VSB – Technical University of Ostrava
Faculty of Electrical Engineering and Computer Science

An Optimised IoT Architecture Based on Fog Computing with a New Method of Data Transfer Control

PHD THESIS

2022 Jakub Jalowiczor

VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

An Optimised IoT Architecture Based on Fog Computing with a New Method of Data Transfer Control

Ing. Jakub Jalowiczor

PhD Thesis

Supervisor: prof. Ing. Miroslav Voznak, Ph.D.

OSTRAVA, 2021

VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

Information

Dissertation Thesis; Delivered in November, 2021
Doctoral Study Programme:
P1807 Computer Science, Communication Technology and Applied Mathematics
Doctoral Study Branch:
2601V018 Communication Technology

Student: Ing. Jakub Jalowiczor
Department of Telecommunications
Faculty of Electrical Engineering and Computer Science
VSB – Technical University of Ostrava
17. listopadu 2172/15, Ostrava-Poruba, 708 00
jakub.jalowiczor@vsb.cz

Supervisor: prof. Ing. Miroslav Voznak, Ph.D.
Department of Telecommunications
Faculty of Electrical Engineering and Computer Science
VSB – Technical University of Ostrava
17. listopadu 2172/15, Ostrava-Poruba, 708 00
miroslav.voznak@vsb.cz

OSTRAVA, 2021

Declaration

I declare that this doctoral thesis was written independently by me, under the guidance of the doctoral thesis supervisor and using the technical literature and other sources of information, all of which are quoted in the text and detailed in the list of literature at the end of the thesis.

As the author of the doctoral thesis, I furthermore declare that, with respect to the creation of this doctoral thesis, I have not infringed any copyright or violated anyone's personal or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation §11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced in the amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll., Section 2, Head VI, Part 4.

....

(author's signature)

Acknowledgement

I would like to express my gratitude to my doctoral thesis supervisor, prof. Ing. Miroslav Voznak, Ph.D., for his kind support during my doctoral studies and related research. I very much appreciate the opportunity to work with him; his advice and the continuous support he has given me have been objective and invaluable.

I thank all the LIPTEL team members, with whom it has always been a pleasure and satisfaction to collaborate in solving research activities and related tasks.

Finally and importantly, I would like to thank my family and girlfriend for their immense support and motivation during my studies and over the course of my life.

....

(author's signature)

Abstract

Over the years, distributed and grid computing paradigms have evolved to cloud computing, which has become a common approach applied in the Internet of Things (IoT). The growing popularity of the cloud computing paradigm lies mainly in the simple management of end devices, uniform access to many services, elasticity of available resources and cost savings. In addition to these advantages, the expansion of IoT devices and the demand for speed and data volume have provided an opportunity for the emergence of new computing paradigms. The fog computing paradigm brings data processing nearer to the end devices while preserving the cloud connection, leading to lower latency, higher efficiency and location awareness.

The overall aim of the dissertation is the design and implementation of an optimised IoT network architecture which adopts the fog computing paradigm. To eliminate the need to build completely new infrastructure, the optimised network architecture is based on LoRaWAN, which has already been deployed at many locations and offers long-distance communication with low-power consumption. This raises several challenges which need to be overcome. For correct functioning of the fog computing paradigm, it was necessary to explore a new method of controlling the data transfer between IoT gateways and the cloud service. The methods explored in this dissertation are both static (based on predefined values) and dynamic (based on machine learning).

Keywords: computing paradigms; Internet of Things; LoRaWAN; fog computing; network architecture

Abstrakt

V průběhu let se výpočetní modely vyvíjely od distribuovaných a gridových ke cloud computingu, který se stal nejčastěji používaným přístupem v oblasti Internetu věcí. Rostoucí popularita cloud computingu spočívá především v jednoduché správě koncových uzlů, jednotném přístupu k velkému počtu služeb, elasticitě dostupných zdrojů a šetření jednotlivých nákladů. Přes všechny své přínosy však narůstající počet připojených zařízení a nároků na rychlost dávají příležitost vzniku nových výpočetních modelů. Fog computing model přenáší výpočetní výkon blíže ke koncovým zařízením při zachování spojení s cloudem, což vede ke snížení latence, zvýšení efektivity a umožnění reagovat na základě aktuálních podmínek.

Výsledným cílem této disertační práce je návrh a implementace optimalizované síťové IoT architektury s podporou pro fog computing. Pro eliminaci nutnosti budovat kompletně novou infrastrukturu počítá výsledné optimalizované řešení s integrací do LoRaWAN, která je již nasazena na mnoha místech a nabízí komunikaci na velké vzdálenosti při nízké spotřebě energie. Tato integrace však přináší několik úskalí, jež je potřeba překonat. K dosažení správné funkčnosti fog computingu bylo potřeba provést výzkum metody pro řízení přenosu dat mezi síťovou bránou a cloud službou. Zkoumané metody jsou jak statické (založené na předdefinovaných hodnotách), tak dynamické (využívající strojového učení).

Klíčová slova: výpočetní modely; Internet věcí; LoRaWAN; fog computing; síťová architektura

Contents

1 Introduction **1**

2 State of the Art **5**

2.1 Existing Studies 5

2.2 Prevalent IoT Technologies 7

2.2.1 Short-Range Wireless Networks 8

2.2.2 Wireless Local Area Networks 8

2.2.3 Conventional Cellular Networks 8

2.2.4 Low-Power Wide-Area Networks 8

2.3 LPWAN Technologies 9

2.3.1 NB-IoT 9

2.3.2 Sigfox 10

2.3.3 LoRaWAN 11

2.4 LPWAN Architectures 15

2.4.1 Cloud Computing Architecture 15

2.4.2 Edge Computing Architecture 15

2.4.3 Fog Computing Architecture 16

3 Machine Learning **17**

3.1 Machine Learning Sub-Classes 17

3.1.1 Supervised Learning 17

3.1.2 Unsupervised Learning 18

3.1.3 Semi-Supervised Learning 18

3.1.4 Reinforcement Learning 18

3.1.5 Deep Learning 18

3.2 Classification Algorithms 18

3.2.1 Decision Tree 19

3.2.2 Random Forest 19

3.2.3 K-Nearest Neighbor 20

3.2.4 Support Vector Machines 20

3.2.5 Naive Bayes 21

3.2.6 Artificial Neural Networks 21

3.3 Model Evaluation Metrics 23

3.4 Cross-validation 25

4 Aims of Dissertation **27**

5 Proposed Network Architectures **29**
5.1 General Fog Computing Architecture . . . 29
5.2 Standard LoRaWAN Architecture . . . 30
5.3 Proposed Network Architectures . . . 31
5.3.1 Architecture A . . . 31
5.3.2 Architecture B . . . 32
5.3.3 Architecture C . . . 33
5.4 Comparison of the Proposed Architectures . . . 35
5.4.1 Comparison in Terms of Queuing Theory (Execution Time) . . . 35
5.4.2 Comparison in Terms of Functional Properties . . . 42
5.4.3 Summary of Results from the Comparison . . . 43

6 Methods to Control Data Transfer **45**
6.1 Controlled Data Transfer . . . 45
6.1.1 Dataset . . . 46
6.1.2 Payload Structure . . . 47
6.1.3 Classification of Data Privacy . . . 47
6.2 Discussion of the Results . . . 56

7 Implementation of Key Components **59**
7.1 Chirpstack . . . 59
7.2 Implementation of the Architecture . . . 59

8 Testbed and Verification of Results **67**
8.1 Current Solution at the VSB-TUO Campus . . . 67
8.2 Verification of Results . . . 68
8.2.1 Comparison of Execution Times . . . 68
8.2.2 Data Privacy Test . . . 70
8.2.3 Offline Processing Test . . . 70
8.3 Discussion of the Results . . . 72

9 Conclusions and Expected Contributions **73**

10 Research Results and Activities **87**
10.1 Participation in Research Projects . . . 87
10.2 Results of Research Activities . . . 88
10.3 Other Achieved Results . . . 88

List of Acronyms

3GPP	–	3rd Generation Partnership Project
ABP	–	Activation By Personalization
AES	–	Advanced Encryption Standard
AI	–	Artificial Intelligence
ANN	–	Artificial Neural Network
API	–	Application Programming Interface
AppSKey	–	Application Session Key
AUC	–	Area Under the Curve
BCET	–	Best-Case Execution Time
CF	–	Carrier Frequency
DevEUI	–	Device Extended Unique Identifier
DBPSK	–	Differential Binary Phase-Shift Keying
FFNN	–	Feed-Forward Neural Network
FHDR	–	Frame Header
FN	–	False Negative
FP	–	False Positive
HTTPS	–	Hypertext Transfer Protocol Secure
IaaS	–	Infrastructure as a Service
IoT	–	Internet of Things
ISM	–	Industrial, Scientific, and Medical
JSON	–	JavaScript Object Notation
LoRaWAN	–	Long Range Wide Area Network
LPWAN	–	Low-Power Wide-Area Network
LTE	–	Long Term Evolution
M2M	–	Machine to Machine
MAC	–	Medium Access Control
MHDR	–	Medium Access Control Header
MIC	–	Message Integrity Code
ML	–	Machine Learning
MQTT	–	Message Queuing Telemetry Transport
NFC	–	Near Field Communication
NwkSKey	–	Network Session Key
OTAA	–	Over-the-Air Activation
OS	–	Operating System
PaaS	–	Platform as a Service
PoE	–	Power-over-Ethernet

