Linköping University | Department of Computer Science
Master thesis, 30 ECTS | Computer Technology
2017 | LIU-IDA/LITH-EX-A--17/054--SE

Investigating the practical performance of the LoRaWAN technology

Joakim Eriksson
Jonas Skog Andersen

Supervisor : Vengatanathan Krishnamoorthi
Examiner : Niklas Carlsson

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

New innovations, technologies, ideas and businesses are driving the realisation of the Internet of Things (IoT) vision. As with many other fields in technology comes competing protocols and standards, ranging from modulation schema used for transmitting data to security standards used to ensure safe operation and the privacy needs for all involved entities. This thesis looks into one of the competing modulation schema and network protocols for IoT applications: the LoRaWAN protocol. The main contribution of this thesis is a data-driven empirical study that helps verify theoretically obtained results from other authors. Our results also suggest that as long as other signals on the same frequency band uses different modulation techniques (or just other parameters for the same modulation technique), then only the signal to noise ratio is affected without introducing collisions. This affects the scalability and overall practical distance covered by a LoRaWAN. Our general conclusion is that the LoRaWAN as a technology/protocol has its disadvantages, mainly how heavily different traffic profiles may affect the scalability of it and a general lack of hard quality of service guarantees.

Acknowledgments

Firstly, we would like to thank our supervisor Niklas Carlsson for his guidance and support during this thesis. It has been most greatly appreciated. And also, our supervisors at Etteplan Sweden AB, Erica Dahlberg and Lars Karlsson, for their ideas, guidance and feedback. The thesis was carried out at Etteplan Linköping, again it has been a privilege to work among you all.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Modern wireless technology is the catalyst for connecting new objects and devices to the internet. People want to use a mobile app to preheat their cars on a cold morning, unlock their house from a remote location so their friend can drop by and feed the cat, monitor the energy usage of various devices in their homes and much more, all through the use of wireless technology. Industrial environments also utilise wireless mediums to connect sensors and actuators enabling automatic planning, monitoring, machine to machine communication, big data analysis and several other use cases.

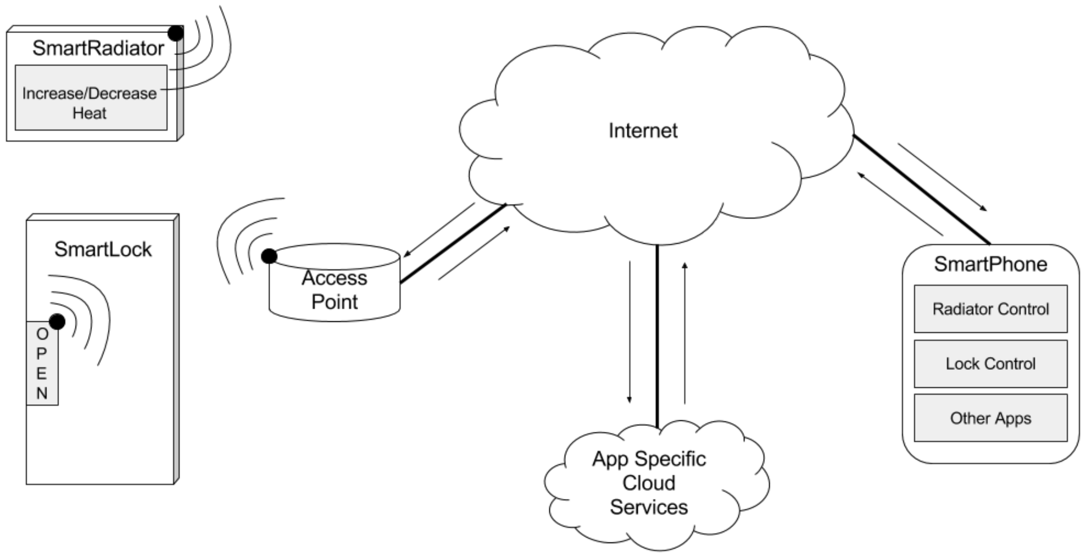


Figure 1.1: An abstract view over some IoT applications that may be found in a modern home.

The growing trend of connecting various *things* to a wireless network and thus being able to remotely access them is starting to realise what is often referred to as the Internet-of-Things (IoT). To achieve this, various techniques are used, all depending on the use case. One growing technique for long distance communication is the Low-Power Wide-Area Networks (LPWANs). LPWANs are wireless protocols developed for constrained devices such as those often utilised in IoT services. The tradeoff is the lower bitrate in exchange for lower energy consumption. One such LPWAN is the Long Range low power Wide Area Network (LoRaWAN). LoRaWAN is a bidirectional IoT communications protocol with long reach but low data rate.

Since LoRaWAN operates in the unlicensed spectra around 868MHz in Europe and 915MHz in North America no licence or permission is needed to setup a LoRaWAN. The combination of the long range and low threshold to get started makes LoRaWAN a great candidate to use in many IoT-applications.

The drawback of using the unlicensed spectrum is that there are several applications operating within the same frequency band, something which may cause interference between the applications. To not flood the a frequency band, LoRa modulation schema in Europe enforce a per sub-band Duty Cycle (DC) limitation to cope with the regulations issued by the European Telecommunications Standards Institute (ETSI). The DC specifies how much of the total time a transmitter is allowed to transmit per hour on that sub-band. For example, if a sub-band has a DC of 1%, the total amount of time the transceiver spends transmitting data must not exceed 36 seconds/hour.

The LoRa modulation schema is based on a chirped spread spectrum technique making it robust to channel noise and also resistant to multi-path fading. Due to the limitation of the DC and the low throughput, somewhere between 250 to 50000 bits/s, it is important to keep the collision rate as low as possible.

1.2 Aim

The aim of this thesis is to estimate and measure the average throughput in a LoRaWAN for each Data Rate (DR) and also investigate the upper bound for transmitting data with a Multitech xDot LoRA card[1]. This thesis also aims to investigate performance in a LoRaWAN, namely how LoRa and LoRaWAN performs under interference from devices utilising the same frequency bands.

1.3 Research questions

1. What is the practical throughput for each data rate in a LoRaWAN?
2. How does LoRa and LoRaWAN operate under interference from other devices on the same frequency sub-band?

1.4 Approach

To answer the research questions an IoT-environment was created and utilised for experiments. The environment contained one access point, called gateway, and two IoT end-devices. The end-devices were placed 3 meters apart and the gateway was placed between them.

In order to measure and estimate the throughput for each data rate(DR) in LoRaWAN, one of the IoT end-devices was configured to broadcast data to the access point. The execution time of the data-broadcasting function on the end-device was then measured, which helps derive the throughput.

To investigate how LoRa and LoRaWAN operate under interference, one of the end-devices was configured to act like a normal IoT-device while the other device acted as a "jammer". By doing this on each and every sub-band the interference can be estimated with the help of the

[1]Multitech xDot Micro Developer Kit MTMDK-XDOT-EU1-A00 - http://www.multitech.com/models/94558024LF

Received Signal Strength Indication (RSSI), Signal to Noise Ratio (SNR), and the packet loss rate. All test cases were compared and tested against a common benchmark.

1.5 Delimitations

This thesis does not take multipath propagation into consideration at all, mainly since the results were evaluated against benchmarks which should face the same problem. Since the authors had only two end-devices available, it was not possible to scale the test cases with more devices. Neither were any range test be done since the end-devices used was a developer unit not properly shielded from the force of nature. Past work regarding LoRaWAN also already investigates this matter.

It should be noted that LoRaWAN is a "connectionless technology" which means that there are no streams of packets from one device to another. Thus, when measuring the throughput for data rates in LoRaWAN, what is really looked into is the throughput for *each* broadcasted packet.

2 Theory and related work

This section covers the theory necessary for this thesis and provides an overview of related works. At a high level the chapter defines and explains important concepts related to the IoT, Low Power Wide Area Networks (LPWANs), LoRaWAN and general signal characteristics.

2.1 Internet of things

The concept of Internet of Things (IoT) is a broad term that captures the connection of any physical object to the internet. There are different means of connecting a physical object to the internet, but e.g. it can be done by attaching an RFID tag to the object (connecting it through some scanning device) or by integrating an embedded circuit with some kind of communication interface. While classic desktop computers, laptops, smart phones and smart tablets are all captured in the concept of IoT, it makes sense in the context of this thesis to only refer to weaker computational entities, or rather constrained devices when talking about end-devices or end-devices in IoT. A Constrained Device has the following characteristics as described by RFC-7228[8]:

- constraints on RAM
- constraints on processing power
- constraints on Flash/ROM
- constrained energy supply
- constraints on user interface and accessibility during deployment (or bootstrapping)

Miorandi et al. [21] described the term IoT as the resulting global network unifying objects by means of extended Internet technologies in between objects. They also claim that IoT can also refer to the set of supporting technologies needed to implement such solution, e.g. RFIDs, sensor/actuators, machine-to-machine communication etc. This general definition of IoT does not always add up with the image of IoT that others have. For example, the Internet of Things Architecture project by the European Commission [10, 9], Vermesan et al. [38, 37], and Sicari et al. [29] all have slightly different definitions about what exactly the IoT definition and vision is. However, the core concept shared by these experts seems to be that some constrained devices communicate through some wireless technique with some type of access point, that is in some way connected to the internet. The later definition is the one utilised in this thesis.

2.2 Wireless techniques for IoT

The wireless techniques can mainly be divided into two categories, short range and long range. The general trade-off between short range and long range techniques being that throughput and general performance is reduced the larger the range of the technique. [14].

2.2.1 Short range techniques - Wi-Fi and Zigbee

An example of a short range technique is Wi-Fi which is based on the IEEE 802.11 standard, used to create wireless local area networks (WLAN). The IEEE 802.11 standard contains a set of specifications for implementing the media access control (MAC) and physical layer (PHY). The typical architecture of a WLAN contains one access point and one or several devices communicating with the access point. Wi-Fi operates on the 2.4 GHz and 5.0 GHz Industrial, Scientific, and Medicinal (ISM) radio bands [15].

Zigbee is another technique for short range communication. Zigbee is based on the IEEE 802.15.4 specification. Zigbee is designed for smaller projects that demands low-power consumption while not having high throughput as a necessity. Zigbee is supposed to be simpler and less expensive than wireless networks such as Wi-Fi. Today Zigbee is widely used in home automation mostly for energy monitoring, light switches, temperature measurement etc. Zigbee also utilises the 2.4 GHz band, but also operates on the sub-GHz ISM bands, $863 - 870MHz$ in Europe [3].

2.2.2 Long Range techniques - Low-Power Wide-Area Networks

In relation to IoT the term Low-Power Wide-Area Network (LPWAN) is often encountered. IoT end-devices usually have a constrained energy supply, meaning that the general energy consumption needs to be restricted. For the communication interface, this means that the transmitter and receiver has to restrict the amount of power to use when sending data.

On the market there are a number of competing standards that can be found. The Ultra Narrow Band (UNB) modulation technique Sigfox is one of the LPWAN techniques found on the market today. Sigfox utilises the ISM band, $863 - 870MHz$ in Europe, and allows communication up to 3-10 kilometers in urban areas. Sigfox claims that with good line-of-sight the distance can reach up to 50 kilometers [2]. Since Sigfox is based on UNB modulation it can send at 0.1 to 0.6 kbits/second at each 100Hz of the signal, which allow high spectral efficiency. The radio modulation hence make it possible to achieve communication over long distances and still be robust against noise [30].

One other competing LPWAN technique is LoRaWAN. Since this thesis focuses on LoRaWAN, the next section is dedicated to it.

2.3 Long Range, low power Wide Area Network (LoRaWAN)

Low-Power Wide-Area Networks (LPWANs) are wireless protocols developed for constrained devices such as those often utilised in IoT services. The tradeoff is usually slower bit rate in exchange for lower energy consumption and/or greater coverage. Most LPWANs need a gateway to communicate with the Internet since the network is not based upon IP. One such LPWAN is the Long Range, low power Wide Area Network (LoRaWAN) [31], a low power, bidirectional IoT communications protocol with long reach but low data rate.

2.3.1 LoRa and LoRaWAN - A general description

LoRaWAN is a network protocol that utilises the modulation schema LoRa. They are thus two different things although they have an intertwined relationship.

LoRa - The modulation schema

LoRa is a PHY-layer modulation technique based on Chirp Spread Spectrum with forward error correction, making the signals robust to interference and channel noise. It is developed by Semtech[1] and the modulation technique is proprietary, thus no open standard nor detailed description is available (to the best of the authors knowledge). The closest thing found is the description provided in various datasheets for LoRa transcievers [2,3] and an official LoRa FAQ [4] released by Semtech. The modulation technique utilises three different parameters that impact mainly robustness and bit rate: the Spreading Factor (SF), the Bandwidth (BW), and the error Correction Rate (CR). A benefit of the spread spectrum modulation is that signals with orthogonal SFs can coexist with other LoRa-modulated signals on the same channel without interfering, as well as coexist with Frequency Shift Keying (FSK) modulated signals.

LoRaWAN - The network protocol

LoRaWAN is an open standard network protocol managed and updated by the LoRa-Alliance[5]. Put short, the LoRaWAN protocol [31] defines how to build a scalable network utilising the LoRa modulation schema; such as what components make up a network, their corresponding relationships and the architecture, how the PHY-payload in LoRa packets should be formatted, how channel access is handled, what frequencies to use for transmissions, and much more.

2.3.2 Components, general architecture, and network topology

The LoRaWAN architecture presented in [31] can be seen in Figure 2.1. A LoRaWAN mainly consists of the following components:

- End-devices : These are devices that either receive downlink traffic from the network server or generate uplink traffic.
- Gateways : These are devices that demodulate the LoRa traffic and relay traffic between the network server and end-devices.
- Network Server : The network server is the central backend of a LoRaWAN, gathering the traffic from all end-devices in the network and forwarding the traffic to an application server.

These type of network may consist of many end-devices. The end-devices may communicate with the network server via the gateways using either the LoRa modulation schema, or FSK modulation. Gateways scan the spectrum and capture LoRa (and FSK) modulated packets. The gateway(s) then forwards the traffic to the network server using some type of network, possibly regular TCP/IP network or some cellular network technology.

[1]Semtech Corporation - http://www.semtech.com/
[2]LoRaSX1272 - http://www.semtech.com/images/datasheet/sx1272.pdf
[3]LoRaSX1276 - http://www.semtech.com/images/datasheet/sx1276.pdf
[4]LoRa FAQ - http://www.semtech.com/wireless-rf/lora/LoRa-FAQs.pdf
[5]LoRa-Alliance - https://www.lora-alliance.org/

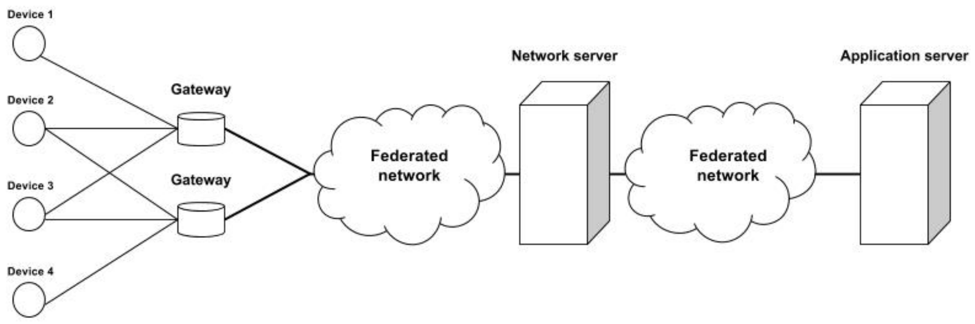


Figure 2.1: The LoRaWAN architecture with end-devices, gateways, a network server and also an application server present.

The function of the gateway can thus simply be described as a packet forwarder. The key function of the network server is to interpret the data that end-devices send, but also to handle LoRaWAN features and the overall management of the network. For example: one important function of the network server is to de-duplicate packets received by several gateways[6], a scenario which occurs if an end-device is located within the coverage of two gateways. Both gateways may pick up a broadcasted packet and forward it to the network server [31], thus introducing the need to eliminate duplicates. The network server also keeps track of information regarding each individual end-device in the network in order to help optimise the routing of traffic to them.

2.3.3 General channel access plan and modulation parameters

The LoRaWAN protocol defines the modulation parameters to use for the LoRa protocol, as well as what frequency channels that are to be used based on what region (Europe, North America, Asia...) the LoRaWAN is situated in. LoRaWANs utilise the unlicensed ISM frequency bands, which vary depending on the region the network is localised in. The protocol thus also defines how to abide by local rules and regulations concerning the utilised channels.

General channel access

The unlicensed ISM frequencies that LoRaWANs operate in are usually subject to rules and regulations. This paper will only cover information regarding use cases in Europe, where the ISM band in question is $863 - 870MHz$. As per LoRaWAN specification v1.0.2 [31], LoRaWAN devices in Europe enforce a per channel Duty Cycle (DC) limitation to cope with the regulations issued by the European Telecommunications Standards Institute (ETSI).

