Evaluating the Efficiency, Capabilities, and Scalability of a LoRaWAN Network in a Fenceless Grazing Implementation

By
Ryan Alder

A THESIS

submitted to

Oregon State University

Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Computer Science
(Honors Scholar)

Presented June 3, 2020
Commencement June 2020

AN ABSTRACT OF THE THESIS OF

Ryan Alder for the degree of Honors Baccalaureate of Science in Computer Science presented on June 3, 2020. Title: Evaluating the Efficiency, Capabilities, and Scalability of a LoRaWAN Network in a Fenceless Grazing Implementation

Abstract approved:

Bechir Hamdaoui

LoRaWAN networks are becoming more popular, and it is becoming common for developers to look at solutions utilizing Internet of Things concepts. In this paper, I introduce a Fenceless Grazing System utilizing the LoRaWAN network stack and discuss the limitations of this theorized network to better understand the scalability prior to production. The goal of this system is to define ”invisible fences” and prevent livestock from crossing these boundaries. Using smart collars on animals, ranchers are able to define boundaries and herd animals from the comfort of their home. This system utilizes the LoRaWAN network stack, and this technology is further discussed. Hardware is introduced, and the capabilities of this system and greater LoRaWAN networks in general is evaluated. Utilizing NS-3, various simulations were run to evaluate the packet loss across the entire network in an effort to determine the constraints of the system. Finally, these results are presented and a theorized maximum distance is evaluated and tested with the hardware on hand.

Key Words: LoRa, Fenceless Grazing, Internet of Things, Optimization

Corresponding e-mail address: alderr@oregonstate.edu

Evaluating the Efficiency, Capabilities, and Scalability of a LoRaWAN Network in a Fenceless Grazing Implementation

By
Ryan Alder

A THESIS

submitted to

Oregon State University

Honors College

in partial fulfillment of
the requirements for the
degree of

Honors Baccalaureate of Science in Computer Science
(Honors Scholar)

Presented June 3, 2020
Commencement June 2020

Honors Baccalaureate of Science in Computer Science project of Ryan Alder presented on June 3, 2020

APPROVED:

Bechir Hamdaoui, Mentor, representing Electrical Engineering and Computer Science

---

Lizhong Chen, Committee Member, representing Electrical Engineering and Computer Science

---

Yeongjin Jang, Committee Member, representing Electrical Engineering and Computer Science

---

Toni Doolen, Dean, Oregon State University Honors College

Ryan Alder, Author

Evaluating the Efficiency, Capabilities, and Scalability of a LoRaWAN Network in a Fenceless Grazing Implementation

Ryan Alder

***Abstract*— LoRaWAN networks are becoming more popular, and it is becoming common for developers to look at solutions utilizing Internet of Things concepts. In this paper, I introduce a Fenceless Grazing System utilizing the LoRaWAN network stack and discuss the limitations of this theorized network to better understand the scalability prior to production. The goal of this system is to define "invisible fences" and prevent livestock from crossing these boundaries. Using smart collars on animals, ranchers are able to define boundaries and herd animals from the comfort of their home. This system utilizes the LoRaWAN network stack, and this technology is further discussed. Hardware is introduced, and the capabilities of this system and greater LoRaWAN networks in general is evaluated. Utilizing NS-3, various simulations were run to evaluate the packet loss across the entire network in an effort to determine the constraints of the system. Finally, these results are presented and a theorized maximum distance is evaluated and tested with the hardware on hand.**

I. INTRODUCTION

Internet of Things (IoT) implementations are becoming more common in the world today. The number of businesses that utilize Internet of Things devices has increased from 13% in 2014 to 25% in 2019. The number of connected IoT devices is projected to increase to 43 billion by 2023, increasing threefold from 2018 [1]. With this increase in popularity, many groups are looking at ways to utilize an IoT network to solve their problems. One of the many inefficiencies in modern day labor is the herding of animals in a ranch setting. Ranchers must spend considerable time managing herds of animals by moving them from pasture to pasture to prevent overgrazing. This sort of problem can be addressed with an IoT implementation where each animal is a node in a network where all communicate back to a central server managed by the rancher. My senior capstone team and I implemented a product to address this inefficiency, which aimed to decrease the total amount of time spent managing and herding animals. This product, the Fenceless Grazing System (FGS), implements collars on each animal which then communicates over LoRa to a gateway hosted in a central location. The Fenceless Grazing System allows for ranchers to define so-called invisible fences to both herd animals and observe their grazing habits. While the design is complete, there are many lingering questions still associated with this implementation that need to be answered before hardware moves out of a preliminary testing phase and is actually implemented in a large-scale setting.

The large scale implementation of the Fenceless Grazing System is untested and unknown. Thorough evaluation into the scalability of the product in a real environment is necessary before any sort of wide-scaled production begins. Many different variables must be estimated, including the max allowable distance, average bandwidth as distance between the end-nodes and the gateway increases, average packet loss, battery life, and more. One of the most significant requirements for this project is it's ability to be widely scaled, as the idea is to significantly decrease the amount of time a rancher has to spend managing herds of animals. This thesis will attempt to demonstrate the capabilities and constraints of the FGS in terms of the number of end-nodes, maximum size of the grazing area, communication frequency, and more by using NS-3 to estimate the average values of these variables. An evaluation of the resulting data will be presented that demonstrates the theoretical capabilities of the Fenceless Grazing System, and results will be verified with the hardware currently on hand.

II. TECHNOLOGY BACKGROUND

A. LoRa Technology

The primary form of communication for the Fenceless Grazing System implementation is LoRa. LoRa (stands for Long Range) is a physical level communication protocol patented by Semtech which utilizes the unlicensed band of the radio spectrum. This protocol exists solely for IoT devices, providing extremely long range capabilities with low battery requirements. With these capabilities, however, the bandwidth is extremely limited and decreases significantly as range increases. This is less of a problem for IoT implementations as most end-nodes will only need to send a very small amount of data per communication cycle. This, of course, is dependent on the network implementation but for most IoT networks does not prove to be an issue. For the Fenceless Grazing System, bandwidth is not anticipated to be a significant factor in limiting the size of the network, as the size of each communication is small at 12 bytes, and frequency of communication can be extremely limited. Depending on the network and requirements of the rancher, communication can be restricted to once every few hours per end-device while still achieving the goals of the project.

