University of Zurich^UZH

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Simulation and Efficiency Improvements of the IoT Communication Protocols Used in the Supply Chain Monitoring Systems

Timo Surbeck
Hettlingen, Zurich, Switzerland
Student ID: 15-701-733

Supervisor: Dr. Eryk Schiller, Sina Rafati
Date of Submission: April 25, 2019

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

Der Fachbereich um Supply Chain Monitoring hat in vergangenen Jahren stark an Bedeutung zugenommen. In diesem Feld eingesetzte Geräte, welche der Klasse des Internets der Dinge zugehören, können stets kleiner gestaltet werden und verbrauchen immer weniger Energie: Dies liess eine Nachfrage für Low Power Wide Area Network (LPWAN) Technologien, welche der Übermittlung von verteilt gesammelten Wertschöpfungsketten-Daten dienen, entstehen. Im Zuge dieser Arbeit wird ein bestehendes Framework für LPWAN-Simulation erweitert, um das Protokoll der simulierten drahtlosen Netzwerke Blockchain-konform zu machen. Zudem wird ein effizienzsteigernder Mechanismus implementiert, um die Performanz simulierter Netzwerke zu steigern. Abschliessend werden verschiedene Szenarien miteinander verglichen, indem sowohl die räumliche als auch zeitliche Dichte simulierter Netzwerke schrittweise heraufgesetzt wird.

The area of supply chain monitoring has gained a lot of importance in recent times. In this field, a variety of devices belonging to the domain of the Internet of Things are becoming increasingly lean and energy-efficient: This let the demand for the Low Power Wide Area Network (LPWAN) technologies, which are applied to transport distributedly collected supply chain data, arise. In the course of this thesis, an existing framework for LPWAN simulation is enhanced by implementing the requirements necessary to make wireless communication technologies blockchain-compliant. Moreover, an efficiency improvement mechanism is integrated into the simulation framework to increase the performance of simulated networks. Finally, to compare several scenarios, the spatial as well as temporal density of simulated networks is gradually increased.

Acknowledgments

I would like to express my sincere gratitude:

To Prof. Dr. Burkhard Stiller, head of the Communication Systems Group at the University of Zurich, for making it possible for me to work on a current and interesting topic.

To my supervisors, Dr. Eryk Schiller and Sina Rafati, for their support and shared expertise.

To my family, for their constant hospitality during this intense time.

To Jonas Pescatore and Alessandro Smider, for their friendship and beautiful music.

Contents

Chapter 1

Introduction

Nowadays, supply chain monitoring is a field which has become increasingly important in several different sectors. Not only is it of interest for producers but also for suppliers, distributors, retailers as well as for customers to gather a variety of data *resp.,* measurements about the production of goods as they traverse different stages of value chains. The most obvious way to distribute generated data would be to use wired networks. However, this solution would only be sustainable if the considered supply chain was static in terms of its value-adding stages' geographical distribution as well as of the constancy of production procedures; Establishing wired networks within large areas as well as adjusting them at frequently evolving supply chain stages leads to high expenses. Another issue becomes clear, when considering industries operating in very crude, remote territories lacking commercial sources of energy as well as network coverage: Should supply chain monitoring expand into such spacial conditions, it occurs the demand for low-power communication technologies running on battery storages as well as featuring wide communication range. The class of Low Power Wide Area Network (LPWAN) technologies is specialized for fulfilling exactly these needs. Among several LPWAN standards, LoRaWAN (Long Range Wide Area Network) [1] is the most popular and an excellent candidate for being used in supply chain monitoring, as it is freely usable and deployable.

Not only are the communication protocols used in supply chain monitoring an emerging domain but also the field of data producing sensor devices themselves — namely the Internet of Things (IoT). The fact that more and more computational power can be provided by increasingly small, cheap as well as low-energy devices makes it attractive for producing companies to distribute many such data collecting 'things', *e.g.*, temperature sensors, among their supply chain.

1.1 Motivation

It seems reasonable to investigate on how novel wireless transmission technologies can exist as a remedy to the aforementioned issues, as by featuring large communication range as well as very low energy consumption they realize an unprecedented level of flexibility. However, these advantages come at the expense of drastically lower transmission speeds, which is a fact counteracting the aim of companies to collect and transmit as much supply chain data as possible. Therefor, many current research projects work on different techniques allowing for efficiency improvements in LPWAN technologies.

Not only improvements in terms of efficiency, but also data integrity oriented blockchain technology in conjunction with supply chain monitoring has recently become an emerging subject: Blockchain technology for itself merely ensures, that stored data cannot be altered afterwards, however, guaranteeing, that genuine data was issued to the blockchain originally, remains a responsibility to its users. For the domain of supply chain monitoring, this means, that technologies ensuring modification-proof wireless communication have to be implemented, if collected data will ultimately be forwarded to blockchains.

1.2 Description of Work

Initially, for addressing real world use cases, an industrial sector is analyzed, for which supply chain monitoring using IoT in combination with LPWAN communication can be implemented usefully. The gathered use cases are combined in a design of a functional architecture presenting various IoT/LPWAN implementations spread over the entire supply chain of an exemplary representative of the considered industry sector.

It follows the principal task of this thesis, namely to analyze the performance-related limits of LoRaWAN networks applicable for supply chain monitoring by conducting simulations. Thematically, this thesis is a part of a group of several theses currently worked at at CSG, which all provide output for a project combining the fields of IoT and blockchain technology. A fellow student, Ile Cepilov, will provide the requirements for making real-world LoRaWAN networks blockchain-compliant; These necessary characteristics will be implemented, *resp.*, represented in an existing simulation framework for LoRaWAN, such that simulation scenarios can be designed, in which data transported within simulated LPWANs is guaranteed to remain integer. This way, conducted simulation scenarios will be tailored to the strategy of collecting supply chain data using LoRAWAN in a secured way and forwarding them onto a blockchain database.

Eventually, the simulation framework will be further enhanced by adding the logic of a distributed efficiency improvement mechanism, which is expected to increase the performance of simulated network scenarios.

1.3 Thesis Outline

This thesis is divided into the following chapters:

Chapter 2 provides a functional IoT/LPWAN architecture design for the supply chain of a considered industrial branch.

Chapter 3 gives insight on prevailing communication technologies currently used in IoT as well as on available simulation frameworks.

In chapter 4 the design and implementation of this work's contributions concerning LoRaWAN simulations are presented.

Chapter 5 covers the implemented methods as well as results of an intermediately conducted sub-project, in which a real data set was processed to gain information about the transmission frequency of existing LoRaWAN devices.

In chapter 6, the gained simulation results are displayed and discussed.

Finally, chapter 7 comprises a summary over the entire project and feature a section on future work in the direction of advanced efficiency improvements for LPWANs.

Chapter 2

Functional Architecture

2.1 Introduction

Nowadays, the oil and gas industry (O&G) is under constant observation due to growing importance of environmental concerns. The enforcement of increasingly harsh regulations lead to the omnipresent interest of agents in this industry to mitigate the risk of accidents and to keep financial losses due to penalties as small as possible. For mainly two reasons, it was decided to choose this industry as a basis for envisioning IoT applications using LPWAN communication and designing a functional architecture in the course of this thesis: Firstly, at every supply chain stage in O&G, goods being processed should be managed with the utmost precaution, which provides the need for collecting sensor data to be put into different rapid alert systems. Secondly, the O&G supply chain's production as well as transportation phase mostly take place in uncivilized environments where taking advantage of LPWAN technology's characteristics is beneficial [42].

The next section covers potential IoT/LPWAN applications at different stages of the O&G supply chain, while section 2.3 presents a diagram of a functional architecture of IoT/LPWAN applications spread over the O&G supply chain.

2.2 IoT and LPWAN in the Oil and Gas Industry

Seismic Acquisition

As a result of progressively strict conditions, operators at the upstream section of the O&G supply chain (crude oil gathering) are required to adhere to environment-related precautionary measures. To better predict environmental impacts as early as possible, more real-time sensor data gathered about the seismic activity in the area of oil fields is required. Shell and Innoseis associates conducted a proof of concept by introducing a LoRaWAN-based network of wirelessly transmitting sensor instruments used for seismic quality control. They not only justify the usage of LoRaWAN due to the advantage of wireless networks over inflexible cable-based systems but also approve the fact that LoRaWAN features extremely low energy consumption; Both factors are expected to reduce the need for maintenance (*e.g.* cable repair work, battery changes) in dangerous, harsh environments [37].

Upstream Monitoring

In a 2017 article, G. Dixit of the Oil and Natural Gas Corp. [32] presented the advantages of upstream monitoring in terms of planning shutdowns in the downstream (refineries): The more stations at the upstream (*e.g.*, well pumps) are equipped with wirelessly communicating IoT sensors, the earlier refineries can be warned about upcoming outages leading to temporary downtimes. For refineries, downtimes as well as recoveries thereof are extremely costly — especially if they appear unexpectedly: At the least, using LPWAN-driven IoT sensors for upstream monitoring allow for forewarning the downstream sector about upcoming raw material bottlenecks, such that by pre-planning, downtimes can be reduced, and the overall performance of refineries be improved [32].

Not only outage detection but also sensors able to determine types of incoming oil blends find good usage at the upstream: The earlier incoming blends are categorized and communicated to the downstream, the better refineries can adapt to changing raw material and reduce adjustment overhead [42].

Pipeline Leakage Detection

A very critical stage in the O&G downstream is the transportation of substances in pipelines: Not only causing a huge, negative effect on climate change but also on the operator's financial situation as well as reputation, leakages are to avoid at any cost. The better pipelines are covered with leakage detection instruments communicating in real-time, *e.g.*, using LPWANs, the earlier operators can react and fix leakages as fast as possible [42].

Cargo Shipping

The ultimate step of the O&G supply chain, namely the transportation of completed goods, again takes place in remote areas — should the substances be shipped by sea. For letting ships transmit data to mainland, LPWANs are not an option, as there is no possibility, *e.g.*, to setup stationary (intermediate) LoRaWAN gateways in the open sea. However, transportation ships themselves might operate an LPWAN to monitor parts of the ship which are either difficult to access or completely unreachable. For forwarding collected IoT sensor data (*e.g.*, tank temperatures and pressures *etc.*) to the outside world, satellite technology can be used as a remedy [42].

Remote Tank Level Monitoring

Today, still a lot of companies as well as private households possess proper oil tanks which have to be checked and re-filled regularly for avoiding interruptions in production *resp.*, heating. With increasing consumption, the effort to measure tank levels and ordering refills by hand becomes obviously higher. As a solution to this problem there are several implementations involving sensor devices as well as wireless communication systems: For example, Swiss manufacturer for isolated pressure transducers and transmitters, KELLER AG, presented a system called EasyOil®, in which oil tanks are equipped with pressure sensors: As soon as measured pressures drop below a certain threshold, oil tanks are considered almost empty and a connected GSM transmission module automatically places a replenishment order [33]. Evidently, this system could also be envisioned by using LPWAN communication instead of GSM: Not only would there be no more reliance on coverage by a GSM provider but also could communication expenses be economized.

2.3 Functional Architecture Design

For effectively simulating IoT/LPWAN applications in the O&G supply chain, the functional supply chain architecture design at the end of this section was developed. The chronologically ordered numbers wrapped in green circles correspond to different supply chain stages (from raw material extraction to end consumer) with individually configured applications:

1. The first stage is set at a production site of the O&G supply chain, namely an oilfield of 400 km^2. There are two different IoT/LPWAN applications used in this area: Ten randomly distributed monitoring sensors used for seismic acquisition as well as 20 randomly distributed upstream monitoring instruments installed at oil pumps. All 30 sensor stations equip LoRaWAN end devices for transmitting collected data. There are 9 equally distributed LoRaWAN gateways in the area. It is noteworthy to mention, that in the entire figure, only the oilfield gateways' geographical positions are true to scale and scope.

2. The second stage is a crude oil pipeline of 100 km of length, which is equipped with 21 geographically equally distributed leakage detection sensors. 21 LoRaWAN end devices send data collected by their respective leakage detectors to 10 LoRaWAN gateways, which geographically are equally distributed alternatingly 600 m above as well as 600 m below the pipeline.

3. The third setting takes place inside a cargo ship, where five randomly distributed sensors measure various dimensions (*e.g.*, pressures, temperatures, *etc.*) of unreachable parts of the ship. LoRaWAN end devices transmit collected data to one central gateway.

4. The fourth and last supply chain stage takes place directly at a customer holding three oil tanks equipped with pressure sensors. Measured pressures at the bottom

of the oil tanks are converted into tank level metrics and — using LoRaWAN end devices — communicated to the customer's single central gateway.

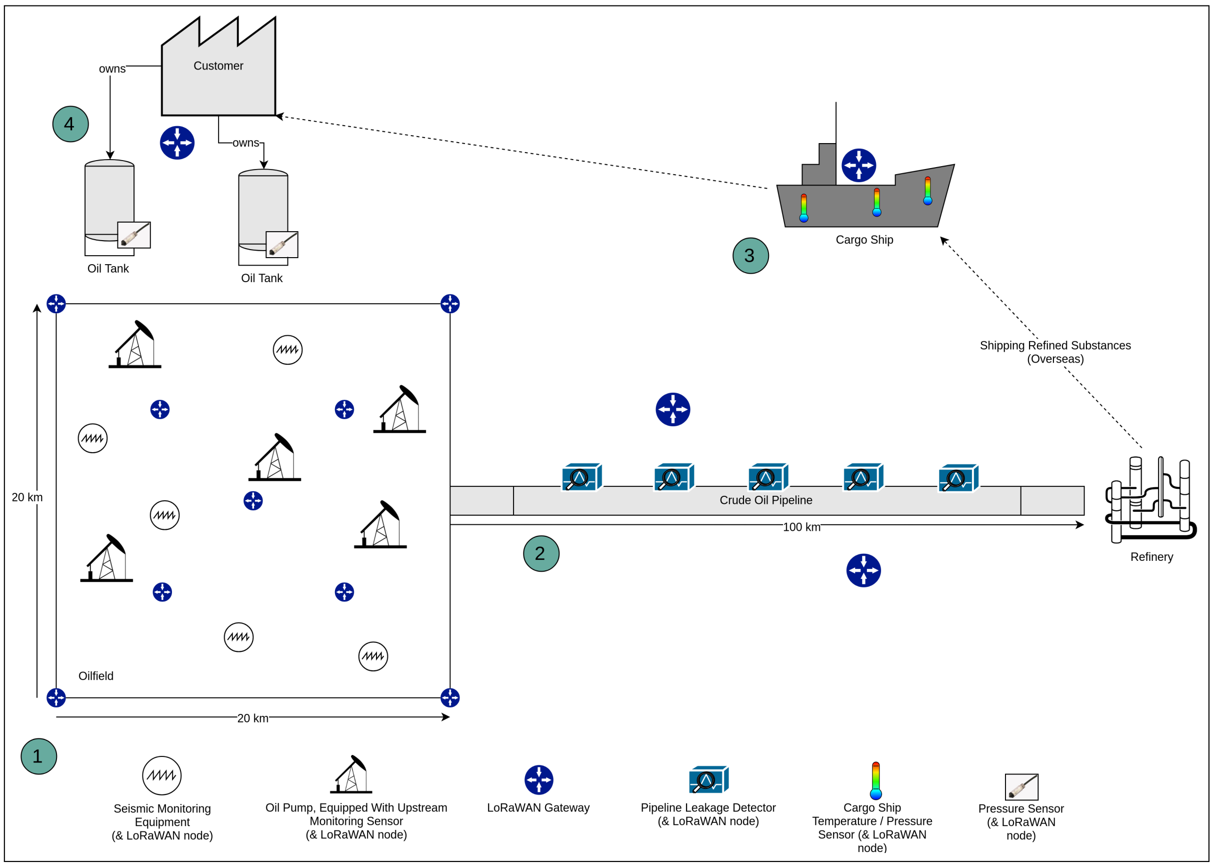
Customer
owns
owns
4
Oil Tank
Oil Tank
Cargo Ship
3
Shipping Refined Substances (Overseas)
20 km
Oilfield
20 km
Crude Oil Pipeline
100 km
2
Refinery
1
Seismic Monitoring Equipment (& LoRaWAN node)
Oil Pump, Equipped With Upstream Monitoring Sensor (& LoRaWAN node)
LoRaWAN Gateway
Pipeline Leakage Detector (& LoRaWAN node)
Cargo Ship Temperature / Pressure Sensor (& LoRaWAN node)
Pressure Sensor (& LoRaWAN node)

Figure 2.1: Functional Architecture Design O&G Supply Chain

2.4 Supply Chain Monitoring and Blockchain

All gathered data at several different places within the O&G supply chain can eventually be forwarded to a blockchain. This blockchain could be maintained by *e.g.*, a government, non-profit organizations or even by the general public (public blockchain) — to name a few entities putting great emphasis on O&G companies' accountability. For example, if a dishonest O&G company measured unfavourable seismic conditions in an area and still continued extracting oil, for lowering the risk of getting penalized, they might purposely obscure the measured data afterwards. However, if sensed data was directly (*i.e.*, unchangedly) added to a blockchain, there would not be any possibility for the company to cover up data later on. In the best sense, collaborating in such a blockchain will be very beneficial for O&G companies' reputation, as their willingness of being identifiable and accountable for any kind of prohibited action will be publicly known and appreciated.

Chapter 3

Related Work

3.1 Introduction

In this chapter, the state of the art in terms of communication technologies allowing Internet of Things (IoT) devices to communicate is elaborated. Whereas in total seven different technologies, *resp.*, standards are broached, LoRaWAN is discussed most in-depth: At the beginning of the process time of this thesis, it was decided, that the focus should only go into the direction of LoRaWAN, to keep the workload in the realms of possibly and to let the output of several theses running concurrently be compatible for later re-use.

LoRaWAN as well as other communication technologies in IoT belong to the class of Low Power Wide Area Networking (LPWAN) [25, 27] offering long-range communication at low power consumption. Due to the deployment of several technologies, batteries powering IoT devices using LPWAN to communicate may run for several years without replacement [40].