The DC specifies how much of the channel a transmitter is allowed to occupy. For example, if a channel has a DC of 1%, the total amount of time the transceiver spends transmitting data on that channel must not exceed 36 seconds per hour. To make sure that a device stays within an issued DC, a wait-timer T_{off}^{CH} is calculated after transmission. The device must then wait T_{off}^{CH} seconds before transmitting on that specific channel again, but is free to use other available channels in the meantime. Equation (2.1) explains how T_{off}^{CH} is calculated. If the DC is 1% and a transmission takes $1s$, then $T_{off} = 99s$.

$$T_{off}^{CH} = \frac{TimeOnAir}{DutyCycle} - TimeOnAir \quad \text{[s]} \tag{2.1}$$

[6]LoRa network structure - https://docs.mbed.com/docs/lora-with-mbed/en/latest/intro-to-lora/

The wait timer is something that applies to *all* LoRaWAN transmitters, including the gateways. The way end-devices in a LoRaWAN schedule their uplink messages is application specific. An end-device in a LoRaWAN can at any point want to broadcast data, and will do so unless the DC limitation hinders it. There is no listen before talk or Carrier Sense (CS) in a LoRaWAN, so the channel access is much like ALOHA. Uplink messages can be of two different types: confirmed and unconfirmed, where a confirmed message prompts an ACK from the gateway.

Modulation parameters for wireless access

Recall that when using the LoRa modulation schema, mainly three parameters are present: the Spreading Factor (SF), the Bandwidth (BW), and the error Correction Rate (CR). The LoRaWAN protocol introduces the concept of Data Rates (DR) to more easily differentiate between different combinations of these parameters. Eight different Data Rates (DR0-DR7) are defined, and their respective LoRa-configurations can be found in Table 2.1 together with the maximum transmission unit for each data rate. While the modulation parameters SF and BW are set for all data rates, the CR is configurable for some of them. The possible values for CR in LoRa are $\frac{4}{4+n}$ where $n \in \{1,2,3,4\}$ [5], however CR in LoRaWAN is defaulted to either $4/6$ or $4/5$.

Data rate	Modulation			Maximum transmission unit		Indicative bit rate
DRy	SF	BW	CR	Total	App Payload	x bit/s
DR0	12	125kHz	4/6	64B	51B	250 bit/s
DR1	11	125kHz	4/6	64B	51B	440 bit/s
DR2	10	125kHz	4/5	64B	51B	980 bit/s
DR3	9	125kHz	4/5	128B	115B	1760 bit/s
DR4	8	125kHz	4/5	255B	242B	3125 bit/s
DR5	7	125kHz	4/5	255B	242B	5470 bit/s
DR6	7	250kHz	4/5	255B	242B	11000 bit/s
DR7	FSK			255B	242B	50000 bit/s

Table 2.1: The table illustrates the spreading factor, bandwidth, maximum transmission unit and the physical maximum bit rate.

2.3.4 Managing downlink traffic

Depending on how end-devices in a LoRaWAN should schedule the reception of downlink traffic, they can be configured to operate as one of three different classes: Class A, Class B, and Class C. Depending on which device class an end-device is configured as, the downlink communication between the gateway and the transceiver is handled differently. All LoRaWAN end-devices have the ability to operate as Class A devices. The application running on the end-device can change the operation of the device depending on its needs, if the hardware supports operation as Class B or Class C.

Class A device - baseline

End-devices operating as Class A are only able to receive data after first transmitting an uplink message. After transmitting an uplink message, there are two scheduled receive windows for the end-device named $RX1$ and $RX2$. After finishing an uplink transmission, the end-device waits $RECEIVE\_DELAY1$ seconds (+/- 20 ms) before opening up the first receive window $RX1$. By default $RX1$ uses the same channel and Data Rate as the uplink transmission.

$RECEIVE\_DELAY2$ seconds (+/- 20ms) after the end of the uplink message, receive window $RX2$ is opened. $RX2$ uses a fixed but configurable channel and data rate. The length of $RX1$ and $RX2$ is configurable but must be guaranteed to be longer than the time required by end-devices radio transceiver to detect the preamble of the downlink packet. If a message arrives at a gateway and is destined for a device whose $RX1$ and $RX2$ windows have timed out, that message cannot be delivered until that end-device decides to send a new uplink message, and must thus be buffered for an unknown amount of time. Figure 2.2 shows the receive windows and their timers.

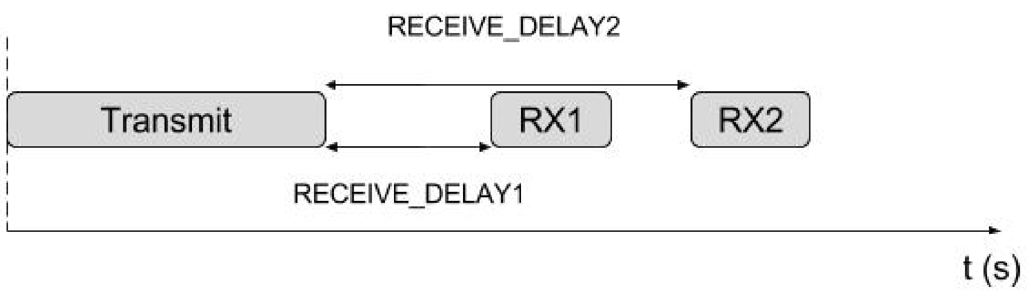


Figure 2.2: The scheduling of receive windows $RX1$ and $RX2$ in Class A LoRaWAN devices.

Class A operation is meant for devices that need to optimise their battery lifetime.

Class B device - beacon

Class B is currently considered to be experimental as per LoRaWAN Specification v1.0.2 [31]. Class B adds synchronised reception windows, and class B operation should be implemented when there is a requirement to open such receive windows at fixed time intervals so that server initiated downlink messages are made possible.

In order to synchronise the reception windows, the gateway sends a beacon on a regular basis to the end-devices in a LoRaWAN network. The beacon thus contains information that helps synchronise the end-devices of the network so that they can open an extra reception window (named "ping slot") at predictable times during periodic time slots.

Class C device - continuous

Class C operation is meant for end-devices with continuous power supply. Class C end-devices continuously listens for data when not sending. Technically, this is achieved by extending the $RX2$ receive window until the next uplink transmission is scheduled. After transmitting data, the class C device opens the $RX2$ receive window for $CLS2\_RECEIVE\_DELAY1$ seconds. After $CLS2\_RECEIVE\_DELAY1$ seconds $RX2$ is closed in order to open receive window $RX1$. After $RX1$ closes, receive window $RX2$ is once again opened for reception and remains open until the next uplink transmission. Figure 2.3 shows how $RX2$ is scheduled in class C devices.

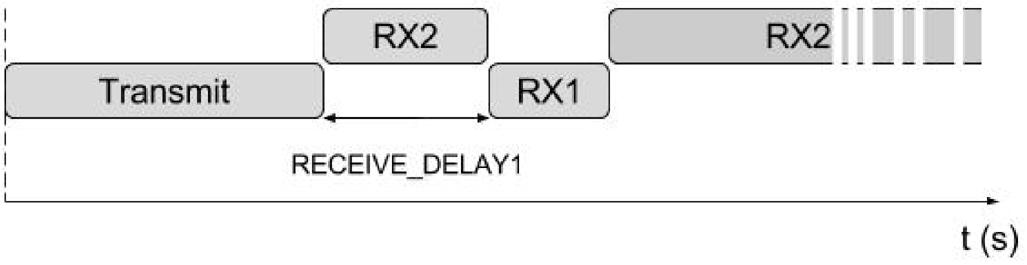


Figure 2.3: The Class C LoRaWAN device has window $RX2$ open almost constantly.

2.3.5 Security in LoRaWAN

In order to discuss the security in LoRaWAN, the LoRa-MAC layer format must first be described. Further, Confidentiality, Integrity, and Availability (CIA) enforcing mechanisms in LoRaWAN are described.

MAC-Layer format in LoRaWAN

The MAC Layer, technically the physical payload (PHYPayload) on the physical layer, consists of three fields: the MAC Header (MHDR), the MAC Payload of a message (MACPayload), and the Message Integrity Check (MIC). The MAC Header specifies the message type and metadata about the MACPayload. The MIC field contains the Message Integrity Code calculated for the MHDR and MACPayload.

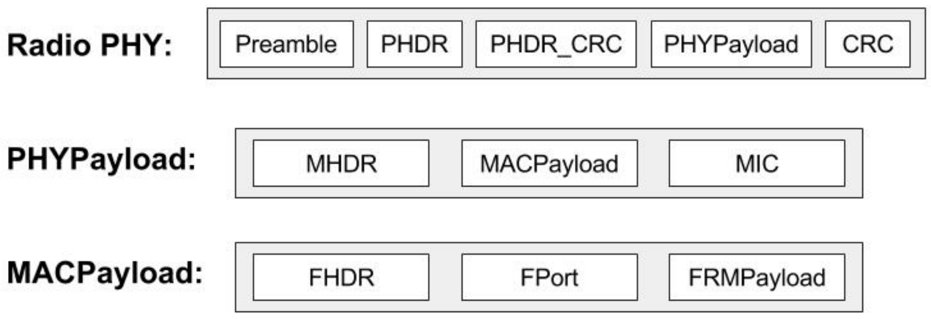


Figure 2.4: LoRaWAN Link/MAC Layer format.

The MACPayload contains three important fields: The Frame Header (FHDR), the Port field (FPort), and the MAC Frame Payload (FRMPayload). The FHDR contains the source device address, a frame control field as well as a frame counter field and an options field. The FPort field is a value $0 - 255$ and is application specific, where the value 0 is reserved and indicates that the MACPayload is a MAC command message for network control. The FRMPayload may be either MAC commands for the network or application specific payload destined to an application server. Figure 2.4 gives an overview of the MAC layer format.

The port field (FPort) is only required when the frame payload is not empty. Thus the MACPayload is 12 bytes if the FPort is not included in the packet and 13 bytes plus the size of FRMPayload when it is included. A FPort value of 0, indicate that the FRMPayload contains a MAC command. FPort values of 1 to 233 are application specific while 224 to 255 are standardised application extensions [31].

LoRaWAN encryption and integrity check

The confidentiality of LoRaWAN messages is protected by AES-128 encrypting the FRMPayload field. LoRaWAN differentiates between MAC messages destined for the network server and application messages destined for the application server through the FPort field (which is not encrypted), thus two different encryption keys are used for the FRMPayload field depending on the intended destination. End-devices in LoRaWAN thus needs two 128-bit encryption keys, one Network Session Key (NwkSKey) for encrypting MAC messages and one Application Session Key (AppSKey) for encryption application messages. The default AES version used in LoRaWAN is AES-CTR.

The AppSKey is an encryption key shared between an end-device and the application server, while the NwkSKey is an encryption key shared between the end-device and the network server. How the keys are used is illustrated in Figure 2.5 [12].

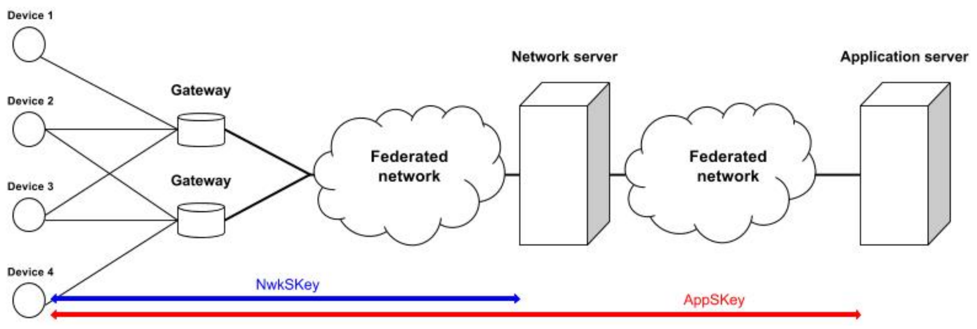


Figure 2.5: The LoRaWAN architecture illustrating how the AppSKey and the NwkSKey are shared and used.

In order to provide message integrity, a Message Integrity Code (MIC) is computed for each packet using AES-CMAC and the NwkSKey. The MIC is computed over the entire PHYPayload (after the FRMPayload has been encrypted) in order to detect packet tampering[31]. The composition of the encrypted and integrity checked message is explained in Figure 2.6 [12].

The message integrity check is only present between the end-device and the network server, and thus messages between the network server and the application server have no integrity mechanisms. Technically this means that there is no integrity mechanism available for the application server to detect whether the network server has tampered with data before sending it to the application server [31]. A similar problem applies when the application server is connected to the network server through a federated network, and some entity present in the federated network tampers with data destined for the application server. The same situation also applies in the reverse scenario when the application server sends data towards the end-device. LoRa-Alliance[7] suggests using secure transport solutions such as HTTPS or VPN between the network server and the application server for this reason [12]. It should be

[7]LoRa-Alliance - https://www.lora-alliance.org/

noted though that such a solution requires trust in the network server.

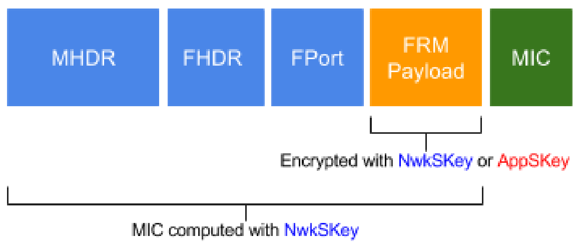


Figure 2.6: The MIC is computed over the PHYPayload. The FRMPayload is encrypted with either the AppSKey or the NwkSKey, while the MIC is computed with NwkSKey.

2.3.6 Join procedure and key distribution in LoRaWAN

The LoRaWAN specification defines two different procedures for end-devices to join a LoRaWAN, which differ in how the encryption keys are derived and distributed. The join procedures are named: Over-the-Air Activation and Activation By Personalisation.

In both join procedures the keys are generated with AES1 cryptographic algorithms and have a length of 128 bits, a so called AES-128 key. The keys are also approved by NIST, a security framework [32] for constrained devices and networks [12]. Each LoRaWAN-device is also equipped with a globally unique identifier (EUI-64-based DevEUI), which is used during the device authentication process [31].

Over-the-air activation

Over-the-Air Activation (OTAA) is based on a globally unique identifier. The commissioning in OTAA is similar to how the 4-way handshake in a WiFi-network works. The DevEUI of an end-device has a similar function to that of a MAC-address in WiFi, the AppEUI has a similar function to the SSID in WiFi, and the AppKey in OTAA has a similar function to that of a password in WiFi. The OTAA uses an over the air message handshake that is carried out as follows [31]:

1. The end-device transmits a join-request message to the network server (indicated by setting the MType field in the MHDR to 000). The join-request message is not encrypted but it is protected by the MIC, in this case calculated with the Application Key (AppKey), which is an AES-128 key. A join-request contains:
    - Globally unique end-device Identifier (DevEUI)
    - Application Identifier (AppEUI)
    - A nonce of 2 octets (DevNonce). The DevNonce is a random value that the network server keeps track of for each end-device.
2. The end-device receives a join-accept message (indicated by the MType field in the MHDR set to 001) from the network server if the end-device is permitted to join the network. The join-accept message contains two important values: a random 24-bit value AppNonce from the network server and a 32-bit device address (DevAddr). The join-accept also contains a network identifier (NetID).

3. The join-accept is authenticated at the end-device, done by calculating the MIC for the message. This is possible since both the network server and the end-device have the AppKey that is used for calculating the MIC. Also note that the AppKey is never sent over the network.

4. The join-accept message is then decrypted by the end-device, and the Device Address (DevAddr, which is distributed by the network server), is extracted and stored. The network server uses an AES decrypt operation in ECB mode to encrypt the join-accept message. The end-device can thus use an AES encrypt operation to decrypt a join-accept message. This way an end-device only has to implement AES encrypt but not AES decrypt for join-accepts.

5. The end-device obtains the NwkSKey, the AppSKey, and metadata regarding the network (RX Delays etc.). The keys are calculated as follows:

    - NwkSKey=aes128_encrypt(AppKey,0x01 | AppNonce | NetID | DevNonce | pad_{16})
    - AppSKey=aes128_encrypt(AppKey,0x02 | AppNonce | NetID | DevNonce | pad_{16})

The NwkSKey and AppSKey are thus unique for each device [31]. A problem that arises in a federated LoRaWAN is that the network server has all the information to derive the AppSKey used for application data encryption. In a federated network an application provider has to support the network operator in the process of end-devices joining and establishing the encryption keys [31].

The LoRaWAN specification [31] also mentions that the AppKey should be specific for each end-device, ensuring that if the AppKey is extracted from one end-device, the integrity of the remaining network devices is still intact.

Activation by personalisation

The alternative way of joining a LoRaWAN and sharing keys is the Activation By Personalisation (ABP) procedure. ABP relies on using pre-shared network and application keys, mainly stored at production time. These keys are locked to a specific network server and application server. The following information is pre-shared in ABP:

- Device Address (DevAddr)
- Network Session Key (NwkSKey)
- Application Session Key (AppSKey)

When implementing ABP, it is required to give each device a unique set of NwkSKeys and AppSKeys. Compromising the keys for one end-device should not compromise the security of the communication for other devices in the system. It is also important that the keys are computed in such a way that they cannot be derived from "public information" [31].

2.4 Signal characteristics

There are differences between wired and wireless communication: the signal strength will decrease when the signal passing through walls and other objects. If the distance between the sender and the receiver is too great the signal will also decrease to an unreadable signal or not reach the receiver at all. One other parameter impacting the signal is interference from other sources, mainly sources sending in the same frequency band. A signal can also be reflected on the ground or on objects, causing the the same signal to reach the receiver multiple times, each instance slightly offset in time from the other. The phenomenon is called multipath propagation and will cause the receiver to register the additional signals as noise, causing the signal to be less clear [15].