1) Chirp Spread Spectrum: LoRa is based on the Chirp Spread Spectrum (CSS) modulation technique, which is a series of upchirps and downchirps that utilizes the entire bandwidth defined by the data rate (see Table 1). As a result, LoRa is resistant to channel noise and multi-path fading and allows for long distance communication with minimal power requirements. Upchirp refers to an increase in frequency over a time period, while a downchirp decreases in frequency. LoRa starts each communication with a preamble, which is a series of repeated upchirps followed by a downchirp which signals the start of frame delimiter [2]. LoRa builds upon Direct Sequence Spread Spectrum (DSSS) and Chirp Spread Spectrum to provide their own implementation of a modulation technique that greatly reduces the complexity of the receiver. LoRa allows for scalable bandwidth, able to be utilized for both narrowband frequency hopping and wideband direct sequence applications [3]. Not only this, but LoRa allows for high robustness, very low power use, Doppler resistance, and long range capabilities.

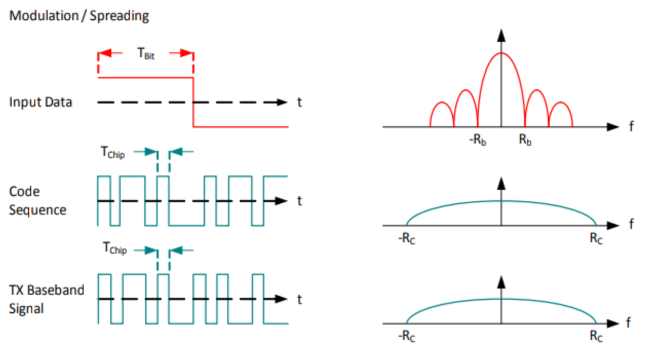


Fig. 1. Modulation / Spreading Process (DSSS) [3]

2) *Spreading Factor:* The spreading factor simply refers to the duration of the chirp sent by the LoRa transceiver. The LoRa protocol has defined spreading factors from 7 to 12, where increasing the spreading factor by one results in double the time a chirp spends on the air. As a result, adaptive data rates can be achieved by increasing the spreading factor as end-nodes move further away from a gateway or decreasing it when a high SF is unnecessary to reduce the total time on air and increase the data rate. This way vast distances can be achieved while still managing total time on air, as well as optimizing a network when the stability of the end-nodes is consistent.

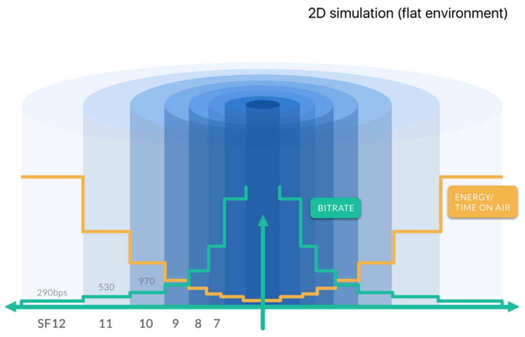


Fig. 2. LoRa SF demonstration, illustrating how changes in SF affect time on air and bitrate [4]

B. LoRaWAN Technology

LoRaWAN is an open source MAC protocol created by the LoRa Alliance that is found on top of the LoRa physical protocol. LoRaWAN handles medium access control, encryption, authentication, and more to seamlessly integrate LoRa devices into a network. LoRaWAN provides an adaptive data rate protocol that optimizes battery life and medium usage as the characteristics of the network change, and defines several different data rates which differ from each other by bandwidth and spreading factor. LoRaWAN is necessary for any network, and is implemented in the Fenceless Grazing System through several LoRaMAC in C (LMIC) libraries and The Things Network as a free option for a network server.

1) *Data Rates:* In order to best adjust to various different environments and networks, LoRaWAN provides several different data rates. These data rates adjust bandwidth and spreading factor to allow for consistent communication while managing air time and battery usage. Swapping between data rates also results in a change in the bits being sent per second as well as the maximum payload size. As SF increases, the data rate will decrease as each bit spends more time on air. Likewise, as the BW widens the data rate will increase [5]. The LoRaWAN defined data rates for the United States frequency of 915Mhz is displayed in Table 1. DR4 is identical to DR12, and DR5-7, 14, and 15 are reserved for future applications [6].

TABLE I
LoRaWAN Data Rates (US)[6]

Data Rate	SF	BW (kHz)	bit/s	Max payload size (bytes)
0	10	125	980	11
1	9	125	1,760	53
2	8	125	3,125	125
3	7	125	5,470	242
4	8	500	12,500	222
8	12	500	980	33
9	11	500	1,760	109
10	10	500	3,900	222
11	9	500	7,000	222
12	8	500	12,500	222
13	7	500	21,900	222

2) *Adaptive Data Rate:* LoRaWAN implements an algorithm to allow for Adaptive Data Rates (ADR), which is toggled on or off by each end-node at an opportune moment. ADR allows the network server to tell each end-node which data rate and transmission power should be used to communicate with the gateway. This results in optimal data rates, airtime, and energy consumption in the network [7]. The Things Network (TTN) recommends only enabling ADR once an end-nodes RF conditions are stable. For example, if an end-node stops moving for a set amount of time then it should enable ADR and disable it once the device begins to move again. This prevents the end-node from being silenced as a result of moving too far away from the gateway while still using a data rate that was previously optimal. ADR should also be disabled by the end-node if RF conditions become unstable, even if the device is still stationary, to prevent being silenced. Once an end-node is confident that RF conditions are stable, it sends a message to the gateway with an ADR bit set. TTN then measures

the signal-to-noise ratio (SNR) and number of gateways that received each uplink throughout the next 20 uplinks. Based on this information, the algorithm informs the end-node of the new optimal data rate. Fig. 3 shows The Things Network's implementation of ADR, which is currently what is used in the Fenceless Grazing System implementation.

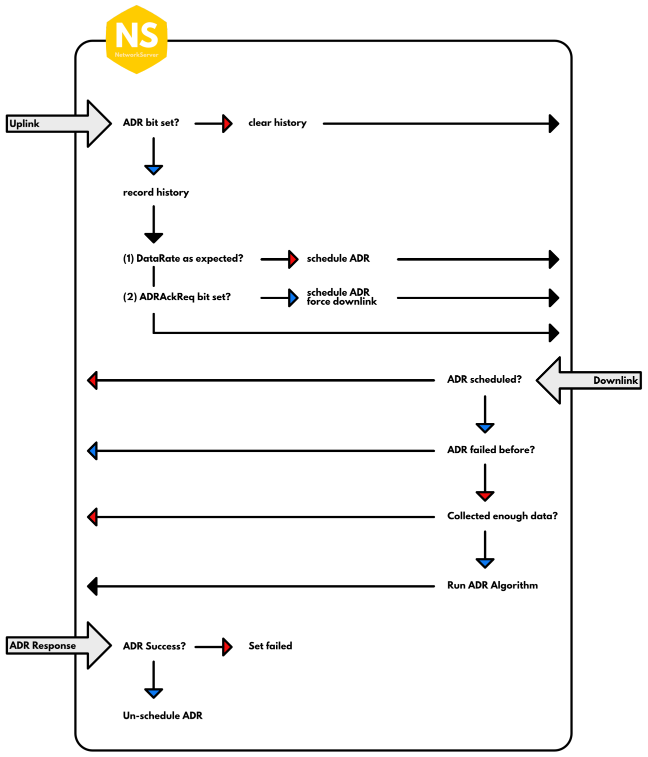


Fig. 3. TTN ADR Algorithm [7]

3) Classes: LoRaWAN defines three different classes that can be implemented on each end-node in an effort to provide options for various networks. Each class handles communication slightly differently, where some excel in low latency others will excel in low battery life. While groups like The Things Network only support class A, classes B and C are gaining popularity and are likely to be implemented in basic network servers like TTN soon.

