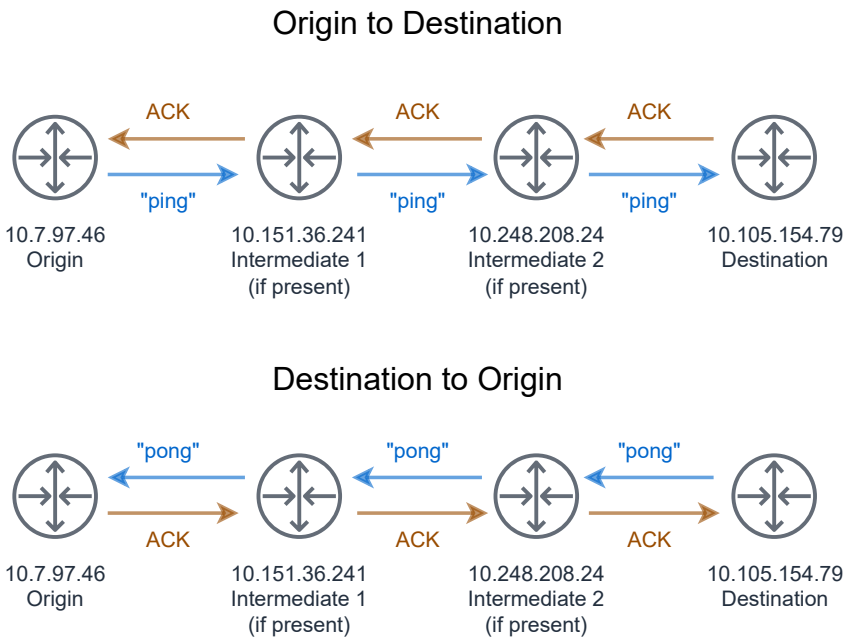


Figure 25 – Test 2 diagram.

This experiment consisted in sending *ping* messages and measuring the time it took for the *pong* replies to be received by the originating node. The originating node already had an active route for the destination before the beginning of the test. Again, the experiment was done with 1, 2, and 3 hops between the origin and destination, and repeated 400 times.

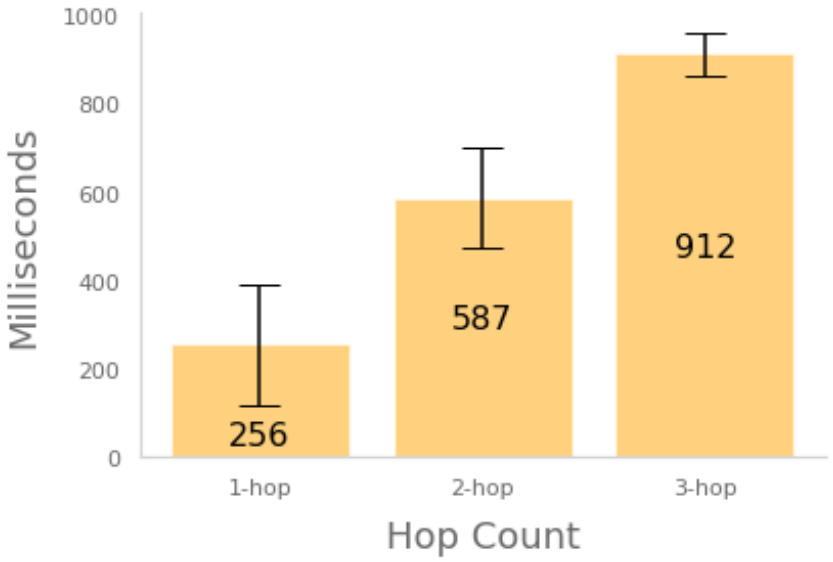


Figure 26 – Test 2 results.

4.4 Memory Usage

In embedded systems, memory and CPU resources are limited, so it is important to evaluate the impact of running LoRaNet under these conditions. Using the `ESP.getFreeHeap()` call, table 2 was built, showing the measured impact of LoRaNet on the device's DRAM (Data RAM).