2.4.1 Signal-to-noise ratio

The Signal-to-Noise Ratio(SNR) is a ratio between the level of the signal and the level of noise. The ratio is represented in the logarithmic decibel scale. Equation 2.2 describes how SNR is calculated. P_{signal} is the power of the signal and P_{noise} is the measured power of the noise. Note that (as equation (2.2) implies) the SNR is a unitless ratio, although often converted to dB.

$$SNR = \frac{P_{signal}}{P_{noise}} \tag{2.2}$$

A SNR greater than 1:1, implies that the power of the received signal is higher than the power of the noise. A higher positive SNR therefore indicates a better reception of a signal while a SNR of less than 0dB, a ratio lower than 1:1, makes the noise overrepresented [15]. It is also the case that a larger SNR means a lower probability of error, and thus a signal with more power is more likely to be received and decoded correctly [26].

2.4.2 Received signal strength indication

Received Signal Strength Indication (RSSI) is the total signal power received in milliwatts. RSSI is measured in the decibel-milliwatts (dBm). The received RSSI also include the interference from other sources then the part sending. RSSI is often represented in negative dBm, which means that a value closer to 0 indicate a better signal. A RSSI of -60 dBm can be considered rather good while -100dBm is considered less good [28].

The companion document [11] also specifies how the receive sensitivity diversify between the data rates. The receive sensitivity can be found in Table 2.2.

Data rate	Receive sensitivity
DR0	-136dBm
DR1	-133dBm
DR2	-132dBm
DR3	-129dBm
DR4	-126dBm
DR5	-123dBm
DR6	-120dBm
DR7	-108dBm

Table 2.2: This table state the receive sensitivity for each data rate.

2.5 Related work

This section will shed some light on the related work. Not much academic literature is presented regarding LoRaWAN specifically, but the literature of acceptable quality is presented in this chapter.

2.5.1 LoRaWAN - Simulations, calculations and general discussion

Vangelista et al.[35] studied LoRaWAN performance with the help of statistical models and the usage of the simulation tool ns-3[8]. Their simulations show that a LoRaWAN can scale up to the order of 10^4 devices with a packet success rate of 95% given that care is taken when selecting traffic model and when increasing coverage area with multiple gateways. Simulations show that the addition of multiple gateways significantly helps improve the reliability of uplink traffic. The authors also mention that a "densification" of gateways may result in more devices opting to use the same spreading factor(s) which in turn may result in a higher probability of collisions (and thus a lower success rate), unless for example the Adaptive Data Rate mechanism in LoRaWAN mitigates this problem.

Bankov et al.[6] also analysed the performance and network capacity of LoRaWAN through simulations with 100-5000 devices, using the three main prescribed channels and data rates 0-5, both the Packet Error Rate (PER) and the Packet Loss Rate (PLE) was estimated. They also explain the problems like the duty cycle limitations and the recommended behavior for re-transmissions, a random delay between 1 and 3 seconds should be selected. The logic being that the duration of the transmission of an acknowledgement packet can be more than 1 second, and thus the authors claim that the probability for a repeated collision is high. The authors also drew the conclusion that one solution to these problems is to increase the density of gateways within the network to help offload the otherwise very busy gateways. A conclusion backed by that the amount of devices connected to a gateway is clearly correlated to the amount of collisions and thus the number of re-transmissions.

Adelantado et al.[1] has tried to bring understanding of the limitations of LoRaWAN. By making a simulation similar to that of Bankov et al. [6] performed by using 250-5000 devices, utilising the 3 main channels and data rate 0-5. The authors draw the conclusion that LoRaWAN faces several problems with the actual capacity of large-scale deployments. The largest problem the authors brought up being the duty-cycle regulations in the ISM bands, which limits the network capacity and scalability. Since LoRaWAN is connectionless and reliability is achieved by sending acknowledgements from the gateway, the duty cycle limitation becomes a limiting factor for downlink traffic (and acknowledgements) since the off-period time also applies to the gateway. The authors point out that this quickly becomes a problem

[8]https://www.nsnam.org/

as the network scaling. According to their simulations, the collision rate increases rapidly as the number of devices increases. The results for the maximum throughput per device (measured in packets/hour), clearly indicates this. The same correlation can be found between the amount of payload sent by each device. Sending larger payloads entails more time on air and thus increases the probability of packets colliding with each other during transmission. The problem of deploying ultra-reliable services on top of LoRaWAN is also pointed out by the authors. A real time system in general needs low latency and bounded jitter, which are two variables that can be very hard to forecast in LoRaWAN. The authors also mention that the ALOHA-based access is not optimal for deterministic traffic. An idea presented by the authors is that of implementing a hybrid of Time Division Multiple Access (TDMA) on top of LoRaWAN, which could lead to the technology becoming more suitable for more use cases.

Mikhaylov et al. [19] provides a mathematical model for calculating the time on air t_{ToA} for transmitting data in a LoRaWAN network. This model is then used to calculate performance values for different LoRaWAN data rates. Table 2.3 summarises some interesting values taken from Table II and III in Mikhaylov et al. [19], where the time on air t_{ToA} for maximum and minimum transmission units are presented (for each Data Rate), as well as the physical layer throughput. Last, the maximum inferred duty cycle is presented under the assumption that a LoRaWAN end-device must wait for $ACK\_TIMEOUT$ seconds before being able to transmit data again. Further calculations highlights the limitations of LoRaWAN imposed by the ETSI regulations (and the duty cycles), and the theoretical evaluation of LoRaWAN devices communicating under Europe-regulations in the 863-870MHz frequencies is shown to not exceed $2kbit/s$ per channel when accounting for duty cycle limitations. The authors also present calculations for how EDs should be distributed in relation to access points or gateways for optimal scalability (under the assumption that end-devices utilise pure ALOHA access). Equation (2.3) presents the equations used for calculating time on air for packets in a LoRaWAN using either the LoRa modulation or GFSK modulation.

$$\begin{aligned}
&\textbf{LoRa Modulation:}\\
&t_{ToA} = \frac{2^{SF}}{BW}\left((NP+4.25)+\left(SW+max\left(\left\lceil\frac{8PL-4SF+28+16CRC-20IH}{4(SF-2DE)}\right\rceil(CR+4),0\right)\right)\right)\\
&\textbf{GFSK Modulation:}\\
&t_{ToA} = \frac{8}{DR}(NP+SW+PL+2CRC)\\
&\textbf{Where:}\\
&NP = \begin{cases}8, & \text{if LoRa}\\ 5, & \text{if GFSK}\end{cases}\\
&SW = \begin{cases}8, & \text{if LoRa}\\ 3, & \text{if GFSK}\end{cases}\\
&CRC = \begin{cases}1, & \text{if uplink packet}\\ 0, & \text{if downlink packet}\end{cases}\\
&IH = 0\\
&DR = 50kbit/s\\
&SF \in \{7,8,9,10,11,12\}\\
&PL \in \mathbb{N}, \text{where PL is the PHY\_Payload bytes}\\
&BW \in 125kHz, 250kHz, \text{ where BW is the bandwidth}\\
&DE: \text{ Indicates use of data rate optimisation}\\
&CR: \text{ Indicates the Coding Rate}
\end{aligned} \tag{2.3}$$

Data Rate	Shortest Uplink Frame $t_{ToA}[s]$	Longest Uplink Frame $t_{ToA}[s]$	PHY Throughput $[b/s]$	Max Duty Cycle [%]
DR0	1.155	2.793	183.3	56.4
DR1	0.578	1.561	328.1	41.9
DR2	0.289	0.698	733.1	24.4
DR3	0.144	0.677	1512.9	23.8
DR4	0.082	0.707	2885.1	24.6
DR5	0.041	0.400	5104.9	15.6
DR6	0.021	0.200	10 209.8	8.5
DR7	0.0035	0.0424	48 113.2	2.1

Table 2.3: Performance calculations for LoRaWAN derived by Mihaylov et al. [19].

Margelis et al. [18] looks into the evolution of some LoRaWAN competitors and major differences between them. The technologies compared were Sigfox[9], OnRamp[10], and LoRaWAN. Potential security vulnerabilities in the OTAA join procedure of LoRaWAN were highlighted, such as the usage of non-uniformly random nonces causing a degraded encryption outcome, as well as eavesdroppers potentially gaining information about network topology due to the join-request packets not being encrypted. They also highlight that there is no PHY-level CRC[11] on downlink packets in LoRaWAN (which is also correct as per LoRaWAN specification v1.0.2 [31]) which supposedly also makes downlink packets vulnerable to integrity attacks and bit corruption in general. It should be noted though that the MAC-layer packet still contains the MIC which still helps mitigate the threat of end-devices accepting false data on the application layer.

Varsier and Schwoerer [36] builds a detailed simulator in MATLAB to evaluate a smart-metering application scenario in Paris. The simulator accounts for both inter and intra spreading factor collisions as well as taking into consideration the signal propagation, shadowing, and fading, which makes for a rather sophisticated simulator. An interesting result is for example that 19 gateways would be needed to reach a QoS of 98%[12] when trying to cover an area of $17km^2$ for collecting power consumption data in their scenario. A significant assumption during these simulations is that there is no other external traffic on the utilised channels, which could mean that QoS may be much worse in a real scenario.

Mikhaylov et al. [20] discusses the pros and cons of device to device (d2d) communication in LoRaWAN and also propose, implement, and evaluate a network assisted d2d communications protocol, and show that the proposed d2d protocol can reduce time and energy for data transfers. However, the authors do not discuss the security implications of employing such a protocol in a LoRaWAN in any detail other than mentioning that such a d2d protocol may impose security and privacy concerns.

Reynders et al. [27] propose an algorithm for optimising the power and spreading factor for each device in a LoRaWAN cell while avoiding what they call the "near-far problem". The algorithm thus helps mitigate the near-far problem while decreasing the packet error rate in a cell. The algorithm is validated using the simulation tool ns-3 (the same tool used by

[9]Sigfox - https://www.sigfox.com/en

[10]OnRamp Wireless - Turned into Ingenu: https://www.ingenu.com/2015/09/ingenu-launches-from-on-ramp-wireless-m2miot-technology-and-networks/

[11]Cyclic Redundancy Check (CRC) - An error detection code used for detecting corrupt or changed data

[12]The QoS value is in their work an average value over the considered period and corresponds to the total number of demodulated packets compared to the total number of emitted packets. It is close to that of an average of the Packet Success Rates for all devices.

Vangelista et al.[35]) and is shown through simulations that edge devices in a network can have their packet error rate reduced by up to 50% for certain scenarios.

Li et al. [16] provides a flexible way of modeling packet collisions in LPWAN technologies such as Sigfox and LoRaWAN. The model considers both time and frequency overlap and is based on stochastic geometry. One interesting thing about their findings in evaluating and applying their model is that cell edge devices seem to bottleneck the networks performance.

2.5.2 LoRaWAN - Security

Naoui et al. [22] propose a reputation-based encryption key management solution to enhance the security of the LoRaWAN architecture, mainly to derive new session-based keys for network level encryption (NwkSKey). The solution is argued to provide authentication, integrity and confidentiality on network security level, however the schemes impact on the overall network performance is not discussed, nor is any other practical evaluation done.

Zulian [40] investigates the OTAA join procedure and shows that there are situations where the generation of DevNonce can a) cause a join request being dropped due to the DevNonce already having been used or b) switch off the end-device for the same reason, due to policy, and further shows how the probability of these events occurring as a function of the number of devices N_D in the network and the number of connection attempts K.

Tomasin et al.[33] bring up some weaknesses in the OTAA procedure. To mitigate replay attacks during the join procedure the server checks if the same DevNonce has been sent from the end-device before, meaning that the server has to store all the previously sent DevNonce's. According to the authors the probability of generating a previously used DevNonce is 11%, which will make the join procedure to fail. The authors also point out that it is hard to implement a random number generator for these type of devices, that result often is pseudo random rather than true randomness. They also point out a weakness in the SX1272[13] transceiver where the random number generator uses the least significant bit from the receiver signal strength indicator (RSSI). This make it possible for an attacker to narrow down the amount of possible solutions.

In Toussaint et al.[34] the performance of a OTAA procedure based on the markov chain model is evaluated. More precise, a simulator is used to compute the expected delay and energy consumption for the procedure. The energy consumption is based on the SX1272 transceiver from Semtech. According to their result the channel quality has a large impact on the delay during the join procedure.

2.5.3 LoRaWAN - signal characteristics

A range test experiment is to be found in Aref et al. [4]. This rigorous range test presents the packet loss, RSSI, SNR and number of received packets for various sizes of payloads at distances ranging from 200 meters to 8500 meters. The test utilises a gateway at a fixed position where an end-device is moved around to perform the measurements at varying distances. The tests presented in Aref et al. [4] were not clear line of sight tests since buildings and other obstacles are in between the transmitter and the receiver. The tests were conducted with payload sizes of 10, 50 and 100 bytes, using the LoRa spreading factor 10 (SF10), and a bandwidth of 250KHz. For the test case where the payload was 10 bytes, invalid packets

[13]Semtech SX1272 - http://www.semtech.com/wireless-rf/rf-transceivers/sx1272/

were encountered above 8,1km. When increasing the payload to 50 bytes invalid packets were experienced at 2.3km.

The full result of the test case with payload of 50 bytes is presented in Table 2.4. It should be noted that spreading factor 10 combined with the bandwidth 250kHz is not part of the defined data rates that can be found in Table 2.1 (which is the specification for the data rates used in Europe). Neither is it part of the set of combinations of spreading factors and bandwidths presented in the LoRaWAN specification [31] or the LoRaWAN regional parameters for Europe [11]. In fact, this combination of spreading factor and bandwidth can not be found for any region, although the test was carried out in Germany at the Offenburg University of Applied Sciences.

Wendt et al. [39] performs a similar long-distance line of sight test as Aref et al. [4] where what is measured and presented is the packet success ratio, the number of sent packets, the number of lost packets, and the amount of transmission errors. What is interesting in this study is that the physical transmission environment is altered between different scenarios. In a scenario similar to [4], 81.58% of the packets were received correctly at a distance of 9.75km. This test was carried out using the data rate DR0 but the size of the transmission unit is unspecified.

A similar test can be found in Petäjäjärvi et al. [24] where they managed to receive a packet success rate of 62% at a distance of 15-30km with line of sight. A minor issue with the works of Aref et al. [4] and Petäjäjärvi et al. [24] is that the size of the transmissions unit is not clearly defined, making it hard to compare their results to each other as well as with other works such as Aref et al. [4].

Distance (km)	Invalid packets	RSSI μ (average)	RSSI σ (standard deviation)	SNR μ (average)
0.276	0%	-76.8dBm	0.44	>0
0.580	0%	-81.9dBm	0.30	>0
0.959	0%	-100.1dBm	2.18	>0
1.346	0%	-100.4dBm	1.65	>0
2.302	9.8%	-115.7dBm	5.56	-3.2
3.575	0.5%	-124.9dBm	1.58	-10.9
5.031	26.3%	-127.6dBm	2.42	-13.4
6.056	13.1%	-127.9dBm	1.22	-13.9
6.667	5.9%	-126.2dBm	2.40	-12.2
7.482	19.7%	-129.8dBm	3.77	-15.8
8.149	94%	-130.7dBm	1.10	-16.7
8.519	n/a	<-150dBm	n/a	n/a

Table 2.4: The table concluding the test case with a payload of 50 bytes from Aref et al. [4].

Petrić et al. [25] also performs a variety of range measurements. In one of the tests, they place a gateway in a TV tower and measure packet error rate, RSSI, and SNR. The end-device is placed at six different locations A, B, C, D, E, and F all situated circularly around the gateway at a radius of 3km. The physical environment between the end-device and the gateway was different for each location, and thus each location represents a slightly different environment ranging from urban to rural.

During the experiment, an end-device transmitted a packet containing 25 bytes of payload data utilising the spreading factors 7, 9 and 10 as well as a bandwidth of 125kHz for each location. Each spreading factor at each location was tested by sending 500 packets. The results of their tests are presented in Table 2.5 (for details, please visit [25]. Note that a recurring result is that the higher the spreading factor was the lower the number of invalid packets, the higher the RSSI, and the higher the SNR, independent of location.

These tests show how different types of obstacles can influence the performance of a network. The authors also point out that it is hard to find a correlation between decreasing RSSI and decreasing number of valid packets. For the measurements with spreading factor 10 the rate of invalid packets increase when the RSSI become higher, but the same test indicates that a higher SNR decreases the risk of packet error. For the other tests, the tests with spreading factor 9 and 7 shows that higher SNR implies a better signal and thus less packet errors.