PSM	–	Power Saving Mode
QoE	–	Quality of Experience
QoS	–	Quality of Service
RF	–	Radio Frequency
RFID	–	Radio Frequency Identification
RMSE	–	Root Mean Square Error
ROC	–	Receiver Operating Characteristic
RSSI	–	Received Signal Strength Indicator
SaaS	–	Software as a Service
SF	–	Spreading Factor
SNR	–	Signal to Noise Ration
SPI	–	Serial Peripheral Interface
SVM	–	Support Vector Machines
TN	–	True Negative
TP	–	True Positive
TTN	–	The Things Network
UNB	–	Ultra Narrow Band
WCET	–	Worst-Case Execution Time
WPAN	–	Wireless Personal Area Network

List of Figures

1 LoRaWAN technology stack. . . . 11
2 LoRaWAN Class A communications scheme. . . . 12
3 LoRaWAN Class B communications scheme. . . . 12
4 LoRaWAN Class C communications scheme. . . . 13
5 Diagram of a neural network. . . . 22
6 Confusion matrix. . . . 23
7 Example of an ROC curve. . . . 25
8 k-fold cross-validation principle. . . . 26
9 Three-layer fog computing architecture. . . . 29
10 Standard LoRaWAN architecture. . . . 31
11 Diagram of network Architecture A. . . . 32
12 Diagram of network Architecture B. . . . 33
13 Diagram of network Architecture C. . . . 34
14 Diagram of the simulation model of Architecture A. . . . 38
15 Diagram of the simulation model of Architecture C. . . . 39
16 Comparison of architectures A and C (95% private messages). . . . 40
17 Comparison of architectures A and C (100% private messages). . . . 40
18 Histogram for Architecture A (95% private messages). . . . 41
19 Histogram for Architecture C (95% private messages). . . . 41
20 Histogram for Architecture A (100% private messages). . . . 41
21 Histogram for Architecture C (100% private messages). . . . 41
22 Dataset records example. . . . 46
23 Training set example - Periodicity 48
24 Comparison of ROC curves for different classification algorithms. . . . 51
25 Training set example – Payload. . . . 52
26 Comparison of ROC curves for different classification algorithms – Payload. . . . 54
27 Comparison of ROC curves for different classification algorithms – Payload (modified dataset). . . . 55
28 Comparison of ROC curves for different classification algorithms – Payload Combined (modified dataset). . . . 55
29 LoRaWAN message structure. . . . 60
30 Block structure. . . . 62
31 example 67
32 example 68
33 Results of service times comparison. . . . 69
34 Offline processing test results. . . . 71

List of Tables

1 Execution times of the components. 36
2 Execution times of the fog gateway. 37
3 Comparison of results from the simulation. 41
4 Summary of features of the proposed architectures. 44
5 Basic statistics of the dataset 47
6 Classification results for different FFNN topologies – periodicity. 50
7 Decision Tree confusion matrix – Periodicity. 50
8 SVM confusion matrix – Periodicity. 50
9 k-NN confusion matrix – Periodicity 51
10 Feed-Forward NN confusion matrix – Periodicity 51
11 Decision tree confusion matrix – Payload. 53
12 Naive Bayes confusion matrix – Payload. 53
13 k-NN confusion matrix – Payload. 53
14 Random forest confusion matrix – Payload. 53
15 Naive Bayes confusion matrix – Payload (modified dataset). 54
16 MAC message types. 61
17 Relationship between FPort and session keys. 61
18 Privacy test results. 70

1 Introduction

The Internet of Things (IoT) is becoming increasingly prevalent, although the concept of interconnecting computers and networks in a manner to monitor and control devices has existed for decades. Considerable interest in this area has resulted from new technologies and market trends that allow the interconnection of greater numbers of devices which are smaller, simpler and less expensive.

With this expansion of the IoT and the unique requirements of IoT devices, the development of new technologies created exclusively to provide wireless connectivity is meaningful. These technologies aim to achieve the minimal possible energy consumption while maintaining quality of transmission. The selection of suitable technology for a particular application is crucial in any IoT deployment. Technologies such as Bluetooth Low Energy and ZigBee are popular for application in smart home solutions and wearable electronics. The energy consumption of these technologies is low, yet their limited coverage makes them unsuitable for use in urban-wide coverage, which is necessary for many smart city applications. Low-Power Wide-Area Network (LPWAN) technologies especially designed for long-distance wireless communication and the low power consumption are more suited to this use. LPWAN technologies are primarily designed to transfer a small amount of data at once, and they include technologies such as LoRa, SigFox and NB-IoT. These network technologies implement a cloud computing paradigm which has several drawbacks, and it is not sufficient for some IoT use cases.

Although precise predictions for the future expansion of connected IoT devices vary, the increasing trend is clear: as the number of IoT devices increases, the demands for speed and data volume become greater. This situation could lead to a certain threshold where cloud computing is no longer an effective solution nor sufficient for specific IoT use cases.

As the title of the dissertation suggests, this work examines a fog computing approach, which is a combination of cloud computing and edge computing providing the benefits of both. While the major part of processing is performed by edge devices which additionally work as data filters, the cloud performs less urgent or more complex computing. From that perspective, the fog computing paradigm can be seen as an extension of the cloud computing paradigm in applications and domains which are not suitable for cloud computing. Location awareness allows fog computing to better adapt to end device needs. The involvement of a mechanism to control the distribution of computational resources between the end devices in the fog layer and cloud service is also a significant component of fog computing. This mechanism can be either static, predefined by the data owner, or dynamic based on machine learning to automatically control the required computational and storage resources. Suitable for a heterogeneous environment, a dynamic mechanism considers the current conditions and applies decisions accordingly.

The above-mentioned benefits of fog computing are significant, and equally so in the IoT. Let us imagine a company which applies LPWAN technology to collect specific information in a production hall. Fog computing can extend the possible use cases with many valuable features.

The company can define which data are processed and stored locally, while the rest are transferred to the cloud. The fog layer can implement a connection failure detection method. If a connection failure is detected, all resources are computed and stored in the fog layer until the connection is re-established. The main advantage of fog computing is a shorter response time, which is crucial in many applications. Examples of application are primarily in smart city environments, where many extraordinary conditions such as car accidents or fires can occur. Quick response to such situations is imperative and cannot be expected from cloud computing. The closer proximity of network elements actively involved in computation, however, is potentially a suitable solution. As mentioned above, existing LPWAN technologies are mainly based on cloud computing and a suitable substitute is not available, even though it has been the subject of many studies.

As the Department of Telecommunications at the VSB-Technical University of Ostrava operates its own campus LoRaWAN network, the proposed solution can be tested and verified in a real-world traffic scenario. This was the main motivation for my study and proposal of a solution which functions with LoRaWAN technology. Additionally, LoRaWAN technology offers long-range communication [1], low energy consumption [2] and many existing deployments over a large number of locations. However, the main conclusions of this dissertation can be generalised to other LPWAN technologies.

Because LoRaWAN technology is deployed widely, network operators need only update individual components of existing networks to support fog computing; the end-devices and hardware remain unchanged. Fog computing features are therefore available at low overall cost. An example is the campus at the VSB-Technical University of Ostrava, where LoRaWAN is already deployed, and the addition of fog computing optimisations for further research is desirable.

The dissertation is structured as follows:

- Chapter 2: State-of-the-Art – summarises existing studies and technologies pertaining to the IoT and describes LPWAN technologies and architectures in detail.
- Chapter 3: Machine Learning – contains a theoretical introduction to machine learning and related methods.
- Chapter 4: Dissertation Aims – contains the aims and delimitations of the dissertation.
- Chapter 5: Proposed Network Architecture – suggests three different network architectures based on LoRaWAN and the fog computing paradigm. The architectures are compared according to several characteristics to select the optimal architecture. This chapter addresses Aim 1.
- Chapter 6: Machine Learning Methods to Control Data Transfer Between a Gateway and the Cloud – includes several approaches for controlling the selection of network elements designated to processing data according to actual data traffic. This chapter addresses Aim 2.

- Chapter 7: Implementation of Key Components of the Architecture for Verification – describes the implementation of the selected optimal network architecture.
- Chapter 8: Testbed and Verification of Results – describes the testbed located at the VSB - Technical University of Ostrava campus and the verification process of the obtained results. Together with Chapter 7, this chapter addresses Aim 3.
- Chapter 9: Conclusions and Expected Contributions – summarises the conclusions and expected contributions of the results.
- Chapter 10: List of the Candidate's Research Results and Activities – contains a summary of results obtained during the Ph.D. study.

2 State of the Art

This section summarises the related knowledge in IoT technologies and reports the most recent developments. The chapter is organised as follows: (i) a discussion of the existing studies of IoT, fog computing, and Quality of Service (QoS); (ii) a description of the technologies prevalent in the IoT, followed by a closer specification of LPWAN technologies; (iii) an introduction to various network architectures.

2.1 Existing Studies

The state of the art in the IoT and various network architectures shows a trend away from the cloud computing paradigm to fog computing. While cloud computing in some situations cannot keep pace with the growing volume and speed of transmitted and processed data, the fog computing paradigm can improve efficiency and attain lower latency and location awareness.

Surveys in [3] and [4] examined LPWAN technologies such as LoRaWAN, Sigfox and NB-IoT. In [5], the authors provided a detailed introduction to cloud computing. In [6], Weisong Shi et al. presented a definition of edge computing and its benefits over cloud computing, some case studies, and related challenges and opportunities. Before the idea of fog computing, Cloudlet [7] designed and developed a similar concept. It consists of a virtual machine-based server or cluster of servers and is placed in single-hop proximity of mobile devices to respond with low-latency. The studies [8], [9] and [10] provide overviews of fog computing. The paper [11] provides a further survey of the fog computing paradigm along with recent research trends, multiple definitions and architectures.