Table 2 – DRAM usage by LoRaNet.

	Free Heap (bytes)	**DRAM usage (bytes)**	**DRAM usage (percentage - ESP32)**
On Boot	374,884	0	0%
After LoRaNet initialization	370,552	4,332	1.1%
After building routing table with two routing entries	370,000	4,884	1.3%

5 Discussion

Although budget and time constraints have prevented expanding the amplitude of the experiments performed, this document has shown that mesh networking using only LoRa end-devices is feasible.

The prototype library developed has successfully implemented the proposed network model, with minimum required features for route establishment and message exchanging. Long-distance routes of tens of kilometers could theoretically be achieved using only a few LoRaNet devices.

5.1 Protocol overhead

Taking into account figures 9, 11, 14, 21, and 22, it is possible to perform a sample simulation of the protocol overhead involved in a LoRaNet transmission.

Suppose node **A** wants to send the message "ping" to node **B**, and none of them or any intermediate nodes have any knowledge of any routes to communicate with one another. Table 3 shows the header sizes from the prototype implementation, and table 4 shows the number of bytes and overhead of this communication using 1, 2, and 3-hop transmissions.

Table 3 – Header sizes in prototype implementation.

	Length (bytes)
Link Layer header	10
Modified IPv4 header	11
RREQ	13
RREP	10

Table 4 – Overhead for sending a 5-byte "ping" message.

Number of Hops	**Route establishment (bytes)**	**Message exchange (bytes)**	**Overhead (percentage)**
1	33	36	92.7%
2	66	72	96.3%
3	99	108	97.5%

Through the analysis of the previous tables, and of figures 24 and 26, it is possible to conclude that very long multi-hop routes could be problematic in terms of latency and overhead. Therefore, application developers must plan carefully if using LoRaNet

or any other mesh network model for the LoRa physical layer, avoiding latency-bound applications if possible.

5.2 Node mobility

Node mobility is also a contributing factor on the latency and overhead of the proposed network. If nodes move frequently, then peer-to-peer links will change frequently as well. This means that existing routes will break and have to be mended, and data packets will have to be queued while the nodes rebuild those routes. Extreme node mobility must be avoided then, if low latency is important to an application using LoRaNet.

5.3 Duty cycle concerns

Due to the long range of LoRa radios and the unlicensed ISM radio frequencies used, in most countries LoRa devices are severely limited by regionally enforced restrictions. The European Telecommunications Standards Institute (ETSI), for example, imposes the maximum duty cycle for the EU868 frequency plan (863MHz - 870MHz) at 1%. Each day, the maximum allowed time on air for a device is of only 864 seconds.

The use of multi-hop routes has the potential to decrease time on air, as faster data rates can be used in place of slower data rates which have longer range. On the other hand, nodes in a mesh network must forward messages for other members of the network and configure routes, making compliance to duty cycle limitations a potential problem in certain countries.

5.4 Power requirements

Battery usage is always a major factor in IoT applications. The power requirements of operating a LoRaNet device must be taken into account when planning an application. Similarly to the duty cycle issues, a node may save up battery when transmitting on a faster data rate, however it will also have to receive and process incoming packets from other nodes, and that can cause battery drain to increase. Further investigation needs to be conducted in order to determine what kind of impact LoRaNet would have on a device's power usage.

6 Future work and conclusion

This section will cover the main research possibilities for improving this work, and review the introduction to assess if the proposed objectives were met.

6.1 Improvements

There is a lot of further work that could be done to improve LoRaNet, both theoretical and software development. Some of them are listed below.

6.1.1 Features

In the LoRaNetRouter module, most of the missing features are related to route maintenance and route repairing. The **Lifetime** feature of AODV is a field in the routing table entry that is created when a new valid route is discovered. As defined by RFC3561, the created route will be active for ACTIVE_ROUTE_TIMEOUT milliseconds, and "*each time a route is used to forward a data packet, its Active Route Lifetime field of the source, destination and the next hop on the path to the destination is updated to be no less than the current time plus ACTIVE_ROUTE_TIMEOUT*". This feature is missing from the current implementation.