3.2 Long Range Wide Area Network (LoRaWAN)

LoRaWAN is arguably the most adopted among current LPWAN standards. It features simple network structures as well as management and provides ubiquitous connectivity, which is advantageous for outdoor IoT applications [27]. LoRa (short form for LoRaWAN) has become an interesting technology for lightweight IoT sensor devices [1]. The LoRaWAN standard defines a radio layer based on the Chirp Spread Spectrum (CSS) modulation as well as a simple channel access method. LoRa operation depends on the following parameters [41]:

1. Bandwidth: A range of frequency spectrum used for transmitting data
2. Spreading Factor (SF): The chirp rate influencing the bit rate as well as reliability (*i.e.*, higher SF result in lower bit rate as well as bit error rate)

3. Coding Rate: The ratio of redundant information used for forward error correction

The CSS modulation used in LoRa enables low reception sensitivity allowing for transmissions over wide distances. It manages to let end devices communicate within a range of several kilometers outdoors and hundreds of meters indoors [46]. In the outdoors case, there is a loss rate (*i.e.*, the ratio between lost packets and packets sent in the network totally) of less than 10% over a distance of 2 km for SF 9-12, and a more than 60% loss rate over 3.4 km for SF 12. Depending on the duty cycle (*i.e.*, how often the spectrum is used to transmit data) of LoRa devices, their lifetimes can be significantly extended — for instance, up to 17 years for a node (an end device) sending 100 B once in a day [40].

LoRaWAN defines an access method [41] to the radio channel similar to ALOHA [26]: For sending data, a device 'wakes up' and immediately transmits a packet to reachable gateways. LoRaWAN and pure ALOHA only differ in terms of the variable packet length used in LoRa in comparison to the fixed packet size in ALOHA.

The European Telecommunications Standards Institute (ETSI) regulations of the 868 MHz ISM frequency band set the limits on the maximum duty cycles between 0.1% and 10% in the 863—870 MHz of the Industrial, Scientific, and Medical (ISM) band (depending on the selected sub band). This is a result of the pure ALOHA implementation of LoRa devices which does not conform to the Listen Before Talk (LBT) schema required by ETSI (*i.e.*, if LBT was applied, transmitters would be relieved from the maximum duty cycle regulations). This circumstance obviously limits the throughput of devices and the overall network capacity. Moreover, the LoRaWAN operation similar to ALOHA results in an increasingly high level of packet losses due to collisions as the number of devices grows [28]. For a larger number of devices, the performance of LoRaWAN strictly follows ALOHA with maximum channel capacity of 18% and an increasingly high collision ratio. As an example, for the link load of 0.48, the collision ratio is around 60% (*i.e.*, the ratio between lost packets due to interference and packets sent in the network totally) [28]. The impact of collisions is — however — significantly mitigated by the capture effect and orthogonal spreading factors, such that some transmissions benefiting from a stronger signal are successful despite collisions [44].

Adelantado et al. [27] realized that for low duty cycles, network throughput is limited by collisions, whereas for greater duty cycles, the maximum duty cycle set by the ETSI regulations (i.e., 0.1% - 10%) prevents devices from increasing their packet transmission rates and limits the overall throughput of the network.

T-H. To et al. [46] improve the performance of LoRaWAN networks by not impacting energy consumption at the same time. They provide a simple LBT enhancement to LoRaWAN which effectively lowers the collision ratio. Their results show that Carrier Sense Multiple Access (CSMA) considerably lowers the collision ratio, while only slightly increasing energy consumption. Moreover, CSMA is implemented through an LBT mechanism preceding every transmission, therefor, the devices are relieved from the restrictive 0.1% - 10% duty cycle regulations by ETSI, allowing for higher achievable throughput. Furthermore, they observed that CSMA featured lower energy consumption than LoRaWAN for a large number of devices. In conclusion, the work of T-H. To et al. significantly increases data rates as well as the probability of successful transmissions for low density networks at the expense of slightly higher energy consumption. At the same time, the probability of successful transmissions, throughput as well as energy efficiency for high density networks are improved.

3.3 Competing LPWAN Technologies

Next to LoRaWAN, there is a series of currently other known LPWAN standards, which are briefly addressed in the following paragraphs as well as compared in terms of performance characteristics in Table 3.1 at the end of this section.

SigFox [27] is a very commonly used LPWAN solution basing on Ultra Narrow Band technology operating in the 869 MHz (Europe) as well as in the 915 MHz (USA) frequency bands. Due to the fact that SigFox is a proprietary technology and per the effective business model, the company possesses the network itself, a large part of the interest group in the LPWAN domain migrated towards more open standards such as LoRaWAN.

Ingenu Random Phase Multiple Access is a proprietary solution for operating private as well as industrial Machine-to-Machine (M2M) networks [27], which transmits signals on the 2.4 GHz frequency band. The outstanding advantage of this technology are the exceptionally high data rates (*cf.* the comparison table below).

Weightless-N is an open LPWAN standard of a set of technologies provided by the Weightless Special Interest Group (*i.e.*, Weightless-W, Weightless-N, and Weightless-P) [27]. Among all three standards Weightless-N provides the longest battery lifetimes of up to 10 years as well as extended transmission ranges, making it a noticeable LoRaWAN competitor.

3GPP The 3GPP (3rd Generation Partnership Project) [27] — most commonly known for mobile telecommunication — developed a series of inexpensive, simple devices intended to be used for eMTC (enhanced Machine Type Communications) [29,36]. More specifically, 3GPP addresses the IoT domain by standardizing the following three technologies:

- Long Term Evolution for Machines (LTE-M), operating in LTE frequency bands within a 1.4 MHz bandwidth
- Enhanced Narrow Band IoT (NB-IoT), an alternative to LTE-M with reduced complexity and cost (at the expense of lower data rates)
- Extended Coverage GSM IoT (EC-GSM-IoT), a standard based on EGPRS and tailored on IoT use cases

It should be mentioned, that due to the fact that cellular IoT communication means operate in licensed frequency bands, participation of mobile operators while setting up private networks cannot be circumvented, which is a profound disadvantage in comparison to openly deployable technologies, such as LoRaWAN.

The main technology characteristics of the reviewed LPWAN technologies are summarized in the following table [25, 43]. MAC MTU stands for Medium Access Control Maximum Transmission Unit.

Table 3.1: LPWAN Performance Comparison Table

Technology	**Range**	**Throughput**	**MAC MTU**
LoRaWAN	2 – 5 km urban, < 15 km suburban	0.3 to 50 kbps	256 B
SigFox	< 10 km urban, < 50 km suburban	100 bps	Fixed 12 B
IngenuRPMA	20 – 65 km	up: 624 kbps, down: 156 kbps	64 B
Weightless-N	< 5 km urban, < 30 km suburban	30 kbps to 100 kbps	max. 20 B
LTE-M	< 12 km	up: < 1 Mbps, down: < 1 Mbps	1500 B
NB-IoT	< 15 km	200 kbps	1600 B

3.4 Simulators of LoRaWAN

In recent years, various simulators have been developed which allow for simulating LoRa-oriented scenarios. The majority of them base on common C++ network simulation frameworks such as Network Simulator Version 3 (NS-3) [3] or OMNeT++ [13]. This section covers a number of these implementations.

Basing on the OMNeT++ network simulator, FLoRa [2] is a framework capable of simulating end-to-end communication in LoRa networks. It divides the domain of LoRaWAN networks into the following modules (*i.e.*, device classes): Nodes (LoRa end devices), gateways as well as network server. In FLoRa, it is possible to connect application logic modules independently with the simulated network server. By making use of applying adaptive data rate technology, FLoRa is capable of dynamically managing simulation parameters. In addition, FLoRa provides a feature to collect energy consumption data in each individual simulated end device. [2].

Magrin et al. [38, 39] assessed the performance of LoRaWAN. They implemented a C++ NS-3 module to simulate the whole LoRaWAN network consisting of tens of thousands of end devices. Their link model is based on the underlying sub models:

1. Link Measurement Model estimating the signal strength at the receiver site
2. Building Penetration Loss Model modelling the losses caused by external as well as internal walls of buildings
3. Correlated Shadowing modelling fading of the signal with various variables, *e.g.*, time, geographical position as well as radio frequency
4. Link Performance Model modelling the reception sensitivity and signal to interference ratio taking into account partial orthogonality of spreading codes used for encoding the signal with different SFs

Van den Abeele et al. provide modelling of LoRaWAN networks in NS-3, which consists of a number of different elements (*i.e.*, Error Correction Encoder/Decoder, Digital

Interleaver/Deinterleaver, Data Whitening/De-Whitening, Gray Encoder/Decoder, LoRa Modulator/De-Modulator as well as Additive White Gaussian Noise channel): Firstly, an error model for the LoRa modulation was implemented in NS-3 based on base band simulations of a LoRa transceiver over an additive white Gaussian noise channel. Secondly, the LoRaWAN physical OSI layer (PHY) and Medium Access Control OSI (MAC) layers were added in NS-3 to represent LoRaWAN gateways and simple class A end devices [41]. Thirdly, NS-3 applications were developed to represent the behaviour of class A end devices and gateways. Finally, a simple Network Server (NS) was added to NS-3 [30, 31].

T-H. To et al. [45, 46] present an NS-3 module which simulates the behavior of LoRaWAN in an accurate way. To assess the module, they compared the simulation results with measurements on a real-world testbed and measured values reported by the work by Haxhibeqiri et al. [35]. The model description is not extensively presented in the paper, however, it is demonstrated that the module correctly represents the capture effect lowering the packet loss ratio due to collision. The simulation of the capturing effect with orthogonal spreading factors is — however — unclear. To estimate energy consumption of battery powered end devices, the energy framework by Wu et al. [47] is used, which is included in NS-3.

3.5 Evaluation of NS-3 LoRa Simulation Modules

In this chapter, the result of an evaluation conducted for a subset of LoRaWAN simulation projects introduced in section 3.4 is presented. For keeping the scope of work reasonable, only NS-3 based simulation projects were considered. To come to the choice of the simulation module on which all further work concerning LoRaWAN simulation and efficiency improvements in the course of this thesis will base, the evaluation worked according to four criteria, which are explained in subsection 3.5.1. The following table gives an overview of the evaluation results:

Table 3.2: Comparison Table of NS-3 Based LoRaWAN Simulation Modules

Module	Usability	Documentation	Implementation	Energy Framework
Magrin et al. [38]	100%	90%	acceptable	available
Abeele et al. [30]	100%	60%	acceptable	not available
Duda & To [45]	70%	20%	unacceptable	not available

3.5.1 Evaluation

Usability

This criterion describes the quality in terms of applicability and flexibility of provided code, *resp.*, simulation scripts. That is, a high usability percentage signifies, that the considered simulation module features concise code with great readability, such that it is expected to be well adaptable to the parameters which will be provided for the simulation scenarios conducted for this work.

Magrin et al. [38,39] provide a series of useful, well-structured simulation example scripts, which are all thoroughly commented. It is estimated, that this precondition will prove itself to be very helpful, when it comes to implant custom simulation parameters (*e.g.,* geographical positions of LoRa end devices) into the provided example scripts. The example scripts by Abeele et al. [30, 31] are similarly well-structured, however, their in-code documentation is notably scarcer than the annotations in the scripts by Magrin et al. [38,39]. To mention last here is the project by Duda and To [45, 46], which unfortunately does neither provide more than one example script nor feature a good structure, *resp.*, code documentation.

Documentation

The percentages in the column 'Documentation' grade the amount and quality of available documentation about the regarded simulation module (*i.e.*, its source code). The higher this percentage, the better does the documentation give insight about which assumptions the simulation is based on. Clearly, the most advanced and extensive documentation is provided by the module by Magrin et al. [38, 39] and it is to mention, that this documentation also features a section about the model's usage, which is great for reusing the provided simulation examples. The module documentation by Abeele et al. [30, 31] presents a similar conciseness in terms of NS-3 model description, however, there are no usage instructions provided. The simulation module by Duda and To [45, 46] does not deliver more than sparse setup instructions; There is no information given, on how their simulation model is effectively reproduced as an NS-3 module.

Implementation

For this criterion, the actual source code files of all three modules were surveyed and it was decided — based on the perceived maturity of the code — whether it is acceptable to reuse in the simulation part of this thesis. 'Maturity of code' refers to the degree of clearness of the code's structure and whether it appears proper and well-formatted. Especially the former criterion will be important, when it comes to learning about the module's source code to eventually enhance it.

Energy Framework

This criterion indicates, if a simulation project features the possibility to measure energy consumption of LoRa devices. Magrin et al. [38,39] have implemented an energy model as well as a simulation script in which it is possible to enter energy consumption rates during different end devices states (*i.e.,* transmission, reception, standby and sleep modes). The module by Magrin et al. [38,39] is the only one of the selection of three simulation modules to feature an energy framework.

3.5.2 Concluding Choice

Clearly, the simulation project by Magrin et al. [38, 39] is the preferred choice to use as a starting point for this thesis' LoRaWAN simulation. While the project by Duda and To [45,46] features very significant results concerning the enhancement of LoRaWAN networks due to collision reduction using LBT, the module code is unfortunately only very scarcely documented. However, the conference paper provided by Duda and To covering this work [46] contains an exact description of their LBT strategy, which will serve as guidance for implementing this feature in the simulation module by Magrin et al.

Chapter 4

Design and Implementation

4.1 Introduction

This chapter covers the design of an existing simulation framework as well as implemented enhancements towards the simulation of blockchain-compliant LoRaWAN networks. The following illustration gives an overview over the different interconnected domains being part of an architecture connecting LPWAN networks with blockchain databases. Moreover, it displays the different device types as well as interfaces in terms of data flow present in a LoRaWAN network. The simulated LoRaWAN characteristics represented in the used framework are discussed in subsection 4.2.1.

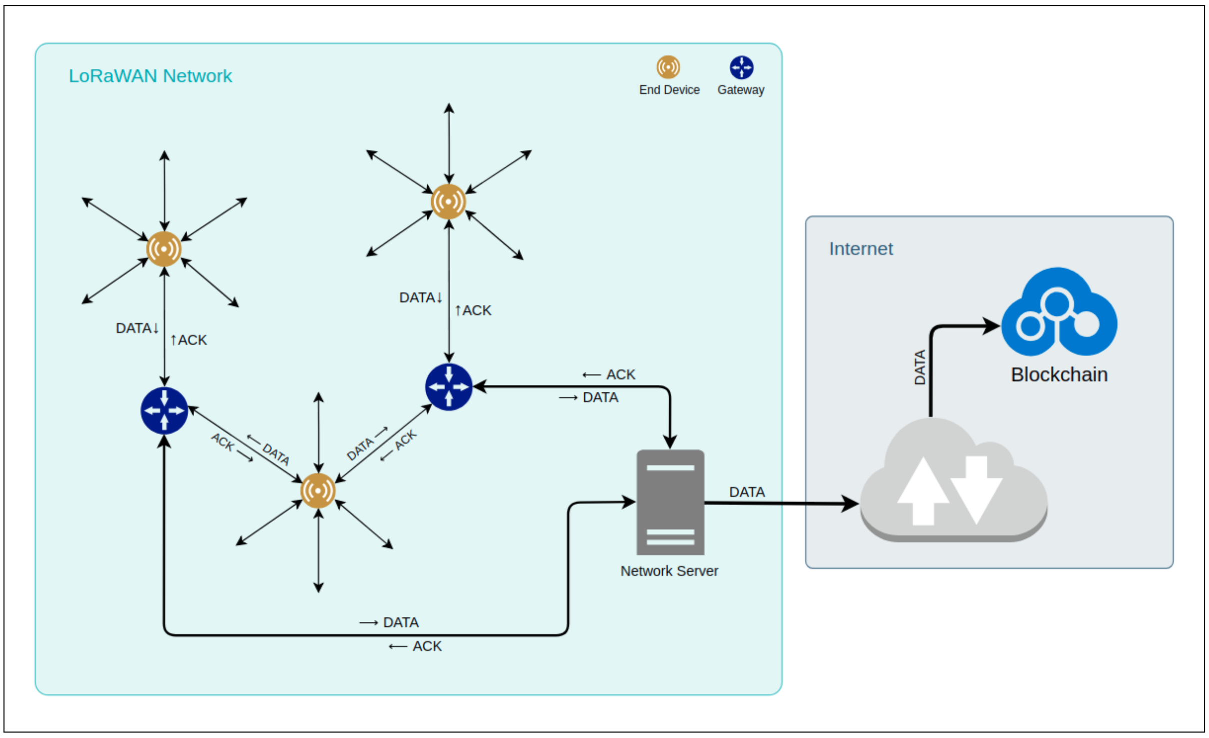

Figure 4.1: Overview of Simulated LoRaWAN Domain

4.2 NS-3 LoRaWAN Simulation Module

As elaborated in subsection 3.5.2, the LoRaWAN simulations conducted in the course of this thesis will be based on the NS-3 module by Magrin et al. [38, 39]. This section gives an overview of the architecture, made assumptions as well as fixed configurations of the considered module.

4.2.1 Module Design

Essentially, the NS-3 module by Magrin et al. depicts three kinds of devices as part of a LoRaWAN network:

End Devices (EDs) adhere to the to the characteristics of basic class A LoRa devices and communicate completely asynchronously, which means that besides conducting transmissions, they can only receive data (*i.e.*, acknowledgements) during two fixed-length receive-windows.

Gateways (GWs) are significantly more sophisticated devices, most prominently in terms of reception ability: By featuring eight parallel receive paths, each gateway can receive up to eight non-interfering packets simultaneously.

Using a robust, high speed link, gateways can forward data to a **Network Server** upon successful packet reception. This third device type's purpose is to keep track of incoming packets and command its affiliated GWs to send acknowledgements (ACK) back to the EDs, should the received packet be marked as to be confirmed. Furthermore, the Network Server can be regarded as the final stage of the LoRaWAN network, where all incoming links are joined and the collected data from within the LoRa network might eventually be forwarded to another network.

Typically for NS-3 modules, the Physical (PHY) as well as the Medium Access Control (MAC) OSI layer of the simulated network are represented as models (classes), which have the responsibility to adhere to the same behaviours as both layers operating in real LoRa networks.