ARM, Cisco, Dell, Intel, Microsoft and Princeton University funded the OpenFog Consortium in 2015 to define an open fog computing architecture. The consortium released the document OpenFog Reference Architecture document [12] in which fog computing is defined as:

> "A horizontal, system-level architecture that distributes computing, storage, control, and networking functions closer to users along a cloud-to-thing continuum."

According to this definition, fog computing is an extension of the regular cloud-based computing paradigm, and all advantages of the cloud should be preserved.

Fog computing and its use in IoT applications are discussed in [13]. A variety of research opportunities in the fog computing field are mentioned in [14]. In [15], Gia et al.introduced a system and gateway architecture along with the experimental results of a fog computing case study of healthcare IoT and ECG feature extraction.

In [16], the authors analysed and described fog computing and other computing paradigms from a security perspective. Security and privacy issues in fog computing and the IoT, for example, authentication, trust, access control and data protection, are presented in [17].

In [18], the authors described the design and experimental deployment of a LoRaWAN infrastructure and the deployment of fog computing nodes at the smart campus of the University

of A Coruña. However, this paper focused primarily on the planning of signal radiation and coverage.

A LoRaWAN architecture with autonomous base stations was introduced in [19], in which the authors also gave an example of use in areas with limited internet access. The authors proposed a master/slave base station architecture in which the master base station acts as a central point for many slave base stations, and in this manner, an internet connection is not required for the slave base stations.

The authors in [20] described the potential of combining fog computing with LoRaWAN in smart cities. At the beginning of the paper, they introduced fog computing and its general three-layer architecture, followed by a comparison with cloud computing. In the next section, the authors described a case study in which fog computing was combined with LoRaWAN for water distribution networks. The proposed network architecture included edge servers and LoRa gateways connected to a LoRa network server which communicated with an application server located in the cloud.

A comparative study of LPWAN technologies in [21] showed that while Sigfox and LoRa have superior assumptions for applications which require longer battery lifetime, higher capacity and lower cost, NB-IoT offers advantages in latency and QoS. However, LoRa end devices can operate in class C (continuous) mode to handle bidirectional communications with lower latency although with greater energy consumption.

In [22], the authors proposed and implemented an energy-efficient downlink communication mechanism for LoRaWAN, called TRILO. The mechanism is based on the IEEE 802.11 Power Saving Mode (PSM) adopted from IEEE 802.11 WLAN and is designed to inform end devices of the downlink traffic waiting to be transmitted. After successfully informing the end device, transmission of downlink traffic can be executed. The authors also compared the performance of the TRILO mechanism with the Class B downlink scheme in simulations and demonstrated that the TRILO mechanism's efficiency was higher. The authors of [23] used the ns-3 simulation tool to analyse the scalability of a single-channel multigateway LoRaWAN. The simulation results indicated that a suitable choice of network parameters is crucial to LoRaWAN performance and that increasing the gateway density can improve the packet delivery ratio of confirmed upstream messages. A Monte Carlo simulation presented in the paper [24] demonstrated a sub-optimal SF allocation method under a pure ALOHA protocol to maximise massive connectivity in LoRaWAN. The results showed that the proposed method can increase massive connectivity compared to the other methods the authors applied in their study. In the paper [25], the authors presented a new method of optimisation for efficient spreading factor selection by individual end devices, thereby increasing the probability of data delivery in LPWAN networks by 20–40%. Although this method of optimisation led to an increase in energy consumption, the increase was minor (1–8%). The work in [26] explored the performance and QoS optimisation of a LoRa network by fine-tuning the optimal setting of the Spreading Factor (SF) and Carrier Frequency (CF) parameters to reduce the packet collision rate and energy consumption. The

results validated by simulations showed that the proposed methodology optimised the SF and CF settings with an average increase of 6% in Data Extraction Rate and that collisions were 13 times lower. A new MAC layer for RS-LoRa with two-step lightweight scheduling was proposed in [27]. Its performance was evaluated with the ns-3 simulation tool, and the results showed that in a single- gateway scenario with 1000 nodes, the proposed MAC layer reduced the Packet Error Ratio by nearly 20%. The authors in [28] designed and developed a LoRa traffic generator based on software-defined radio technology (SDR) which emulated thousands of LoRa sensors. The authors analysed the communication collisions while applying different SFs at different channel bandwidths on a single communication channel. The study presents an empirical evaluation of LoRa modulation and an experimental analysis of the LoRa traffic generator in a semi-anechoic chamber. The results indicated that collisions dramatically decreased overall performance. In [29], the authors presented a smart city network architecture called Fog Computing Architecture Network. The study showed that the network provided lower latency and improved energy provisioning. In [30], Gupta et al. produced an iFogSim software simulation of the IoT and a fog environment to evaluate resource management policies in terms of their effect on latency, energy consumption and other properties. In [31], the authors used iFogSim to evaluate a proposed QoE-aware application placement policy in fog computing environments.

Many researchers have measured and analysed QoS parameters for NB-IoT. The authors of [32] obtained empirical measurements of the Signal-to-Noise Ratio (SNR) and Received Signal Strength Indicator (RSSI) on different floors of a university building. Their results indicated slight variations in the SNR and RSSI values throughout the day, possibly caused by increases in mobile network user activity during peak times. In [33], the authors measured the QoS parameters (reliability, signal quality, throughput and latency) of a commercial NB-IoT network. The real network parameters were compared to theoretical parameters, and the subsequent results revealed that lower signal levels produced higher latency and reduced throughput. The overall results demonstrated reliable performance of the NB-IoT network. The authors in [34] measured the physical and application layer QoS parameters in a public NB-IoT network. Compared to previous studies, their analysis of physical network parameters and their effect on the application layer QoS was deeper.

The studies mentioned above indicate the topicality of the IoT and fog computing paradigm and highlight that QoS should be considered in the optimisation of IoT architecture. The dissertation extends these studies and provides the design for an optimised architecture based on LoRaWAN, with a practical implementation. Although some studies describe a similar topic, the practical aspects and many challenges, such as the end-to-end payload encryption of a standard LoRaWAN solution, were not addressed.

2.2 Prevalent IoT Technologies

Many various network technologies are used in the IoT, but the most common are short-range wireless networks, wireless local area networks (WLANs), cellular networks and LPWANs [3].

All have advantages which make them suitable for different use cases, and they are designed to complement, not replace each other.

2.2.1 Short-Range Wireless Networks

IoT technologies falling into this category operate primarily in personal applications, in a so-called Wireless Personal Area Network (WPAN). A critical feature of IoT applications is low energy consumption, as the end devices in most cases are battery-powered. The short range of these technologies eliminates their use in applications requiring urban-wide coverage. Namely, this category includes Bluetooth Low Energy, Zigbee, ZWave, Near Field Communication (NFC) and Radio Frequency Identification (RFID) [35].

2.2.2 Wireless Local Area Networks

This category is not principally designed for the IoT, evidenced by the higher energy consumption of end devices. When power supply to an end device is not a problem, this can be a suitable solution for smart home applications [Vas18]. Its advantage is its presence in almost all households and that building a new network infrastructure explicitly dedicated to IoT communication is unnecessary. The physical and media access layers are specified in the IEEE 802.11 standard (Wi-Fi) [35].

2.2.3 Conventional Cellular Networks

Current cellular networks have sufficient coverage, but they are not very suitable for IoT applications in a default state. The most significant problem is the continuous synchronisation of end devices (mainly cellular phones) with the network given by the nature of the service, causing higher energy consumption. Cellular networks such as 1G, 2G, 3G, 4G and 5G fall into this category.

Because of the extended coverage capabilities of cellular networks and unsuitable features for IoT applications, the 3GPP (3rd Generation Partnership Project) project attempted to make adjustments in these features to minimise energy consumption. The result of this effort was NB-IoT technology, which is integrated into the LTE standard and removes many features to remain as simple as possible and thereby lower price and minimise battery consumption. Although NB-IoT was adapted from the cellular network, its types of features place it under the LPWAN category [37].

2.2.4 Low-Power Wide-Area Networks

LPWAN is a generic term for a group of technologies with common features to enable long-distance wireless communication with low power consumption. LPWAN technologies are primarily designed to transfer state information of small size in the uplink. These technologies do not require constant synchronisation with the network since they are not designed primarily for

continuous downlink communication. Therefore, end devices can sleep most of the time and send data only a few times per day, preserving battery life, which is essential, and for some technologies, the battery can last over ten years [38]. End devices designed for this kind of network technology are mainly placed in difficult-to-access areas and are battery-powered. Battery replacement in these areas may be difficult or even impossible. LPWAN technologies are classified into two categories: technologies which function in an unlicensed frequency spectrum and technologies which communicate in a licensed frequency spectrum [3]. The first category is most often represented by LoRa (Long Range) or Sigfox. NB-IoT technology represents the second category [1].

2.3 LPWAN Technologies

Because the dissertation discusses architecture which supports fog computing, which is especially relative to the range of urban-wide networks, it is also relevant to examine LPWAN technologies in the following section. According to the nature of this work and the reasons described in the introduction, most of the content relates to LoRaWAN technology, which is discussed in detail.

2.3.1 NB-IoT

As the name of the technology suggests, NB-IoT is based on narrowband radio technology, and it was standardised in Release 13 of the 3GPP, issued in 2016. This release introduced several new technologies optimised for application in the IoT. The NB-IoT communication protocol is based on the LTE protocol and reduces many features to make it suitable for the IoT. According to the specification, NB-IoT lowers the price and energy consumption to track IoT trends [39]. An important feature of NB-IoT is its ability to coexist with GSM and LTE. NB-IoT operates in a frequency bandwidth of 200 kHz (one resource block) and supports flexible deployment with three different operation modes [40]:

- In-band operation mode – uses one physical resource block within an LTE carrier,
- Guard-band operation mode – uses the unused resource blocks in guard bands within an existing LTE carrier,
- Standalone operation mode – adopts idle spectrum resources, for example, by reframing the GSM carrier to carry NB-IoT traffic.