- Class A (All) is the default class and must be implemented on every end-node on a LoRaWAN network. Class A provides the foundation for classes B and C, which build upon the implementation of class A. This class excels in restricting the battery usage, and boasts the lowest power requirements out of all of the classes. In class A, end-nodes are the ones to always initiate communication, and if an application needs to communicate with an end-node through the gateway it must wait until the end-node initiates the communication. An uplink transmission from an end-node to the gateway can be initiated at any time, which may result in collisions as this implementation is an ALOHA type of protocol. Once an end-node has initiated communication, the gateway only has two windows in which to respond to the end-node. Fig. 4 shows these two windows, and if the end-node does not receive a response from the gateway in these windows it can assume the packet was lost. The response sent by the gateway can be adjusted to fit the needs of the application, but must set an ack bit in the response to acknowledge the receipt of the previous packet. If necessary, the end-node can continue to attempt to re-transmit the packet if no acknowledgement was received up to a certain number defined by the network administrator. During this re-transmission process, the end-node will hop between frequencies to try to find an available channel on the gateway. Note that LoRaWAN doesn't implement listen before transmit as it is an ALOHA protocol, so collisions are likely to occur especially as the number and density of end-nodes increases.

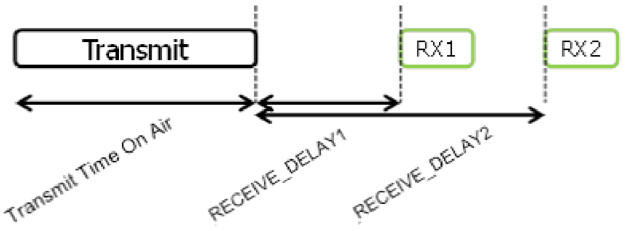


Fig. 4. LoRaWAN Class A Receive Windows [6]

- Class B (Beacon) is ideal for battery-powered end-nodes that are either mobile or stationary. This class, which is built upon class A, schedules specific receive slots for each end-node. This increases the battery usage of the end-devices, as they are forced to turn on their transceivers and listen to the medium during every defined receive window. While this decreases battery life, it increases the flexibility of the network, allowing for the gateway to communicate with the end-nodes without requiring the end-nodes to initiate communication. This is accomplished through the use of a beacon, which is a signal sent by the gateway once every 128 seconds. The end-nodes listen for this beacon, and use it to synchronize with the gateway. Then, end-devices are made available for reception at a predictable time based on the beacon value, and during this time the gateway has the opportunity to send messages to a specific end-node. Class B implements slot randomization in order to prevent collisions between many different end-nodes, which randomizes the receive slots across all separate end-devices. Fig. 5 demonstrates this feature, showing the gateway's network beacon transmission and the scheduled RX windows on the end-node based on the time of the beacon. In the US specification of LoRaWAN, the gateway is allowed to utilize frequency hopping for transmitting the beacon, meaning the beacon will be transmitted across all frequencies. This allows the end-nodes to be spread out across those frequencies, which increases the number of potential devices as well as decreases the chance of packet loss due to collision. Each end-device can be set to operate on a specific frequency, which allows for frequency diversity across the entire network.

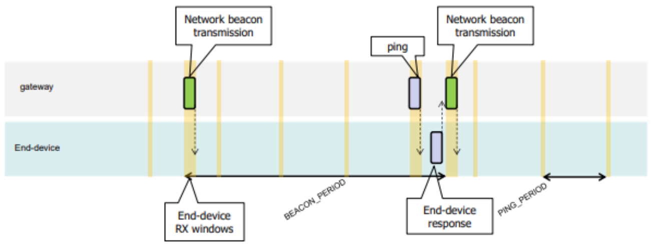


Fig. 5. LoRaWAN Class B Receive Windows [6]

- Class C (Continuously listening) significantly affects the battery life of the end devices while greatly decreasing the latency. As the name suggests, this class implements class A except instead of two limited receive windows, the last receive window is extended until the next transmission. That is, if an end-node is not currently transmitting a message to the gateway then it is listening for communication from the gateway. Fig. 6 demonstrates how it extends class A implementation by extending the second receive window until the next uplink. This class is ideal for end devices that do not have any power requirements that could benefit from extremely low latency in regards to gateway to end-node communication when compared to the other two classes.

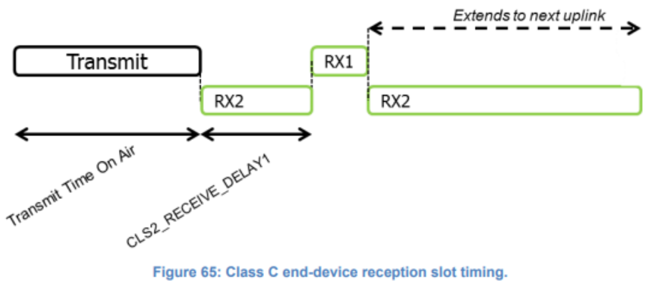


Fig. 6. LoRaWAN Class C Receive Windows [6]

III. FENCELESS GRAZING SYSTEM

A. System Overview

The Fenceless Grazing System is made up of four distinct components: a collar device, a LoRaWAN gateway / HTTP server, a network server, and an Android application. Collars are placed on each animal, and communicate over LoRa and LoRaWAN to the gateway. The gateway then forwards these packets to the network server hosted by The Things Network (TTN) over Ethernet or Wi-Fi. A network server is required for any LoRaWAN implementation, and these servers manage encryption, adaptive data rate algorithms, packet parsing, server-side LoRaWAN class implementation, and more. TTN was chosen for simplicity and ease of use, and provides both uplink and downlink capabilities for each end device. The HTTP API server hosted on the same machine as the gateway is the main application, and utilizes LoRaWAN to send and receive these transmissions. Lastly, the Android application authenticates using JSON Web Tokens, and communicates with the HTTP API server. Fig. 7 demonstrates the high level overview of this network and shows the communication between these elements. In this figure, the reference to other hardware connected to the collars encompasses any form of desired stimulus (whether it be auditory or electrical) to prevent livestock from leaving the set bounds.