4.2.2 PHY Layer Model

Magrin et al. focus on two main aspects for declaring a signal as being received successfully, or not, namely device sensitivity as well as signal orthogonality:

Device Sensitivity

The model for device sensitivity makes assumptions on how strong incoming signals of certain spreading factors (SF) at receiving gateways have to be at least, such that the signal can still be interpreted correctly. The model uses the following sensitivity threshold matrix [38, 39]:

Table 4.1: Gateway Reception Sensitivity per Spreading Factor of Incoming Signal

Spreading Factor	**Sensitivity [dBm]**
SF 7	-130
SF 8	-132.5
SF 9	-135
SF 10	-137.5
SF 11	-140
SF 12	-142.5

The power of signals being received is computed by using NS-3's built in two propagation models, `PropagationLossModel` as well as `PropagationDelayModel` [10]. For propagation loss, the `LogDistancePropagationLossModel` [7] is chosen, which computes the remaining signal power at a receiving entity by applying a logarithmic formula to the distance between sender and receiver. Magrin et al. set the following parameters required for this model's formula, which are directly re-used for the simulations conducted for this thesis:

- Path loss distance exponent = 3.76
- Reference distance = 1 m
- Path loss at reference distance = 7.7 dB

For propagation delay, the `ConstantSpeedPropagationDelayModel` [6] is used. In this model, each signal spreads out at constant speed, while the delay between signal emission *resp.*, reception is computed using the direct distance between sender and receiver.

Signal Orthogonality

Not only is the propagation loss due to the distance between sending and receiving antenna a critical factor for reception success but also the effects of interference caused by simultaneously present signals in the area of the receiving entity. To determine, whether a signal being received is interfered by another signal on the channel, Magrin et al. consider the following matrix giving the minimum signal to interference Ratio thresholds deciding over successful reception, *resp.*, signal destruction due to interference [34, 39].

$[dB]$	SF7	SF8	SF9	SF10	SF11	SF12
SF7	6	-16	-18	-19	-19	-20
SF8	-24	6	-20	-22	-22	-22
SF9	-27	-27	6	-23	-25	-25
SF10	-30	-30	-30	6	-26	-28
SF11	-33	-33	-33	-33	6	-29
SF12	-36	-36	-36	-36	-36	6

The rows represent the SF of the aimed-at signal, while columns stand for current interfering signals on a certain SF. For a considered SF combination, the ratio of the energy (= reception power $*$ signal duration) of the signal, to which the receiver is currently tuned into, against the aggregated energy of all interferers on some SF has to be above the threshold, such that the desired signal still wins against its interferers.

Using a popular NS-3 programming paradigm — Callbacks [8] — each gateway's PHY layer immediately emits the occurrence of each unsuccessful packet reception. These packet reception failures are categorized into the following causes, such that a global overview on the different issues at the PHY layers can be produced:

- No. of packets lost due to interference
- No. of packets lost due to reception under sensitivity
- No. of packets lost because no more receivers available
- No. of packets lost because receiving gateway was transmitting during packet arrival

4.2.3 MAC Layer Model

The medium access control layer modeled in the simulation module by Magrin et al. reproduces the LoRaWAN standard according to prevalent duty cycle regulations in the EU: As standard LoRaWAN emits signals in license-free frequency bands without performing LBT, the transmission periodicity of individual devices has to adhere to the maximum duty cycles for the current sub band which is defined by ETSI. The following tables show the ETSI sub bands regulations as well as the implemented channels to operate within the sub bands [38]:

Table 4.2: Prevalent ETSI Sub Band Regulations

Frequency Range [MHz]	**Max. Duty Cycle**	**Max. TX Power**
868 – 868.6	1%	14 dBm
868.7 – 869.2	0.1%	14 dBm
869.4 – 869.65	10%	27 dBm

Magrin et al. defined three different frequency channels to transmit signals on: 868.1 MHz, 868.3 MHz as well as 868.5 MHz. They all belong to the 868 — 868.6 MHz sub band, thus they have to adhere to the 1% duty cycle regulation. If an end device wants to transmit a packet, it chooses randomly one of these three channels. The following formula is then used to compute the time the transmission will be postponed until it actually may be started, such that the max. duty cycle can be complied with [38]:

$$t_{\text{off}} = \frac{t_{\text{air}}}{\text{dc}} - t_{\text{air}}$$

The time the end device is enforced to wait for to transmit is denoted t_{off}, while t_{air} is the air time needed for the planned transmission and dc is the duty cycle, *e.g.*, 1%.

Furthermore, the MAC layer manages the re-transmission of packets in case end devices were not yet able to receive any acknowledgement (ACK) for some confirmed packet from any gateway. Similarly as the PHY layer, the MAC layer emits the number of required transmissions for each packet using Callbacks. In general, as shown in Figure 4.1, confirmed message handling is accomplished by the Network Server sending back ACKs via the Gateways to the End Devices as soon as it has received a data packet.

4.3 Module Enhancements

As introduced, this thesis aims at analyzing, how the performance of simulated LoRaWAN reveals itself, if its transmission scheme is adjusted to become blockchain-compliant. This can be achieved by letting end devices send signed multi-packet transactions, which ultimately will be forwarded to a blockchain. The module by Magrin et al. simulates indistinguishable packets, making it impossible to evaluate the throughput in terms of successful transactions per time. This section covers the various implementational advancements, which were required to achieve the desired figures, especially in terms of attainable throughput, *i.e.*, multi-packet transactions per hour. The enhanced NS-3 module source code is available at `www.github.com/timolex/ns-3-dev-with-signetlabdei-lorawan-module` as well as on the attached DVD.

4.3.1 Transmission Scheme

Magrin et al. provide a C++ class for periodically issuing transmissions, which is called `PeriodicSender`. It operates at a settable, steady periodicity and uses a fixed packet size for all transmissions. On the contrary, this work aims at simulating — besides single-packet transmission scenarios — end devices sending signed multi-packet transactions, which is not possible with the provided artifacts of the NS-3 module by Magrin et al.

The following transactional transmission scheme was delivered by Ile Cepilov, who by working with a real-world LoRaWAN testbed investigated, in how to convert larger data records to be sent by end devices into a signed series of packets, namely a transaction. Each transaction consists of 10 data packets as well as of a signature, which is divided into 2 packets. This signature packets contain the hash value of a cryptographic function which is eventually used to validate the integrity of transmitted data. The following scheme shows the structure of a transaction in terms of individual packet sizes:

42 B	**42 B**	**42 B**	**42 B**	**42 B**	**42 B**	**42 B**	**42 B**	**42 B**	**42 B**	34 B	34 B

In reality, this system requires a counter of 2 B per packet allowing to distinguish sub-transaction-packets from each other at their destination, which is the reason, why all packets (also partial signature packets) are simulated with additional + 2 B in size. Thus, the remaining payload per data packet is 40 B, resulting in a total data payload of $10 * 40\text{ B} = 400\text{ B}$ per transaction.

Along the lines of the module's `PeriodicSender` class, the class `TransactionalSender` was developed taking the following additional parameters into account:

- `dataPacketSize`: The size in B of data packets
- `sigPartPktSize`: The size in B of partial signature packets
- `intraTransactionDelay`: The time delay between issuing the transmission of two consecutive packets
- `interTransactionDelay`: The time delay between completion of an old, and beginning of a consecutive transaction

4.3.2 Transaction Success Assessment

Packet Distinguishability

As mentioned before, the simulated packets in the plain module by Magrin et al. are anonymous, meaning that they do not carry along any information about *e.g.*, their source in the payload. This circumstance was changed by serializing the following fields into each payload making each packet distinguishable after source, transaction as well as packet itself:

- `node_uid`: The unique ID of the packet's source (end device)
- `packet_id`: The ID of the current packet
- `transaction_id`: The ID of the transaction to which the current packet belongs to (only used for simulation of end devices transmitting multi-packet transactions)

The serialization of these data into packets' payloads was accomplished by implementing two classes, *i.e.*, `TransactionalPacketHeader` as well as `PeriodicPacketHeader` (without transaction-aspect), which both inherit from `Header` [9] provided by the standard NS-3 library.

The handling of packet sizes was solved in such that after adding the respective `Header`-classes' serialized fields to the packet payload, the remaining payload is filled with zero-bits until the packet size matches the pre-defined requirement (*e.g.*, 34 B per signature packet, as described in subsection 4.3.1).

Transaction Completeness Analysis

In a discrete event network simulation environment, there is no necessity to verify the intactness of transmitted packet content; As soon as at least one receiving entity computed a packet reception as successful, it can be assumed, that the data would also have arrived intact in the real world case. This is also true for the simulations featuring multi-packet transactions which are investigated in this work: The additional counter-related payload (2 B) as well as the two-part signature are only simulated for the sake of taking into account all physical preconditions which are necessary to comply with a transactional transmission scheme in reality.

A multi-packet transaction is declared as having passed the simulated LoRaWAN network successfully if (and only if) all of its associated packets were fully received by at least one (arbitrary) gateway. The class modelling the device type Network Server (introduced in subsection 4.2.1) was chosen to be augmented with the feature of transaction success analysis, as it represents a stage in the simulated network where all gateways' ends meet.

For each unique ID of an End Device (packet source) there are two separate data structures (associative containers) in the `NetworkServer` class — one for registering successful and another for recording unsuccessful transactional packet receptions. As soon as some gateway's PHY layer calls back either the successful, or, the unsuccessful reception of a packet transmission, it is registered in the respective container based on the IDs (`node_uid, packet_id, transaction_id`) gathered by de-serializing the packet's `Header`-fields. It should be mentioned that due to the fact, that gateways do not have any knowledge of each other, it appears the case in which *e.g.*, there are several gateways calling back the unsuccessful reception of the same packet despite there was already a gateway, which called back this (same) packet's reception as successful. Also, due to the associative nature of the chosen C++ data structures, multiple registrations of the same packet-IDs (and the same outcome – *i.e.*, successful or unsuccessful) are idempotent.

At the end of simulations, for each packet source (end device), the containers for successful, *resp.*, unsuccessful transactional packet registrations are analyzed: If (and only if) for some transaction ID all required packet IDs are present in the container for successful packets, the transaction counts to the global no. of successful transactions. Immediately, this transaction is erased from the container of unsuccessful packets to avoid false negatives. All remaining transactions in the container for unsuccessful packets directly increase the global no. of unsuccessful transactions.

The C++ code on the following page was developed to be used in class `NetworkServer` to determine the no. of successful, *resp.*, unsuccessful transactions at the end of every simulation using either `scenario3` or `scenario4`:

The variables `m_successfulTransactionalPackets` as well as the equivalent for unsuccessful transactional packets (`m_unsuccessfulTransactionalPackets`) are associative containers of the following type: `std::map<int, std::map<int, std::set<int>>>`, a map indexed after packet sources (`node_uid`) containing as values maps which are organized after transactions (`transaction_id`) holding sets of already registered packets (`packet_id`) as value.

```
/**
* Computing the number of entirely received transactions
* (all packets + signature packets received)
*/
int successfulTransactions = 0;
int unsuccessfulTransactions = 0;
// Iteration over all end devices in m_successfulTransactionalPackets
for (auto edIte = m_successfulTransactionalPackets.begin ();
     edIte != m_successfulTransactionalPackets.end (); ++edIte)
{
  for (auto transIte = edIte->second.begin ();
         transIte != edIte->second.end (); ++transIte)
  {
    /**
    * Counting this transaction to the no. of successful transactions,
    * if its size fulfils the requirement
    */
    if (transIte->second.size () ==
         (unsigned int) m_numberOfPacketsPerTransaction)
    {
      NS_LOG_DEBUG ("Successful transaction nr.: " << transIte->first <<
       ", size: " << transIte->second.size () << ", node :" << edIte->first);
      ++successfulTransactions;

      /**
      * Removing the transaction from the m_unsuccessfulTransactionalPackets
      * map, if present there
      */
      auto edIte2 = m_unsuccessfulTransactionalPackets.find (edIte->first);
      if (edIte2 != m_unsuccessfulTransactionalPackets.end ())
      {
        edIte2->second.erase (transIte->first);
      }
    }
  }

  auto edIte2 = m_unsuccessfulTransactionalPackets.find (edIte->first);
  // Checking, if there is a transaction map for the current node
  if (edIte2 != m_unsuccessfulTransactionalPackets.end ())
  {
    for (auto transIte2 = edIte2->second.begin ();
           transIte2 != edIte2->second.end (); ++transIte2)
    {
      NS_LOG_DEBUG ("Unsuccessful transaction nr.: " << transIte2->first <<
       ", size: " << transIte2->second.size () << ", node :" << edIte2->first);
      ++unsuccessfulTransactions;
    }
  }
}
```

Single-Packet Transmissions For simulating the periodic transmission of single, unsigned packets (for certain scenarios introduced in subsection 4.4.2), the process of success analysis is somewhat simplified: For each end device, successful *resp.*, unsuccessful packet receptions are registered into associative containers organized after packet sources (end devices). After simulations have ended, the size of the container holding successful transmissions is directly considered as the no. of successful transactions for the currently analyzed end device, while an entry in the container for unsuccessful transmissions only counts to the no. of unsuccessful transactions, if the same packet ID does not appear in the container holding successful transmissions.

Transaction Failure Due To Short Simulation Time It appears the case that due to the hard defined end time of NS-3 simulations, ongoing transactions are interrupted for the sole reason of not having had enough time to complete. Since this is a circumstance which would not appear in reality, this effect was leveraged by letting all simulations continue during an extra time window in which every end device is allowed to finish conducting its current transaction (*resp.*, transmission for single-packet transmission scenarios).

4.3.3 Virtual Packet Queue Management

By default, in the NS-3 simulation module by Magrin et al. [38], the process of issuing a new transmission works like follows: A virtual NS-3 sender application [4] is installed on every simulated end device. It is this sender application, which adheres to the transmission periodicities (`inter-/intraTransactionDelay` *resp.*, `interTransmissionDelay`) configured in the simulation scripts. At this predefined pace, the sender applications call the end devices' MAC layer to start the transmission of a new, *i.e.*, the next packet. In some cases, during the time the next command by the sender application arrives, a packet is still being held back by the ED MAC layer to not violate the max. duty cycle, or, to eventually re-transmit, if the MAC has not yet received any acknowledgement. In such a case, the module by Magrin et al. simply cancels the ongoing transmission of the current packet and gives the new packet incoming from the sender application precedence. This practice revealed itself very disadvantageous for scenarios using high transmission frequencies; Especially for scenarios using multi-packet transactions, success rates suddenly dropped to 0% at a certain level of transmission frequency: The more often the EDs' MAC layer tries to deliver new packets, the higher becomes the probability, that packets cannot be immediately sent due to the implemented mechanism adhering to max. duty cycles. Because of this fact, held-back packets were regularly dropped, leading to incomplete multi-packet transactions.

In any case, the 'cancelling' habit implemented by Magrin et al. does not properly represent LoRaWAN end devices existing in reality: Rather, real existing LoRaWAN nodes would feature data storage for caching rapidly incoming data packets (*e.g.*, from sensor instruments). Typically, these storages are organized in data structures adhering to the principle of FIFO- (first in, first out) queues: New packets to be transmitted have to 'stand in line' until the MAC layer has finished processing the transmission of the previous packet.

The behaviour of a FIFO-packet-queue was implemented the following way: If the ED MAC layer is called by a sender application to start the transmission of a new packet, it verifies, if there is currently still a packet being processed. If so, it does not proceed to send the new packet but rather calls back (using NS-3 Callback [8]) the sender application: After the sender application was notified about a packet having been rejected, it resets the current IDs (`node_uid, packet_id, transaction_id`), such that after the next transmission delay, the application can re-send a packet with the same IDs as the previously rejected packet. This strategy basically emulates the characteristics of a FIFO-packet-queue, therefor it is referred to as 'Virtual Packet Queue'.

4.3.4 Efficiency Improvement Through Listen Before Talk

It was shortly mentioned in subsection 4.2.3, that LoRa end devices have to obey to the mandatory duty cycle regulations, if they do not assess the channel before sending. Duty cycle enforcement is a simple, but effective means, when it comes to protecting a wireless network from excessive congestion. However, postponing transmissions to comply with max. duty cycles obviously attenuates the up-scaling of simulation scenarios in terms of end devices' transmission frequency; The more frequent a devices attempts to issue data, the longer will it have to postpone each transmission to not violate duty cycle regulations.

However, if end devices spend the additional energy to listen into the channel before transmitting on it (*i.e.*, Listen Before Talk (LBT)), they are exempt from duty cycle commandments. This fact sounds promising, keeping in mind the intention to scale up end devices' transmission frequency. To be able to show the (expected) positive effects of LBT in the course of simulations, the NS-3 module by Magrin et al. [38] was extended by an `LBT-mode`, which — if activated — lets end devices' PHY- and MAC-layer suppress duty cycle enforcement and operate according to LBT. It was decided to implement the scheme presented by To et al. [46], which they refer to as CSMA-x.