Mobile network operators can add NB-IoT support to the existing LTE infrastructure by upgrading the software, which aids quick deployment. Because operators manage the NB-IoT network, it is impossible to build private networks. In the Czech Republic, the first operator to offer NB-IoT connection was Vodafone company.

As the NB-IoT operates in a licensed frequency spectrum and employs an LTE-based protocol, it is a better choice for guaranteed quality applications than LoRa or Sigfox. However,

NB-IoT end devices consume additional energy because of QoS manipulation and more energy-intensive features inherited from LTE [41].

2.3.2 Sigfox

Sigfox [42] is a radio technology designed especially for the IoT. It uses the Ultra Narrow Band (UNB) modulation technique to provide scalable, high-capacity, and low energy consumption for end devices with Differential Binary Phase-Shift Keying (DBPSK) at 100 bps. Powered with a batteries, devices connected to the Sigfox network should last several years, and in many cases, up to ten years. A significant feature is the ability to design simple, small antennas at the end of the network infrastructure and cheap, easy-to-match antennas on the device side. The use of lower data transfer rates has narrower bandwidth requirements, thereby reducing the interference level and increasing the receiver's sensitivity. This permits coverage of large geographical areas with fewer base stations [21]. The technology operates in the ISM (Industrial, Scientific and Medical) band. This band is an unlicensed frequency band initially intended for industrial, scientific and medical applications. In Europe, it is a frequency band in the area of 868 MHz.

The unlicensed frequency band has a lower price for data transfer but suffers greater interference than a licensed frequency band. Devices which communicate in unlicensed bands must be designed to coexist with other technologies on these frequencies without the risk of collision.

Sigfox Cloud can be used for device management or data collection and provides a web application interface. Once an end device sends a message to the Sigfox network, one of the three main ways of handling the received data can be used:

- Graphical user interface – available via a web interface,
- REST API – this interface can be used to reach data and related information,
- Callback – a mechanism to forward each received message to the user's application server. Callbacks are fully configurable HTTPS requests. For example, we can set the method, headers, type of content and content structure within the message body.

The Sigfox technology is suitable for more straightforward applications in which end devices send uplink messages only a few times per day. In this frequency band, the number of uplink messages for each end device is limited to 144 and the transmit power is limited to 25 mW. Sigfox uses the duty cycle method as a mechanism to share the spectrum in the ISM band. In Europe, the duty cycle is 1%, allowing end devices to transmit only 36 seconds every hour. With a time-on-air of six seconds per packet, the maximum is six transmitted messages per hour with a payload of 4, 8 or 12 bytes [43]. Radio messages handled by a Sigfox network are small (12 bytes payload in the uplink, 8 bytes in the downlink) because of the lightweight protocol. The above implies that the Sigfox technology is not suitable in applications where bidirectional and frequent communication throughout the day are necessary. Sigfox is a closed technology and

does not have many freely available detailed technical specifications. A service provider manages the network infrastructure and data storage for all received messages [Jal19a, Gre19].

Taking into account the properties mentioned above, it can be concluded that Sigfox is not very well prepared for the future. While NB-IoT excels primarily in providing QoS, and LoRaWAN technology allows anyone to build a private network, Sigfox technology offers nothing special or outstanding as an alternative.

2.3.3 LoRaWAN

LoRaWAN defines the communication protocol and system architecture for the network, while the LoRa physical layer provides a communications link. LoRa provides wireless modulation for long-range, low-power and low-data-rate communication. LoRa modulation is based on Chirp Spread Spectrum (CSS) modulation, which is used by the military and for communications in space, however LoRa is the first implementation for commercial use [21]. LoRa modulation has been patented by Semtech Corporation. Figure 1 illustrates a LoRaWAN technology stack.

Protocol and network architecture have the most significant effect on end device battery life, network capacity, QoS, security and applications. LoRaWAN architecture is typically deployed in a star-of-stars topology, in which messages sent by each end device can be received by multiple gateways, not just a single gateway. The technology operates in the ISM band [39].

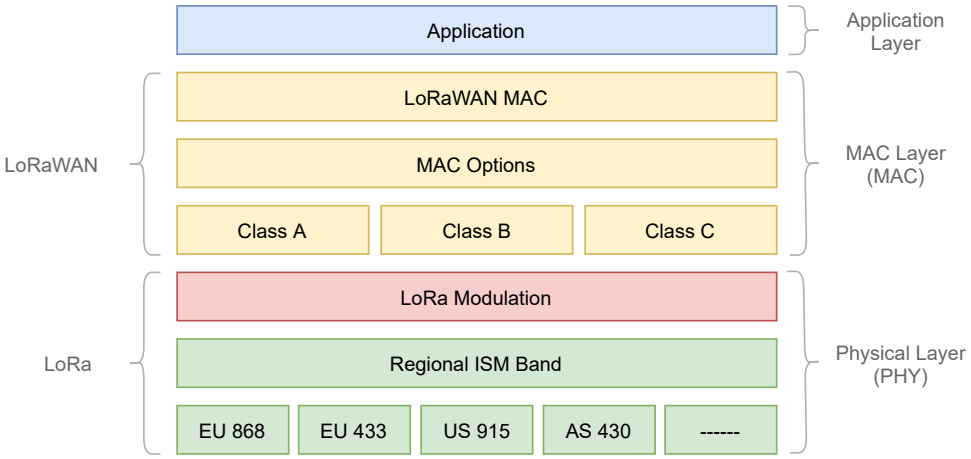

Figure 1: LoRaWAN technology stack.

Because end devices are used in applications which have different requirements, the LoRaWAN specification defines three end device classes. The three classes and their features are as follows [46]:

- **Class A** (Bidirectional end devices) – Class A end devices can communicate bidirectionally whereby each uplink message from the end device is followed by two short downlink receive windows, both with a specified delay. These windows use mutually different frequencies

and data rates. End devices of this class have the lowest power consumption. This type of communication is suitable for communication initialised by the server shortly after the end device has sent an uplink message. If the server sends a message at any other time, the downlink communication must wait until the next scheduled uplink transmission. Implementation of Class A features is mandatory for all LoRaWAN end devices. Figure 2 illustrates the communication scheme of Class A, in which RX1 and RX2 represent the two receive windows with different parameters.

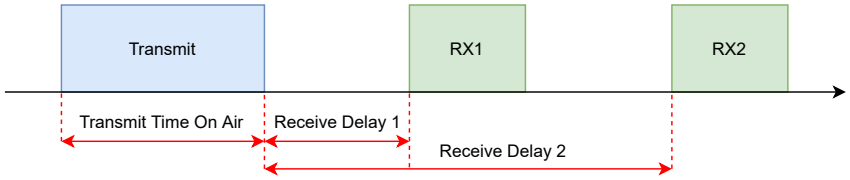

Figure 2: LoRaWAN Class A communications scheme.

- **Class B** (Bidirectional end devices with scheduled receive slots) – Class B end devices have more receive slots. Because the uplink messages are random, downlink transmission in the case of Class A end devices cannot does not occur predictably. Class B end devices therefore open extra receive windows at scheduled times in addition to the Class A random receive windows. For this scheduled timing to function correctly, end devices of this class periodically receive a signal from the gateway to synchronise all end devices in the network. In this manner, end devices can open a short receive window at scheduled times to receive downlink communications. This reception window is called a ping slot, and the downlink which uses one of these ping slots is called a ping. Figure 3 depicts the communications scheme of Class B.

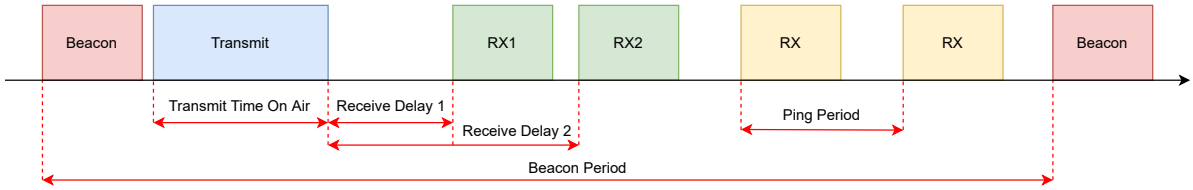

Figure 3: LoRaWAN Class B communications scheme.

- **Class C** (Bidirectional end devices with maximum receive slots) – Class C end devices allow bidirectional communication nearly continuously because their receive windows with RX2 parameters are opened almost constantly and close only during transmission or while listening on RX1. Figure 4 shows that the end device must open a short window with RX2 parameters immediately after the end of uplink transmission and keeps this window open until the beginning of the receive window with RX1 parameters. The end device must

listen on RX2 immediately after the receive window with RX1 has been closed and continue listening until the end device transmits another uplink message. The disadvantage of Class C is that end devices consume more energy than Class A and Class B end devices. In this manner, Class C communication is designed for end devices only with sufficiently available power. However, this class offers the lowest latency for server-initiated communications.

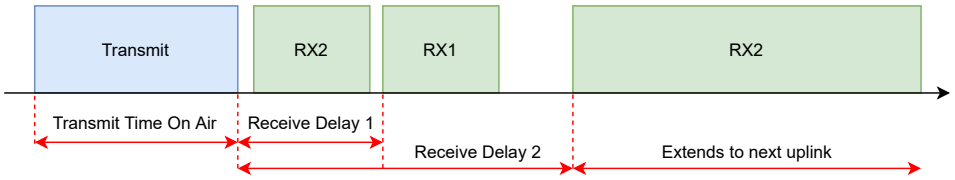

Figure 4: LoRaWAN Class C communications scheme.