Location	Spreading factor	Invalid packets	RSSI μ (average)	RSSI σ (standard deviation)	SNR μ (average)
A	SF10	44%	-106.5dBm	4.74	5.17
A	SF9	51%	-113.7dBm	3.71	-6.08
A	SF7	90%	-117.8dBm	1.42	-7.08
B	SF10	3%	-100.5dBm	3.64	6.47
B	SF7	47%	-107.8dBm	3.93	2.86
C	SF10	53%	-106.2dBm	5.89	5.10
C	SF9	35%	-109.7dBm	3.55	-0.72
C	SF7	34%	-111.6dBm	3.33	0
D	SF10	44%	-109.9dBm	4.15	3.71
D	SF9	87%	-116.1dBm	2.82	-9.33
D	SF7	100%	n/a	n/a	n/a
E	SF10	42%	-109.5dBm	4.68	4.02
E	SF9	43%	-113.5dBm	3.74	-4.85
E	SF7	57%	-115.7dBm	2.48	-4.93
F	SF10	34%	-107.7dBm	4.33	5.03
F	SF10	3%	-106.3dBm	4.82	5.46
F	SF9	22%	-110.9dBm	4.72	-2.40
F	SF7	49%	-114.0dBm	2.80	-2.13
G	SF10	35%	-106.8dBm	5.55	4.73
G	SF9	49%	-114.3dBm	3.65	-6.08
G	SF7	42%	-114.0dBm	2.73	-2.51

Table 2.5: The table concluding the results from Petrić et al. [25].

Neumann et al. [23] provides a case study where a small LoRaWAN was deployed indoor. Measurements were done at various indoor locations where packet loss ratio, RSSI, SNR and power consumption was monitored for uplink traffic going from an end-device to a gateway.

2.5.4 LoRa collision rate

An estimation of the collision rate in LoRa was done by Augustin et al.[5]. They define a collision in LoRa to occur when two packets were sent during an overlapping in time using the same frequency and the same spreading factor.

To estimate the collision rate a simulator was built with a poisson process for generating packets and assigns the packet size according to a uniform distribution over 51 bytes. If a collision occurs between two packets, none of the packets are said to reach the gateway. The simulator uses spreading factor 7 and with bandwidth of 125KHz (corresponding to data rate 6).

The simulations show that LoRa capacity usage peaks as channel load is about 0.5 (where channel load is defined as the average amount of devices trying to send data), and declines as channel load increases. At peak capacity usage, the collision rate is as above 0.6 in their simulations. The authors tries to shed some light on the CSMA mechanism to counter the problem with channel load. In this case it should be the collision avoidance in CSMA that they argue LoRa could benefit from in order to minimise the number of re-transmissions needed. An issue with collision avoidance (CSMA/CA) solution that should be noted is the increase in energy consumption for all devices in the network. Further, the gateway then needs to implement the CSMA mechanism since hidden terminal problem[14] may be present in the network, and thus the end-devices can not perform proper "carrier sense" on the channel to see if the channel is busy. When the CA-part is implemented on gateways to mitigate the hidden terminal problem, the gateway would need to send a "clear to send" message every time an end-device is about to send data. Not only does this further introduce higher energy consumption for devices (due to them having to listen for the clear-to-send), it may also become a problem due to the LoRa duty cycle limitations affecting the gateway.

2.5.5 Interference measurement in WiFi-networks

Not much work is to be found when it comes to measuring interference in LoRa networks however a study of interference in WiFi-networks is done by Mahanti et al.[17] and shows that during certain times of the day, almost 80% of the Wi-Fi channels were occupied by "unintentional interference". This "unintentional interference" comes from several different devices not communicating over Wi-Fi, e.g. microwave ovens, analog cordless phones and bluetooth headsets. The authors also investigate how an "intentional" source of interference, such as a jammer, influences the network. The jammer used can create a power level of 1mW to 30mW and thus ensure that the RF medium is never clear, preventing devices from using the jammed channel(s). In order to measure the interference in this paper, the authors utilise an off the shelf spectrum analyser.

[14]The hidden terminal problem is explained in detail in the book "A top down approach" [15]

3 Method

This chapter describes the methodology that was used for measuring the average throughput of different data rates in a LoRaWAN, as well as the packet loss rate for LoRaWAN devices sending data under high load conditions. While it is tempting to copy the methodology used by Mahanti et al.[17], a stressing issue (apart from the differences between a Wi-Fi network and a LoRaWAN) is the price and availability of a spectrum analyser which is a key component of their experiments. This means that replicating their methodology for a LoRaWAN is troublesome, and a different approach is needed.

This chapter thus covers the following: first a brief overview of the experimental environment and the tests to be conducted, followed by detailed sub-chapters covering the experimental setups in detail, how to estimate the average throughput for each data rate, and lastly how to simulate high load conditions and measure loss rate in it.

Table 3.1 lists all variables used in the original research of this thesis in the order they are introduced.

Variable:	**Definition:**
$MaTU$	Maximum Transmission Unit
$MiTU$	Minimum Transmission Unit
p_{sent}	The number of sent packets
p_{recv}	The number of received packets
p_{loss}	The packet loss ratio between p_{sent} and p_{recv}
SNR	Ratio between the power of the signal P_{signal} and the measured noise P_{noise}
$\mu_y(x)$	Average throughput in bits/s sending x bits over data rate y
t_{off}^{CH}	Time to wait for send on channel CH again
$t_y^{ToA}(x)$	Average time on air sending x bits over data rate y
t_{other}	The amount of extra time imposed by the send-function
$t_y^{exec}(x)$	Execution time for the send function sending x bits over data rate y
$T_y^{max}(x)$	Ratio between time $t_y^{ToA}(x)$ and $t_y^{exec}(x)$ sending x bits over data rate y

Table 3.1: Definition of the variables used in the thesis.

3.1 Setup

Figure 3.1[1] shows the setup for measuring the traffic overhead which contains several components.

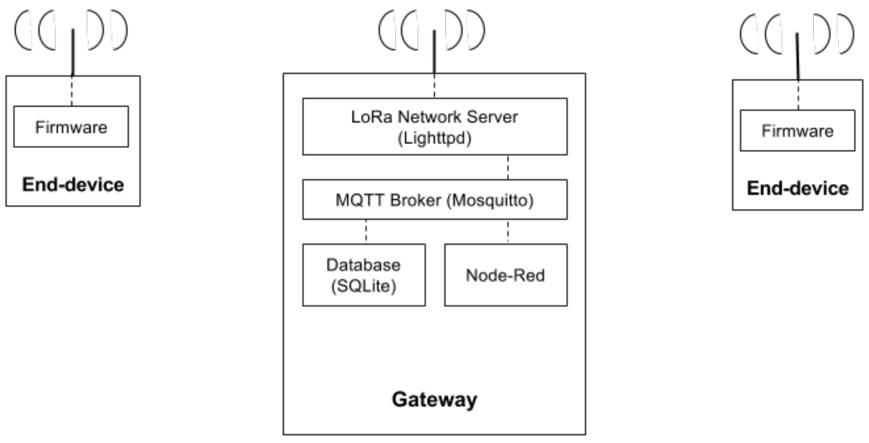


Figure 3.1: Overview of the setup.

The end-device is a programmable, low-power RF module used for LoRaWAN communication. The Gateway utilises the wireless LoRaWAN technique to communicate with the end-device. The Gateway has several interfaces, so it can communicate utilising several different communication techniques. The end-devices ran simple applications to send and receive data. The devices were positioned 1.5 meters from the gateway.

The Gateway has a more complex structure since it provides several different network interfaces and a user interface. The main purpose of the gateway to translate the traffic from LoRa and forward the traffic to a network server, but in this setup a simple network server also runs on the same device as the gateway.

3.1.1 End-devices

The experiments conducted feature a LoRaWAN module: the microcontroller xDot from Multitech [2]. The xDot card can be configured through a Digital I/O channel. It is possible to write C/C++ applications for the xDot card using the library libmxDot-mbed5 [3] and the mbed-os[4]. When using these libraries, the compiled application (a .bin-file) is loaded and run on the xDot card. By developing an application for the xDot card it is possible to configure the xDot cards (end-devices) through it and control its behavior.

Specifications of the end-devices

The specification for the xDot card is listed below:

CPU:	ARM Cortex-M3 (ST32L151CCU6)
Max Clock:	32 MHz (configurable to power use)
Flash Memory:	256 KB
RAM:	32 KB

[1]Internal structure of the Gateway - http://www.multitech.net/developer/software/lora/lora-network-server/

[2]Multitech xDot Micro Developer Kit MTMDK-XDOT-EU1-A00 - http://www.multitech.com/models/94558024LF

[3]MultiTech libxDot-mbed5 - https://developer.mbed.org/teams/MultiTech/code/libxDot-mbed5/

[4]ARMmbed-os - https://github.com/armmbed/mbed-os

3.1.2 Gateway

The Gateway used for setting up the LoRaWAN is the Multitech MultiConnect Conduit MTCDT-LEU1-210A-EU-GB[5]. The gateway runs the operating system YOCTO 1.6 (based on the Linux 3.12 kernel), and comes with the packages SQLite, Ligttpd web server, Mosquitto MQTT broker, Node-Red, and the BusyBox core utilities.

The gateway was connected to a MultiConnect LoRa mCard[6] which is a LoRa transceiver, giving the conduit a LoRa interface allowing it to run as a LoRaWAN gateway. Along with the LoRa mCard the gateway also features an RJ-45 socket for ethernet connection for configuration, user interface and terminal access. The gateway also permits full root console access via SSH.

LoRa network server

The LoRa network server can be described as a network server receiving and transmitting LoRa-messages passed up from the gateway or down to it. The network server then distributes incoming data to the targeted application server. In these experiments both the network and application server is running on top of the gateway. This differs slightly from the architecture described in Section 2.3.2 (Theory - LoRaWAN). The network server uses the MQTT [7, 13] protocol to communicate with the application server (using JSON[7] data formatting), so therefore the gateway too runs an internal MQTT Broker. The reason that the gateway uses MQTT/JSON is that the user interface Node Red, uses them to display information. More about the network server can be found in Section 2.3.2.

Node red

Node Red is a graphical interface (or programming tool) for wiring together hardware devices, APIs and online services. It is built on Node.js and allows the user to write JavaScript functions to help manipulate, present, and/or generally handle data. It provides a browser-based editor that makes it easy to wire together flows using a wide range of devices and functions that comes with it.

Specification of the gateway

The specification for the gateway is listed below:

CPU:	ARM9 processor with 32-Bit ARM
Max Clock:	400MHz
Flash Memory:	256MB
RAM:	128X16M DDR RAM
Operating System:	Linux 3.12 Kernel, Yocto 1.6
Database:	SQLite
Core Utilities:	BusyBox

[5]Multitech MultiConnect Conduit MTCDT-LEU1-210A-EU-GB - http://www.multitech.com/models/94557209LF
[6]MultiConnect MTAC-LORA-868 94557296LF - http://www.multitech.com/models/94557296LF
[7]JavaScript Object Notation (JSON) - http://www.json.org/

3.2 Measuring the average throughput for LoRaWAN data rates

This section covers how to estimate the average throughput for sending a LoRaWAN packet. The mbed.os LoRaWAN library can be used to configure an xDot card to print debugging logs through its USB interface and thus expose them to external analysis. The debugging logs contain information of interest for this study, namely the Time-on-Air (ToA) for each TX-window and the corresponding signal stats: Received Signal Strength Indication (RSSI) and Signal to Noise Ratio (SNR).

3.2.1 How to measure average throughput

In order to measure the average throughput the average time on air $t_y^{ToA}(x)$ for sending x bytes of data with data rate y will be measured through the debugging logs. The time on air as a function of the amount of bytes sent, x, was then approximated as first and second order functions: $t_y^{ToA}(x) = k \cdot x + C$ and $t_y^{ToA}(x) = k_1 \cdot x^2 + k_2 \cdot x + C$ using Matlabs polyfit[8].

Once the functions $t_y^{ToA}(x)$ have been estimated, the average throughput $\mu_y(x)$ in kbit/s can be estimated as:

$$\mu_y(x) = \frac{x}{t_y^{ToA}(x)} \cdot 1000 \cdot 8 \ \text{[kbit/s]} \tag{3.1}$$

3.2.2 Measurement setup and application

This section describes the setup used for measuring the average throughput, as well as the application running on the end-devices.

For these measurements, only the Gateway/Network Server device and one xDot card is used. The Gateway only receives data during this test, while the xDot card acts as a data publisher device, sending data to the gateway. Pseudo code for the software running on the xDot card can be found in listing 3.1.

Listing 3.1: Pseudo code for measurementApp.cpp

```
#include mbed.h
/* Other includes */

/* Initializations */

//Set logging level, provides Time on Air for transmssions
mts::MTSLog::setLogLevel(mts::MTSLog::TRACE_LEVEL);

dot = mDot::getInstance(); //pointer to class providing internal functionality

/* Join Network */

dot->setTxDataRate(datarate); //configures node to use Date Rate "datarate" when
    transmitting

uint8_t bytes;
payload = generateData(bytes);     //generates data of 0-242 bytes
data = loadData();
dot->send(data);                   //sends 12-255 bytes to the gateway/network
    server
/* Debug printing - Time On Air for transmission of data */

return 0;
```

[8] Matlab polyfit - https://se.mathworks.com/help/matlab/ref/polyfit.html

3.2.3 Average throughput measurements - experimental details

The xDot card is configured (through the software running on it) to transmit data on frequency $868.3MHz$. The variables changed in these experiments is the Data Rate y, ranging from 0 to 7 and the amount of bytes sent to the gateway in each transmission, x. In order to retrieve the time on air (ToA) for each transmission, the xDot card has to be configured to provide trace-level logging. The xDot card is also configured to operate as a Class A device.

The data points x used as input for measuring the time on air and estimating the time on air functions $t_y^{ToA}(x)$ is presented in Table 3.2. Each data rate y imposes a specific Maximum Transmission Unit ($MaTU$) and Minimum Transmission Unit ($MiTU$), and thus 11 data points were chosen, spaced from $MiTU$ to $MaTU$. $MiTU$ for all MAC-Layer LoRaWAN packets is 12 bytes due to the MAC formatting, while $MaTU$ differs depending on the data rate y that is used.

Data rate y	Data points: $x_0, ..., x_{10}$ bytes
$DR0 - DR2$	12,18,23,28,33,38,43,48,53,58,64
$DR3$	12,25,37,49,61,73,85,97,109,121,128
$DR4 - DR7$	12,38,63,88,113,138,163,188,213,238,255

Table 3.2: Data points x, the amount of bytes sent to the gateway/network server, for each data rate y investigated.

In order to gain some reliability, the mean of 20 measurements for each data point is calculated and used.

3.3 Estimating the upper bound for how much time the xDot cards can be broadcasting data

The mbed.os LoRaWAN library can be used to configure the xDot cards to ignore the duty cycle limitations imposed by the LoRaWAN standard (for testing purposes).

While this removes the criteria of waiting t_{off}^{CH} seconds (as described in equation (2.1), section 2.3) before transmitting data on the same channel again, the send function provided by the mbed.os LoRaWAN library still has to wait for two RX-windows to time out. This behavior imposes an upper bound for how often an xDot card can be programmed to send data to the gateway/server.

A similar approach to that used for measuring the average throughput is adapted in an attempt to measure the amount of "extra time" t_{other} imposed by the send-function in the xDot cards. The execution time t_{exec} for the send-function should thus consist of the time $t_y^{ToA}(x)$ (where x is the amount of bytes sent) spent broadcasting data, time spent listening for data in $RX1$ and $RX2$, as well as executing other internal events in the micro controller. The execution time $t_y^{exec}(x)$ for the send function is thus assumed to have the following relationship:

$$t_y^{exec}(x) = t_y^{ToA}(x) + t_{other} \iff t_{other} = t_y^{exec}(x) - t_y^{ToA}(x) \quad \text{[s]} \tag{3.2}$$

In order to retrieve an estimation of t_{other}, t_{exec} is measured for each data rate y sending the same amounts of data as presented in Table 3.2. Once measurements for each data point x and data rate y have been done, t_{other} is approximated as:

$$t_{other} \approx \frac{1}{8 \cdot 11} \sum_{y=0}^{7} \sum_{n=0}^{10} \left(t_y^{exec}(x_n) - t_y^{ToA}(x_n) \right) \quad [\mathrm{s}] \tag{3.3}$$

The ratio $T_y^{max}(x)$ between time $t_y^{ToA}(x)$ spent transmitting data and the execution time $t_y^{exec}(x)$ of the send-function can be seen as an upper bound of total time the application can send data. Thus, despite being able to turn off duty cycle limitations, an xDot card cannot spend more time transmitting than $100 \cdot T_y^{max}(x)\%$ per hour. Equation (3.4) shows how $T_y^{max}(x)$ is calculated:

$$T_y^{max}(x) = \frac{t_y^{ToA}(x)}{t_y^{exec}(x)} = \frac{t_y^{ToA}(x)}{t_y^{ToA}(x) + t_{other}} \tag{3.4}$$

The mean of 20 measurements for each data point is calculated and used in order to gain some reliability.

3.3.1 Application - estimating upper bounds for broadcasting data

The application used for measuring t_{exec} is similar to the one presented for estimating the average throughput. The difference is that the application used in this experiment utilises a timer function in order to calculate the execution time for the send function (instead of logging the time-on-air). Pseudo code for the program *timeOfsend.cpp* running on the xDot card is presented in listing 3.2.