Clear Channel Assessment (PHY Layer)

For end devices to be able to determine, whether the channel they plan to transmit on is occupied or not, a means for their PHY layer to listen to ongoing transmissions is required. Besides, it has to be settled, for which time interval the devices should listen. To et al. use a Clear Channel Gap (CCG) of 10 ms during which end devices probe the channel [46]; The 'x' within the name CSMA-x is intended to be set to the duration of the CCG in milliseconds. This listening mechanism for CSMA-10 was reproduced as follows:

In the module by Magrin et al. [38], each `LoraPhy`-object is associated with an instance of `LoraInterferenceHelper`, which holds a list of recent, ongoing transmissions on all channels. In `LBT-mode`, before every transmission this list is iterated over by method `EndDeviceLoraPhy::IsChannelOccupied`: This method immediately stops the list iteration and reports channel occupancy, if for another ongoing transmission, *resp.* its signal, the following three conditions hold:

- The signal has to appear on the same channel on which the ED plans to transmit
- The transmission has to either partially or fully take place during the CCG of 10 ms
- The signal must be detectable over the device's reception sensitivity threshold

The last criterion depends on the detected signal's spreading factor; The following sensitivity threshold matrix is used for all simulated end devices [38]:

Table 4.3: End Device Reception Sensitivity per Spreading Factor of Incoming Signal

Spreading Factor	**Sensitivity [dBm]**
SF 7	-124
SF 8	-127
SF 9	-130
SF 10	-133
SF 11	-135
SF 12	-137

Backoff Strategy (MAC Layer)

While the channel sensing during a CCG is plainly PHY-layer-related, it is the end devices' MAC layer, which is responsible to react correctly to the PHY layer's reported answer concerning channel occupancy: Should the channel be busy, the MAC layer has to stop the current transmission attempt and postpone a next one until the end of a backoff interval. To et al. use a random backoff-interval of k seconds, while the bound of the random number to pick depends on the n^{th} attempt to transmit [46]:

$$k \in [0, 2^{\mathrm{n}} - 1]$$

If after $n = 3$ attempts, the channel is still busy, the transmission is cancelled. The unsuccessful transmission is called back to the Network Server class where the packet associated to this transmission is registered and the "No. of packets dropped due to reaching the maximum allowed transmission attempts when using CSMA-x" is incremented.

The following sequence diagram illustrates the implemented LBT mechanism, which was discussed in the previous sections:

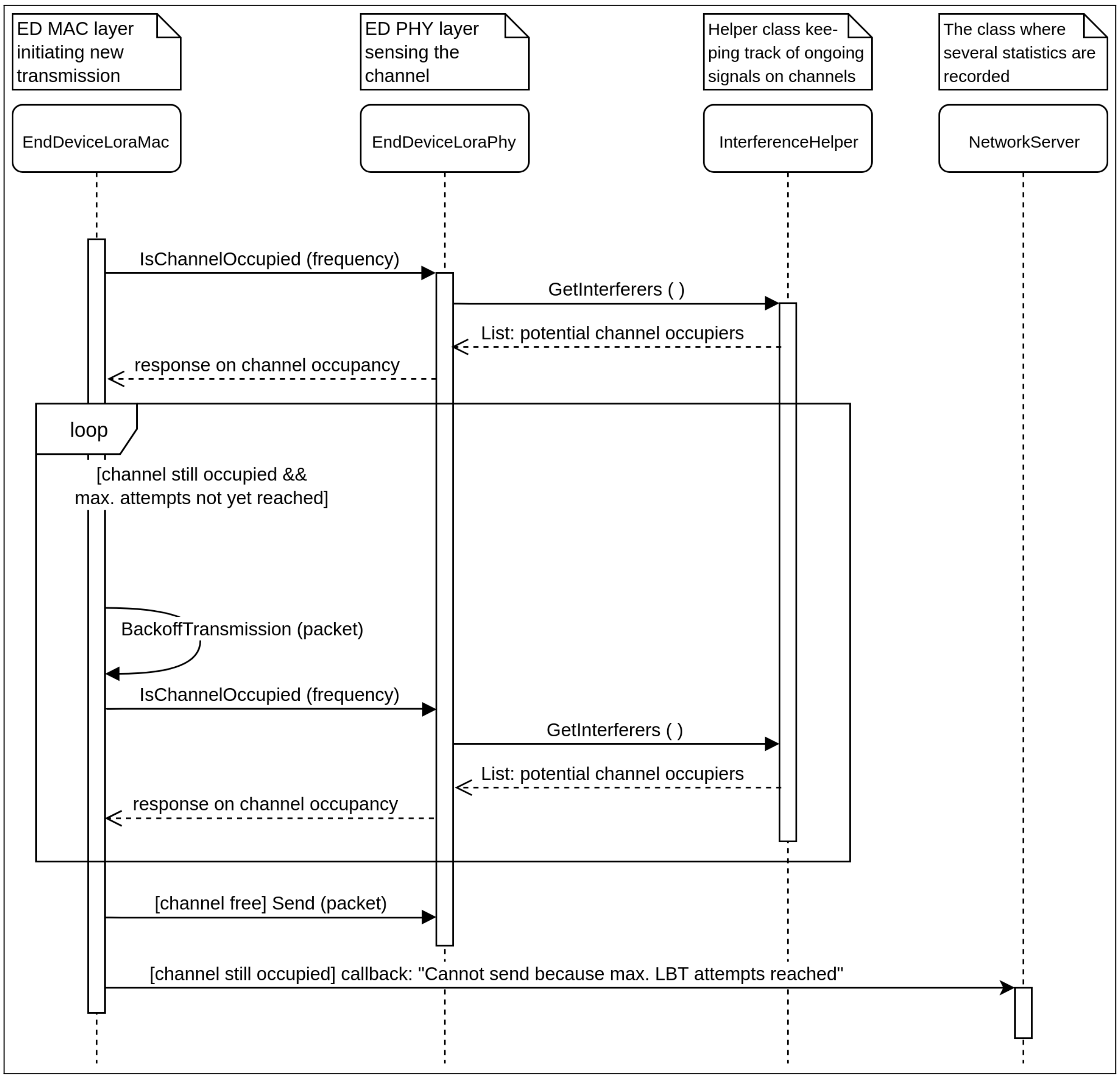
ED MAC layer initiating new transmission
ED PHY layer sensing the channel
Helper class kee-ping track of ongoing signals on channels
The class where several statistics are recorded
EndDeviceLoraMac
EndDeviceLoraPhy
InterferenceHelper
NetworkServer
IsChannelOccupied (frequency)
GetInterferers ( )
List: potential channel occupiers
response on channel occupancy
loop
[channel still occupied && max. attempts not yet reached]
BackoffTransmission (packet)
IsChannelOccupied (frequency)
GetInterferers ( )
List: potential channel occupiers
response on channel occupancy
[channel free] Send (packet)
[channel still occupied] callback: "Cannot send because max. LBT attempts reached"

Figure 4.2: Sequence Diagram for Implemented LBT Mechanism

4.4 Simulation Scenarios

In this section, the four fundamental simulation scenarios, which were used to perform several simulations, are presented. Whereas the four scenarios differ in terms of transmission schemes, they all share the following common parameters:

4.4.1 Common Assumptions

Simulation Time

The total simulation time is set to 30 min for all scenarios. For the conducted simulations, this is enough time for letting each end device transmit at least one multi-packet transaction while keeping the computation time reasonable. It is assumed that choosing longer simulation time is useless, as due to the steady transmission pace of end devices, the same effects in the network would be repeated over and over.

PHY- & MAC-Layer Related Assumptions

For all simulation scenarios — except for the efficiency-improved LBT-scenarios — the preconditions concerning simulated devices' PHY- and MAC-layer models provided by Magrin et al. [38, 39] (subsection 4.2.2, subsection 4.2.3) were left unchanged.

4.4.2 Introduction of the Four Simulation Scenarios

Single-Packet Transaction Scenarios `scenario1` & `scenario2`

Simulation scenarios `scenario1` as well as `scenario2` demonstrate the outcome of letting all end devices transmit unsigned, single-packet transmissions (42 B each). While `scenario1` works with unconfirmed packets, `scenario2` re-transmits packets up to seven times if no acknowledgement was received yet. The parameter to adjust the time interval between two successive single-packet transmissions is called `interTransmissionDelay`.

Multi-Packet Transaction Scenarios `scenario3` & `scenario4`

The scenarios `scenario3` as well as `scenario4` adhere to the transaction scheme described in subsection 4.3.1, letting every end device transmit signed multi-packet transactions. Analogous to the single-packet scenarios, end devices in `scenario4` expect gateways to send back acknowledgements, whereas `scenario3` does not use confirmed packets. The adjustable delay between two successive multi-packet transactions is referred to as `interTransactionDelay`, while with `intraTransactionDelay`, the interval between two consecutive packet transmissions within a transaction can be set. It should be mentioned, that for all multi-packet transaction scenarios simulated in the course of this work, for `interTransactionDelay` usually the same time values as for `intraTransactionDelay` were chosen.

4.5 Collection of Measured Performance Metrics

NS-3 was configured to log performance metrics as well as statistics about executed simulations into console output, which easily can be redirected into text files; Additionally, the module code was enhanced in such that after every simulation, computed statistics are also written into (appended to) comma-separated files (CSV). For each of the four scenarios (subsection 4.4.2), there is a separate CSV file generated to which the results of successively run simulations are appended. For multi-packet transaction scenarios, the result files feature the following fields (marked fields (*) are only relevant for multi-packet transactions (`scenario3`, `scenario4`)):

- Configuration Data
  - `NEndDevices`: No. of simulated EDs
  - `NGateways`: No. of simulated GWs
  - `SimulationTime` in s
  - `InterTransactionDelay`, *resp.*, `InterTransmissionDelay` in s
  - `IntraTransactionDelay` in s *
  - `PacketsPerTransaction`: No. of data packets per multi-packet transaction *
  - `SignaturePacketsPerTransaction`: No. of signature packets per transaction *
  - `DataPacketSize` in B
  - `PartialSignaturePacketSize` in B *
- Result Data
  - `SuccessfulTransactions`, *resp.*, `SuccessfulTransmissions`: No. of successfully received transactions (*resp.*, transmissions)
  - `UnsuccessfulTransactions`, *resp.*, `UnsuccessfulTransmissions`: No. of unsuccessful transactions (*resp.*, transmissions)
  - `SuccessRate`: The ratio of successful transactions against all transactions (*resp.*, transmissions)
  - `Throughput`: The no. of successful transactions (*resp.*, transmissions) achievable per hour

Additionally to the data written into CSV files, the recorded NS-3 console output contains gateway MAC & PHY-layer statistics. Most prominently, these figures represent the different reasons, why gateways called dropped packets back to the NetworkServer; Noteworthy examples for such figures are:

- No. of packets lost due to interference
- No. of packets lost due to reception under sensitivity
- No. of packets lost because no more receivers *(all of the GW's receive paths were occupied)*
- No. of packets lost because GW was transmitting during packet arrival *(only relevant for scenarios using confirmed packets)*

4.6 Bottleneck Identification

For identifying technical limits of different LoRaWAN networks adhering to the scenarios introduced in subsection 4.4.2, several simulations were conducted by constantly increasing the no. of end devices present in the network as well as by decreasing the delay in between letting end devices' sender applications release two consecutive packet transmissions. The results of these simulations are discussed in section 6.1.

4.6.1 Scaling Up Simulation Scenarios

As scaling up all four simulation scenarios by hand would be cumbersome, Bash-scripts allowing for running batches of simulations for all four scenarios were developed. Using two nested for-loops, the simulation programs were scaled up after no. of end devices (`nEndDevices`) for different time intervals (s, `delays`) in between end devices' consecutive packet transmissions. This was accomplished by making advantage of the possibility in NS-3 to run simulations by passing arguments via the command line [5]. The following Bash code is used to run a batch of simulations using `scenario3`. :

```bash
#!/bin/bash
nEndDevices="200 400 600 800 1000 1200 1400 1600"
delays="120 95 65 35 14 9"

for delay in $delays
do
  for number in $nEndDevices
  do
    mkdir -p logs
    ./waf --run "scenario3 --nDevices=$number --intraDelay=$delay
      --interDelay=$delay" > logs/scen3_${delay}_${number}.txt 2>&1
  done
done

echo "Batch simulation for scenario 3 complete."
date +%d.%m", "%X
```

4.6.2 Constant Parameters

The consistent configuration used during scaling up all four scenarios (subsection 4.4.2) is presented in the following table:

Table 4.4: Parameters Used For Bottleneck Identification Simulation Scenarios

Parameter	Value
`SimulationTime`	30 min
`PacketsPerTransaction`	10
`SignaturePacketsPerTransaction`	2
`dataPacketSize`	42 B
`sigPartPktSize`	34 B
`NGateways`	6

4.6.3 Device Distribution

It was decided to keep the number of simulated gateways low, to not end up with excessive computational effort. This decision bases on the following idea: If hypothetically, the no. of end devices and gateways is scaled up equally, such that for each increment, the ratio between EDs to GWs remains similar, it becomes clear that by following such a strategy, the same effects in terms of network exhaustion would be repeated over and over. Therefor, each simulation was conducted using only six gateways; Their geographical positions correspond to real data provided by TTN [22] and are chosen in such a way that each gateway has the smallest Euclidean distance to one of the six regularly transmitting end devices in the Zurich area shown in Fig. 4.5. The positions of six end devices are taken from section 5.3.3 in chapter 5, where the determination of real device positions is thoroughly described. If the no. of simulated end devices exceeds six, additional nodes with randomized positions are added, such that their coordinates differ +/- 1000 m from the six real end devices' coordinates. The following figure shows an up-scaled scenario with 400 end devices in red and gateways in blue (map: © OpenStreetMap contributors [14]):

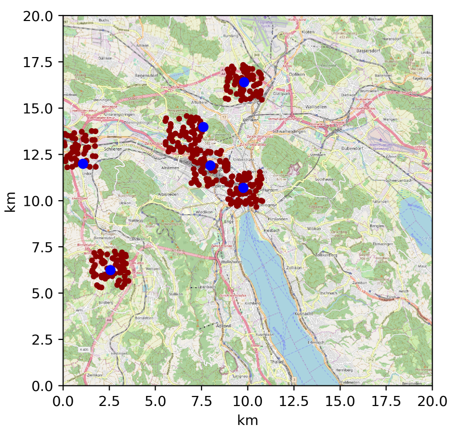

Figure 4.3: 400 EDs Clustered Near 6 TTN GWs in the Zurich Area

4.7 Oil and Gas Supply Chain Simulation Scenario

According to the specifications defined in section 2.3 covering the functional architecture design for the O&G supply chain, an NS-3 simulation scenario was configured. In essence, it corresponds to the same characteristics as `scenario4` introduced in subsection 4.4.2 (confirmed, signed multi-packet transactions). The following table shows the used parameters (according to their definitions in subsection 4.3.1):

Table 4.5: Parameters Used For Functional O&G Supply Chain Architecture Simulation

Parameter	**Value**
`SimulationTime`	12 h
`PacketsPerTransaction`	5
`SignaturePacketsPerTransaction`	2
`dataPacketSize`	42 B
`sigPartPktSize`	34 B
`inter-/intraTransactionDelay` for oilfield EDs	5 min
`inter-/intraTransactionDelay` for pipeline EDs	66 s
`inter-/intraTransactionDelay` for cargo ship EDs	2 min
`inter-/intraTransactionDelay` for end customer EDs	15 min
total no. of end devices	59
total no. of gateways	21

The results of a simulation scenario using the above parameters can be found in section 6.2.

Chapter 5

Traffic Patterns in TTN

5.1 Introduction

During the first third of the processing time of this thesis, it became clear, that working on the simulation part could not be started as early as planned. The reason therefor was, that output from the work by Ile Cepilov — namely the design of a transactional transmission scheme making real-world LoRa devices blockchain-compliant, which was a precondition for configuring the simulation framework — could not be delivered in time. Therefor, it was decided intermediately to invest the time on hold in analyzing traffic patterns of real-world LoRaWAN devices. Due to the waiting time delay as well as the additional effort spent for the implementation covered in this chapter, it was agreed to bypass the milestones concerning experiments and measurements on a real-world LoRaWAN testbed.

As addressed in chapter 1, the simulation scenarios developed in this work should be compatible with the requirement that data produced by end devices will — after being collected at gateways — eventually be sent onto a blockchain. Since one of the fundamental tenets of blockchains is data integrity, it must be ensured, that the communication between end devices and gateways is secure, too and is not prone to any malicious modification. Therefor, at the end devices data records records to send are split into several packets, which all are processed using a cryptographic hash function. By adding the resulting signature (the hash value) to the set of data packets, a signed multi-packet transaction is formed. At the receiving sites (*i.e.*, gateways) the data packets are again processed using the same hash function. This allows for comparing the computed hashes at the gateways with the one sent by an end device, such that the gateways can verify the integrity of received data.

The initial aim for the sub-project documented within this chapter was to gather statistical information about real LoRa devices which currently are following transaction-like transmission schemes.

5.2 TTNMapper Data Set

The freely available database dump by TTNMapper.org [24] revealed itself useful to conduct the task of analyzing a real existing LoRaWAN network for traffic patterns and eventually obtaining statistical information about devices following transactional transmission schemes. TTNMapper is a project providing a global map displaying the coverage of LoRaWAN gateways being part of The Things Network (TTN) [23], a community-based, public LoRaWAN-network. For TTNMapper.org to work, every LoRa device engaging in it has to announce its exact geographical position. Therefor, TTNMapper only covers a subset of the entire TTN, as by far not all TTN end devices reveal their positions.

5.3 Implementation

This section covers the implementation of a Python 3.0 [17] application, which was developed for the purpose of processing the data set provided by TTNMapper.org. This implementation is available on the attached DVD as well as under the URL mentioned in the installation guidelines:

5.3.1 Procedure

The following approach was applied to get at the desired information:

1. Using a Python version [19] of the S2 geometry library [18], a circular area with radius of 10 km around the center of the city of Zurich was defined. This circular shape is used for the first filtering stage based on geographical positions; Only end devices from within this area are considered for further processing.

2. The second filtering stage performs an analysis of each end device's lifespan as well as of their total number of transmissions and removes those, which only existed for less than 24 h or transmitted less than 15 packets. As in the data set being processed, every data point corresponds to a transmission of some end device, these two figures are determined by computing the time difference between a device's first and last transmission as well as by counting the number of data points for each device. It is assumed, that end devices (nodes), which did not pass this filtering stage, were exclusively used for testing purposes and therefor are not interesting for this analysis.

3. At this stage, a distinction between nodes with regular transmission frequencies and nodes, which send irregularly, is carried out. The transmission scheme of irregularly sending nodes can eventually be analyzed and the findings be reused in the simulation, as it is assumed, that these nodes are involved in transactions with their gateways, which is detectable at irregular transmission periodicities. The algorithms used to analyze transmission frequency are presented in subsection 5.3.2.