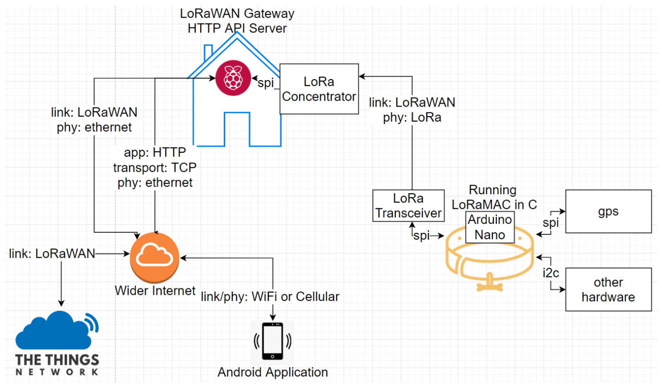


Fig. 7. FGS Network Topology

B. Collar

The collar component of the FGS project will be the physical collar placed upon the animal. Each collar will be an end-node in the greater LoRaWAN network, communicating with the gateway to update the user of the animal's status and location as well as gathering user defined fence boundaries. The main purpose of this device is GPS acquisition, LoRa communication, and bounds checking. Fig. 8 shows the current state of the collar, currently being developed on a breadboard for ease of use during the prototype stages. From the left to the right the figure shows the Arduino Nano acting as the main controller, the logic converter (5v to 3.3v), the GPS module, and lastly the LoRa transceiver. The logic converter is required as the Nano operates at 5 volts, while the GPS module and LoRa transceiver operates at 3.3 volts. In the final product more connections to a speaker and electrical stimulus module will be added. Currently, we are utilizing the TinyLoRa library by Adafruit for LoRaWAN capabilities on the end-node [14]. This manages the link layer communication with the gateway, and the application layer is written in C++ for Arduino and flashed onto the Nano. The application manages sending updates to the gateway, and receiving communications from the gateway for updated fence constraints. This device also periodically checks the GPS location and based on that value determines whether or not the collar is within the defined bounds. Lastly, all communication is first encoded using Google's protocol buffers (Protobuf) [15] to decrease the total length of the message which in turn increases the max number of end-nodes and decreases the chances of packet loss due to collisions. Each communication from the end-nodes is only 12 bytes, which is achieved through the use of protocol buffers.

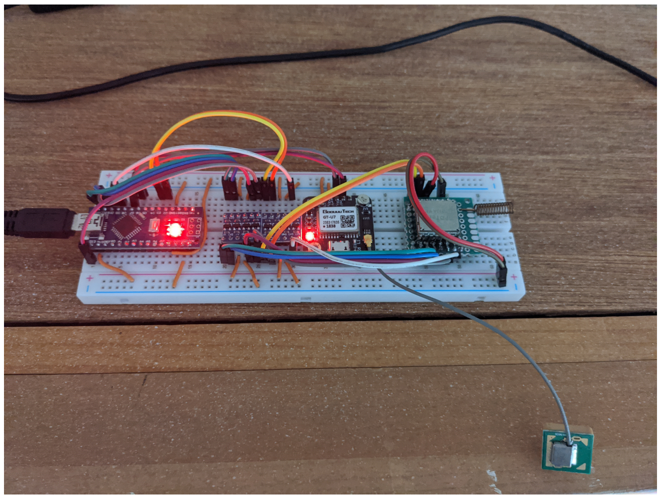


Fig. 8. FGS End-node (Collar)

C. Gateway

The gateway component of the FGS project acts as two separate entities: the LoRaWAN gateway and the HTTP API server used to communicate with the Android application. This gateway is designed to be hosted in a central location with connection to internet through Ethernet. Fig. 9 shows the current state of the gateway, with the primary sub-component being a Raspberry Pi. The Pi is a single board Linux computer running a custom Raspbian image provided by The Things Network which comes with all of the necessary programs for running the LoRaWAN gateway including a packet forwarder to the network server [16]. The Raspberry Pi is capable of interfacing with the LoRa concentrator seen in Fig. 9, which is the set of transceivers capable of listening on eight different channels at the same time. As previously mentioned, the Raspberry Pi also takes on the responsibility of hosting the API server in order to reduce the total amount of hardware necessary. The API server was written in Python using the Sqlalchemy library, and interacts with an SQLite database also hosted on the Pi. This database holds all of the received information from each end-node, and is queried by the API whenever the user requests information to be displayed. Lastly, the ideal back-haul for the gateway would be Ethernet due to increased reliability, but also supports Wi-Fi depending on the hosting requirements. In the final product, a secure outer shell will be used to encapsulate this hardware to allow for it to be hosted outside if necessary to provide better communication strength with all of the end-nodes.

D. Network Server

The network server is being hosted by The Things Network, providing an easy to use interface that allows for a simple setup for a LoRaWAN network. While not ideal

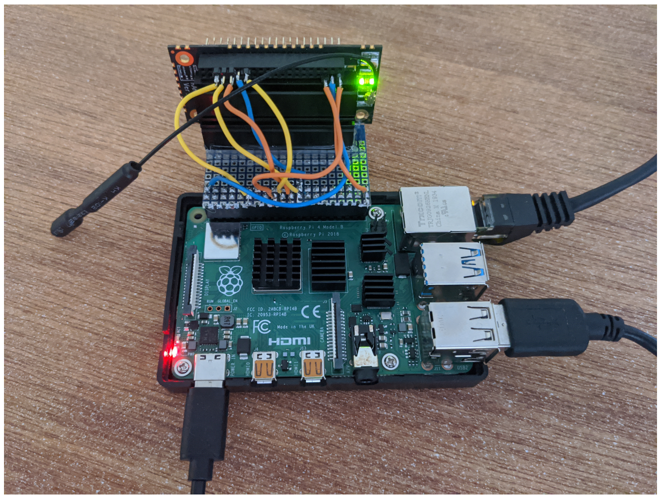


Fig. 9. FGS Gateway

due to off-site hosting and privacy concerns, TTN provides an extremely robust and simple system allowing for quick development time and testing. In an ideal world, each implementation would host an internal network server to provide security and remove the complexity of having an off site server. TTN manages many aspects of a LoRaWAN network, including adjusting the data rate as defined by the Adaptive Data Rate algorithm, end-to-end encryption, and encoding / decoding of packets for the LoRaWAN protocol. Fig. 10 shows the network stack, where the Network Server is hosted by TTN and the gateway server and application server are both hosted on the Raspberry Pi.