Listing 3.2: Pseudo code for timeOfsend.cpp

```
#include mbed.h
#include time.h
/* Other includes */
/* Initializations */
dot = mDot::getInstance(); //pointer to class providing internal functionality
/* Join Network */
dot->setTxDataRate(datarate); //configures node to use Date Rate "datarate" when
    transmitting
uint8_t bytes;
payload = generateData(bytes);     //generates data of 0-242 bytes
data = loadData();

Timer timer; //timer class providing us and ms granularity
uint32_t begin;
uint32_t end;

timer.start();                  //starts timer
begin = timer.read_ms();        //read timer value

dot->send(data);                //<----- Return time is more than just Time-on-
    Air

timer.stop();                   //stops timer
end = timer.read_ms();          //read timer value
timeOfSend = end-begin;         //calculate time t_execution of send function in
    ms
/* Debug printing - print return time of send function transmitting data */
return 0;
```

3.4 Creating high load conditions and measuring packet loss rate

This section describes how high load conditions in a LoRaWAN network were created using two xDot cards and a gateway, which also served as a LoRaWAN network server.

3.4.1 Creating high load conditions

To create a high load LoRaWAN network with only a few components, one xDot card is used as a *yeller* and one xDot card is used as a *talker*. The goal of the yeller is to create traffic on the network, simulating a network with many devices wanting to transmit data. The goal of the talker is to behave as a "regular device" in the network, wanting to transmit data to the network server every now and then. The yeller device is configured to broadcast data as often as it can (as per investigations conducted in section 3.3).

3.4.2 Measuring packet loss rate

In order to measure the packet loss rate, the talker device is configured to print its debug log through its serial USB interface. The debug log will contain internal events, as well as a counter for each packet sent to the network server, p_{sent}. The network server keeps its own log over all its internal events, including all packets received, p_{recv}, and their corresponding source (which is the LoRaWAN DevEUI, somewhat equivalent to that of a MAC-adress). If both the network server log and the talker device debug log is cleared before an experiment, then the packets sent from the talker device can be correlated to the packets received by the network server. The *packet loss ratio* p_{loss} can be calculated as:

$$p_{loss} = 1 - \frac{p_{recv}}{p_{sent}} \tag{3.5}$$

In the presence of additional traffic on the network generated by the yeller device, the packet loss ratio p_{loss} for the talker device should be affected especially when the talker and yeller use the same multiplexing and modulation techniques (or rather, the same data rates y) when transmitting data. If two devices use the same channel frequency but are configured to transmit data with non-matching data rates y, then the transmissions done by the yeller device should only be regarded as noise when the gateway is receiving data from the talker device.

3.4.3 Packet loss ratio measurements - experimental details

This section describes the different tests conducted to calculate the packet loss as well as the applications running on the two devices and the corresponding device configurations. Four different tests were conducted. The first one as a benchmark without any influence from the *yeller*. The other tests then alter the data rate y and payload size x to gauge the impact of the *yeller*. To gauge the impact, the Received Signal Strength Indication (RSSI) and the Signal-to-Noise Ratio (SNR) are measured at the talker and the gateway. The SNR is used to gauge whether the yeller device has increased the perceived noise on a channel, and the RSSI indicates whether the strength of the received signal has changed significantly. One could say that if the RSSI stays the same while the SNR decreases, the implication is that the decrease in SNR is caused by more noise on the channel. These measurements were mainly used because they were accessible through the network log on the gateway. The mean value μ and standard deviation σ was calculated for the RSSI and SNR.

Scenario 1 - the benchmark

During the first scenario the application on the talker was configured to send the maximum size of the transmission unit on each data rate. The transmission unit consists of random integers and the MAC-Layer LoRaWAN packets of 12 bytes. Even though the application was set to ignore the duty cycle, the talker was programmed to only send the data every 20 second. The yeller was not in use during this scenario. These tests were conducted in order to see the performance of a LoRaWAN device under (close to) ideal conditions.

This type of benchmark test was the closest thing that could be done with the available equipment to assess the (potential) influence of noise created by devices in the vicinity. There was (to the best of the authors knowledge) no way of using the equipment to measure the intensity of the noise on a channel without using a talking node, since SNR and RSSI can only be seen for packets received by the access point (or network server).

Scenario 2 - maximum transmission unit

In scenario 2, both the talker and the yeller were programmed to send the Maximum Transmission Unit ($MaTU$) on each data rate. Doing so maximises the probability of both devices being active on the channel simultaneously.

Scenario 3 - minimum transmission unit at talker

In scenario 3, the talker was programmed to send the Minimum Transmission Unit, which is only the MAC-Layer LoRaWAN fields (12 bytes) with no additional payload. The transmission unit for the yeller is still the Maximum Transmission Unit ($MaTU$) for each data rate.

Scenario 4 - different data rates

In scenario 4, the yeller and talker never used the same data rate, and the data rate for the yeller was set to DR0 for all test cases. The reason for the yeller utilising only DR0 for broadcasting data was the following: it is the data rate where it takes the longest time to transmit data, thus maximising the relationship between time on air and time spent waiting for RX timeouts. The talker sent the maximum transmission unit for each data rate, maximising the likelihood of transmissions overlapping.

Application - yeller and talker

The applications running on the two devices in the network were simple. They both had some initial code for setting internal variables and configuring the channel frequency and data rate used by the transceiver when transmitting. The devices also had a simple program-loop running which generated data, sent it to the network server, printed debugging information and logs, and then repeated the process. Pseudo code illustrations of the two C++-programs *yeller.cpp* and *talker.cpp* is found in Listings 3.3 and 3.4. Also note that both the talker and the yeller was programmed to ignore the duty cycle.

Listing 3.3: Pseudo code for yeller.cpp

```
#include mbed.h
/* Other includes */

/* Initializations */
mts::MTSLog::setLogLevel(mts::MTSLog::TRACE_LEVEL); //Set logging level, provides
     RSSI, SNR stats

dot = mDot::getInstance(); //pointer to class providing internal functionality
dot->setDisableDutyCycle(true) // now ignores internal dutycycle limitations

/* Join Network */

dot->setTxFrequency(frequency); //configures node to transmit on "frequency" Hz
dot->setTxDataRate(datarate); //configures node to use Date Rate "datarate" when
    transmitting

while(true){
        /* generate data of x Bytes */

        data = loadData();
        dot->send(data); //sends data of x Bytes to network server

        /* Debug printing - Packets sent, average RSSI, average SNR */
}
return 0;
```

Listing 3.4: Pseudo code for talker.cpp

```
#include mbed.h
/* Other includes */

/* Initializations */
mts::MTSLog::setLogLevel(mts::MTSLog::TRACE_LEVEL); //Set logging level, provides
     RSSI, SNR stats

dot = mDot::getInstance(); //pointer to class providing internal functionality
dot->setDisableDutyCycle(true) // now ignores internal dutycycle limitations

/* Join Network */

dot->setTxFrequency(frequency); //configures node to transmit on "frequency" Hz
dot->setTxDataRate(datarate); //configures node to use Date Rate "datarate" when
    transmitting

while(true){
        /* generate data of x Bytes */

        data = loadData();

        start = timer.start(); //Starts timer to clock time of send-function
        dot->send(data); //sends data of x Bytes to network server
        stop = timer.stop(); //Stops timer
        time = stop-start;

        /* Debug printing - Packets sent, average RSSI, average SNR */

        wait(20-TimeOnAir); //Begin transmission of data every 20 seconds

}
return 0;
```

4 Results

In this section the results of the test cases defined in Section 3 will be presented.

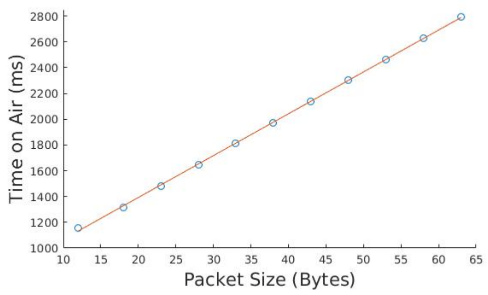
4.1 Throughput - measurements and estimations

This section aims to display measurements and calculations done to estimate the average throughput $\mu_y(x)$ for transmitting a packet in LoRaWAN. Appendix A shows the measurements for t_{ToA}, and Table 4.1 shows the coefficients of the estimated functions $t_y^{ToA}(x)$.

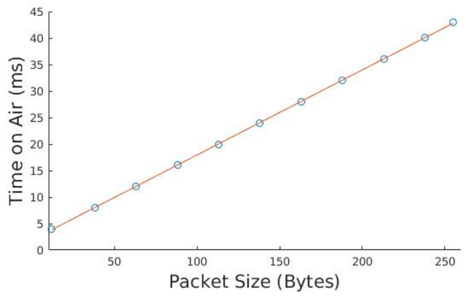
Function	**1st order approx.**			**2nd order approx.**			
$DRy : t_{ToA}(x)$	k	C	R^2	k_1	k_2	C	R^2
$DR0 : t_{ToA}(x)$	32.77	728.64	1.000	0.00	32.76	728.30	1.0000
$DR1 : t_{ToA}(x)$	18.30	363.93	0.9931	-0.03	20.70	325.97	0.9910
$DR2 : t_{ToA}(x)$	8.12	183.54	0.9998	-0.01	8.57	176.54	0.9998
$DR3 : t_{ToA}(x)$	4.62	91.77	0.9988	-0.00	4.95	83.22	0.9986
$DR4 : t_{ToA}(x)$	2.57	48.85	0.9998	-0.00	2.57	48.62	0.9997
$DR5 : t_{ToA}(x)$	1.47	23.82	0.9998	-0.00	1.48	23.40	0.9998
$DR6 : t_{ToA}(x)$	0.74	11.60	0.9997	-0.00	0.74	11.36	0.9996
$DR7 : t_{ToA}(x)$	0.16	1.88	1.0000	0.00	0.16	1.97	1.0000

Table 4.1: k, k_1, k_2 and C is listed for the different data rates for 1st and 2nd order approx. R^2 is the statistical coefficient of determination.

By examining the R^2 values in Table 4.1 it appears that a second order approximations of the function $t_y^{ToA}(x)$ gives no real benefit over a first order approximation. This is also evident from examining the k_1 values for the second order approximation which are either 0 or very close to 0. Figures 4.1a and 4.1b shows first order approximations of $t_0^{ToA}(x)$ and $t_7^{ToA}(x)$ plotted against the measured data points, and for all future calculations involving $t_y^{ToA}(x)$ the first order approximations presented in Table 4.1 are used.



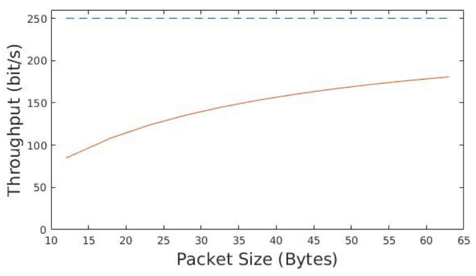
(a) Graph displaying time on air for DR0.



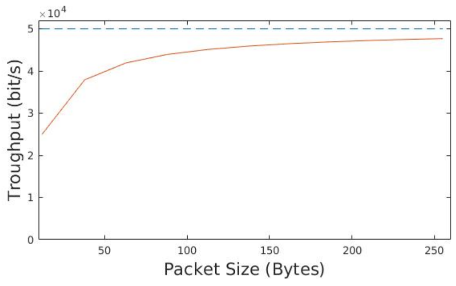
(b) Graph displaying time on air for DR7.

Figure 4.1: The estimated function for time on air for DR0 is displayed in graph (a) and for DR7 in graph (b). The measured points are displayed as plotted circles.

Figures 4.2a and 4.2b show the calculated average throughput for DR0 and DR7 together with the indicative max bit rates taken from the LoRaWAN specification. The maximum bit rate is visualised by a blue dashed line. The same tendency that is seen in Figures 4.2a and 4.2b is seen for all data rates. In particular, it is seen that more bytes sent causes a higher average throughput for the transmission. To make it easier to compare the average throughput for each data rate, Figure 4.3 illustrates this for the transmission units 12, 50, 100, and 200 bytes. Have in mind that the scale on the y-axis is $10 \cdot log_{10}(bit/s)$.



(a) The average throughput for DR0.



(b) The average throughput for DR7.

Figure 4.2: The average throughput for data rate 0 and 7 together with the maximum bit rate (visualised by a blue dashed line).

Figures 4.4a and 4.4b shows Mikhaylov et al. [19] model plotted against the functions $\mu_0(x)$ and $\mu_7(x)$. Note that the measured points (circles) are also predicted by the more complex model, something that the model in Mikhaylov et al. does better than μ_y for data rates $1-6$.

To summarise the results from this section, the time on air function $t_y^{ToA}(x)$ is seemingly linear since Table 4.1 shows that the x^2 constants k_1 are all very small (< 0.05) for the second order estimations. The linear behaviour is further reinforced by examining figures 4.1a-4.1b, suggesting that the sought function is on the form of $t_y^{ToA}(x) = k \cdot x + C$. Figure 4.4a shows the average throughput inferred by equation (2.3) together with $\mu_y(x)$. This figure shows that there is a "steplike" behaviour described by Mikhaylovs model that neither the first nor second order approximations of $t_y^{ToA}(x)$ capture.

4.2 Upper bound for transmitting data - timing the send function

This section presents the results from timing the send function and estimating the upper bound ratio $T_y^{max}(x)$ for sending data on the xDot cards, as described in Section 3.3. Ap-

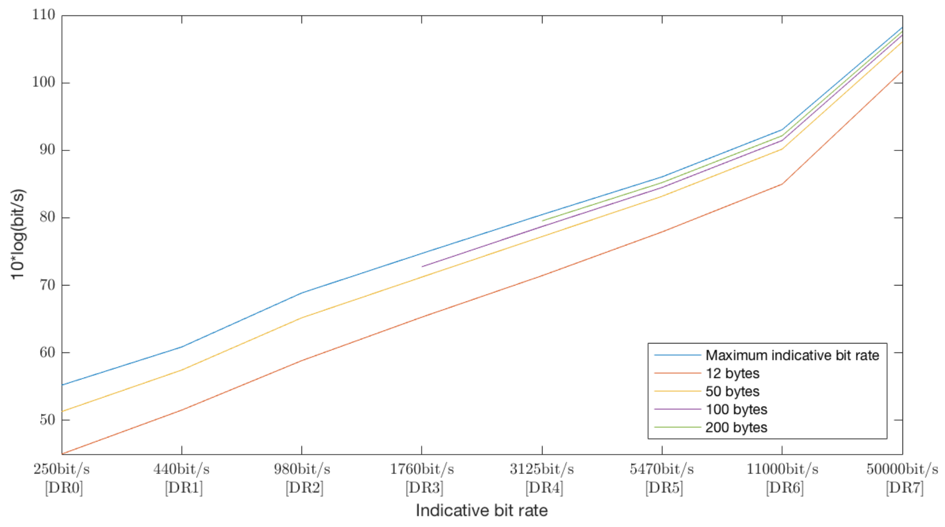
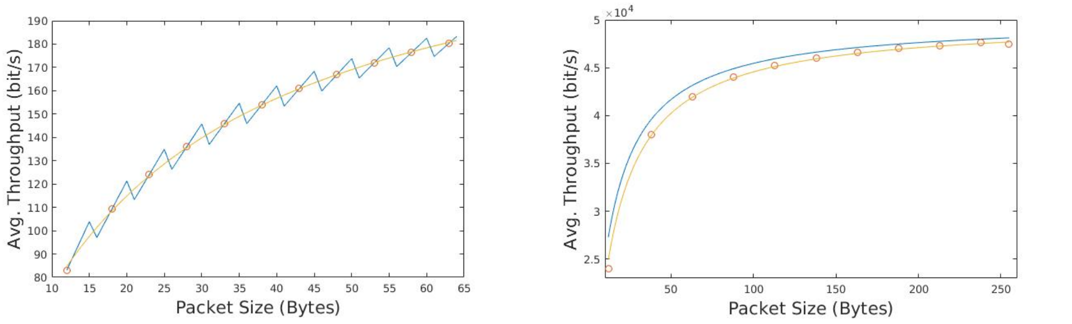


Figure 4.3: Comparison transmission unit of 12, 50,100 and 200 bytes for each data rate.



(a) The average throughput (blue) inferred from equation (2.3) describing t_{ToA}, plotted against $\mu_0(x)$ (orange) and the measured throughput (circles).

(b) The average throughput (blue) inferred from equation (2.3) describing t_{ToA}, plotted against $\mu_7(x)$ (orange) and the measured throughput (circles).

Figure 4.4: Average throughput estimated for data rates 0 and 7 by equation (2.3) describing t_{ToA}, and $\mu_y(x)$, as well as the measured throughput.

pendix B contains the measurements $t_y^{exec}(x)$ from timing the send function seen in listing 3.2, as can the values t_{other} calculated for all data rates y and data points x_n.