4. As the TTNMapper.org data set features geographic positions of end devices, it was decided to reuse this data as realistic input for the simulation. Therefor, the sphere-based coordinates (latitude-/ longitude) present in the data set are mapped to NS-3's metric Cartesian coordinate system. This task was accomplished by the Python-version of the PROJ.4 library [15,16], which allows for mapping coordinates from one coordinate system to another. In this context, coordinates of the commonly known sphere-based WGS84 system [21] are mapped to the metric 2D-plane projection CH1903 [20] covering Switzerland and Liechtenstein. This implementation only focuses on devices from within a rather small area — bearing in mind the used map projection. Therefor, the converted coordinates are normalized by defining a new origin point (X, Y) = (0, 0), which is located 10 km to the south and 10 km to the east of the center of the city of Zurich.

5.3.2 Determination of Transmission Frequencies

This section covers the two algorithms (*resp.* methods,) which were developed to distinct regularly transmitting from irregularly transmitting TTNs end devices. To apply conversions from the time- to the frequency-domain, both methods make use of Fast Fourier Transformation (FFT), which was applied using the NumPy package [11, 12] providing functions for numerical computing in Python.

Sine Method

This algorithm is used to determine those end devices, which send at one clearly detectable peak frequency. It is assumed that these devices are serving one application only (*e.g.*, transmission of air humidity measurements every three hours). Also due to a found peak frequency, it can be assumed, that these devices are not involved in transactions with the network, as transactions would yield irregular traffic, for which no such strong major frequency could be found. The algorithm was developed by the author of this thesis and consists of the following phases:

1. For every end device present in the data set, a transmission array is built, which ranges from the timestamp of its first transmission to the timestamp of its last, such that every index of the array corresponds to a second in that time interval. If for a specific second, one or more transmissions are found in the data set, the value at this second's index is set to 1, the values of the remaining indices are set to 0. The following figure shows a plot (of a section) of an end device's transmission array:

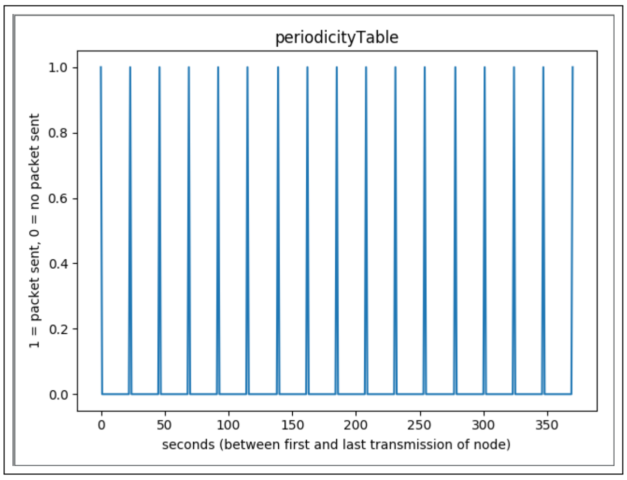

Figure 5.1: Example Plot of an End Device's Transmission Array

2. As sine waves are fundamental signals, which do not show harmonics, they produce proper and clearly detectable representations in frequency analysis such as FFT. This intuitive idea is applied in this context in such that the transmission array is modified by filling in one full period of a sine wave in between the indices of every two subsequent transmissions. This conversion yields the following representation:

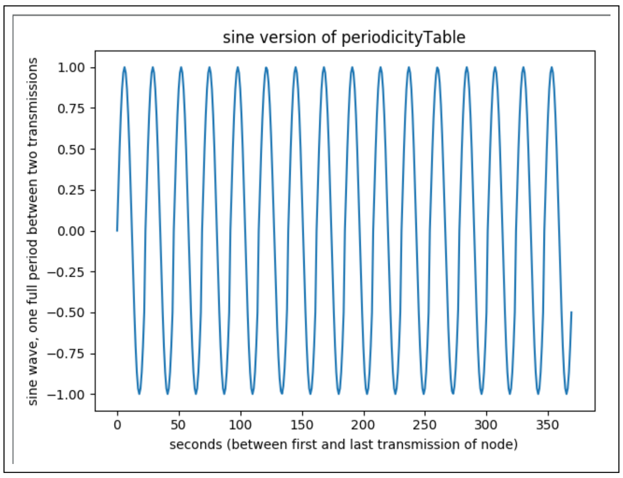

Figure 5.2: Sine Version of an End Device's Transmission Array

3. Lastly, the sine version of the transmission array is transferred into the frequency domain by processing it using Fast Fourier Transformation. Clearly, the plot of the FFT-array shows a strong peak at index 17, which correctly corresponds to a periodicity of approximately 23 s. A frequency is interpreted as peak, if the ratio of its value against the length of the FFT-array is above 50%.

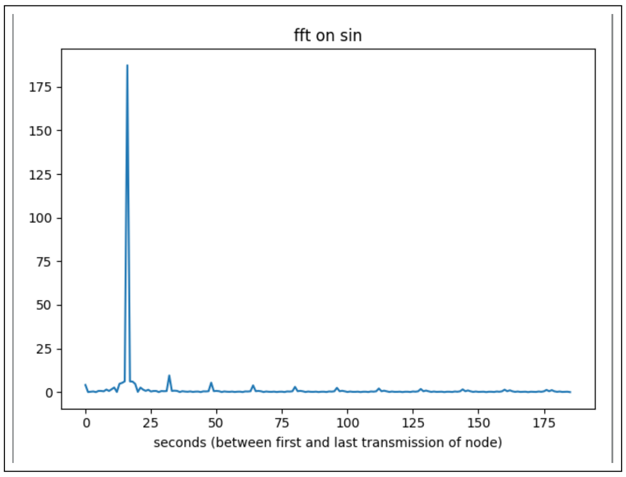

Figure 5.3: Fast Fourier Transformation of a Sine Transmission Array

Direct Method

This method operates on the remainder of the Sine method and is applied to detect nodes with several regular transmission frequencies. The same argumentation as in the Sine method is used here for declaring nodes as not being involved in transactions; End devices showing a set of significant transmission periodicities cannot possibly be following a transaction scheme, as this would result in an incoherent, undetectable set of transmission periodicities. The following algorithm was kindly advised by Dr. Eryk Schiller's colleague at University of Bern, Mostafa Karimzadeh:

1. The same way as for the Sine method, for each node a transmission array is constructed (*cf.* Fig. 4.1).
2. This *direct* method computes FFT directly on the transmission array — hence its name. However, due to the character of this array, the resulting FFT shows an entire 'forest' of harmonics, which makes it difficult to determine the most dominant frequencies.

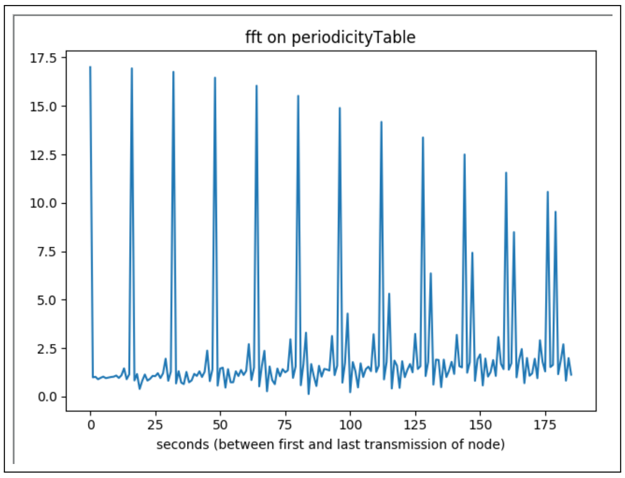

Figure 5.4: Fast Fourier Transformation of a Transmission Array

3. The FFT-array is only analyzed for frequencies within a certain cutoff, which is computed after predefined periodicities, *e.g.*, transmission every 2 h up to every two weeks. The found peak-frequency from within this cutoff is then checked against the transmission array: The transmission array is divided in intervals, such that the number of intervals corresponds to the peak-frequency. In each interval, the occurrence of at least one transmission is verified. End devices only pass this check, if for a considerable percentage (*i.e.*, 99%) of intervals there really appeared transmissions.

5.3.3 Results

Unfortunately, filtering end devices from the data set after the methods presented so far did not yield any leftover, which possibly could be examined further for gathering information about transactional transmission schemes applied in reality. This might be due to the fact that TTNMapper.org only keeps data on LoRaWAN end devices communicating their geographical position, such that only a set of nodes broadcasting data in simple transmission schemes are present for the data points in the considered area.

Nevertheless, this implementation provides value in being able to extract geographical positions of ordinarily operating TTN end devices, which is data serving as realistic input for the LoRaWAN simulation configuration as discussed in subsection 4.6.3. Also, Dr. Eryk Schiller expressed interest for reusing the implementation and results of this subproject in future work.

Results Zurich Area

The next figure shows the geographical positions of TTN end devices, which — according to the previously introduced filtering stages — are transmitting data frequently (map: © OpenStreetMap contributors [14]). The following parameters were used to run the Python application:

Table 5.1: Parameters for Analysis of Node Transmission Periodicity in the Zurich Area

Parameter	**Value**
Radius of circle (cap) around Zurich	10 km
Min. node life span	24 h
Min. no. of packets transmitted	15
Min. FFT-peak against FFT-array-length ratio (Sine Method)	50%
Frequency-cutoff used in Direct Method	2 h - 2 weeks
Min. % of intervals to be checked back (Direct Method)	99%
No. of used data points (transmissions)	2M transmissions

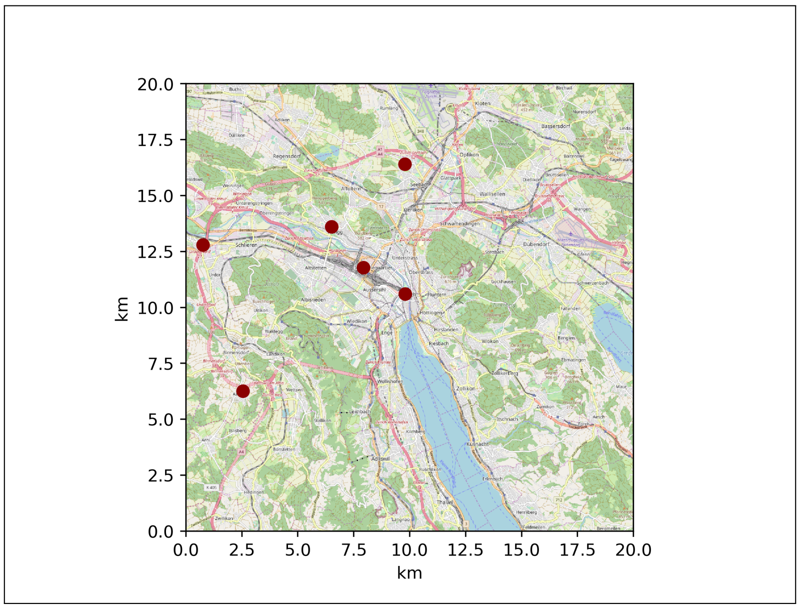

Figure 5.5: Regularly Transmitting TTN Nodes in the Zurich Area

Global Analysis

By removing the geographical restrictions comprising the Zurich area, an analysis over the entire TTNMapper.org data set [24] was conducted. The following histogram shows the distribution of end devices in terms of transmission periodicity. Periodicities range from two hours (at index 0) up to two weeks. The histogram is overlaid with a kernel density curve:

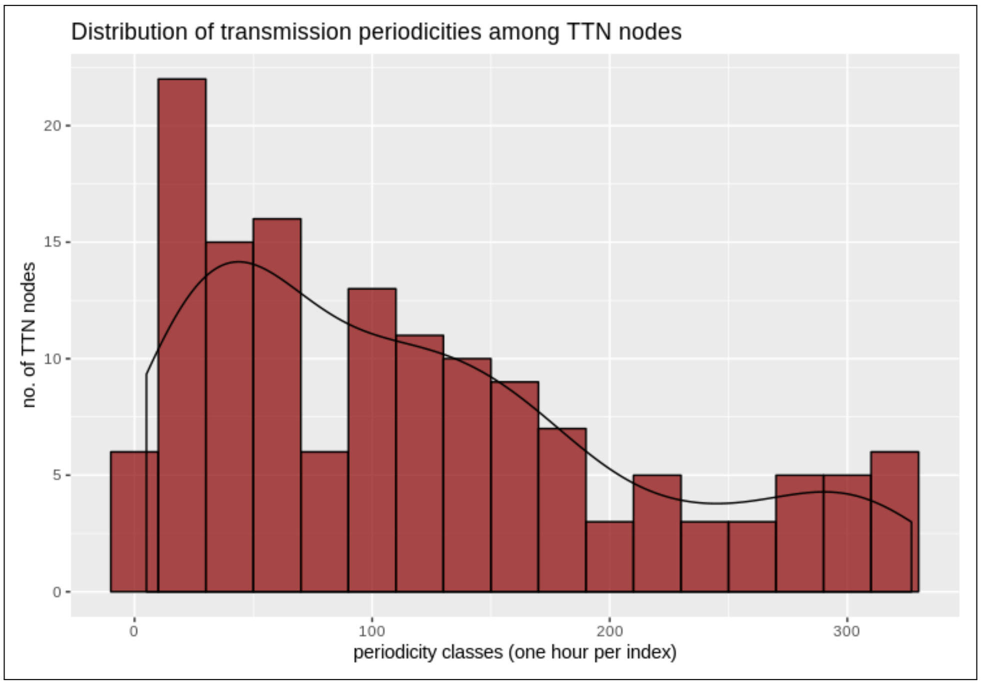

Figure 5.6: Global Transmission Periodicity Histogram

Chapter 6

Simulation Results

This chapter presents and discusses the results for simulations which, were performed after the various scenarios declared in sections 4.6 and 4.7. The full simulation result artifacts in terms of CSV-files as well as log-text files can be found on the attached DVD.

6.1 Results for Bottleneck Identification

This section covers the results for simulations conducted to identify technical network limits as introduced in section 4.6. For both strategies, namely Duty Cycle enforcement (DC), as well as LBT, all four simulation scenarios (subsection 4.4.2) were run for for every combination of the following two parameters to scale up network density and nodes' transmission frequency:

- `interTransactionDelay / interTransmissionDelay`: [120, 95, 65, 35, 14, 9] s
- `NDevices`: [200, 400, 600, 800, 1000, 1200, 1400, 1600]

The thereby generated 48 results per scenario were used to create color maps depicting the development of success rates in terms of intactly transported data packets *resp.*, completely received multi-packet transactions. The y-axis denotes the no. of end devices against the no. of gateways, which is derived by dividing the no. of end devices by the fixed no. of gateways (6). For the x-axis, `interTransmissionDelay`, (*resp.*, `inter-/intraTransactionDelay`) was converted to the no. of issued packet transmissions per end device per hour.

The second plot-type shows for each network density (*i.e.*, no. of end devices per gateway) the transmission frequency in terms of issued packet transmissions per ED and h for which the highest throughput (*i.e.*, single packet transmissions *resp.*, multi-packet transactions per hour) could be achieved. The purpose of these graphics is to demonstrate, that for certain scenarios, the simulated networks are exhausted by scaling up network density such that transmission frequencies need to remain moderately small to retain high throughput.

6.1.1 Duty Cycle Enforcement

Simulation Results `scenario1`

Max. throughput: 178'046 packets per h, attained by 257.15 issued transmissions per h and end device (`interTransmissionDelay` = 14 s), at 266.67 end devices per gateway (`NEndDevices` = 1600)

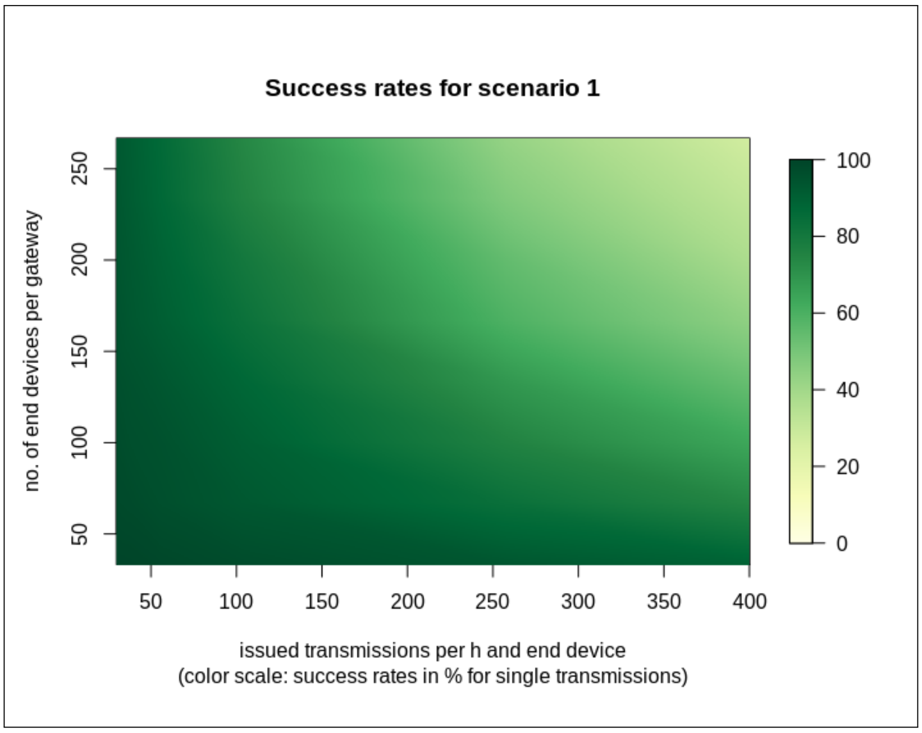

Figure 6.1: Success Rates for `scenario1` using DC

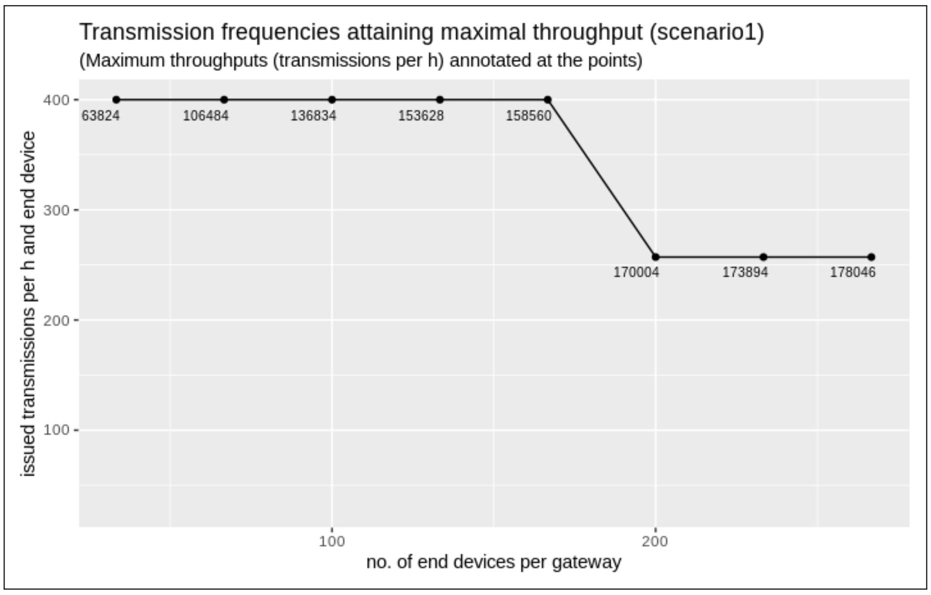

Figure 6.2: Maximal Throughputs for `scenario1` using DC

Simulation Results `scenario2`

Max. throughput: 80'154 packets per h, attained by 400 issued transmissions per h and end device (`interTransmissionDelay` = 9 s), at 266.67 end devices per gateway (`NEndDevices` = 1'600)