As mentioned above, the LoRa physical layer implements CSS modulation. The following configurable parameters determine its characteristics [26], [47]:

- **Bandwidth** (BW) – is the range of frequencies available in a certain transmission band. If a higher BW is selected, the data rates are higher. However, the sensitivity to interference increases with greater BW. BW can be seen as a trade-off between sensitivity and data rate. Channels can have a BW of 125, 250 or 500 kHz, depending on the region and frequency plan. A BW of 125 kHz corresponds to a chip rate of 125 kilo-chips-per-second (kcps) [48].

- **Spreading Factor** (SF) – is the ratio between the symbol rate and the chip rate. SF values can range from 7 to 12, and the different SFs are orthogonal to each other. Although the authors of some works [49]–[51] have shown that the orthogonality is imperfect, it ensures that data packets can be successfully decoded if the gateway receives them from multiple end devices with different SFs concurrently. Higher SF values mean lower sensitivity to interference, but with a reduced data rate, which increases the airtime and power consumption requirements of the end device. If an end device is located near a gateway, the signal does not propagate over a long distance, and therefore a lower SF than the SF used in signal propagation over long distances should be used [24].

- **Carrier Frequency** (CF) – is a central frequency which can be configured between 137 and 1020 MHz in accordance with the legislative regulations of a specific geographical region. The CF value affects the maximum duty cycle per hour according to the ETSI EN300.220 standard [52]. The duty cycle represents the ratio of the maximum time a given end device occupies a communication channel relative to a period of one hour. The duty cycle is expressed as a percentage and represents a certain limitation of fair usage of the shared channel [53]. Hence, each end device must remain silent for a period of time before sending another message, thereby reducing the probability of potential collisions.

- **Transmission Power** (TXPower) – is the maximum transmit power the end device can operate. It is a regional-specific parameter. The value of this parameter is between 0 and 15, where 0 corresponds to a power of 20 dBm and 5 corresponds to 2 dBm. The remainder of the values are reserved for future use.
- **Coding Rate** (CR) – represents the ratio of useful bits to redundant bits. Redundant bits are added to protect useful data from transmission interference when the Forward Error Correction method is used. Thus, 5–8 redundant bits are added to every four bits. The coding values are 4/5, 4/6, 4/7 and 4/8. Higher CR values increase the resistance to interference. With a higher CR, both airtime and power consumption increase. In the LoRaWAN protocol, CR is fixed at 4/5 [54].

The symbol and data rates depend on the SF and BW, as follows from the corresponding equations. The symbol rate R_s can be calculated according to [55]:

$$R_s = \frac{BW}{2^{SF}} \; [symbols/sec] \tag{1}$$

The data rate, also known as a nominal bit rate R_b, can be expressed as [55]:

$$R_b = SF \cdot \frac{\frac{4}{4+CR}}{\frac{2^{SF}}{BW}} \; [bits/sec] \tag{2}$$

LoRaWAN implements an Adaptive Data Rate (ADR) mechanism to successfully optimise data rates, airtime, and energy consumption. For static end devices, the network server searches for optimal parameters to evaluate the last 20 uplinks of a specific end device. When the network server has found these optimal parameters, it prompts the end device to apply them through MAC commands. This process is referred to as network-managed ADR or static ADR. The end device itself controls ADR activation by setting the ADR bit in the Frame Header (FHDR). If the ADR bit has not been set, the network server does not evaluate the incoming uplinks or control the data rate of this end device. The goal of ADR is to attain transmission with the highest data rate and lowest airtime and power consumption possible given the conditions at the time. When the ADR optimises the SFs of the end devices, network capacity also increases since data packets with different SFs can be transmitted simultaneously [48]. Network-managed ADR should only be applied to static end devices or end devices with stable RF conditions. Even a mobile end device can be stationary for a longer time, allowing network-managed ADR to be used during those times. If the end device is not stationary, ADR is operated by the end device without collaboration with the network server. This process is referred to as Blind ADR. However, the end device should enable ADR whenever possible to increase battery life and maximise network capacity [56].

When an end device needs to participate in a LoRaWAN network, an activation procedure is required. During this activation procedure, two session keys and a short device address (DevAddr) are assigned to the end device. This information is necessary for subsequent commu-

nication with the network. LoRaWAN supports two methods of activation. When an end device applies over-the-air activation (OTAA), it uses static root keys to negotiate the dynamically generated DevAddr and session keys during a join procedure. These session keys remain the same until a new session is established. With activation by personalisation (ABP), an end device does not contain root keys. Instead, the fixed DevAddr and session keys for the pre-selected network are embedded in the end device. Thus, no interaction is needed to join the network. ABP is a less safe method of activation since an end device must use the same session keys throughout its lifetime, although the power consumption is lower [57], [58].

A benefit of LoRaWAN technology is its openness and the possibility to build private networks. These allow developers to design and deploy their LoRaWAN gateways, network server and application server together with a backend interface for end-users. Deployment of private networks is a special aspect of LoRaWAN which differentiates it from other technology.

2.4 LPWAN Architectures

In this subsection, different network architectures are discussed to describe the state of the art, as this is the main topic of the dissertation.

2.4.1 Cloud Computing Architecture

The cloud computing paradigm offers resources (e.g., computing, storage, services, and applications) over the Internet. It provides several deployment models, for example, public or private cloud, and three service models: SaaS (software as a service), IaaS (infrastructure as a service) and PaaS (platform as a service). SaaS is a service model which gives end-users access to cloud-based software or other applications. IaaS provides virtualized computing resources over the cloud, for example, computing power or storage. PaaS is a cloud environment for the development, deployment and testing of applications. The benefits of cloud computing include reduced management efforts, on-demand resource allocation, convenience and flexible pricing based on the usage of services (pay-as-you-go) [5]. In LPWA networks, this is the most common approach, which is not always suitable. Its fundamental limitation is considerable physical distance between the cloud service and an end device, which increases the average network latency and jitter and causes location-unawareness [59]. Connectivity is obtained via an internet connection. Cloud computing may therefore not be suitable for time-sensitive applications, for example, many smart city applications, healthcare applications and vehicular networks.

2.4.2 Edge Computing Architecture

Because the cloud computing paradigm is not an ideal solution for every application, architecture based on edge computing eliminates the weakness of high latency. This elimination is achieved by transferring the computing power nearer to the end devices, i.e., to the edge of the network. *Edge* is a term for the intermediate element between a data source and cloud. The main idea

of edge computing is to perform computation in the proximity of data providers [60]. The data providers in the IoT or M2M (various sensors in most cases) generate an enormous amount of data in every short period. A good example is Google's self-driving car, which produces nearly 1 GB of data every second [6]. Edge computing can save bandwidth and computational time, which is crucial in applications where transmitting such a large quantity of data to the cloud would be impossible.

2.4.3 Fog Computing Architecture

The concept of fog computing was introduced by Cisco Systems in 2012 [13]. Similar to edge computing, the fog computing paradigm provides computation nearer to the end devices, with the benefits of low latency, high bandwidth and location awareness. The term *fog* is an analogy of the closer proximity of data sources, i.e., a cloud situated nearer to the ground. Fog computing is typically cooperation with a cloud which, together with end devices, forms a three-layer architecture. In this manner, tasks requiring low latency or location awareness can be processed at the edge of the network, while the cloud connection can be used for more complex tasks or to store historical data. The key differences between edge and fog computing are as follows[12]:

- Fog computing cooperates with the cloud, whereas edge computing excludes this type of cooperation.
- Fog computing functions with a hierarchical architecture, whereas edge computing tends to consist of a small number of layers.
- Fog computing provides computation, networking, storage and control, whereas edge computing commonly offers only computation.

The distribution of responsible tasks between the fog and a backend cloud is application-specific, and it relies on many factors. This distribution of tasks could be predefined, but it could also be altered dynamically with the network state as a result of CPU load, storage capacity, fault events, cost targets, etc.

3 Machine Learning

This chapter provides an overview of machine learning and related methods. I have included this chapter to establish the following methods in context with the solution proposed in this dissertation.

Machine learning (ML) is a research area in which a machine acquires the ability to modify or adapt its actions to provide better performance in the future without being explicitly programmed. The term *machine* can be understood here as a computer [61]. The basis for ML is data subsequently separated into three datasets: training (to train the model), validation (to show how well the model has been trained as it learns), and testing (to produce the final results). ML typically consists of three phases [62], [63]:

- **Training Phase** – during this phase, the model is trained by pairing the input with the expected output. The training dataset is applied during this phase.
- **Validation and Test Phase** – this phase tests how well the model has been trained in the previous phase and estimates the model's properties (error measures, precision, etc.). The validation and testing datasets are applied during this phase.
- **Application Phase** – in this phase, the model is exposed to real-world data to obtain results.

As for the data, it can be either labelled or unlabelled. Unlabelled data is most often raw data, which consists of samples of natural or human-made items. If a meaning in the form of labels is attached to the unlabelled data, it becomes labelled. The labels are usually defined by humans and considerably more expensive to acquire than unlabelled data. Both labelled and unlabelled data can be applied to the learning model. However, greater accuracy can be achieved by using a combination of labelled and unlabelled data [62].

3.1 Machine Learning Sub-Classes

ML can be divided into several sub-classes. The following is one method of dividing individual ML algorithms [62].