Equation (4.1) presents the mean value of t_{other}, which is in turn used to calculate $T_y^{max}(x)$ (as described in (3.4)) which is graphically presented in Figure 4.5. Table 4.2 shows $T_y^{max}(MaTU)$, the maximum transmission ratio for each data rate when transmitting the Maximum Transmission Unit ($MaTU$). To remind the reader: having $T_y^{max}(MaTU) = x\%$ means that a LoRaWAN Class A Device *cannot* occupy a channel more than $x\%$ despite deactivating the duty cycle regulations.

$$t_{other} \approx \frac{1}{8 \cdot 11} \sum_{y=0}^{7} \sum_{n=0}^{10} \left(t_y^{exec}(x_n) - t_y^{ToA}(x_n) \right) \approx 2359ms \tag{4.1}$$

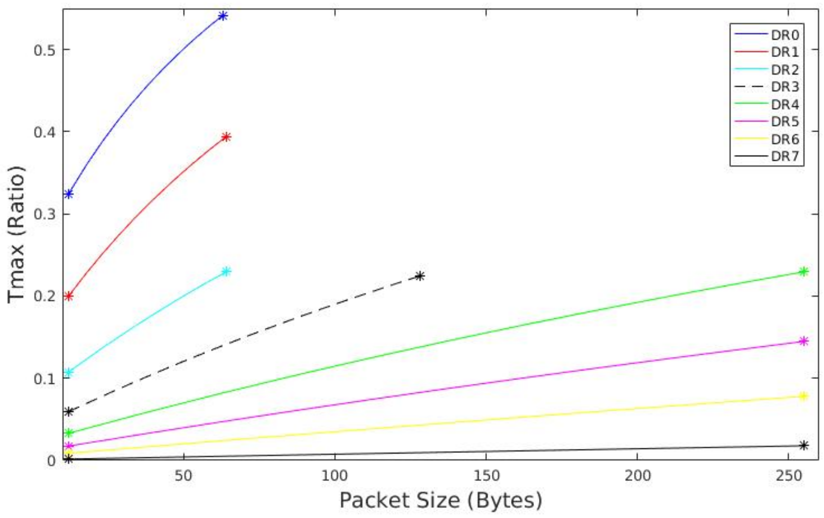


Figure 4.5: The maximum transmission ratio presented for each data rate (DR). The first order approximation of $T_y^{max}(x)$ has been used for calculating these values.

DRy	DR0	DR1	DR2	DR3	DR4	DR5	DR6	DR7
1st order approx. $T_y^{max}(MaTU)$	0.54	0.39	0.23	0.22	0.23	0.14	0.8	0.2

Table 4.2: The maximum transmission ratio for each data rate when transmitting the Maximum Transmission Unit ($MaTU$).

4.3 Creating high load conditions and measuring packet loss rate

4.3.1 Result from scenario 1 - the benchmark

Table 4.3 shows how many packets that were sent, received, and the conditions during these tests. These are the benchmark values when the system is working without the influence of the *yeller*. Figures 4.6 and 4.7 illustrates the results from Test case 1.0 to Test case 1.7. The figures also display the average μ and the standard deviation σ for SNR and RSSI at the gateway.

Test case			**Packets**			**RSSI (dBm)**	**SNR (dB)**
Test	Talker	Yeller	P_{sent}	P_{recv}	P_{loss}	$\mu(\sigma)$	$\mu(\sigma)$
T:1.0	64bytes (DR0)	-	100	100	0	-36 (1.62)	9.52 (0.80)
T:1.1	64bytes (DR1)	-	100	100	0	-37 (2.29)	10.08 (0.80)
T:1.2	64bytes (DR2)	-	100	100	0	-34 (1.84)	10.40 (1.05)
T:1.3	128bytes (DR3)	-	100	100	0	-43 (2.27)	12.18 (0.80)
T:1.4	255bytes (DR4)	-	100	100	0	-44 (3.89)	11.38 (0.64)
T:1.5	255bytes (DR5)	-	100	100	0	-41 (3.25)	10.00 (0.67)
T:1.6	255bytes (DR6)	-	100	100	0	-47 (3.06)	9.89 (0.39)
T:1.7	255bytes (DR7)	-	100	0	1	n/a	n/a

Table 4.3: The results of the benchmark test. The test is performed with the maximum transmission unit on each data rate.

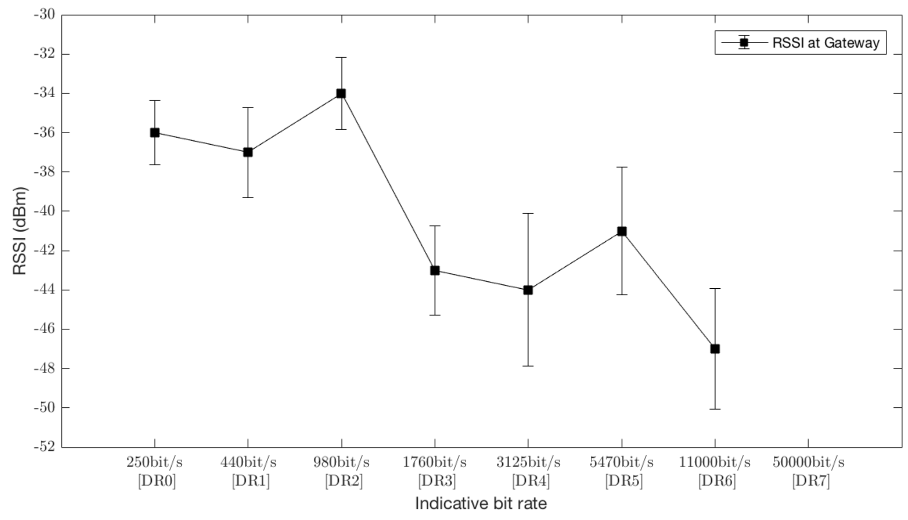


Figure 4.6: The average RSSI for each data rate(DR) measured at the gateway during scenario 1, the benchmark. The error bars indicate the standard deviation σ.

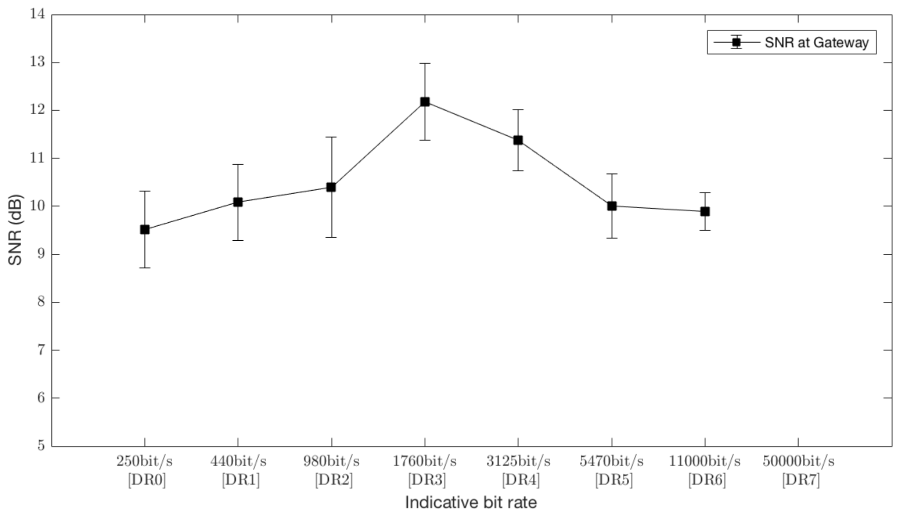


Figure 4.7: The average SNR for each data rate(DR) measured at the gateway during scenario 1, the benchmark. The error bars indicate the standard deviation σ.

4.3.2 Result from scenario 2 - maximum transmission unit

The results from Test case 2.0 to Test case 2.7 can be found in Table 4.4. In short, the tests were carried out with the maximum transmission unit on each data rate under the influence of the yeller. The yeller also sent the maximum transmission unit for each data rate.

Test case			**Packets**			**RSSI (dBm)**	**SNR (dB)**
Test	Talker	Yeller	P_{sent}	P_{recv}	P_{loss}	$\mu(\sigma)$	$\mu(\sigma)$
T:2.0	64bytes (DR0)	64bytes (DR0)	100	8	0.92	-37 (2.85)	7.44 (3.47)
T:2.1	64bytes (DR1)	64bytes (DR1)	100	33	0.67	-37 (2.39)	8.10 (3.33)
T:2.2	64bytes (DR2)	64bytes (DR2)	100	42	0.58	-37 (2.78)	9.78 (0.99)
T:2.3	128bytes (DR3)	128bytes (DR3)	100	59	0.41	-46 (3.03)	11.70 (1.63)
T:2.4	255bytes (DR4)	255bytes (DR4)	100	41	0.59	-39 (2.42)	10.54 (3.08)
T:2.5	255bytes (DR5)	255bytes (DR5)	100	78	0.22	-38 (2.19)	9.01 (2.53)
T:2.6	255bytes (DR6)	255bytes (DR6)	100	79	0.21	-46 (1.94)	10.07 (0.42)
T:2.7	255bytes (DR7)	255bytes (DR7)	100	0	1	n/a	n/a

Table 4.4: The results of the test with maximum transmission unit under influence of the yeller.

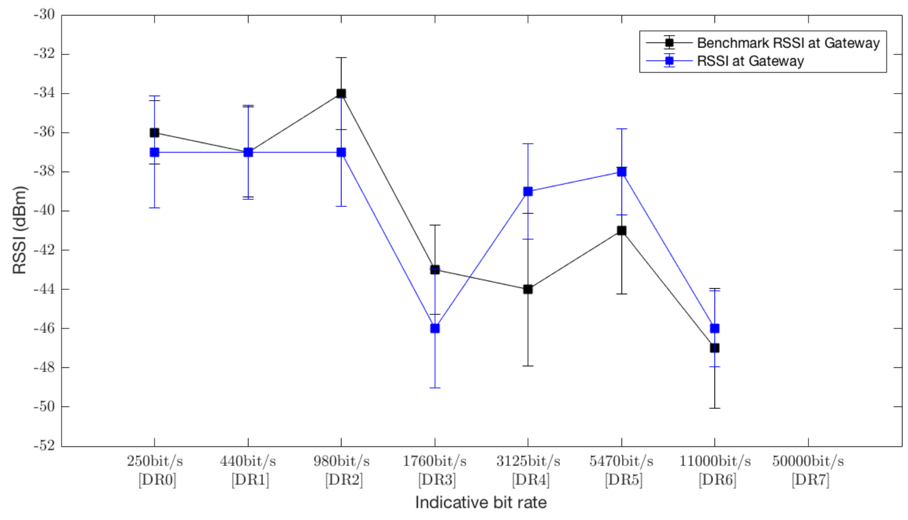


Figure 4.8: The average RSSI for each data rate (DR) measured at the gateway during scenario 2, utilising the maximum transmission unit. The error bars indicate the standard deviation σ.

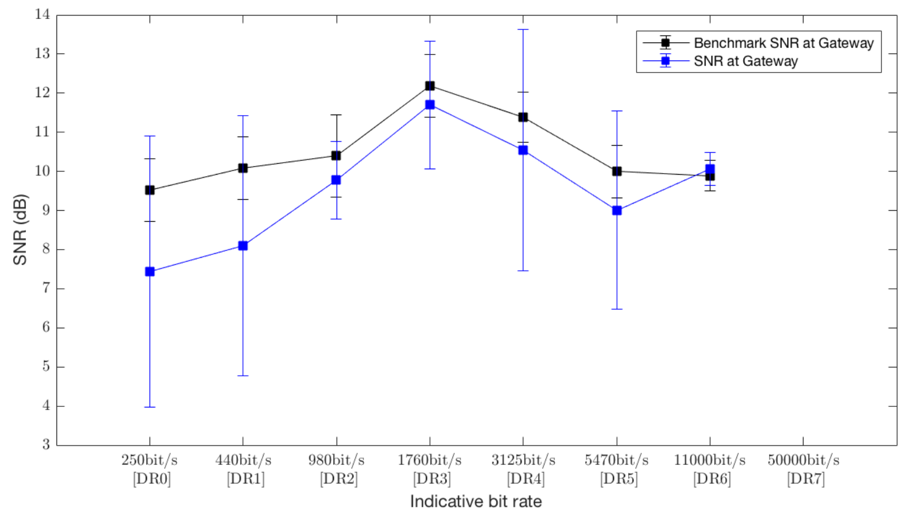


Figure 4.9: The average SNR for each data rate (DR) measured at the gateway during scenario 2, utilising the maximum transmission unit. The error bars indicate the standard deviation σ.

4.3.3 Result from scenario 3 - minimum transmission unit at talker

Table 4.5 shows the results of the test cases 3.0 to 3.7 where the transmission unit in each packet sent from the talker was 12 bytes while the yeller sent the maximum transmission unit for each data rate.

Test case			Packets			RSSI (dBm)	SNR (dB)
Test	Talker	Yeller	P_{sent}	P_{recv}	P_{loss}	$\mu(\sigma)$	$\mu(\sigma)$
T:3.0	12bytes (DR0)	64bytes (DR0)	100	25	0.75	-41 (1.60)	9.09 (0.75)
T:3.1	12bytes (DR1)	64bytes (DR1)	100	67	0.33	-39 (3.18)	9.43 (1.60)
T:3.2	12bytes (DR2)	64bytes (DR2)	100	69	0.31	-38 (3.24)	9.53 (1.24)
T:3.3	12bytes (DR3)	128bytes (DR3)	100	74	0.26	-41 (2.93)	10.36 (0.68)
T:3.4	12bytes (DR4)	255bytes (DR4)	100	77	0.23	-41 (2.94)	10.56 (0.70)
T:3.5	12bytes (DR5)	255bytes (DR5)	100	81	0.19	-41 (3.23)	9.29 (0.61)
T:3.6	12bytes (DR6)	255bytes (DR6)	100	91	0.09	-46 (2.23)	9.27 (0.50)
T:3.7	12bytes (DR7)	255bytes (DR7)	100	0	1	n/a	n/a

Table 4.5: The results of the tests in scenario 3 with transmission units of 12 bytes, under the influence of the yeller.

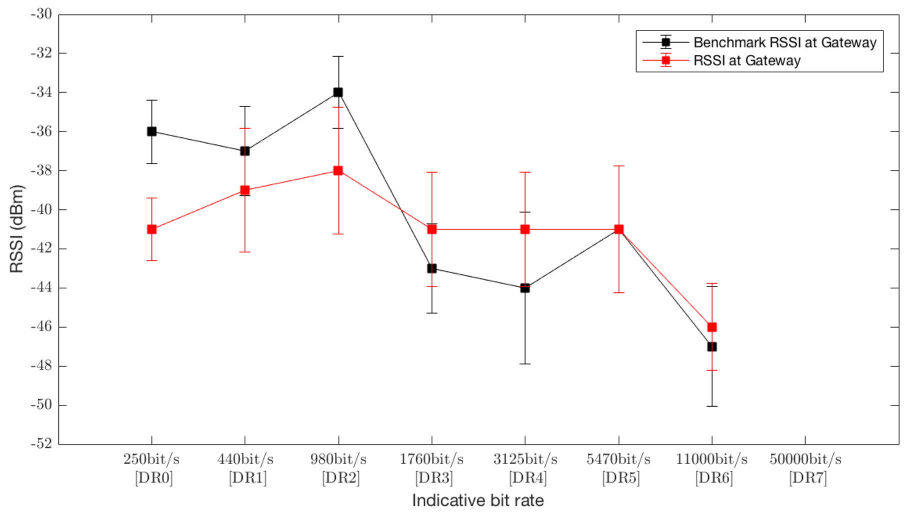


Figure 4.10: The average RSSI for each data rate(DR) measured at the gateway during scenario 3, talker utilising the minimum transmission unit. The error bars indicate the standard deviation σ.

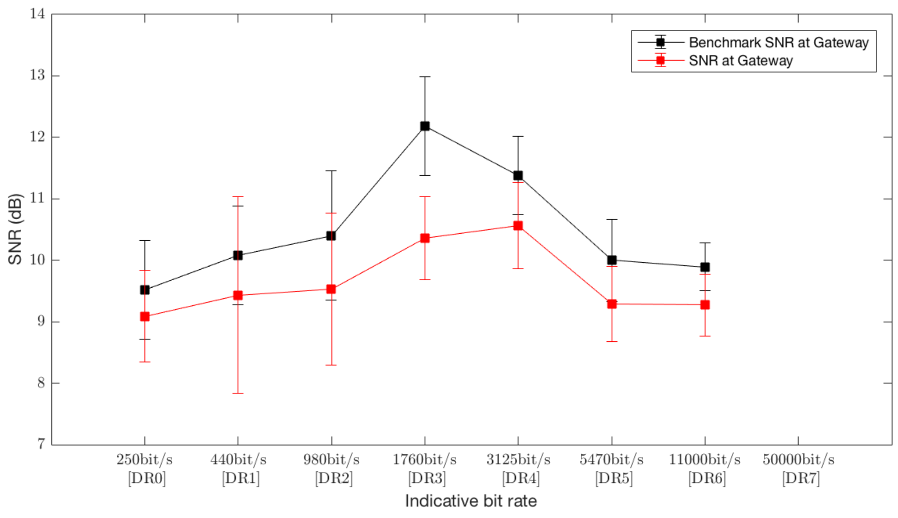


Figure 4.11: The average SNR for each data rate (DR) measured at the gateway during scenario 3, talker utilising the minimum transmission unit. The error bars indicate the standard deviation σ.

4.3.4 Result from scenario 4 - different data rates

In Table 4.6 the result form scenario 4, test case 4.0 to 4.6, can be found. The test cases were performed utilising the maximum transmission unit for each data rate for the packets sent from the talker while the yeller sent the maximum transmission unit of 64 bytes on data rate 0 (DR0). The reason that the yeller sends 64 bytes on DR0 during all tests is that theses parameters create the longest transmission time, see Table 4.2.