The other most important missing features are the RERR message and the route repairing features. Whenever a link break is detected, a node is supposed to attempt to repair a route, before sending a RERR message to all nodes that used the broken link. These features are obviously essential for the correct operation of the network.

6.1.2 Scalability

The main research question that remains is the one of the scalability of the proposed network. The experiments have indicated that route latency and overhead increased linearly with the number of hops, yet further testing is necessary to find out how big could a LoRaNet network be. This could be done using network simulators or using real-world devices.

6.1.3 Security

The proposed model is **very insecure**, there is no guarantee of confidentiality nor of authentication. All data sent on LoRaNet is readable by any LoRa radio, and there is no protection against impersonation or man-in-the-middle attacks.

A lot of work could be done to improve this situation, using asymmetric encryption to ensure the secrecy of communications and the authenticity of received messages, although header size could be a limiting factor.

6.1.4 Internet gateways

A very important feature that would be easier to implement is the compatibility of LoRaNet with the Internet. As explored on section 3.4, gateway nodes would just need to reply to RREQs addressed to public IPs, filling a NAT table to allow for bi-directional communication between Internet hosts and LoRaNet nodes.

6.2 Conclusion

This project has proposed a detailed network stack, called LoRaNet, for the upper layers of the LoRa physical layer, implemented in C/C++ for the ESP32 using the Arduino IDE. The suggested network is not a definitive solution for IoT applications, but rather another tool that can be useful for certain applications.

Among the advantages of using LoRaNet, is the possibility of deploying a LoRa-only network with complete addressing, acknowledgments, and a compatible library interface, allowing for peer-to-peer exchange of messages without the use of the Internet. The possibility of multi-hop routing is another very promising feature that can decrease the time on air of LoRa radios, as well as drastically increase the range of LPWANs without the need to deploy costly infrastructure.

Large LoRaNet LPWANs are theoretically possible, but restricted by latency and duty cycle limitations. Experiments performed have shown that latency will increase linearly with the number of hops on a route, making very long network routes problematic, although a real-world mesh topology was not tested.

The prototype implementation should be used in the future to evaluate the scalability of LoRaNet. Security features such as confidentiality and authenticity must be developed in order for this network to be considered a viable solution for LoRa LPWANs.

References

[1] *A technical overview of LoRa® and LoRaWAN™*. Accessed Feb. 8, 2022. LoRa Alliance. URL: https://lora-alliance.org/wp-content/uploads/2020/11/what-is-lorawan.pdf.

[2] *A Comprehensive Look At Low Power, Wide Area Networks*. Accessed Feb. 8, 2022. LinkLabs. URL: https://cdn2.hubspot.net/hubfs/427771/LPWAN-Brochure-Interactive.pdf.

[3] Ferran Adelantado et al. "Understanding the Limits of LoRaWAN". In: *IEEE Communications Magazine* 55.9 (2017), pp. 34–40. DOI: 10.1109/MCOM.2017.1600613.

[4] David Kushner. "The Making Of Arduino". In: *IEEE Spectrum* (Oct. 26, 2011). URL: https://spectrum.ieee.org/the-making-of-arduino (visited on 03/12/2017).

[5] *How many Arduinos are "in the wild?" About 300,000*. Accessed Feb. 8, 2022. Adafruit Industries. URL: https://blog.adafruit.com/2011/05/15/how-many-arduinos-are-in-the-wild-about-300000/.

[6] *ESP32 - Technical Reference Manual*. Accessed Feb. 8, 2022. Espressif Systems. 2021. URL: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf.

[7] *TTN Mapper*. Accessed Feb. 8, 2022. TTN Mapper. URL: https://ttnmapper.org.

[8] Daniel Lundell et al. "A Routing Protocol for LoRA Mesh Networks". In: *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*. 2018, pp. 14–19. DOI: 10.1109/WoWMoM.2018.8449743.

[9] *802.15.4-2020 Standard*. Accessed Feb. 8, 2022. IEEE. URL: https://standards.ieee.org/ieee/802.15.4/7029.