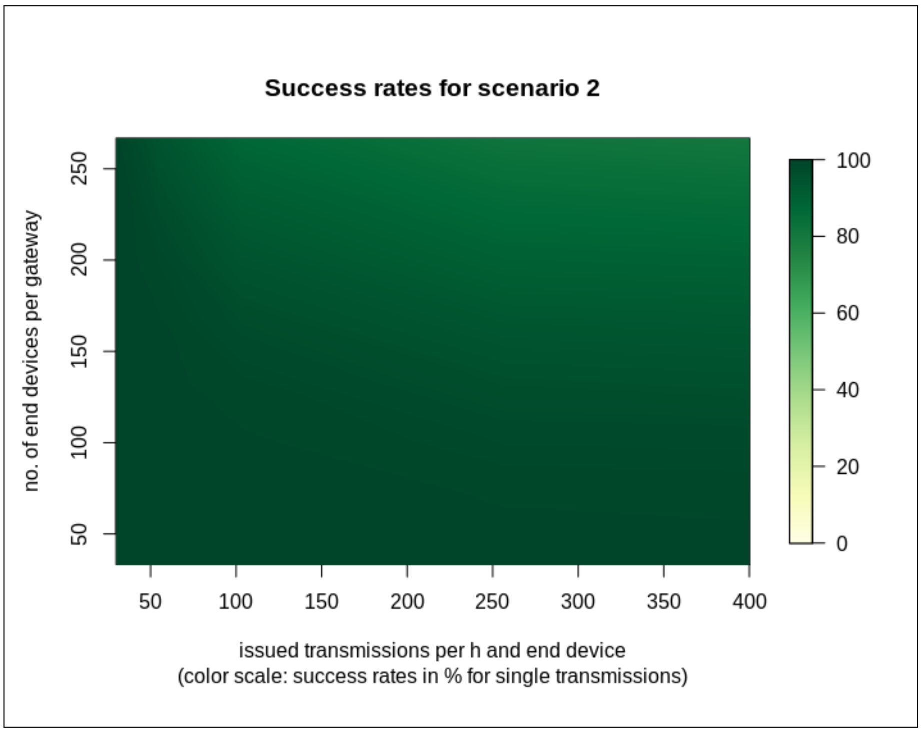

Figure 6.3: Success Rates for `scenario2` using DC

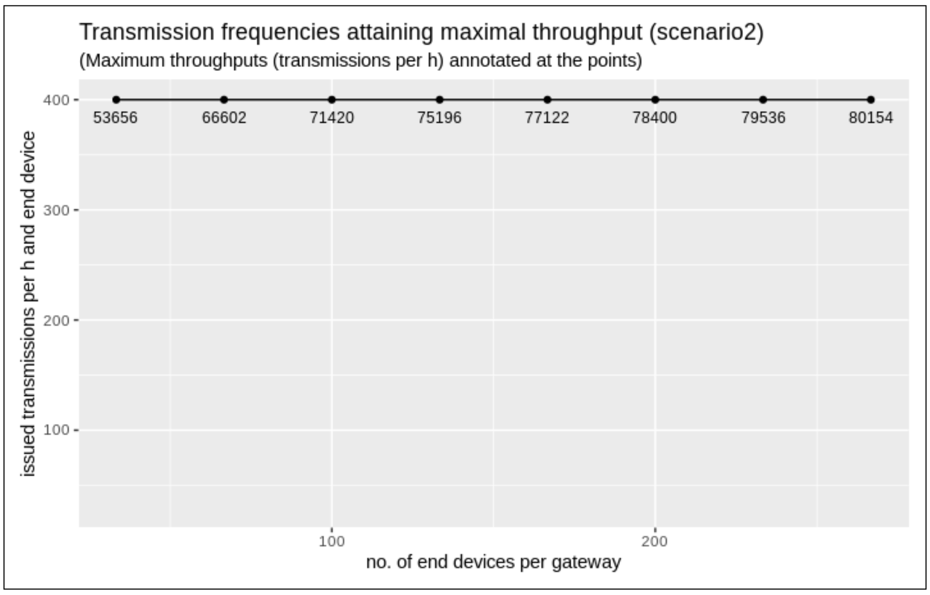

Figure 6.4: Maximal Throughputs for `scenario2` using DC

Simulation Results `scenario3`

Max. throughput: 5'988 signed multi-packet transactions per h, attained by 102.86 issued transmissions per h and end device (`inter-/intraTransactionDelay` = 35 s), at 200 end devices per gateway (`NEndDevices` = 1'200)

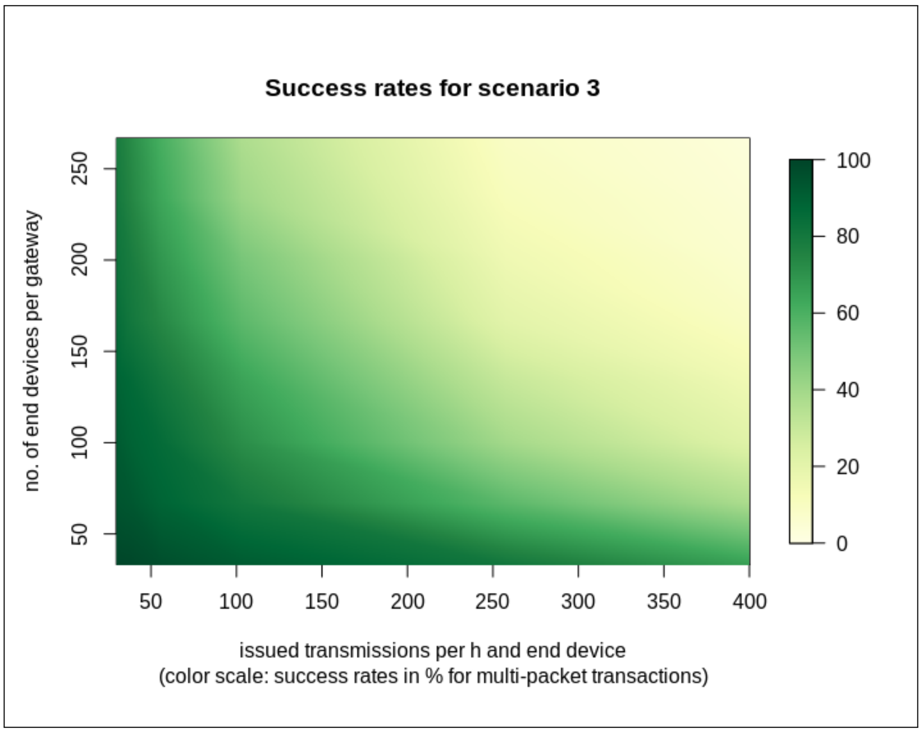

Figure 6.5: Success Rates for `scenario3` using DC

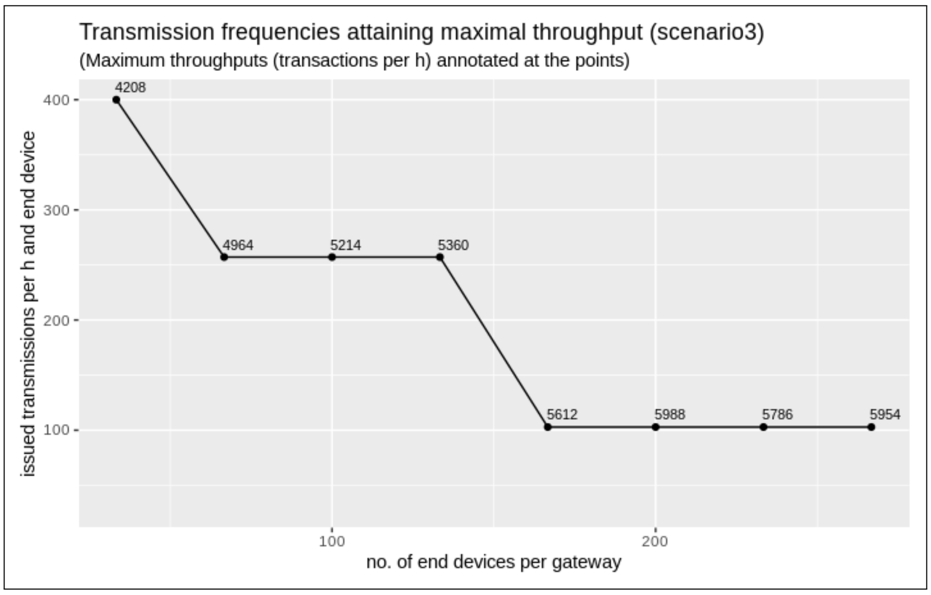

Figure 6.6: Maximal Throughputs for `scenario3` using DC

Simulation Results `scenario4`

Max. throughput: 6'340 signed multi-packet transactions per h, attained by 55.39 issued transmissions per h and end device (`inter-/intraTransactionDelay` = 65 s), at 200 end devices per gateway (`NEndDevices` = 1'200)

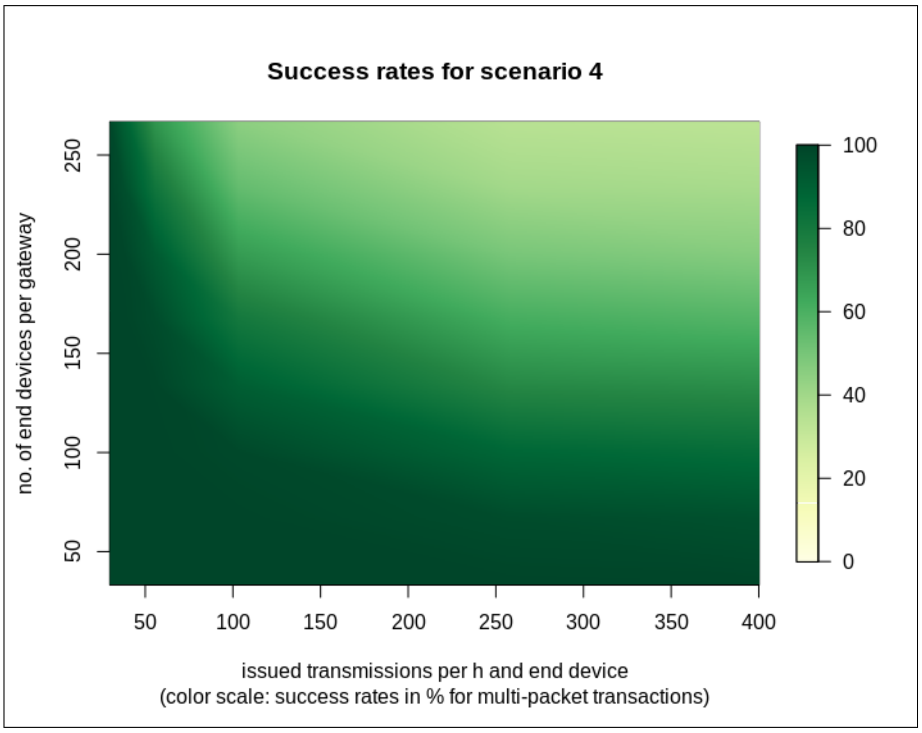

Figure 6.7: Success Rates for `scenario4` using DC

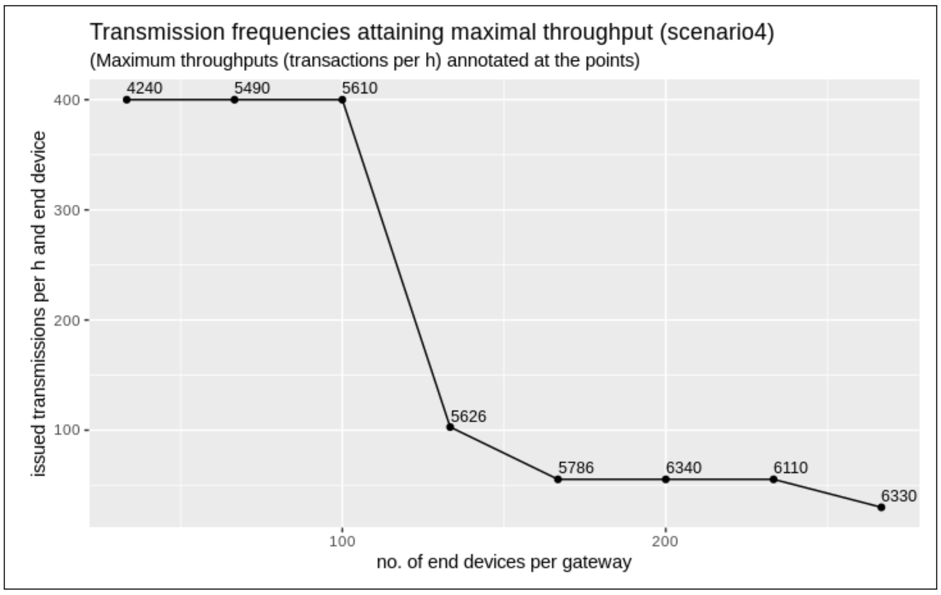

Figure 6.8: Maximal Throughputs for `scenario4` using DC

6.1.2 Listen Before Talk

Simulation Results `scenario1`

Max. throughput: 379’810 packets per h, attained by 400 issued transmissions per h and end device (`interTransmissionDelay` = 9 s), at 266.67 end devices per gateway (`NEndDevices` = 1’600)

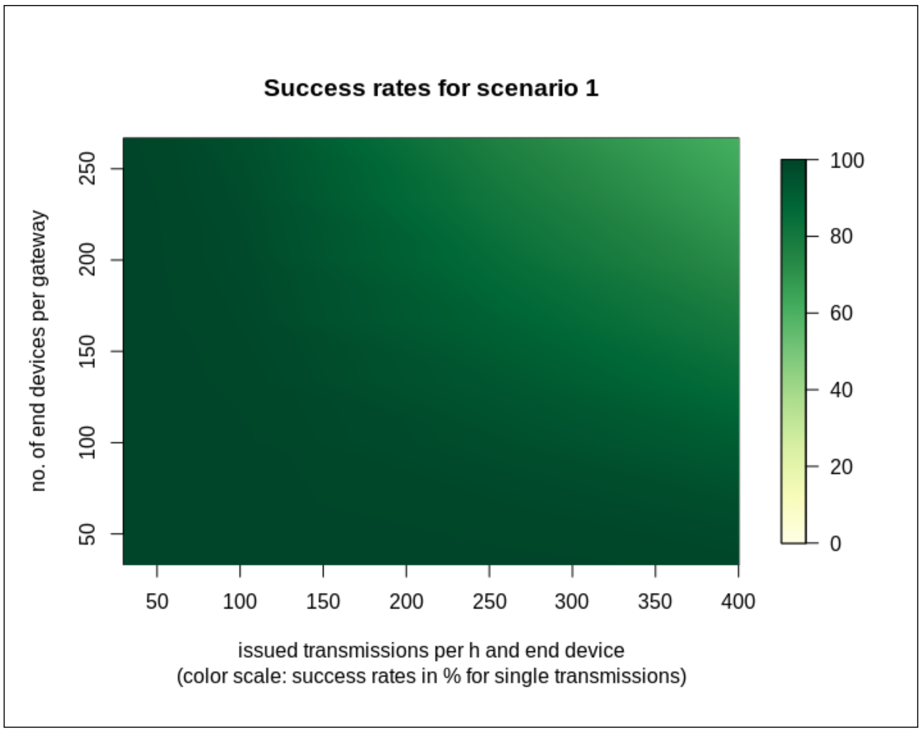

Figure 6.9: Success Rates for `scenario1` using LBT

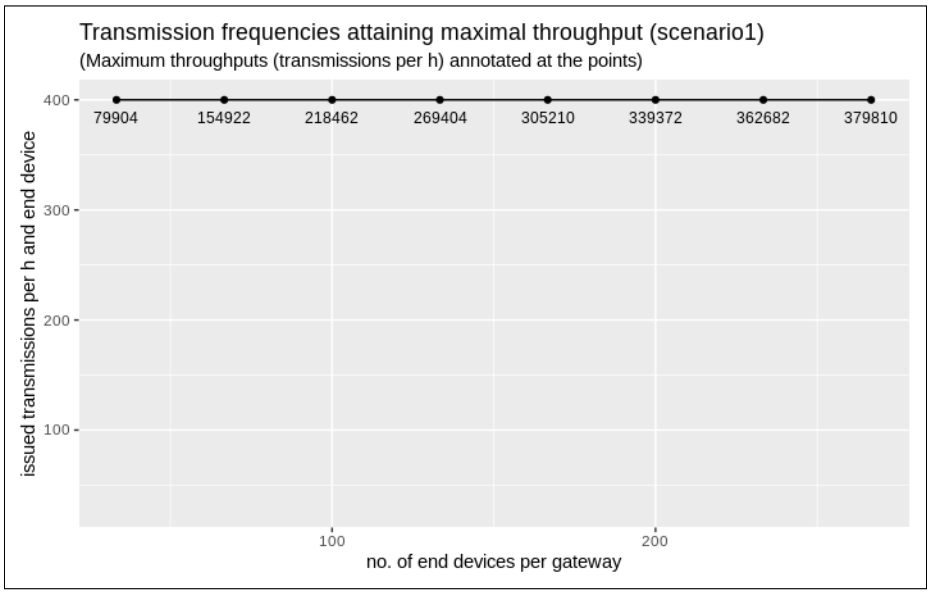

Figure 6.10: Maximal Throughputs for `scenario1` using LBT

Simulation Results `scenario2`

Max. throughput: 63'378 packets per h, attained by 55.39 issued transmissions per h and end device (`interTransmissionDelay` = 65 s), at 266.67 end devices per gateway (`NEndDevices` = 1'600)