Test case			**Packets**			**RSSI (dBm)**	**SNR (dB)**
Test	Talker	Yeller	P_{sent}	P_{recv}	P_{loss}	$\mu(\sigma)$	$\mu(\sigma)$
T:4.0	64bytes (DR1)	64bytes (DR0)	100	100	0	-36 (3.10)	6.21 (5.54)
T:4.1	64bytes (DR2)	64bytes (DR0)	100	100	0	-35 (3.98)	2.62 (6.78)
T:4.2	128bytes (DR3)	64bytes (DR0)	100	100	0	-39 (5.53)	6.83 (5.59)
T:4.3	255bytes (DR4)	64bytes (DR0)	100	97	0.03	-37 (2.49)	9.23 (3.17)
T:4.4	255bytes (DR5)	64bytes (DR0)	100	97	0.03	-36 (2.84)	6.04 (3.95)
T:4.5	255bytes (DR6)	64bytes (DR0)	100	97	0.03	-39 (3.92)	4.26 (6.04)
T:4.6	255bytes (DR7)	64bytes (DR0)	100	0	1	n/a	n/a

Table 4.6: The results of the tests in scenario 4 (with different data rates).

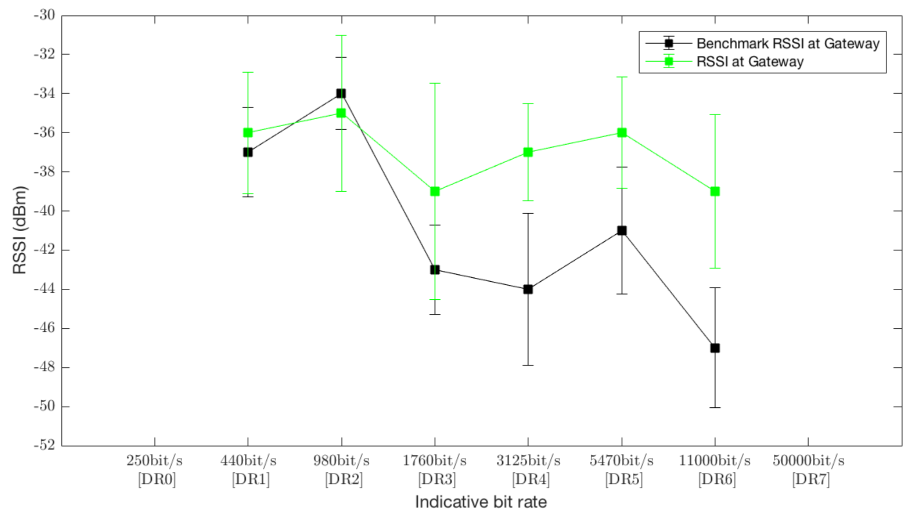


Figure 4.12: The average RSSI for each data rate (DR) measured at the gateway during scenario 4, using different Data Rates. The error bars indicate the standard deviation σ.

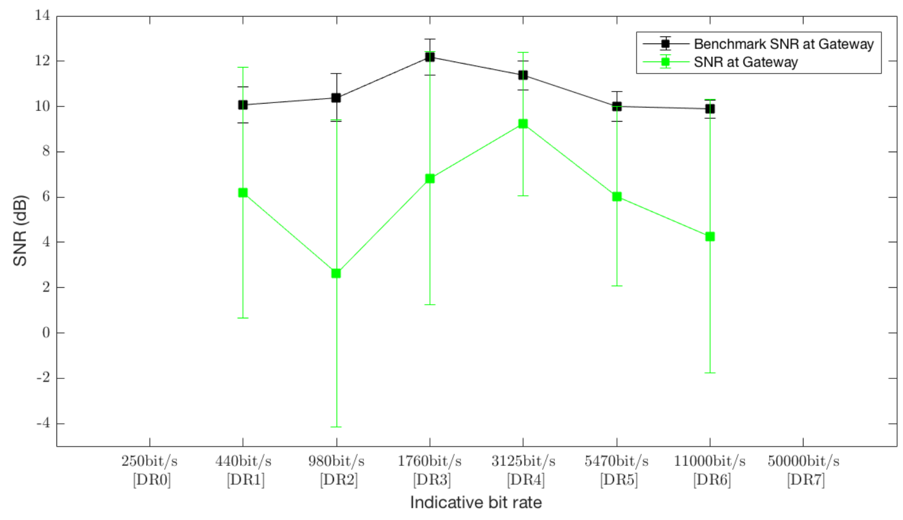


Figure 4.13: The average SNR for each data rate (DR) measured at the gateway during scenario 4, using different Data Rates. The error bars indicate the standard deviation σ.

4.3.5 Comparison of the RSSI and SNR measured at gateway

The results from the simulation of the high load condition were quite as expected. With focus on the three parameters packet loss, average SNR and average RSSI the results from scenario 1, the benchmark, indicate that the conditions on the short range is almost ideal. Zero packets were lost, a good result but nothing else can be expected for the range of 1.5m between the end-device and the gateway. The average SNR is also good. The average SNR will show us how much noise the gateway pick during the benchmark in relation to the signal sent to it. The SNR also seems to be quite stable across the different data rates while the average RSSI for the signal seems to be decreasing.

By comparing the average SNR, see Figure 4.14, it is clear that the yeller has created some noise. The most noise was created (where a lower value is worse than a higher value), in scenario 4 when the yeller was set to always send packets on DR0 with the maximum transmission unit of 64 bytes. The simple explanation of this is that the yeller used the settings to reach the maximum time on air, 54%. The estimation of time on air can be found in Table 4.2.

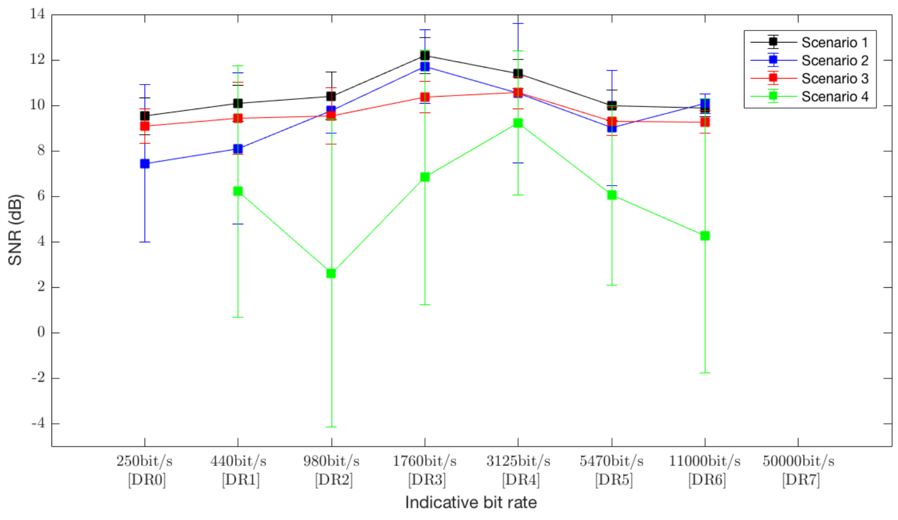


Figure 4.14: Comparison of the SNR measured at the gateway for the different scenarios. The error bars indicate the standard deviation σ.

Regarding the RSSI for the different scenarios it doesn't really provide us with any additional information, see Figure 4.15. During the scenarios where the system is under load from the yeller, the RSSI doesn't deviate notably. All the average RSSI:s measured is still above -50dBm. In Table 2.2 the upper bound for the receive sensitivity of each data rate is displayed. Consider the measured RSSI is between 30% to 40% of the upper bound the received signal strength and it is not deviating during the load test, the following conclusion can be made: RSSI is not impacted by the high load condition at the distance of 1.5m.

While the average SNR and RSSI does not deviate from the benchmark too much, the packet loss does. In scenario 2 and 3 heavy packet loss is present. The packet loss seems correlated with the maximum time on air for each data rate the yeller can achieve, as seen in Figure 4.5 and Table 4.2. The largest packet loss can be found for Test case T:2.0, the test case where both devices use DR0 and the transmission unit is set to the maximum, 64 bytes. The loss in test case T:2.0 is 92%, despite the yeller only having a channel time of 54%. The fact that scenario 2 gets a higher packet loss for DR0 than scenario 3 confirm the hypothesis that the

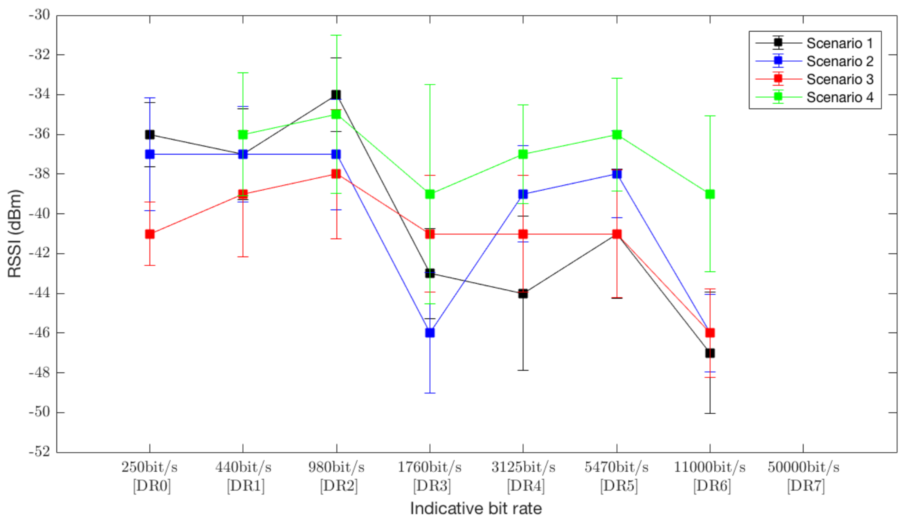


Figure 4.15: Comparison of the RSSI measured at the gateway for the different scenarios. The error bars indicate the standard deviation σ.

time on air is critical for causing collisions. By looking closer at scenario 4 we do not see particular packet loss, the loss for T:4.3, T:4.4 and T:4.5 is unknown. The fact that the test cases in scenario 4 performed for the lower data rates don't indicate any packet loss make it more likely that the loss in T:4.3, T:4.4 and T:4.5 is caused by an external source. Since the talker and the yeller not share the same data rate during scenario 4, the gateway pick up the packets sent from the yeller as noise during the transmission of packets from the talker. This assumption can be supported by deviation of the SNR for scenario 4 seen in the comparison of the SNR in Figure 4.14. The reason that the noise level does not change in scenario 2 and 3 is that the packets sent from the yeller do not create noise but result in packets overlapping causing collisions.

In Figure 4.16 all the data points from scenario 1 and 4 using DR2 are plotted. It is very clear from this plot that in some occasions the packets from the yeller and the talker overlap and in some cases not. When they overlap, a significant reduction of the SNR can be seen.

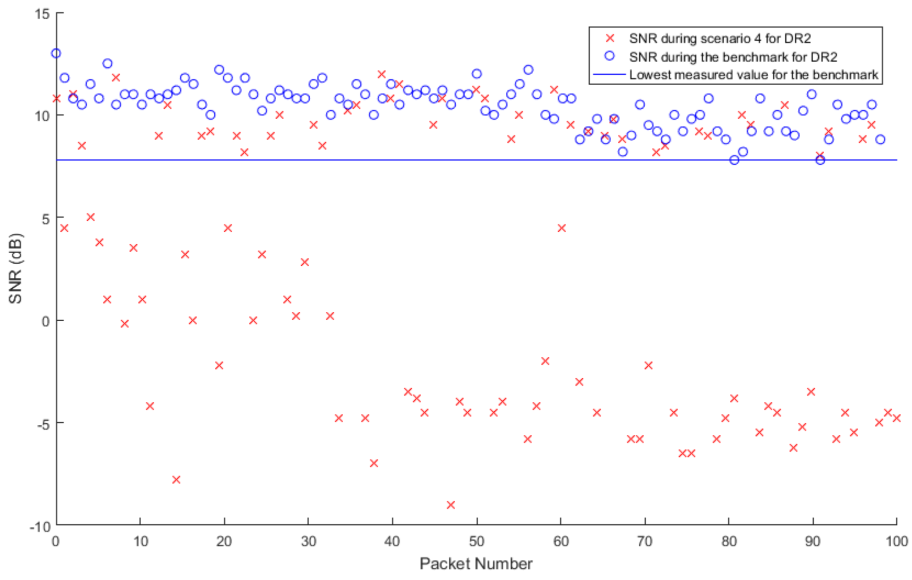


Figure 4.16: Comparison of the SNR measured for DR2 during scenario 1 (benchmark) and scenario 4. The y-axis indicates the SNR(dB) for each packet sent, and the x-axis represents packet number x.

During the experiment some of the tests failed, more specific, the tests where DR7 was used, test case T:1.7, T:2.7, T:3.7 and T:4:6. Since all tests were performed using the frequency 868.3Hz, the end-device was hard coded to use this frequency and a assigned data rate depending on test case. Data rate 7 do not use LoRa-modulation but Frequency-shift keying (FSK), where the data is transmitted through discrete frequency changes of a carrier signal. The best guess for why DR7 does not seem to work at all, even during benchmark, is that hard coding a set frequency on end-devices interferes with the FSK-modulation. This anomaly was not investigated any further though.

5 Discussion

5.1 Results

This section aims to discuss the results obtained from the practical study described in 3 and presented in section 4.

5.1.1 Throughput

The data points used when estimating the functions $t_y^{ToA}(x)$ are all evenly spaced. The effect of this was that the measurements coincidentally appeared to fit on a straight line as seen back in Figure 4.1a, even though Mikhaylovs model seems to capture a more complex behaviour as is indicated by Figure 4.4a.

If one accepts the linear approximations $t_y^{ToA}(x)$ as adequate, an interesting result is that it becomes clear how efficient it is adding or piggybacking extra data sent in a packet, as opposed to transmitting an extra packet with the additional data. Let the data, x, sent in a packet be on the form $x = x_{app} + x_{formatting} = x_{app} + 13$ (since $x_{formatting}$ is 13 bytes when application data is included). Then $t_y^{ToA}(x)$ takes the form given in equation (5.1). Equation (5.2) utilises equation (5.1) to show that increasing the size of the payload x_{app} by r times (under the assumption that it stays within the *MaTU*) is always $(r-1) \cdot k \cdot 13 + (r-1) \cdot C$ [ms] more efficient in transmission time than transmitting r packets each containing x_{app} bytes of application payload data. Table 5.1 shows the time-difference described by equation (5.2) for some values r and the different data rates.

$$t_y^{ToA}(x_{app}) = k \cdot (x_{app} + 13) + C \text{ [ms]} \tag{5.1}$$

$$\begin{aligned} r \cdot t_y^{ToA}(x_{app}) - t_y^{ToA}(r \cdot x_{app}) &= r \cdot \Big(k(x_{app} + 13) + C\Big) - \Big(k(r \cdot x_{app} + 13) + C\Big) = \\ &= \Big(r \cdot k \cdot x_{app} + r \cdot k \cdot 13 + r \cdot C\Big) - \Big(r \cdot k \cdot x_{app} + k \cdot 13 + C\Big) = \\ &= (r-1) \cdot \Big(13 \cdot k + C\Big) ms \end{aligned} \tag{5.2}$$

There are several interpretations of equation (5.2) and Table 5.1: for example one can either utilise these results and optimise the amount of data a device can broadcast over a day, or one can utilise them to minimise the risk of interfering with other devices broadcasting data in the vicinity. Which approach is beneficial depends on what is sought: higher throughput for individual devices, increased scalability allowing for more devices in a network, or a more robust network with lesser probability of collision or interference with other broadcasting devices.

DRy / r	$(r-1) \cdot (13 \cdot k + C)$ **for** DRy **[s]**							
	DR0	DR1	DR2	DR3	DR4	DR5	DR6	DR7
r = 1	0	0	0	0	0	0	0	0
r = 2	1.1546	0.6019	0.2892	0.1518	0.0822	0.0430	0.0212	0.0040
r = 3	1.5805	0.8398	0.3948	0.2119	0.1156	0.0621	0.0308	0.0061
r = 4	2.0065	1.0777	0.5004	0.2719	0.1489	0.0813	0.0404	0.0081
r = 5	2.4324	1.3157	0.6060	0.3319	0.1823	0.1004	0.0499	0.0102

Table 5.1: The time on air saved by appending r times x_{app} bytes of data to one packet, as opposed to sending r packets each containing x_{app} bytes of data. The values for k and C are obtained by selecting the first order approximation constants from Table 4.1.

5.1.2 Timing the send function and estimating T_{max}

One of the more interesting things to highlight about T_{max} (available in Table 4.2) is that $T_{max} < 10\%$ for data rate 6 and data rate 7. This means that despite having access to frequencies whose channel plans allow for a 10% duty cycle, the full potential of it cannot be utilised.

As in the case with the throughput for Maximum Transmission Units, the results obtained from Table 4.2 are in line with the results obtained by Mikhaylov et al. [19] presented in Table 2.3 under column "Max Duty Cycle".

5.1.3 Packet loss rate under simulated high load conditions

In Section 4.3.5 a comparison of the RSSI and SNR measured at gateway from the measurement of the packet loss is compared. The outcome of the experiment is also pointed out in this section. The largest packet loss is 92% and is measured during test case T:2.0. This result is quite remarkable since the yeller device only reaches a duty cycle of 54%.

Since LoRaWAN utilises the unlicensed spectrum below 1GHz several other applications and devices can be found that uses the same spectrum. Several competing techniques like Zigbee, mBus and Sigfox using the same frequency band. LoRa is based on the Chirped Spread Spectrum (CSS) technique to handle the interference problem. But if the spectrum is occupied by other applications LoRa is still facing problems operating without notifiable interference and packet loss.