[10] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach*. 7th ed. Boston, MA: Pearson, 2016. ISBN: 978-0-13-359414-0.

[11] John Moy. *OSPF Version 2*. RFC 2328. Apr. 1998. DOI: 10.17487/RFC2328. URL: https://www.rfc-editor.org/info/rfc2328.

[12] *Internet Protocol*. RFC 791. Sept. 1981. DOI: 10.17487/RFC0791. URL: https://www.rfc-editor.org/info/rfc791.

[13] "IEEE Standard for Ethernet". In: *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015)* (2018), pp. 1–5600. DOI: 10.1109/IEEESTD.2018.8457469.

[14] Henrik Nielsen et al. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. June 1999. DOI: 10.17487/RFC2616. URL: https://www.rfc-editor.org/info/rfc2616.

[15] *Domain names - concepts and facilities*. RFC 1034. Nov. 1987. DOI: 10.17487/RFC1034. URL: https://www.rfc-editor.org/info/rfc1034.

[16] *Domain names - implementation and specification*. RFC 1035. Nov. 1987. DOI: 10.17487/RFC1035. URL: https://www.rfc-editor.org/info/rfc1035.

[17] Dr. John C. Klensin. *Simple Mail Transfer Protocol*. RFC 5321. Oct. 2008. DOI: 10.17487/RFC5321. URL: https://www.rfc-editor.org/info/rfc5321.

[18] Jim Martin et al. *Network Time Protocol Version 4: Protocol and Algorithms Specification*. RFC 5905. June 2010. DOI: 10.17487/RFC5905. URL: https://www.rfc-editor.org/info/rfc5905.

[19] *Transmission Control Protocol*. RFC 793. Sept. 1981. DOI: 10.17487/RFC0793. URL: https://www.rfc-editor.org/info/rfc793.

[20] *User Datagram Protocol*. RFC 768. Aug. 1980. DOI: 10.17487/RFC0768. URL: https://www.rfc-editor.org/info/rfc768.

[21] Bob Hinden and Dr. Steve E. Deering. *Internet Protocol, Version 6 (IPv6) Specification*. RFC 2460. Dec. 1998. DOI: 10.17487/RFC2460. URL: https://www.rfc-editor.org/info/rfc2460.

[22] "IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications". In: *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)* (2021), pp. 1–4379. DOI: 10.1109/IEEESTD.2021.9363693.

[23] Kais Mekki et al. "Overview of Cellular LPWAN Technologies for IoT Deployment: Sigfox, LoRaWAN, and NB-IoT". In: *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. 2018, pp. 197–202. DOI: 10.1109/PERCOMW.2018.8480255.

[24] Edsger W. Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1 (1959), pp. 269–271.

[25] C. Perkins, E. Belding-Royer, and S. Das. *RFC3561: Ad Hoc On-Demand Distance Vector (AODV) Routing*. USA, 2003.

[26] J.J. Garcia-Luna-Aceves. "A Unified Approach to Loop-Free Routing Using Distance Vectors or Link States." In: vol. 19. Aug. 1989, pp. 212–223. DOI: 10.1145/75247.75268.

[27] *MQTT Version 3.1.1 Plus Errata 01*. Accessed Feb. 9, 2022. OASIS. URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html.

[28] *A guide to EEPROM*. Accessed Feb. 10, 2022. Arduino LLC. URL: https://docs.arduino.cc/learn/programming/eeprom-guide.

[29] Pieter Robyns et al. “A Multi-Channel Software Decoder for the LoRa Modulation Scheme”. In: *IoTBDS*. 2018.

[30] Robert Moskowitz et al. *Address Allocation for Private Internets*. RFC 1918. Feb. 1996. DOI: 10.17487/RFC1918. URL: https://www.rfc-editor.org/info/rfc1918.

[31] *WiFi LoRa 32 (V2)*. Accessed Feb. 11, 2022. Heltec Automation. URL: https://heltec.org/project/wifi-lora-32/.