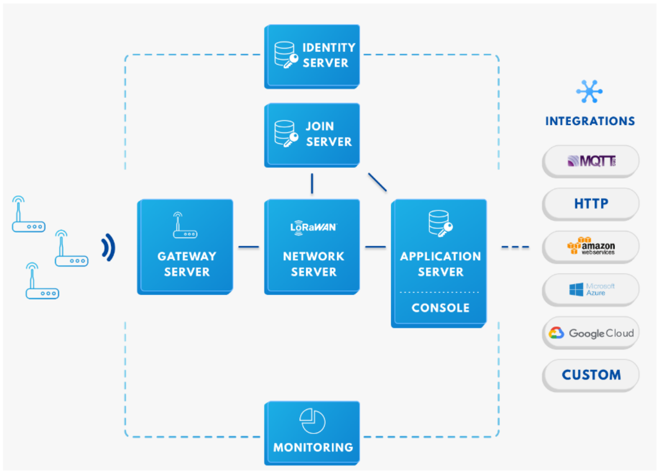


Fig. 10. TTN LoRaWAN Stack [17]

E. Android Application

The primary way of interacting with the system currently is through the use of an Android application. This application connects to the HTTP API server written in Python being hosted on the Raspberry Pi. This API uses JSON Web Token (JWT) technology for authenticating clients, and basic JSON as the data exchange format. The downloaded Android client must first login, and once authenticated allows for the user to monitor the collars and define new grazing boundaries. All information is gathered from the SQLite database hosted on the Pi before being sent back to the client.

IV. System Evaluation

A. Simulation Setup

Simulations were run using NS-3, a discrete networking simulator alongside a LoRaWAN library [8]. While extremely thorough, this library had not yet implemented the US frequency of 915 Mhz. As a result, simulations were run with the EU standard. While this differs from the hardware purchased for the Fenceless Grazing System, the simulation results should vary insignificantly, and still provide a good baseline for determining the scalability of the FGS project. The simulation code was based on an example provided by the LoRaWAN library, adjusted based on the requirements and network topology of the Fenceless Grazing System [9]. The three reasons measured for packet loss are as follows:

- Interfered: This form of packet loss occurs when two packets collide at the gateway. That is, they were both sent at the same time by two different transceivers with the same spreading factor. As a result, the gateway cannot distinguish between the two of them which results in collision and packet loss.
- No Receivers: A LoRaWAN gateway utilizes a LoRa concentrator - essentially a piece of hardware that hosts multiple radio transceivers to allow for multiple downlinks at the same time. A typical LoRa concentrator can receive eight different packets at once given separate spreading factors on different channels. This form of packet loss occurs when there are no longer any receive paths available to lock onto the incoming packet. That is, all of the channels are occupied already so the packet is dropped.
- Low TX Power: This form of packet loss occurs when a packet is received at a gateway at too low a power. That is, the gateway or end-node is not sensitive enough to be able to correctly decode the message, which would be fixed with an increase in the output power, an increase in the spreading factor, or simply a reduction in distance between the end-node and the gateway.

B. Network Topology

Various simulations were run while adjusting several significant parameters of interest including the number of end-nodes in the network, the number of gateways listening to the end-nodes, the rate of transmission, and the mobility models of the end-nodes. The full software stack was simulated, including the network server which implements adaptive data rate (ADR) and Class A implementation of LoRaWAN. Packets were sent from end-nodes to a single gateway at the center of the simulated area, which then forwards the packets along to the simulated network server. In the cases where there were mulitple gateways, the gateways were placed into the operating area in such a way that they were equidistant from each other and provided the best coverage of the field. Each gateway was placed 5 meters above the ground in order to simulate being placed on top of a house or other structure which provides better line of sight to all the collars. Each end-node was randomly placed in the field, and depending on the mobility model given an initial speed ranging from 0.5 to 11 meters/second. Mobility was handled by the RandomWalk2dMobilityModel provided by NS-3. Some simulations were run with no mobility model, where each end-node was randomly placed into the field and given no velocity vector. Eleven meters/second was chosen as the top speed, as that is the top speed of an average cow. Each end-node continued along it's initial vector until 1km or the end of the field was reached, where in the FGS implementation some form of stimulus would be provided to prevent cattle from crossing over. The path loss exponent was set to 3.033 to best simulate an open field with minimal barriers [12]. The default behavior for each end-node was to send a packet every 10 minutes, however some simulations adjusted this value to determine the affect the rate of transmission has on the scalability of the system. Every simulation was ran for one day of simulation time, resulting in a total of 144 packets sent per device with the default transmission rate. The starting time of each end-node was randomized to prevent packet collisions at time 0 of simulation. Simulated packet size is 12 bytes, which is currently what the FGS sends from each end-node to the gateway. Initial simulations were done with 400 end-nodes and a field of 4km by 4km, and the packet loss was measured as the number of end-nodes increased, while subsequent simulations adjusted the number of gateways, rate of transmission, and lack of mobility.

C. Issues

As mentioned, the library used did not provide the ability to mimic the Fenceless Grazing System perfectly. One of the issues was the lack of support for the US LoRa standard, resulting in the use of the EU standard instead. This resulted in a change in the frequency as well as the different data rates implemented by LoRaWAN. The EU has a much smaller available bandwidth than the US (865-868 Mhz vs 902-928 Mhz respectively). As a result, US devices can hop around the band more to avoid interference. With that said, however, there is usually more interference in the US 915Mhz range then there is in the EU range [10]. Lastly, the EU has restrictions on the duty cycle and transmission power of the end nodes. The duty cycle regulation says that you can only be on the air for 1% of the time. The EU also limits the power output, limiting the potential range [11]. However, the differences between the EU and US in regards to simulations is minimal, and while there is a difference in the bandwidth allocated, simulation results should be conservative enough to be applied to the US standard.

D. Results

The first series of simulations adjusted the number of end-nodes present in the system while observing the packet loss across the entire network. The goal of this simulation was to estimate the true average percentage of packet loss, where packet loss is split into three varying reasons: low transmission power, interference at the gateway, and no available receivers. Each simulation was conducted with a different total number of collars and simulated 24 hours of communication. The RngRun value was also incremented for each simulation, ensuring different starting locations and initial velocity vectors were chosen. Each simulation was run with only one gateway hosted exactly in the middle of the operating field 5m above the ground. The total size of the area was 4 kilometers by 4 kilometers, and each device was sending a packet once every ten minutes. Fig. 11 shows the results for this series of simulations, illustrating the total packet loss as the number of end-nodes increased. Setting the maximum percentage of packet loss at 20%, we can host a maximum of 1500 collars in this network configuration. This graph demonstrates the rapid increase in packet loss, eventually ending at 85% of all packets lost with a singular gateway and 5,000 end-devices.