3.1.1 Supervised Learning

Supervised learning is the most common type of ML, which consists in finding the most suitable classifier $f : Y \rightarrow X$ for a given problem [61]. An algorithm which falls into this category is based on creating mappings between input and output attributes, which are then used to generate output for new input data. For supervised learning, the assumptions of what needs to be analysed in the given dataset are known. This type of learning operates with labelled data. If the data label type is categorical, it is a classification problem; if it is numeric, it is a regression problem [64].

3.1.2 Unsupervised Learning

This type of ML is applied if a specific aim that needs to be achieved is unknown, which also determines that the output attributes are not defined. It is used to explore and understand data for building up ML models where the algorithm does not have access to labels because an unlabelled dataset is used. It is therefore associated with the creation of a model after the process of finding similarities in the input data. This learning type is therefore not based on finding mappings between input and output data, but instead analyses the organisation of points in the input data. For example, clustering or outlier detection can be included in the category of unsupervised learning [62], [64].

3.1.3 Semi-Supervised Learning

Semi-supervised learning uses both labelled and unlabelled data. Because the acquisition of labelled data is expensive since the labels are assigned by a specialist, the dataset contains only a small amount of labelled data and a large amount of unlabelled data. The presence of labelled data can significantly improve learning accuracy. The assumptions for unlabelled data must be correct, as any wrong assumptions can invalidate the model. This learning type is inspired by human learning [62], [64].

3.1.4 Reinforcement Learning

Reinforcement learning is based on obtaining a reward or penalty depending on the achieved results. The model is responsible for decisions which maximise performance, for which it then receives feedback in the form of regular rewards. Unlike supervised learning, the results are not immediate and may require a series of steps before the final result is obtained. Ideally, the algorithm produces a sequence of selections that help to obtain the greatest reward [62], [64].

3.1.5 Deep Learning

Deep learning focuses on combining ML with artificial intelligence (AI). It is a widespread technique applied to resolve the most complex AI problems. This area explores the creation of more complex neural networks to solve problems classified under semi-supervised learning and works on datasets which contain only a small amount of labelled data. A network that forms a deep learning model contains multiple layers, which gave this area the adjective *deep*. This category includes, for example, convolutional networks or deep belief networks [62].

3.2 Classification Algorithms

In this section, the selected classification algorithms (classifiers) are further discussed. Later, the selected classifiers are used and validated in the proposed solutions discussed by this dissertation.

Classification is suitable if the data type of a label is categorical and the task of a classification algorithm is to divide the dataset into specific classes.

3.2.1 Decision Tree

The Decision Tree algorithm can be applied to solve both classification and regression problems. It is an algorithm based on a tree-like model, which accepts a vector of attributes at the input and returns a decision, i.e., one of the possible output values. Each inner node of the tree represents a condition and is divided into individual branches. A branch that is not further divided represents the leaf of the tree, i.e., the target value. Alternatively, the leaf may contain a probability vector indicating the probability that the attribute has a particular value [65]. Features in a dataset can be numerical or categorical. The output of a classification decision tree is a particular category (class).

Because most datasets have many features, it is necessary to prevent overfitting. This prevention can be done by defining the minimum quantity of training inputs to use on each leaf. Another option is to determine the maximum depth of the tree, i.e., the longest path from the root of the tree to the leaf. To avoid unnecessary complexity, a decision tree can be pruned to remove branches that use features with low importance.

A significant advantage of decision trees is their simple interpretation, which leads to an easier understanding of the function. However, once we have a dataset with many features, the training time increases as the tree becomes more complex. Another potential problem of decision trees is overfitting, but the improvements described above can be applied to prevent this problem [66].

3.2.2 Random Forest

As already mentioned, the disadvantage of a decision tree is that it is prone to overfitting, which leads to low accuracy when applying a decision tree to a test dataset. A random forest consists of a certain number of decision trees which perform together as an ensemble to achieve better prediction accuracy. The concept behind the random forest classifier is known as the wisdom of crowds [62]. The prediction of the individual trees is combined, resulting in the final prediction of the random forest classifier. Although the predictions of some trees may be wrong, many other trees will predict correctly, leading to the correct decision overall. Random forests can be used with nominal and numerical attributes.

The advantage of a random forest is high efficiency and accuracy even with large datasets. In addition, individual trees are built on bootstrapped subsets, not on the original dataset, which prevents overfitting. Bootstrapping also ensures the uniqueness of individual decision trees in a random forest. The disadvantage of random forest is the more complex interpretation than with a single decision tree, because it aggregates several such decision trees. However, random forests often have superior performance [67].

3.2.3 K-Nearest Neighbor

As with the decision tree and random forest algorithms, k-nearest neighbour (k-NN) can be used to solve classification and regression problems. This algorithm assumes that the similarities are close to each other. Therefore, it is based on finding the nearest neighbour using some of the distance measurement techniques. Since searching for only one nearest neighbour is a problematic task in terms of sensitivity to outliers, the k-NN algorithm is based on searching for k nearest neighbours. However, it is necessary to select an appropriate value of k, as too low values may lead to sensitivity to outliers, while too high values may cause the inclusion of too many neighbours, leading to increased inaccuracy. Selecting an optimal value is therefore necessary. Usually k is selected as a low positive number [62].

Modification of the k-NN algorithm is a weighted k-NN, where neighbours are assigned weights through the kernel function. The kernel function assigns higher weights to neighbours which are closer to each other and lower weights to neighbours which are further away [68].

One of the main parameters affecting the output of k-NN is the distance measurement technique, as there are several commonly used methods. If the algorithm works with numerical attributes, the default method is Euclidean distance, which applies the following formula to calculate the distance [62]. The disadvantage of this method is sensitivity to extreme values within one attribute.

$$d(x, y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}, \tag{3}$$

where x and y are vectors in two-dimensional vector space and n is the number of variables in the vectors.

If the algorithm works with categorical attributes, the default method is Hamming distance. This method is based on comparing values and determining whether they are equal. If the values are equal, the distance is 0, otherwise the distance is 1 [63].

3.2.4 Support Vector Machines

Support Vector Machines (SVM) is a linear classifier used to solve classification problems and is based on the principle of maximising the margin. It is especially suitable for binary classification [69]. It only works with numerical features, for which it is more efficient. In the case of categorical features, it is necessary to convert them into numerical features. Considering two-category classification, SVM divides the space with hyperplane into two regions, each of these regions corresponding to a specific class. The hyperplane can be imagined as a line in two-dimensional space or as a plane in three-dimensional space [62]. The principle is to find a hyperplane for which a maximal margin between this hyperplane and the nearest point of classification remains on both halves. The points located on this hyperplane are also called support vectors.

When a linear boundary dividing the data is not adequate, it is necessary to determine a nonlinear boundary using a kernel function. Kernel-based SVM transforms the input data into a hypothetical kernel (feature) space, where it is possible to perform linear classification [70].

The SVM classifier is one of the best in generalisation and achieves low overfitting. Its main disadvantage is that it is not interpretable.

3.2.5 Naive Bayes

The naive Bayes algorithm is based on the "naive" assumption that all dataset attributes are equally important and independent. Hence, the algorithm received the denomination *naive* because the assumption is met only exceptionally in real data. However, despite violation of the assumption, the naive Bayes classifier is able to achieve high accuracy. Prediction of the classifier is based on the Bayes conditional probability formula, which is used to find the probability of a specific outcome if we know certain other probabilities. The formula has the following form:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{4}$$

where A and B are two events.

The naive Bayes classifier then solves the classification problem to predict the class c based on the provided feature vector $X = [x_1, x_2, \ldots, x_n]$. The equation is as follows:

$$P(c|x_1, \ldots, x_n) = \frac{P(x_1, \ldots, x_n|c)P(c)}{P(x_1, \ldots, x_n)}. \tag{5}$$

However, the simplicity of the naive Bayes classifier may lead to a state in which a given attribute value never occurs in the context of a particular class in a training dataset. In this case, the conditional probability has a zero value, and if it is multiplied with other probabilities, a zero value is returned, leading to an error in the overall result. The solution to this problem is Laplace smoothing, where a certain low number (usually 1) is added to each cell in the frequency table, which ensures that each class-attribute combination is at least 1 [71].

The naive Bayes classifier is used to solve classification problems, and its advantage is that it can build a good model even with a small dataset.

3.2.6 Artificial Neural Networks

The inspiration for the artificial neural networks (ANN) classification technique was the structure of biological neural networks in mammalian brains. The general function of neurons is to receive input signals and produce a response. Each biological neuron has dendrites which receive information from other neurons, at most one axon to transmit the output, and contact points between different neurons called synapses. Synapses increase or decrease the transmitted signal. Artificial neurons (perceptrons) adopt this concept, and each neuron has defined inputs (x) and outputs (y). An activation function is responsible for generating outputs for different neural

inputs. Typically, individual inputs have associated weights (w) which multiply the incoming information by the corresponding value. Output can be further modified by a value called *bias* [72], [73].

Sigmoid neurons are often used in ANNs to provide output which is smooth, real-valued and therefore a bounded function of all inputs (n). The following equations define the output of a neuron which uses the sigmoid activation function [62]:

$$z = bias + \sum_{i=1}^{n}(w_i x_i), \tag{6}$$

$$y = \frac{1}{1 + e^{-z}}. \tag{7}$$

Neurons can be connected to form an ANN, which allows the processing of inputs that the network does not yet know. An ANN consists of several layers which can be divided into three groups. The first group is called the input layer and contains several neurons, each passing a specific attribute of a dataset to the network. The second group is called the hidden layer. A neural network can be formed with only one hidden layer, or it is possible to have multiple hidden layers, where each layer can be responsible for specific learning. The last group is called the output layer and contains the output of the neurons presented in the final hidden layer. Its role is to identify the resulting classes [74]. Neurons from the input layer are connected to each neuron in the first hidden layer. The first hidden layer neurons are connected to each neuron in the next hidden layer. The neurons in the final hidden layer are connected to each neuron in the output layer. An ANN where neurons in the layers do not form a directed cycle and where information moves only forward (from the input layer, across the hidden layers, to the output layer) is called a feedforward neural network [75].