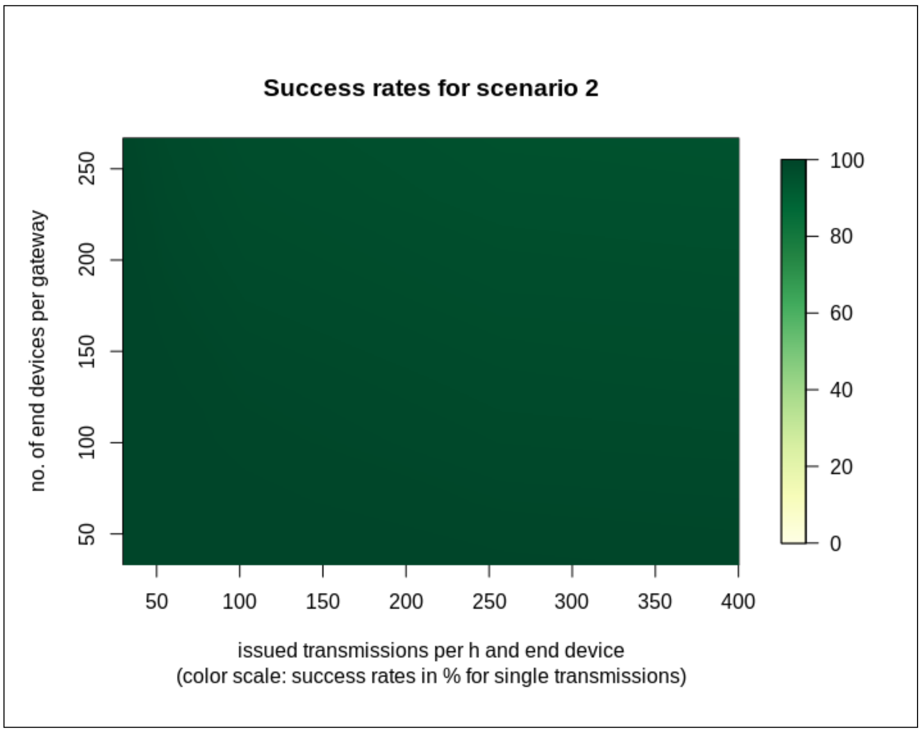

Figure 6.11: Success Rates for `scenario2` using LBT

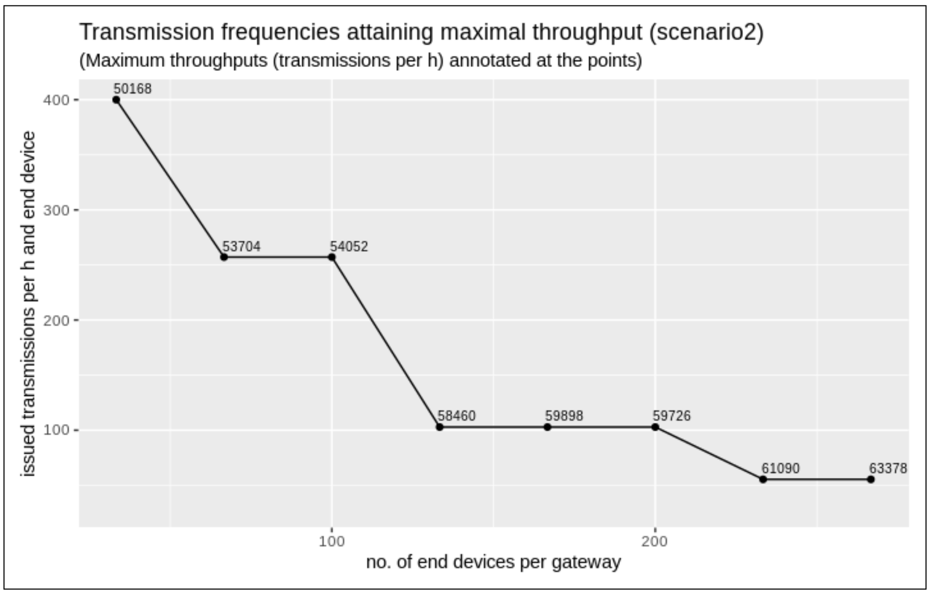

Figure 6.12: Maximal Throughputs for `scenario2` using LBT

Simulation Results `scenario3`

Max. throughput: 11'486 signed multi-packet transactions per h, attained by 102.86 issued transmissions per h and end device (`inter-/intraTransactionDelay` = 35 s), at 266.67 end devices per gateway (`NEndDevices` = 1'600)

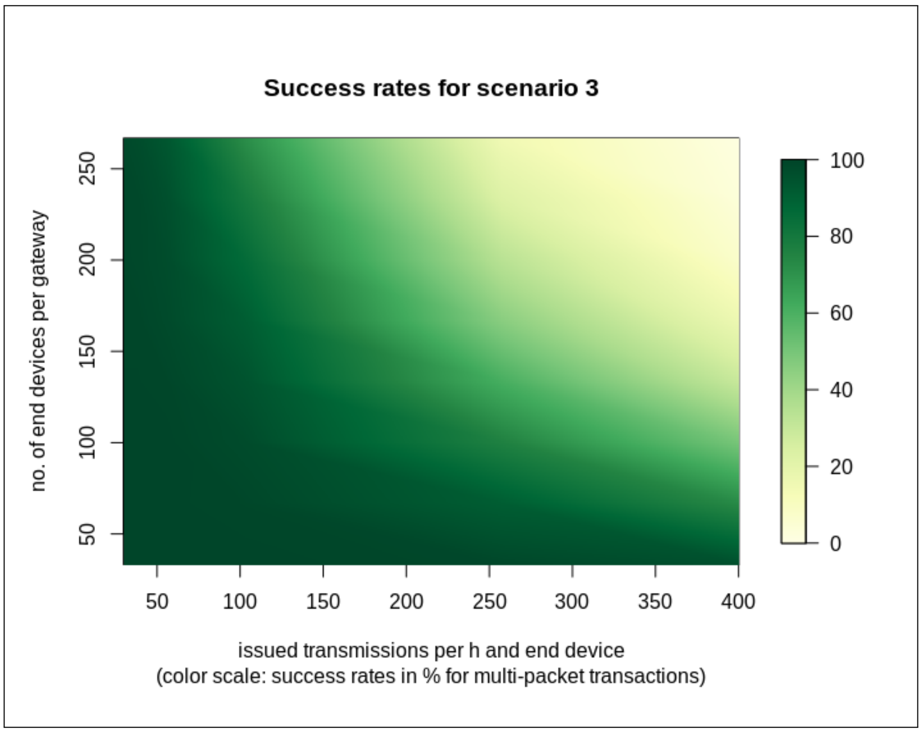

Figure 6.13: Success Rates for `scenario3` using LBT

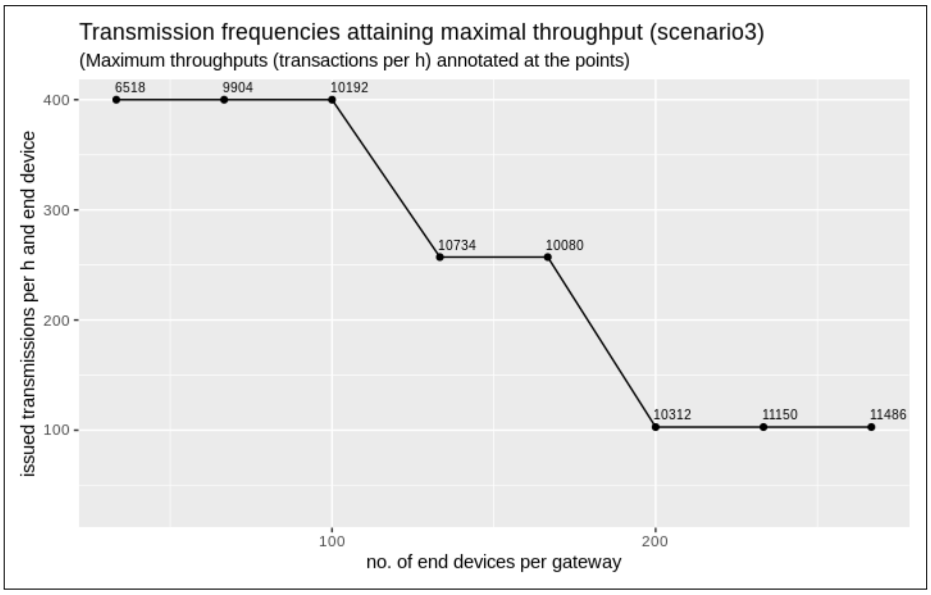

Figure 6.14: Maximal Throughputs for `scenario3` using LBT

Simulation Results `scenario4`

Max. throughput: 6’078 signed multi-packet transactions per h, attained by 55.39 issued transmissions per h and end device (`inter-/intraTransactionDelay` = 65 s), at 266.67 end devices per gateway (`NEndDevices` = 1’600)

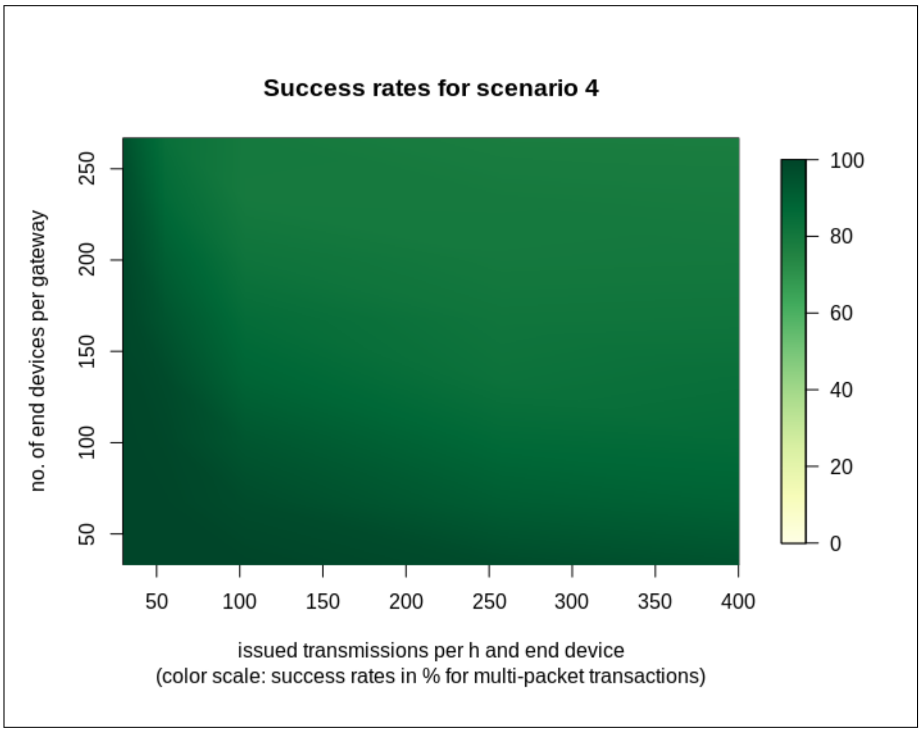

Figure 6.15: Success Rates for `scenario4` using LBT

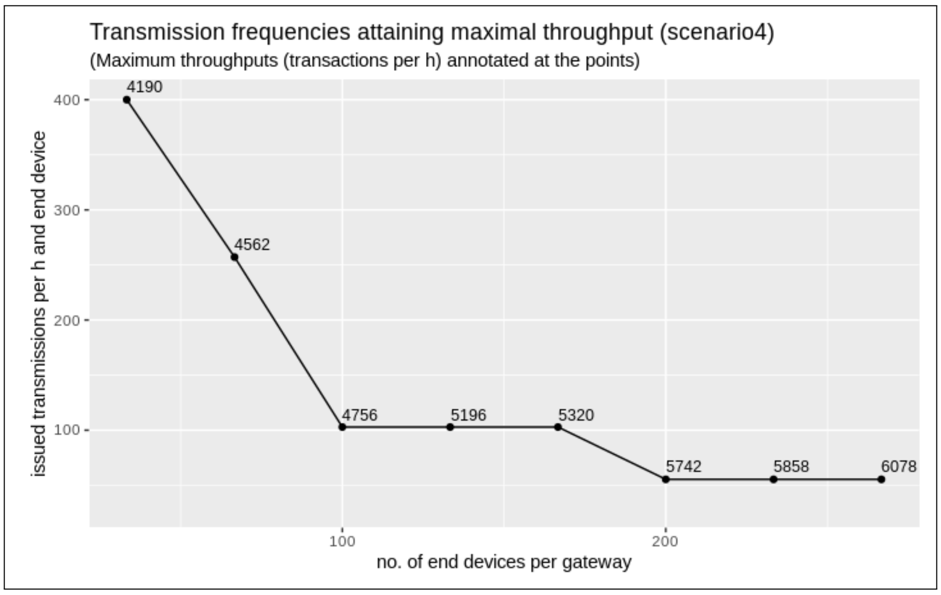

Figure 6.16: Maximal Throughputs for `scenario4` using LBT

6.1.3 Discussion

In general, scenarios simulating the transmission of signed multi-packet transactions scored significantly lower success rates, *resp.*, throughputs, than single-packet transmission scenarios. This is due to the fact, that for multi-packet transactions, all twelve packets belonging to a transaction have to be consecutively received intact, such that a transaction counts as successful, whereas for the other scenarios, a single successful packet transmission increases the success rate.

As expected, implementing and installing the Listen Before Talk variant CSMA-x [46] in the used LoRaWAN simulation framework lead to notably improved performances of networks simulated within the four scenarios. This becomes most apparent by comparing the success rate color maps for `scenario1` (Figure 6.1) and `scenario3` (Figure 6.5) using duty cycle enforcement with the same scenarios using LBT (Figure 6.9, Figure 6.13): Not only success rates but also throughputs in terms of single-packet transmissions, *resp.*, signed multi-packet transactions per hour are improved; Clearly, in `scenario1`, the duty cycle prevention mechanism begins to attenuate throughput from network density of 200 EDs/GW and upwards (Figure 6.2), which is not the case for the LBT-variant in Figure 6.10.

For `scenario2` using confirmed messages, introducing LBT and suppressing the max. duty cycle enforcement did not bring any increase in terms of throughput. This fact can be ascribed to the following effect: In DC, end devices only consider their own transmission behaviour when it comes to the decision, whether to send data immediately, or, to postpone to not violate the max. permitted duty cycle; At the same time, the enforced max. duty cycle remains the same, regardless of the actual total channel usage. On the contrary, when using LBT, end devices actually are aware of other ongoing transmissions, such that the channel usage of receivable devices' signals, too, is a respected factor when it comes to prevention of network congestion. Therefor, in confirmed-message scenarios exhibiting elevated channel occupancy due to transmissions of acknowledgements as well as occasional packet re-transmissions, end devices will sense the channels as being occupied more regularly and transmissions need to be backed off more often, leading to lower throughput.

6.2 Results for O&G Supply Chain Scenario

This section presents the result of a simulated LoRaWAN network scenario which was customized for the functional architecture design of chapter 2 by using the parameters displayed in Table 4.5. Running a simulation with duty cycle enforcement, *i.e.*, without conducting Listen Before Talk, by using confirmed and signed multi-packet transactions (`scenario4`) and setting geographical device positions described in section 2.3 yields the following results:

Table 6.1: Obtained Results Functional O&G Supply Chain Architecture Simulation

Figure	**Value**
`SuccessfulTransactions`	2885 transactions
`UnsuccessfulTransactions`	0 transactions
`SuccessRate`	100%
`Throughput`	240.417 transactions per h

The used parameters concerning network density as well as transmission frequency yield an average no. of 2.81 end devices per gateway and 25.14 issued packet transmissions per end device in one hour. Looking at the success rate color map for `scenario4` using duty cycle enforcement (Figure 6.7), these values are related to the region at the very bottom left of the plot. Therefor, the achieved success rate of 100% for this simulation scenario is justifiable.

Chapter 7

Summary and Conclusions

The contributions achieved in this thesis are twofold:

Firstly, to analyze the transmission behaviour of real-existing LoRaWAN devices, a data set covering TTN devices was processed. While the initial goal — namely to determine end devices in the Zurich area transmitting data as a series of packets forming transactions — could not be realized due to the input data at hand, this implementation can be used to determine the geographical position of end devices in a definable area which are transmitting in a regular fashion. Moreover, the ability to draw the distribution of transmission periodicities among end devices, is realized through this implementation.

Secondly, an existing NS-3 LoRaWAN simulation module was enhanced by several components such that the simulation of blockchain-compliant LoRa networks became possible: At first, the code was extended to reach the goal of determining simulated networks' scored success rates as well as attained throughput in terms of successfully transmitted multi-packet transactions. Later, the simulation framework was equipped with the ability for end devices to conduct Listen Before Talk (LBT), *i.e.*, to only transmit, if the channel was sensed clear beforehand. By running four different simulation scenarios and scaling up simulated networks in terms of end device density as well as transmission frequency, the reused code adhering to duty cycle enforcement (DC) was compared against the LBT implementation: While for all four scenarios, LBT introduced significantly improved success rates, throughput was only raised for scenarios using unconfirmed messages. However, if for certain use cases, it is strictly to avoid losing any of the transmitted data, LBT would definitely be the better choice, as it trades throughput for greater transmission success rates.