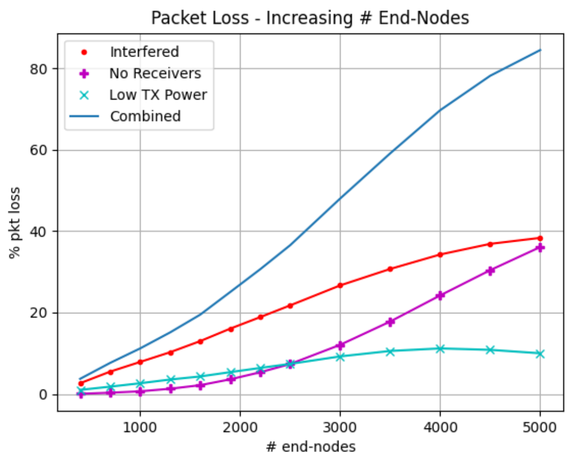


Fig. 11. Packet loss as # end-nodes increases (4km by 4km, one gateway)

Fig. 12 shows how increasing the total number of gateways in our network increases scalability and efficiency with a high density of end-devices. These simulations simply added gateways to the topology of the network at the previous final simulation of 5,000 end nodes. This way the change in packet loss can be determined while easily comparing to the results of the previous simulation set. Each gateway was added to a location in the field such that it was equidistant from the other gateways, and provided the best possible coverage of the operating area. For each new simulation the gateways were moved to different locations. Up to four gateways were simulated, and the results show a drastic decrease in packet loss as the other gateways can pick up the loads and the adaptive data rate algorithm comes more into play. This will prove to be slightly challenging in a real world scenario. Currently the Fenceless Grazing System requires a backend of either Ethernet or Wi-Fi which restricts the locations where a gateway can be deployed. The original thinking was to host it in the home of the rancher where an internet connection would be available. However, if a gateway needs to be hosted in the middle of the field support for another type of backend like 5G will likely be required. With the results in Fig. 12, we were not able to achieve less than 20% of packet loss in an area of 4 kilometers by 4 kilometers.

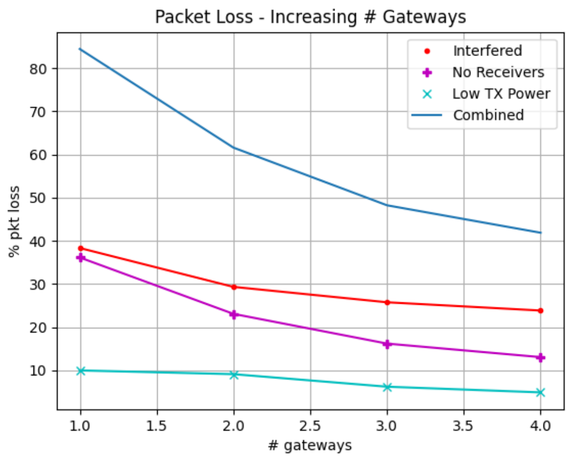


Fig. 12. Packet loss as # gateways increases (4km by 4km, 5,000 end-nodes)

The next set of simulations aimed to see how changing the transmission rate affected the scalability of the network. As we increase the rate of communication for each end-node, the latency for sending a boundary update request will decrease drastically as the FGS product implements Class A of LoRaWAN. Thus, we can expect there to be a perfect ratio of keeping latency down while still maintaining good battery life and network stability. Fig. 13 illustrates the percentage of packet loss as the delay between transmissions increases. That is, the initial value was one minute, indicating that each end-node attempted to send a packet once every minute. As the transmission rate decreases, we see a significant decrease in packet loss. However, this change is the most significant from one minute to about ten minutes, showing that at one point there is no significant decrease in packet loss. Originally, the main cause for packet loss was the lack of available receivers on the gateway as it was overwhelmed by requests. However, this quickly changed as interference became the main reason for packet loss from one transmission every three minutes and onward. Fig. 13 shows that anything less than ten minutes is not decreasing the total packet loss substantially enough to account for the high latency.

Fig. 14 is essentially the same set of simulations as Fig. 11 but with the mobility model disabled. Each end-node was randomly placed in the field and stayed in that location throughout the entire duration of the simulation. Each simulation shows an increase in the total number of end-nodes and the packet loss was graphed against the number of end-nodes. The adaptive data rate algorithm is most impactful when the radio strength of each node remains constant, such as when there is no movement between the nodes. As a result, this set of simulations were conducted to

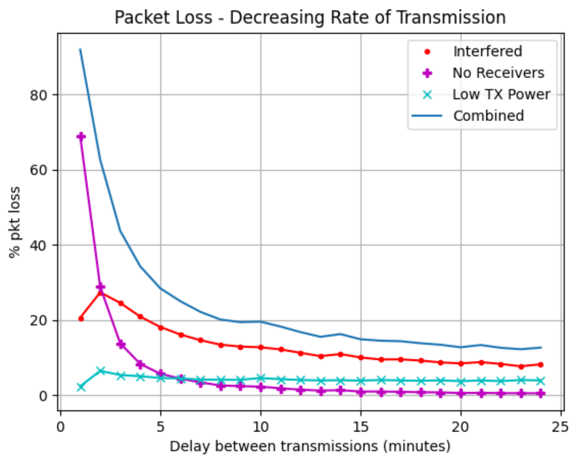


Fig. 13. Packet loss as the rate of transmission for each device decreases (4km by 4km, 1500 end-nodes)

see if there was a significant difference between moving and non-moving devices. In the Fenceless Grazing System it has to be assumed that the end-nodes will be somewhat mobile. However, there will likely be times where each animal stops moving to sleep which would result in the adaptive data rate algorithm being initiated. Comparing the results in Fig. 14 to Fig. 11, there is no significant difference in the total packet loss. While the ADR algorithm decreases time on air, we still see no substantial difference in the rate of interference between these packets. At 5,000 end-nodes the difference between the total amount of packet loss is 0.5%.