In the future when the usage of the ISM band is greater and more devices and applications are using it, LoRa might potentially be facing problems. The benchmark, Scenario 1, indicates the regular conditions in the area where the study was conducted. In our environment, there is no real way of distinguishing background noise from other interfering systems operating on the same spectrum as our Access Point. The benchmark just indicates the sum of background noise and how much interference the gateway picks up from other devices operating with in the same spectrum.

Aref et al. [4] did a range test and 9.8% of the packets were invalid at the distance of 2.3km. The average RSSI registered was -115.7dBm and the average SNR was -3.2dB. The network conditions for our tests are much better, the closest to these conditions is during scenario 4 when the yeller creates as much noise as possible by using DR0 and the maximum transmission unit causing it to take 54% of the time on air. The lowest measured value for SNR is -7.8dB and the average SNR for the same test is 2.62dB, see T:4.1. The largest difference between the measurements is the distance (2.3km) and the RSSI, -115.7dBm respective -35dBm. Scenario 4 mainly indicate how much noise a device on a neighbouring channel can create.

Augustin et al. [5] point out that channel capacity of 18% entail that 60% of the packets were dropped due to collision. These results were estimated in a simulator where the parameters were set to emulate data rate 6 (DR6). By combining the fact that several devices and upcoming techniques using the ISM band increasing the noise level and how the distance between the end-device and the gateway influence the signal, LoRa is facing a grand challenge in the future. Due to the duty cycle limitations it is not always possible to resend data, thus a collision between packets can hence have large impact on the system. The impact of the timeliness on the data is therefore a fact. Running a security-critical system on top off LoRa can therefor become a problem for several reasons.

5.2 Method

5.2.1 Choice of literature

The combination of new technology and a specific question give a limited amount of useful literature. To get the basic information about LoRa and LoRaWAN we turned to the LoRa-alliance. The LoRa-alliance standardise LPWAN the information collected from them can be challengeable. At the same time they are the best source of the information about the standard, no intermediary that can misinterpret the information is introduced. The sources referring to the LoRa-alliance are [31] [11] [12].

5.2.2 Replicability, reliability and validity of experiments

The replicability[1] of the obtained results should be good. One could argue that the pseudo code provided creates space for misinterpretations and differences in implementation, which might affect the replicability of the study. However the provided material should still be enough to reproduce similar experiments, and also provides the opportunity to use other libraries and electronics for conducting a similar study.

Regarding the reliability[2] of the performed experiments there are mainly two issues that deserve highlighting: the execution and overlap timings in packet transmissions, and the ability to recreate the same noise conditions and external circumstances. The first issue may prove hard to address due to limited tools for closely monitoring and controlling transmission overlaps and/or collisions. Since-the authors cannot provide the data for exact overlaps and collisions, neither can anyone else be expected to emulate the very same outcome. Whether two transmissions overlap or not has significant impact on the measured metrics for packet loss. However, the "overlap issue" is not a problem when replicating the throughput measurements and T_{Max} measurements, since only one device is included in those experiments.

The issue of not being able to replicate the same external circumstances (such as other networks in the vicinity) and background noise needs to be addressed. The goal should be to recreate the same RSSI and SNR conditions for the access point during benchmark tests. The benchmark tests can thus be used as a reference as to whether experimental conditions are at least somewhat similar.

[1]Replicability means that someone reading a scientific report should be able to follow the method description and then carry out the same study and check whether the results obtained are similar.

[2]Reliability is a term for whether one can expect to get the same results if a study is repeated with the same method.

Validity[3]: During the experiments measuring packet loss, especially in Scenario 4 (which utilised different modulation parameters used for the transmitting devices), there is no real way of telling whether a packet was lost due to interference created by the yeller (jammer device) or whether it is a result of outer circumstances.

5.2.3 Closing thoughts on choice of methodology

This thesis formulated its methodology drawing some inspiration from Mahanti et al. [17]. What was discovered in the process of studying LoRaWAN was that the attempted methodology had some troubles. For example, maximum channel utilisation using only one device in LoRaWAN has been shown to be 54% (both mathematically and now experimentally), while we had initially hoped to be able to reach 100%. By using more devices it could be possible to reach 100% channel utilisation (in this thesis: T_{max}) instead of the obtained result seen in Table 4.2.

Two other issues were the data collection and trying to create overlap in transmissions between the two devices in the conducted experiments. Data collection was tedious since most of the data had to be manually extracted from logs-files, and cross referenced with other log-files.

To conclude this section, the authors would like to discourage anyone from attempting this type of study unless the following problems can be properly addressed:

- Working around the upper bound of channel utilisation in LoRaWAN electronics.
- Distinguish the background noise from the noise created by your own equipment
- An easier way of collecting the measurement data than manual interpretation.

5.3 The work in a wider context

We have in this thesis tried to provide experimental data regarding performance in LoRaWAN. This is currently a very new technology and research is scarce. In todays society this technique can be used for many different applications within the growing trend of "Smart city applications", but also within industrial sectors or for example farming. Since the technology may potentially end up being utilised in many societal pillars, performance is very important to study.

We would like to point out that one of the more interesting findings in this thesis is the potential of building a very effective LoRaWAN jammer. While jammers are present (and common, although illegal) for technologies such as WiFi, an important difference is the wide reach of LoRaWAN[4]PGN+16 which increases the potential damage made by a single jammer. This becomes especially important to consider when considering LoRaWAN for safety-critical applications.

[3]Validity is, somewhat simplified, concerned with whether a performed measurement actually measures what one thinks is being measured.

6 Conclusion

The purpose of this thesis was to investigate the throughput for LoRaWAN and to study performance relating to interference between LoRaWAN devices using the same frequency. Throughput has been shown to vary depending on the amount of data sent, and when transmitting the Maximum Transmission Unit for each data rate the throughput was close to what is specified in the LoRaWAN standard. It is very important to note that it is the throughput for *one packet* at a time that is measured in this thesis and also advertised in the LoRaWAN standard. It has also been shown that when channel utilisation is above 54% for Data Rate 0, one can expect a very high packet loss rate. In the conducted experiments it was seen that when sending 51 bytes of application data on such a channel, one can receive a packet loss as high as 92%.

The tests in this thesis are not conclusive enough to draw any broad concrete conclusions about LoRaWAN, what can be said however is that devices operating in the same frequency sub-band (but on different data rates) give impact on the RSSI and the SNR. What is interesting about this is that broadcasts on the same channel but with different data rates could both coexist without significant packet loss, which could be considered an expected result since it is one of the major selling points of LoRaWAN.

Bibliography

[1] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne. “Understanding the Limits of LoRaWAN”. In: *IEEE Communications Magazine* 55.9 (2017), pp. 34–40.

[2] K. Al Agha, G. Pujolle, and T. Yahiha. *Mobile and wireless networks*. 2nd ed. Hoboken, New Jersey, Wiley, 2016.

[3] Zigbee Alliance. “Zigbee IP and 920IP - Technical Summary”. In: (). [Online; accessed 2017-10-02].

[4] M. Aref and A. Sikora. “Free space range measurements with Semtech Lora x2122; technology”. In: *2nd International Symposium on Wireless Systems within the Conferences on Intelligent Data Acquisition and Advanced Computing Systems* (Sept. 2014), pp. 19–23.

[5] A. Augustin, J. Yi, T. Clausen, and W. Townsley. “A Study of LoRa: Long Range and amp; Low Power Networks for the Internet of Things”. In: *Sensors* 16.9 (Sept. 2016), pp. 1–18.

[6] D. Bankov, E. Khorov, and A. Lyakhov. “On the Limits of LoRaWAN Channel Access”. In: *International Conference on Engineering and Telecommunication (EnT)* (Nov. 2016), pp. 10–14.

[7] A. Banks and R. Gupta. “MQTT Version 3.1.1 OASIS Standard”. In: (Oct. 2014). [Online; accessed 2017-01-25], pp. 1–81.

[8] C. Bormann, M. Ersue, and A. Keranen. “RFC 7228: Terminology for Constrained-Node Networks”. In: *Internet Requests for Comments, IETF* (May 2014).

[9] European commission. “D3.6 - IoT Protocol Suite definition”. In: *IOT-A WP3 D3.6* (2013). [Online; accessed 2017-02-13].

[10] European commission. “IOT-A - Internet of Things Architecture”. In: *Project ID: 257521* (2013). [Online; accessed 2017-02-10].

[11] LoRa Alliance Technical committee. “LoRaWAN Regional Parameters - companion document to the LoRaWAN1.0.2 Specification”. In: *LoRa Alliance, Inc.* (July 2016), pp. 1–45.

[12] Actility Gemalto and Semtech. “LoRaWAN Security - Full End-To-End encryption for IoT application providers”. In: *LoRa Alliance* (Feb. 2017), pp. 1–4.

[13] ISO/ICE ISO20922:2016. “Information technology – Message Queuing Telemetry Transport (MQTT) v3.1.1”. In: (June 2016). [Online; accessed 2017-01-25], pp. 1–73.

[14] B. Javed, M. W. Iqbal, and H. Abbas. “Internet of things (IoT) design considerations for developers and manufacturers”. In: *IEEE International Conference on Communications Workshops (ICC Workshops)* (May 2017), pp. 834–839.

[15] J.Kurose and K. Ross. *Computer networking : a top-down approach*. 5th ed. Pearson Education, cop. 2013.

[16] Z. Li, S. Zozor, J. M. Drossier, N. Varsier, and Q. Lampin. "2D time-frequency interference modelling using stochastic geometry for performance evaluation in Low-Power Wide-Area Networks". In: *IEEE International Conference on Communications (ICC)* (May 2017), pp. 1–7.

[17] A. Mahanti, N. Carlsson, C. Williamson, and M. Arlitt. "Ambient Interference Effects in Wi-Fi Networks". In: *NETWORKING 2010: 9th International IFIP TC 6 Networking Conference, Chennai, India* (May 2010), pp. 160–173.

[18] G. Margelis, R. Piechocki, D. Kaleshi, and P. Thomas. "Low Throughput Networks for the IoT: Lessons learned from industrial implementations". In: *IEEE 2nd World Forum on Internet of Things (WF-IoT)* (Dec. 2015), pp. 181–186.

[19] K. Mikhaylov, J. Petäjäjärvi, and T. Hänninen. "Analysis of Capacity and Scalability of the LoRa Low Power Wide Area Network Technology". In: *European Wireless 2016; 22th European Wireless Conference* (May 2016), pp. 1–6.

[20] K. Mikhaylov, J. Petäjäjärvi, T. Hänninen, and A. Pouttu. "D2D communications in LoRaWAN Low Power Wide Area Network: From idea to empirical validation". In: *IEEE International Conference on Communications Workshops (ICC Workshops)* (May 2017), pp. 737–742.

[21] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac. "Internet of things: Vision, applications and research challenges". In: *Ad Hoc Networks* 10.7 (Sept. 2012), pp. 1497–1516.

[22] S. Naoui, M. E. Elhdhili, and L. A. Saidane. "Enhancing the security of the IoT LoraWAN architecture". In: *International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN)* (Nov. 2016), pp. 1–7.

[23] P. Neumann, J. Montavont, and T. Noël. "Indoor Deployment of Low-Power Wide Area Networks (LPWAN): a LoRaWAN case study". In: *IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)* (Oct. 2016), pp. 1–8.

[24] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, and M. Pettissalo. "On the coverage of LPWANs: range evaluation and channel attenuation model for LoRa technology". In: *14th International Conference on ITS Telecommunications (ITST)* (Dec. 2015), pp. 55–59.

[25] T. Petrić, M. Goessens, L. Nuaymi, L. Toutain, and A. Pelov. "Measurements, performance and analysis of LoRa FABIAN, a real-world implementation of LPWAN". In: *IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)* (Sept. 2016), pp. 1–7.

[26] D. M. Pozar. *Microwave and RF wireless systems*. ill. New York : Wiley, cop. 2001.

[27] B. Reynders, W. Meert, and S. Pollin. "Power and spreading factor control in low power wide area networks". In: *IEEE International Conference on Communications (ICC)* (May 2017), pp. 1–6.

[28] M. Sauter. *From GSM to LTE. an introduction to mobile networks and mobile broadband*. -. Chichester, West Sussex, U.K. : Wiley, 2011.

[29] S. Sicari, A. Rizzardi, D. Miorandi, C. Cappiello, and A. Coen-Porisini. "Security policy enforcement for networked smart objects". In: *Computer Networks* 108.Supplement C (2016), pp. 133–147.

[30] Sigfox. "Sigfox Technology Overview". In: (). [Online; accessed 2017-10-01].

[31] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent. "LoRaWAN Specification V1.0.2". In: *LoRa Alliance, Inc.* (July 2016), pp. 1–70.

[32] National Institute of Standards and Technology. “Framework for Improving Critical Infrastructure Cybersecurity”. In: (Feb. 2014), pp. 1–41.

[33] S. Tomasin, S. Zulian, and L. Vangelista. “Security Analysis of LoRaWAN Join Procedure for Internet of Things Networks”. In: *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)* (Mar. 2017), pp. 1–6.

[34] J. Toussaint, N. El Rachkidy, and A. Guitton. “Performance analysis of the on-the-air activation in LoRaWAN”. In: *IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (Oct. 2016), pp. 1–7.

[35] L. Vangelista, M. Centenaro, and D. Magrin. “Performance evaluation of LoRa networks in a smart city scenario”. In: *IEEE International Conference on Communications (ICC)* (May 2017), pp. 1–7.

[36] N. Varsier and J. Schwoerer. “Capacity limits of LoRaWAN technology for smart metering applications”. In: *IEEE International Conference on Communications (ICC)* (May 2017), pp. 1–6.

[37] O. Vermesan and P. Friess. “Internet of Things - From Research and Innovation to Market Deployment”. In: (2014). [Online; accessed 2017-03-16].

[38] O. Vermesan, P. Friess, and A. Furness. “The Internet of Things 2012 Cluster Book”. In: (). [Online; accessed 2017-03-03].

[39] T. Wendt, F. Volk, and E. Mackensen. “A benchmark survey of long range (LoRaTM) spread-spectrum-communication at 2.45 GHz for safety applications”. In: *IEEE 16th Annual Wireless and Microwave Technology Conference (WAMICON)* (Apr. 2015), pp. 1–4.

[40] S. Zulian. “Security threat analysis and countermeasures for LoRaWAN join procedure”. In: *Università degli Studi die Padova (Padova Digital University Archive for thesis)* (2016).

A Appendix

Data Rate	t_{ToA} **for Datapoints** $x_0, x_1, ...x_{10}$ **(ms)**										
DRy	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
DR0	1155	1318	1482	1646	1810	1974	2138	2301	2465	2629	2793
DR1	577	659	823	905	987	1069	1150	1232	1314	1396	1560
DR2	288	329	370	411	452	493	534	575	616	657	698
DR3	144	205	267	328	369	431	492	533	594	656	676
DR4	82	143	215	276	338	399	471	532	594	655	707
DR5	41	82	118	153	189	225	266	302	338	374	399
DR6	20	41	59	76	94	112	133	151	169	187	199
DR7	4	8	12	16	20	24	28	32	36	40	43

Table A.1: For each data points x and data rate DRy, the time on air t_{ToA} is presented.

B Appendix

Data Rate	t_{exec}^{DRy} **for Datapoints** $x_0, x_1, ...x_{10}$ **(ms)**										
DRy	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
DR0	3494	3670	3834	3998	4162	4326	4490	4654	4818	4982	5146
DR1	2925	3018	3182	3265	3347	3429	3511	3594	3676	3757	3922
DR2	2636	2689	2730	2771	2812	2853	2895	2936	2978	3018	3059
DR3	2491	2565	2627	2689	2730	2792	2854	2896	2958	3020	3041
DR4	2429	2504	2576	2639	2702	2764	2837	2899	2962	3024	3076
DR5	2380	2434	2470	2508	2544	2582	2624	2660	2697	2734	2760
DR6	2367	2401	2420	2439	2458	2477	2498	2518	2537	2555	2569
DR7	2352	2369	2373	2377	2381	2386	2390	2395	2399	2404	2406

Table B.1: Measurements of $t_{exec}^{DRy}(x)$ for various Data Rates and data points x_n.

Data Rate	$t_{exec}^{DRy} - t_{ToA}^{DRy}$ **for Datapoints** $x_0, x_1, ...x_{10}$ **(ms)**										
DRy	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}
DR0	2339	2352	2352	2352	2352	2352	2352	2353	2353	2353	2353
DR1	2348	2359	2359	2360	2360	2360	2361	2362	2362	2361	2362
DR2	2348	2360	2360	2360	2360	2360	2361	2361	2362	2361	2361
DR3	2347	2360	2360	2361	2361	2361	2362	2363	2364	2364	2365
DR4	2347	2361	2361	2363	2364	2365	2366	2367	2368	2369	2369
DR5	2339	2352	2352	2355	2355	2357	2358	2358	2396	2323	2361
DR6	2347	2360	2361	2363	2364	2365	2365	2367	2368	2368	2370
DR7	2348	2361	2361	2361	2361	2362	2362	2363	2363	2364	2363

Table B.2: Calcuclating t_{other} for various Data Rates and data points x_n.