7.1 Future Work

The implemented and simulated efficiency improvements through Listen Before Talk (subsection 4.3.4) can be described as a distributed solution mitigating the effects of interference. That is, the simulated end devices are given the task to sense the channels for ongoing transmissions and possibly back off in case of a busy channel. However, with this approach, there is by no means a 100% guarantee, that the chosen channel will still be free after an end device eventually started transmitting. It became clearly recognizable in the simulation results, that also by using LBT, at a certain level of transmission frequency *resp.*, network density, the effects of interference have a negative impact on the conducted simulations' success rate. Undoubtedly, packet loss is to avoid at any cost, if there is great emphasis on data completeness as well as integrity in a considered use case (*e.g.*, section 2.2).

For future work, it can be analyzed, how a central controlling entity (*i.e.*, a 'controller') can be engaged to avoid unsuccessful (*resp.*, incomplete) transactions due to packets dropped by interference: Rather than the end devices themselves, only the controller will be permitted to grant individual end devices permission to conduct their subsequent transmission. This way, the controller will be able to leave the channel free for individual packet transmissions to complete while keeping potential interferers on hold.

Bibliography

[1] A Technical Overview of LoRa and LoRaWAN. `https://lora-alliance.org/portals/0/documents/whitepapers/LoRaWAN101.pdf`. Accessed: 2018-11-07.

[2] FLoRa - A Framework for LoRa Simulations. `https://flora.aalto.fi/`. Accessed: 2018-11-20.

[3] NS-3, a Discrete-Event Network Simulator. `https://www.nsnam.org/`. Accessed: 2018-11-20.

[4] NS-3 Applications Documentation. `https://www.nsnam.org/doxygen/group__applications.html#details`. Accessed: 2019-01-22.

[5] NS-3 CommandLine Class Reference Documentation. `https://www.nsnam.org/doxygen/classns3_1_1_command_line.html`. Accessed: 2019-02-25.

[6] NS-3 ConstantSpeedPropagationDelayModel Documentation. `https://www.nsnam.org/docs/models/html/propagation.html#constantspeedpropagationdelaymodel`. Accessed: 2019-01-20.

[7] NS-3 LogDistancePropagationLossModel Documentation. `https://www.nsnam.org/docs/models/html/propagation.html#logdistancepropagationlossmodel`. Accessed: 2019-01-20.

[8] NS-3 Manual: Callbacks. `https://www.nsnam.org/docs/release/3.29/manual/html/callbacks.html`. Accessed: 2019-01-24.

[9] NS-3 Packet Header Class Reference Documentation. `https://www.nsnam.org/doxygen/classns3_1_1_header.html`. Accessed: 2019-01-28.

[10] NS-3 Propagation Model Documentation. `https://www.nsnam.org/docs/models/html/propagation.html`. Accessed: 2019-01-20.

[11] NumPy: Discrete Fourier Transform. `https://docs.scipy.org/doc/numpy-1.15.1/reference/routines.fft.html`. Accessed: 2018-12-17.

[12] NumPy, the Fundamental Package for Scientific Computing with Python. `http://www.numpy.org/`. Accessed: 2018-12-17.

[13] OMNeT++ Discrete Event Simulator. `https://omnetpp.org/`. Accessed: 2018-11-20.

[14] OpenStreetMap. `https://www.openstreetmap.org/copyright`. Accessed: 2019-02-28.

[15] PROJ.4 library: Generic Coordinate Transformation Software. `https://proj4.org/`. Accessed: 2018-12-20.

[16] pyproj: Python Interface to PROJ.4 Library. `https://pypi.org/project/pyproj/`. Accessed: 2018-12-20.

[17] Python 3 Documentation. `https://docs.python.org/3/`. Accessed: 2019-01-29.

[18] S2 Geometry Library. `http://s2geometry.io/`. Accessed: 2018-12-14.

[19] s2sphere: Python Implementation of a Part of the C++ S2 Geometry Library. `https://s2sphere.readthedocs.io/en/latest/`. Accessed: 2018-12-14.

[20] Spatial Reference: The EPSG:21781 Projection (CH1903/LV03). `http://spatialreference.org/ref/epsg/ch1903-lv03/`. Accessed: 2018-12-20.

[21] Spatial Reference: The World Geodetic System 1984. `http://spatialreference.org/ref/epsg/wgs-84/`. Accessed: 2018-12-20.

[22] The Things Network Gateway Data (JSON). `https://www.thethingsnetwork.org/gateway-data`. Accessed: 2019-03-10.

[23] The Things Network (TTN). `https://www.thethingsnetwork.org/`. Accessed: 2018-12-12.

[24] TTNMapper FAQ Page. `http://ttnmapper.org/faq.php`. Accessed: 2018-12-12.

[25] A. Pelov A. Minaburo and L. Toutain. LP-WAN Gap Analysis. `https://tools.ietf.org/html/draft-minaburo-lp-wan-gap-analysis-00`, Feb. 2016. Accessed: 2018-11-28.

[26] Norman Abramson. THE ALOHA SYSTEM: Another Alternative for Computer Communications. In *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*, AFIPS '70 (Fall), pages 281–285, New York, NY, USA, 1970. ACM.

[27] Ferran Adelantado, Xavier Vilajosana, Pere Tuset, Borja Martinez, Joan Melia-Segui, and Thomas Watteyne. Understanding the Limits of LoRaWAN. *IEEE Communications Magazine*, June 2017.

[28] Aloÿs Augustin, Jiazi Yi, Thomas H. Clausen, and William Mark Townsley. A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. In *Sensors*, 2016.

[29] S. Dawaliby, A. Bradai, and Y. Pousset. In Depth Performance Evaluation of LTE-M for M2M Communications. In *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, volume 1, pages 1–8, Oct. 2016.

[30] F. Van den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke. Scalability Analysis of Large-Scale LoRaWAN Networks in NS-3 (Source Code on GitHub. `https://github.com/imec-idlab/ns-3-dev-git/tree/lorawan`. Accessed: 2018-11-22.

[31] F. Van den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke. Scalability Analysis of Large-Scale LoRaWAN Networks in NS-3. *IEEE Internet of Things Journal*, 4(6):2186–2198, Dec. 2017.

[32] Gaurav Dixit. Internet of Things in the Oil and Gas Industry, Feb. 2017.

[33] Marcel Gautschi. KELLER EasyOil® System. `http://www.keller-druck.ch/home_e/painfo_e/berichte_2013_e.asp`, 2013.

[34] Claire Goursaud and Jean-Marie Gorce. Dedicated networks for IoT : PHY / MAC State of the Art and Challenges. *EAI Endorsed Transactions on Internet of Things*, October 2015.

[35] Jetmir Haxhibeqiri, Floris Van Den Abeele, Ingrid Moerman, and Jeroen Hoebeke. LoRa Scalability: A Simulation Model Based on Interference Measurements. In *Sensors*, May 2017.

[36] Maria Hernandez. Connectivity Now and Beyond; Exploring Cat-M1, NB-IoT, and LPWAN Connections. `https://ubidots.com/blog/exploring-cat-m1-nb-iot-lpwan-connections/`. Accessed: 2018-11-28.

[37] Hadi Jamali-Rad, Xander Campman, Ian MacKay, Wim Walk, Mark Beker, Johannes van den Brand, Henk Jan Bulten, and Vincent van Beveren. IoT-Based Wireless Seismic Quality Control. *The Leading Edge*, 37:214–221, Mar. 2018.

[38] D. Magrin, M. Centenaro, and L. Vangelista. Performance Evaluation of LoRa Networks in a Smart City Scenario (Source Code on GitHub). `https://github.com/signetlabdei/lorawan`. Accessed: 2018-11-22.

[39] D. Magrin, M. Centenaro, and L. Vangelista. Performance Evaluation of LoRa Networks in a Smart City Scenario. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–7, May 2017.

[40] É. Morin, M. Maman, R. Guizzetti, and A. Duda. Comparison of the Device Lifetime in Wireless Networks for the Internet of Things. *IEEE Access*, 5:7097–7114, 2017.

[41] T. Eirich T. Kramp O.Hersent N. Sornin, M. Luis. LoRaWAN Specification v1.0.2. `https://lora-alliance.org/resource-hub/lorawantm-specification-v102`. Accessed: 2018-11-05.

[42] Brian Ray. NS-3 Documentation: CommandLine Class Reference. `https://www.link-labs.com/blog/iot-oil-gas-use-cases`, Aug. 2018. Accessed: 2018-11-10.

[43] U. Raza, P. Kulkarni, and M. Sooriyabandara. Low Power Wide Area Networks: An Overview. *IEEE Communications Surveys Tutorials*, 19(2):855–873, June 2017.

[44] Brecht Reynders, Wannes Meert, and Sofie Pollin. Range and Coexistence Analysis of Long Range Unlicensed Communication. In *23rd International Conference on Telecommunications, ICT 2016, Thessaloniki, Greece, May 16-18, 2016*, pages 1–6, 2016.

[45] T. To and A. Duda. Simulation of LoRa in NS-3: Improving LoRa Performance with CSMA (Source Code on GitHub. `https://github.com/drakkar-lig/lora-ns3-module`. Accessed: 2018-11-22.

[46] T. To and A. Duda. Simulation of LoRa in NS-3: Improving LoRa Performance with CSMA. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7, May 2018.

[47] He Wu, Sidharth Nabar, and Radha Poovendran. An Energy Framework for the Network Simulator 3 (NS-3). In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, SIMUTools '11, pages 222–230, ICST, Brussels, Belgium, Belgium, 2011. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Abbreviations

LPWAN	Low Power Wide Area Network
LoRaWAN	Long Range Wide Area Network
LoRa	Short form for LoRaWAN
SF	Spreading Factor
CSS	Chirp Spread Spectrum
NS-3	Network Simulator version 3
LTE-M	Long Term Evolution for Machines
LBT	Listen Before Talk
DC	Duty Cycle enforcement
FFT	Fast Fourier Transformation
TTN	The Things Network
IoT	Internet of Things
ED	End Device
GW	Gateway
PHY	Physical OSI Layer
MAC	Medium Access Control OSI Layer
OSI	Open Systems Interconnection model
ETSI	European Telecommunications Standards Institute
TX	Transmit-
RX	Receive-
ID	Identifier
CCG	Clear Channel Gap
CSMA	Carrier Sense Multiple Access
ACK	Acknowledgement
O&G	Oil and Gas Industry
CSV	Comma-Separated Values (file format)

Glossary

back off The verb used to describe an end device postponing a transmission due to other observed transmissions on the channel.

congestion The state of a network which has to bear too much ongoing traffic such that a significant part of transmitted data packets are dropped due to the effects of interference.

signal to interference ratio The ratio of the reception power of a signal being received against the aggregated signal power of interfering signals simultaneously present on the channel.

transactional The adjective used to describe a data record consisting of several data packets as well as of a signature forming a transaction; The signature contains a hash value gained by encrypting the data packets to ensure integrity.

bit error rate The ratio of incorrectly transmitted bits against the size in b of the totally transmitted data.

forward error correction A technique letting sending entities transmit redundant data allowing receivers to detect errors in transmitted data.

List of Figures

List of Tables

Appendix A

Installation Guidelines

Please note, that the setup instructions below (subsection A.1.1, subsection A.2.1) are only necessary for installing the two provided implementations on a host machine. The Lubuntu 18.10 (www.lubuntu.net) virtual machine image attached on the DVD (compatible with Oracle VirtualBox, www.virtualbox.org) features both implementations pre-setup in directories under `Desktop/` of user `lorasim` (password = `lorasim`).

A.1 NodeExtractor Application

To run the Python-script allowing for determining frequently transmitting TTN end devices in the Zurich area (as described in subsection 5.3.2), the following dependencies have to be available on your UNIX system (it was successfully tested on a Ubuntu 18.04.2 LTS machine). The package names in square brackets refer to the official Ubuntu 18.04 package repositories:

- Python 3.6 [`python3.6`] (www.docs.python.org/3.6/using/index.html)
- Tkinter for Python 3 [`python3-tk`] (www.tkdocs.com/tutorial/install.html)
- Virtualenv [`virtualenv`] (www.virtualenv.pypa.io/en/latest)

A.1.1 Setup

1. Please extract file `nodeExtractor.zip` (included on the DVD) onto your system and launch a terminal in project directory `nodeExtractor`. Alternatively, the code can be obtained by cloning the Git repository available under:
   `https://github.com/timolex/nodeExtractor.git` . (Please note, that the input CSV-file called '`input.csv`' is only available on the attached DVD).

2. Please run the provided Bash-script `setup.sh` and follow its instructions. This script automatically creates a virtual Python environment in the current directory and installs all required Python packages into it using Python's package manager `pip`.

3. As prompted by the script, please verify, if the dependency installation process was successful (pip logs messages in red in case of errors).

A.1.2 Running the Application

1. (If using the attached VM, please open directory `Desktop/nodeExtractor` in a terminal.) Please run `$ source venv/bin/activate` in the project directory to activate the virtual Python environment.

2. Please execute `$ python3.6 nodeExtractor.py` to start the application. Upon completion, a window containing a plot showing the distribution of TTN nodes' transmission periodicities will be displayed. Please note, that given the large input data set, this computation might take up substantial time resources.

A.1.3 Input Data

Please note that raw data are coming from the included file `input.csv`, which is a subset of two million transmissions from the database-dump provided by TTNMapper.org. This file may be exchanged with a new version from TTNMapper.org, or a subset, *resp.*, superset of the included file as long as its format (its CSV signature) as well as the filename remain the same.

A.1.4 Parameters

As mentioned earlier, the `nodeExtractor.py` application uses the same parameters as presented in Table 5.1. However, different parameters can be defined by modifying file `nodeExtractor.py` and changing the fields at the top of the file under the comment "`# general parameters`".

A.2 LoRaWAN Simulation

A.2.1 Setup

1. Please extract file `LoRaNS3.zip` (included on the DVD) onto your UNIX system and launch a terminal in the project (sub-)directory: `ns-3-dev-with-signetlabdei-lorawan-module`. Alternatively, the source code can be obtained by cloning the Git repository available under: `https://github.com/timolex/ns-3-dev-with-signetlabdei-lorawan-module.git` .

2. Please run the following command to launch the configuration process for the Python build automation tool Waf:
   ```
   $ ./waf configure --enable-tests --enable-examples
   ```
   Please read the console output and verify, if `'configure' finished successfully`. A reason for an unsuccessful configuration is usually a missing dependency, *e.g.*, C++ compiler GCC. Please repeat the command above after installing every new dependency until the configuration was reported as successful.

3. Please checkout either of the following two Git branches:
   - `$ git checkout virtualQueueMgmt`
   - `$ git checkout LBTwithQueue`

4. Please run the following command to build NS-3:
   ```
   $ ./waf build
   ```

5. (*Optional:*) Upon completion of a successful build, please run the following command to test the functionality of the LoRaWAN module:
   ```
   $ ./test.py -s lorawan
   ```

A.2.2 Running Simulations

1. (If using the attached VM, please open directory `Desktop/lorawanSim` in a terminal.) For running simulations with end devices adhering to max. duty cycles, please checkout branch `virtualQueueMgmt`; For simulating end devices operating according to Listen Before Talk, checkout branch `LBTwithQueue`. Please note, that after each time, a different branch was checked out, it is necessary to re-build NS-3 with the command: `$ ./waf build`

2. After a successful build, there is the option of running single simulation scenarios: With the following commands, each of the four scenarios (introduced in subsection 4.4.2) may be run with different parameters, namely no. of end devices (X) and delay in s (Y) in between issuing two consecutive packet transmissions:
   - `$ ./waf --run "scenario1 --nDevices=X --delay=Y"`
   - `$ ./waf --run "scenario2 --nDevices=X --delay=Y"`

- `$ ./waf --run "scenario3 --nDevices=X --intraDelay=Y --interDelay=Y"`
- `$ ./waf --run "scenario4 --nDevices=X --intraDelay=Y --interDelay=Y"`

Alternatively, there is the possibility of running the four scenarios in batches: Therefor, please edit the Bash scripts `scenN.sh` (`N` $\in$ [1, 4]) and set the desired set of numbers of end devices (`nEndDevices`) as well as the delays in s in between issuing two successive packets (`delays`) at the beginning of the file. Simulation batches can be started with the command `$ ./scenN.sh` .

Both running single simulations as well as batches will create a CSV file for each scenario (*e.g.*, `scenario3.csv`), where the performances of finished simulations will be written to.

Simulation batches additionally create a sub-directory called `logs/`, where NS-3 console output is bypassed to in terms of text files. The names of these log files feature the following structure: `scenN_Y_X.txt`, where `N` corresponds to one of the four scenarios, Y is the delay in s in between issuing two consecutive packet transmissions and X the no. of end devices in the simulated network.

All simulation scripts work with a fixed no. of six gateways (*cf.* subsection 4.6.3). This, as well as a variety of other assumptions may be changed by modifying the simulation scripts under `scratch/scenarioN.cc` (`N` $\in$ [1, 4]).

Appendix B

Contents of the DVD

- `Abstract.txt`, `Zusfsg.txt`: This thesis' abstract in English and in German.
- `Bachelorarbeit.pdf`: This thesis document in the PDF format.
- `BScThesis.ps`: This thesis document in the PostScript format.
- `BScThesis.zip`: The LaTeX source code of this thesis document.
- `IntermediatePresentation.pptx`: The presentation slides of the midterm presentation.
- `LoRaSim.ova`: An OVF 1.0 image of a Lubuntu 18.10 virtual machine with both main implementations already set up.
- `nodeExtractor.zip`: The source code for the implementation covered in chapter 5.
- `LoRaNS3.zip`: The source code for running LoRaWAN simulations as documented in section 4.6.
- `Simulation_Results.zip`: Results in terms of CSV files as well as logs (subdirectory `logs/`) for the simulation scenarios discussed in section 4.6