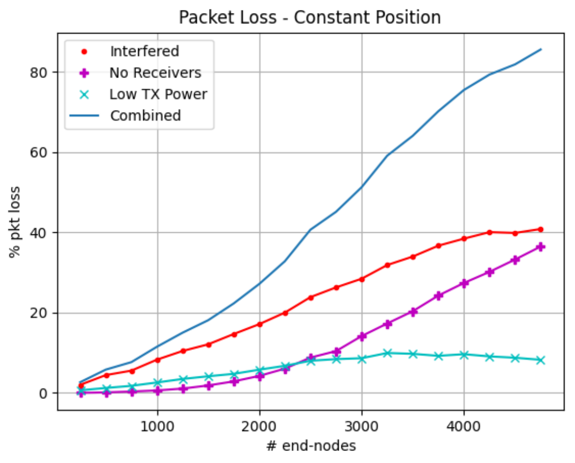


Fig. 14. Packet loss as # end-nodes increases with constant position of each end-node (4km by 4km, one gateway)

Overall, depending on the reliability requirements, one gateway may only be able to handle 1000 - 2000 end-devices. As the current Fenceless Grazing System is only implemented with one gateway, the total number of devices will be limited primarily based on the percentage of packet loss. If a network does not need that many collars the rate of transmission can be increased while still maintaining reliability and network stability at the cost of battery life. If low latency is more desired then the number of end-nodes can be decreased to maintain the same average level of packet loss. These results demonstrate the flexibility of this system and the ability to adjust network parameters based on individual requirements of each implementation.

E. *Maximum Achievable Range*

We have seen how packet loss increases as the number of end-nodes increases within a confined area. The next metric that will constrain the scalability of the Fenceless Grazing System is the maximum achievable distance. There are many things that affect the range of LoRa including the spreading factor or data rate, the transmission power, the hardware itself, and the medium that the radio is trying to penetrate through. For instance, LoRa will not be able to reach nearly as far in a dense forest than it would across open plains with line of sight. While an argument could be made that the majority of farms would be rather flat it is not guaranteed and the achievable distance will adjust for each implementation. Fig. 15 illustrates the required output power to be able to reach a specific distance while still accounting for the various spreading factors. This graph was built using (1) and (2), mathematical representations of the path loss and link budget respectively as detailed in [13], which utilized a loss exponent of 3 to represent an urban area. In (1), L represents the path loss, f is the LoRa frequency, and c is the speed of light. The path loss exponent n was set to 2, and d represents the distance of communication. Equation (2) shows the relationship between the link budget L and the transmitted power P and receiver sensitivity S.

$$L = 10\log_{10}((\frac{4\pi df}{c})^2 d^n) \tag{1}$$

$$L_{Budget}(SF, BW) = P_{TX} - S_{RX}(SF, BW) \tag{2}$$

Then, [13] sets these two equations equal to each other in order to graph the theorized range with regards to spreading factor and output power. These mathematical expressions provides basic assumptions regarding the range estimations for the Fenceless Grazing System. Fig. 15 demonstrates the importance of output power in the maximum achievable range, which will differ across different hardware test-beds. In an effort to compare this model against the FGS implementation, the achievable range of the current test-bed hosting the FGS hardware was tested.

V. TEST-BED IMPLEMENTATION RESULTS

While all of the simulation results could not be verified entirely with the hardware on hand, the theorized maximum communication distance could be tested. The challenging part was to try to find a location that provided similar topology to the simulations. That is, significant distance while still maintaining line of sight. The furthest distance while still maintaining line of sight that was relatively local to my area was 5.25 km, where the gateway was set up at the

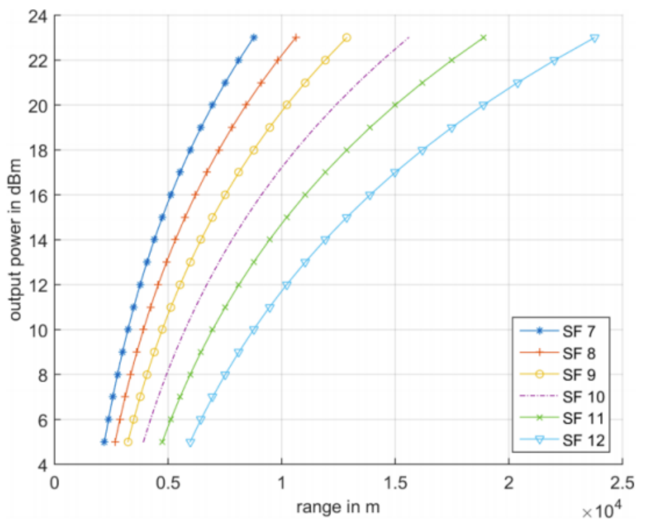
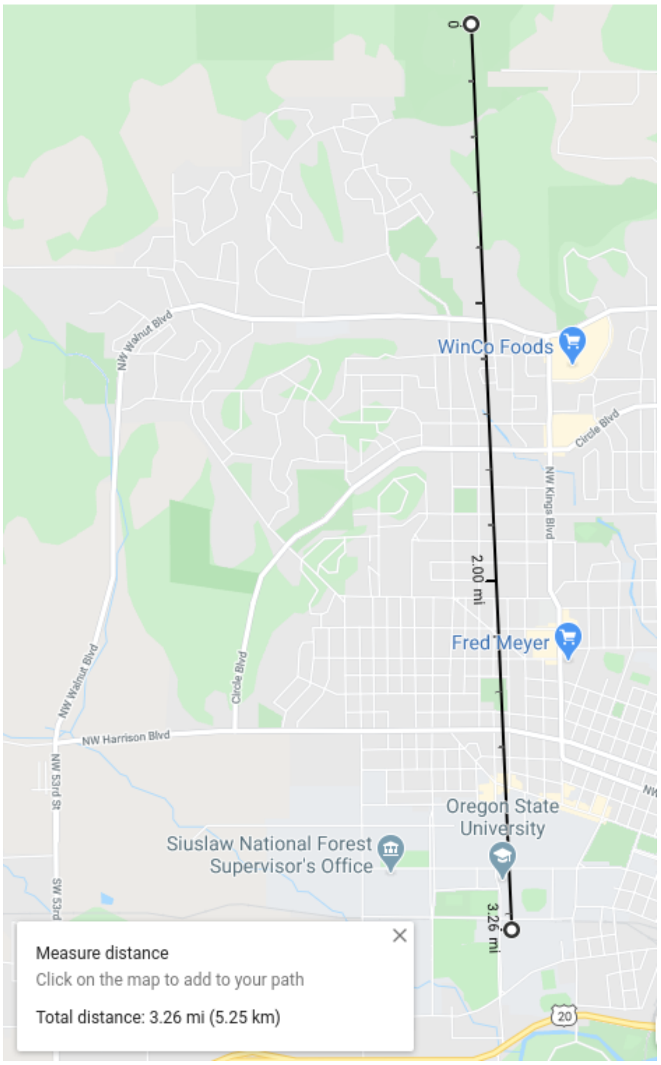