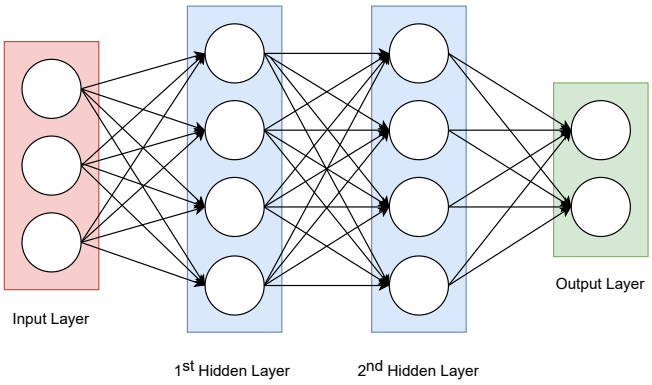

Figure 5: Diagram of a neural network.

For an ANN to provide the most accurate results, the weights of connections between individual neurons must be correctly set. These weights are adjusted during the learning process to determine their correct values. One of the learning methods in a multi-layer ANN is back-

propagation, which attempts to improve the initial randomly generated weights of individual connections. The result of this adjustment is minimisation of the error between the real and expected output of an ANN [72]. The diagram in Figure 5 illustrates a feedforward neural network (FFNN) with two hidden layers, where the nodes represent individual artificial neurons.

Due to the nature of multi-layer ANNs, the correct choice of network topology is essential in achieving optimal results. If an ANN contains too few neurons, it leads to the inability of a neural network to correctly learn the presented patterns. In the case of too many neurons, a neural network tends to reproduce the learned patterns too accurately, limiting its ability to evaluate new input correctly.

3.3 Model Evaluation Metrics

Once a model has been trained, certain metrics are required to evaluate whether the results obtained with this model are correct. For this purpose, the accuracy and error rate can be evaluated. General accuracy can be calculated according to:

$$A = \frac{N_{correct}}{N_{total}} \cdot 100\ [\%], \tag{8}$$

where $N_{correct}$ is number of correctly classified inputs and N_{total} is the total number of inputs [76].

In analysing the results, it is much more appropriate to proceed more deeply than evaluate unweighted accuracy. It is possible to construct a confusion matrix, where the individual columns represent the predictions obtained with a specific model and the rows represent true values. A confusion matrix is a key element in defining the performance metrics of a model. Values on the leading diagonal of a matrix are those which are predicted correctly. Values outside the diagonal are predicted incorrectly [64]. The pattern of a confusion matrix is illustrated in Figure 6.

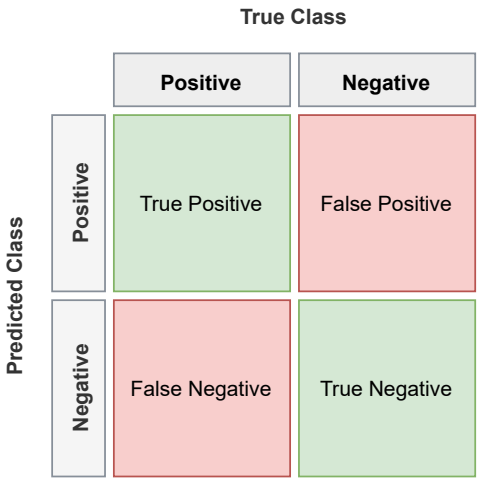

Figure 6: Confusion matrix.

Four different error metrics are the most usual in classification problems [77]:

- **Accuracy** – the percentage expression of correct predictions. The equation for accuracy is:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \cdot 100\ [\%] \tag{9}$$

- **Recall (Sensitivity, True Positive Rate)** – the percentage expression of positive cases that the model was able to predict. The equation for recall is:

$$Recall = \frac{TP}{TP + FN} \cdot 100\ [\%] \tag{10}$$

- **Specificity (True Negative Rate)** – the percentage expression of negative cases that the model was able to predict. The equation for specificity is:

$$Specificity = \frac{TN}{TN + FP} \cdot 100\ [\%] \tag{11}$$

- **False Positive Rate** – the percentage expression of negative cases that are mistakenly predicted as positive, with respect to all negative cases. The equation for false positive rate is:

$$FalsePositiveRate = \frac{FP}{FP + TN} \cdot 100\ [\%] \tag{12}$$

- **Precision** – the percentage expression of positive predictions that were correct. The equation for precision is:

$$Precision = \frac{TP}{TP + FP} \cdot 100\ [\%] \tag{13}$$

In addition to calculating the criteria for evaluating predictions, a graphical visualisation with receiver operating characteristic (ROC) curves can be used to evaluate and adjust a binary classifier. It is a graphical plot where the x-axis shows the percentage of false positives (1-specificity) and the y-axis shows the percentage of true positives (recall). Figure 7 provides an example of this type of curve.

Each run of a classifier generates a single point on the ROC plot. As follows from the nature of this chart, an ideal classifier would produce a point (0, 1), which means 100% true positive and 0% false positive rates. Conversely, the worst classifier would produce a point (1, 0), which represents a 0% true positive rate and a 100% false positive rate. This implies that the nearer the results are to the upper left corner, the better the classifier performance. Any classifier lying on the diagonal from (0, 0) to (1, 1) has a random chance for the accuracy of the results. The area under the curve (AUC) is the area measured under the ROC curve [64].

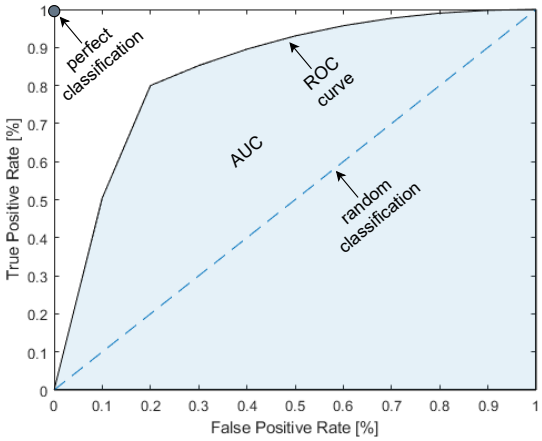

Figure 7: Example of an ROC curve.

3.4 Cross-validation

Cross-validation is a technique for evaluating the effectiveness of ML models. This evaluation is then essential for selecting the appropriate ML algorithm to solve a specific prediction problem. The cross-validation technique does not apply the entire dataset for training and testing, only a certain subset. In this manner, cross-validation results can show whether the ML model is overfitting, underfitting, or well generalised. Several methods can be used to select subsets from a dataset for training and testing [78], [79]:

- **Hold-out** – is the simplest method. The dataset is divided into two subsets, where one subset is used for training the given classifier and the other subset for testing. The advantage of this method is low computational time. However, the results strongly depend on the data contained in the training and testing dataset.

- **k-fold** – improves the hold-out method. The dataset is divided into k sub-sets, where one subset is used for testing and the remaining $k - 1$ subsets for training. This procedure is then repeated, each subset being used for testing only once. Usually, a value of 5 or 10 is given to k. The advantage of this method is a relatively precise estimation of accuracy. The disadvantage is longer computation time. Figure 8 depicts the principle of the k-fold method.

- **Repeated random sub-sampling** – the dataset is randomly divided into training and testing sets according to user-defined sizes. The advantage of this method over the previous method is the independence of the ratio between training and testing subsets on the number of folds. The disadvantage is the randomness that may cause some data to appear multiple times in subsets whereas some data may not be used at all.

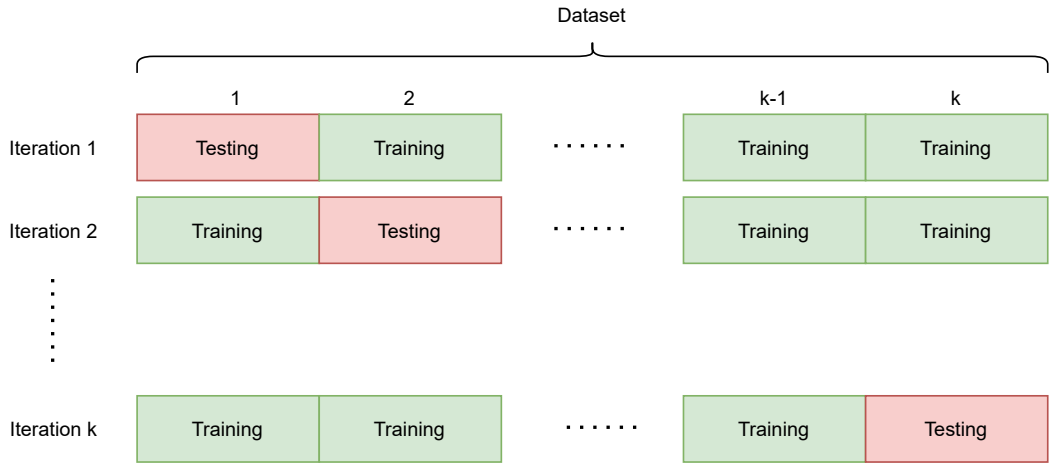

Figure 8: k-fold cross-validation principle.

- **Leave-one-out** – is a modification of the k-fold method, where k corresponds to the number of samples in the dataset. In this manner, one sample of the dataset is selected for testing, and the rest is used for training. As follows from the nature of this method, it is time-consuming, although it achieves the best results.