Fig. 15. Achievable range with adjustments in SF and output power [13]

top of a hill across Corvallis, OR to the end-node on top of a parking garage on Oregon State campus (see Fig. 16). We were able to achieve communication at this distance using DR8 (SF 12, BW 500) with an output power of 15 dBm. This data rate corresponds with 980 bits/s with a maximum payload size of 33 bytes. Theoretically, with our expectations based on the data provided in Fig. 15, the maximum distance achievable with these parameters would be about 12 km, over twice the distance tested with the current FGS test-bed. The next test consisted of using DR3 (SF 7, BW 125) with an output power of 6 dBm and did not reach to the full distance. The output power was increased to see when the communication would go through, but we could not achieve communication with a spreading factor of 7 as our library limited the total output power. The lowest data rate that allowed for communication at this range was DR 11 (SF 9, BW 500) with an output power of 14. The non-conformance with the model may be due to many various issues with the hardware, but the most likely scenario is due to cheap antennas. While local area constraints limited the ability to push the range any further, the ability of the current hardware on hand to reach 5.25 km at least allows for far distance communication which corresponds of a maximum acreage of 21,400 acres. This is assuming a circular field with a radius of 5.25 km which is simply the furthest distance tested with the current hardware. The LoRa protocol easily provides long distance communication making the likely constraint to the Fenceless Grazing System the ratio of the number of end-devices to gateways.

VI. Conclusions

The LoRa communication protocol provides the Fenceless Grazing System with great scalability in terms of both range and number of devices. With the information gathered through the NS-3 simulations, the theoretical maximum number of end-devices is determined to be just under 2,000 with



Measure distance

Click on the map to add to your path

Total distance: 3.26 mi (5.25 km)

Fig. 16. Maximum Achieved Distance

only a singular gateway and a cutoff at 20% of total packet loss. This provides the knowledge necessary to implement the FGS project in a multitude of different situations and based on the topology and structure of different ranches a network administrator can determine whether or not it is feasible to implement this system. Also, the achievable range was confirmed to be on par with other LoRa implementations, and with the maximum tested distance to be 5.25 kilometers the FGS project supports significantly large areas of land. Potential improvements to the Fenceless Grazing System include developing support for multiple gateways as this is the limiting factor to the size of the network both in terms of the number of animals as well as the acreage of the grazing area. With these restrictions and capabilities in mind, ranches can be evaluated and a full scale model of this product can be built and deployed.

Acknowledgment

I thank Dr. Bechir Hamdaoui (Oregon State University) for his assistance and support throughout the design and development of this thesis. His knowledge and experience helped to streamline the development of the Fenceless Grazing System as well as the set of simulations, and his input could not have been more valuable. With his help, the research and testing aspects of this thesis went smoothly.

References

[1] F. Dahlqvist, M. Patel, A. Rajko, and J. Shulman, “Growing opportunities in the Internet of Things,” *McKinsey & Company*, Jul-2019. [Online]. Available: https://www.mckinsey.com/industries/private-equity-and-principal-investors/our-insights/growing-opportunities-in-the-internet-of-things. [Accessed: 06-Apr-2020].

[2] B. Ray, “What Is LoRa? A Technical Breakdown,” *Link Labs: Cost-Effective Connectivity For IoT*, 26-Jun-2018. [Online]. Available: https://www.link-labs.com/blog/what-is-lora. [Accessed: 01-May-2020].

[3] Semtech Application Note AN1200.22, "LoRa Modulation Basics", *Semtech*, May-2015. [Online]. Available: http://wiki.lahoud.fr/lib/exe/fetch.php?media=an1200.22.pdf. [Accessed: 01-May-2020].

[4] Qoitech, “How Spreading Factor affects LoRaWAN device battery life,” *The Things Network*, 18-Nov-2019. [Online]. Available: https://www.thethingsnetwork.org/article/how-spreading-factor-affects-lorawan-device-battery-life. [Accessed: 01-May-2020].

[5] “Modulation & Data Rate,” *The Things Network*, 30-Mar-2020. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/modulation-data-rate.html. [Accessed: 01-May-2020].

[6] N. Sornin, M. Luis, T. Eirich, T. Kramp, O. Hersent, "LoRaWAN Specification V1.0," *LoRa Alliance*, 2015. [Online]. Available: https://lora-alliance.org/sites/default/files/2018-05/2015_-_lorawan_specification_1r0_611_1.pdf. [Accessed: 01-May-2020].

[7] “Adaptive Data Rate,” *The Things Network*, 25-Oct-2019. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/adaptive-data-rate.html. [Accessed: 01-May-2020].

[8] D. Magrin, M. Capuzzo, S. Romagnolo, M. Luvisotto, LoRaWAN ns-3 Module, 29-Apr-2020, Github repository, https://github.com/signetlabdei/lorawan.

[9] R. Alder, Honors Thesis, May-2020, Github repository, https://github.com/ryanalder/honorsthesis.

[10] G. Schatz, “How To Launch An IoT Application In Europe Vs. America,” *Link Labs: Cost-Effective Connectivity For IoT*, 17-Feb-2016. [Online]. Available: https://www.link-labs.com/blog/launch-iot-application-europe-vs-america. [Accessed: 01-May-2020].

[11] Saelens, M., Hoebeke, J., Shahid, A. et al. Impact of EU duty cycle and transmission power limitations for sub-GHz LPWAN SRDs: an overview and future challenges. *Wireless Com Network 2019*, 219 (2019). https://doi.org/10.1186/s13638-019-1502-5

[12] R. E. Chall, S. Lahoud, and M. E. Helou, “LoRaWAN Network: Radio Propagation Models and Performance Evaluation in Various Environments in Lebanon,” IEEE Internet of Things Journal, vol. 6, no. 2, pp. 2366–2378, 2019.

[13] A. Pötsch and F. Haslhofer, "Practical limitations for deployment of LoRa gateways," 2017 IEEE International Workshop on Measurement and Networking (M&N), Naples, 2017, pp. 1-6.

[14] C. Riederer, TinyLoRa, Oct-2019, Github repository, https://github.com/adafruit/TinyLoRa.

[15] “Protocol Buffers | Google Developers,” Google. [Online]. Available: https://developers.google.com/protocol-buffers. [Accessed: 05-May-2020].

[16] “RAK2245 Pi Hat,” The Things Network, Oct-2019. [Online]. Available: https://www.thethingsnetwork.org/docs/gateways/rak2245/. [Accessed: 05-May-2020].

[17] “The Things Network,” The Things Network. [Online]. Available: https://www.thethingsnetwork.org/. [Accessed: 05-May-2020].