4 Aims of Dissertation

LPWAN technologies such as LoRa, SigFox and NB-IoT are based on the cloud computing paradigm, mainly due to on-demand services and scalability features. However, the cloud computing paradigm also has drawbacks which make it insufficient for some IoT use cases. The main problem is that all the computation and storage of data inherently occur in a cloud. Fog computing, however, introduces an intermediate layer which can serve for computation, storage or filtering of data. In this manner, not all data are computed within a cloud, and this solution can thus attain greater efficiency, selective data privacy and correct reactions for time-sensitive applications.

Based on the researched information and state of the art, the aims of the dissertation are as follows:

- Explore and design an optimised IoT network architecture with integrated components, including communication with a cloud service.
- Research a new method based on machine learning to control data transfer between the fog layer and a cloud service.
- Verification of the design features, implementation of the critical components, and comparison with existing IoT architecture solutions.

The aims of the dissertation were approved by a commission on rigorous examination, held in June 2020.

By applying the fog computing paradigm to the architecture, response time and thus the QoS can be influenced in several ways. Because location awareness is supported, the fog gateway can better respond to current RF conditions and adjust the radio parameters individually for different end devices or improve the existing ADR mechanism, which is discussed in Section 2.3.3. Optimisation of these parameters has been explored in many studies, presented in Section 2.1 which discusses the existing studies. With a suitable choice of configurable LoRa modulation parameters for individual end devices, a specific end device can be prioritised to ensure quicker and more reliable network transmission of messages from the device. Another fog computing optimisation method is decoding and decrypting the message payload on a fog gateway, which reduces processing time since messages do not need to be forwarded to a public cloud. Fog message processing also increases reliability since processing can continue even after a connection failure between the gateway and cloud service. Moreover, fog computing allows selective message privacy according to a specific end device.

Because the IoT network can be optimised by applying the fog computing paradigm in many different ways, the detailed exploration, design and implementation of all methods are beyond the scope of this dissertation. Therefore, in the following chapters, I focused mainly on decoding and decrypting the message payload on the fog gateway and researching related optimisations, because the State-of-the-Art section indicated that similar research does not exist.

To simplify the explanations of individual principles and procedures, I apply the term “private data” in the following chapters interchangeably with the term “time-sensitive data”, according to the meaning of the fog computing paradigm. Both private and time-sensitive data need to be processed expeditiously and privately by a fog node, while the remaining traffic is processed in a cloud.

Overall, the dissertation contributes with an optimised IoT network architecture that applies the fog computing paradigm. Because a fog gateway can decode and decrypt the received message payloads and make decisions according to the acquired data, I have proposed new methods of controlling the transfer of data computation and storage between the fog and cloud layers. These methods operate according to static and dynamic information. Static information is predefined by the end device owner, and the dynamic information depends on the result of a machine learning classifier.

5 Proposed Network Architectures

This chapter deals with the first aim of the dissertation. It contains a description and comparison of three proposed IoT network architectures and a discussion of which one is optimal. That is an essential component which affects the overall properties available to end nodes.

5.1 General Fog Computing Architecture

Because the resulting architecture adopts the fog computing paradigm, it contains three layers. Generally, the bottom layer includes IoT end devices designed to collect information from the surrounding environment. The middle layer is called the fog layer and includes fog nodes designed for autonomous decision-making based on processed data. The upper level is the cloud layer, which consists of a cloud server as an addition to the fog nodes [12]. Figure 9 illustrates the general three-layer architecture.

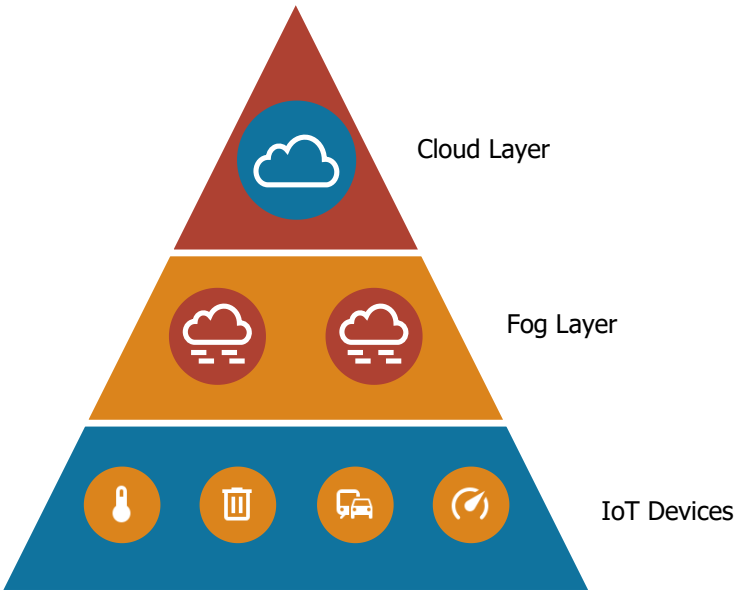

Figure 9: Three-layer fog computing architecture.

The following properties must be preserved to adopt the fog computing paradigm:

- Location-awareness – the closer proximity of end devices allows the network to react to various end device needs and adapt to their requirements.
- Efficiency – is given by full cooperation between the fog and cloud layers. If required, computation and storage can be transferred from the fog layer to the cloud layer and vice versa. This transfer can happen in many circumstances, for example, during a situation when resources are unavailable.

- Lower latency – lower latency can be achieved through processing at the network's edge and applied to add support for time-sensitive applications, for example, alarm systems or health monitoring systems.

The properties mentioned above are the main reasons for implementing fog computing in the IoT, and their preservation is therefore essential.

5.2 Standard LoRaWAN Architecture

As mentioned in Section 2.3.3 describing LoRaWAN technology, LoRaWAN enables the creation of private networks, and because it is open technology, it allows modification of the network architecture. Standard LoRaWAN network architecture consists of the following components [80]:

- **End devices** – sensors or actuators capable of wireless communication with gateways using LoRa RF modulation. The end devices are mostly battery-powered.
- **Gateway** – serves as a bridge between a low power LoRaWAN network and high bandwidth IP network such as Ethernet, Wi-Fi or Cellular. The gateway works as an entry point to the network for end-nodes, converts their RF packets into IP packets, and relays them.
- **Network server** – manages the network, filters redundant received packets, performs security checks and controls adaptive data rate, etc. It serves as an interface to the application server.
- **Application server** – a final destination for data. It determines what to do with the received data; for example, insertion into a database or visualisation in a certain web application.

The LoRaWAN standard network architecture depicted in Figure 10 shows that a gateway is the closest component of the architecture from the point of view of the end device. The gateway acts as a passive area of the network from the perspective of data processing and computing. It simply receives and transforms data into a form suitable for subsequent transmission and processing, then forwards the data to the network server. The application server is located somewhere in the Internet and solves data processing, computing and storage. The network architecture corresponds to the cloud computing paradigm scheme.

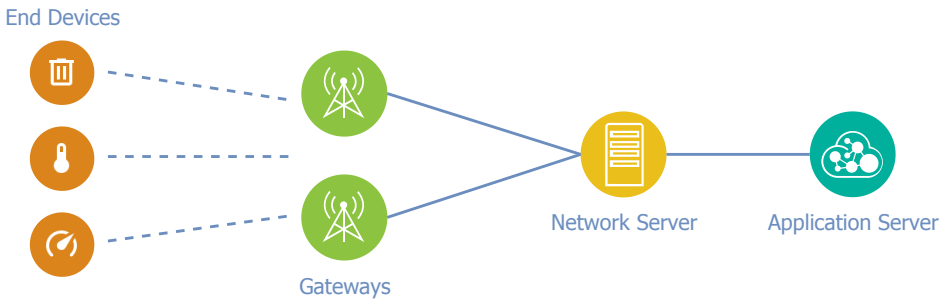

Figure 10: Standard LoRaWAN architecture.

5.3 Proposed Network Architectures

The hypothesis of the dissertation is as follows. If we apply specific changes to basic LoRaWAN elements, we can transform them into an architecture which supports the fog computing paradigm. The main challenge is application payload encryption, which is end-to-end encrypted between the end device and the application server. The gateway itself does not have access to decrypted data, and as a result, data storage and processing are not possible.

Two keys are stored in end device memory:

- **Network Session Key (NwkSKey)** – This key is used by both the network server and end device to calculate and verify the message integrity code (MIC) and ensure the data integrity in data messages. It is stored in the end device and network server memories after a successful activation process.
- **Application Session Key (AppSKey)** – This key is designed for the decryption and encryption of the application payload. It is stored in the end device and application server memories after a successful activation process.

The integrity of the application payload is not protected, and therefore, the risk arises that the network server may change the content of a data message. However, network servers are commonly considered trustworthy. Both keys are specific to each end device. More information about keys and the decryption process is presented in 7.2.

The following section of the dissertation describes three different proposed network architectures. These are further compared from multiple perspectives to determine the optimal one.

5.3.1 Architecture A

In the standard LoRaWAN architecture, all gateways are separated units consisting of an RF concentrator, an antenna and a computing unit. The RF concentrator is able to receive packets sent by different end devices while the computing unit converts and relays the packets. The

network and application servers are separated from the gateways, and they both run on dedicated hardware accessible via the gateway through an internet connection.

The proposed Architecture A is based on the idea that all standard LoRaWAN network elements can be integrated into a single unit, called a *fog gateway*. In this manner, each gateway, in addition to receiving, converting and relaying packets, performs the functions of private network and application servers. This adjustment could lead to quicker responses to current conditions according to actual data.

However, fog processing is not always necessary, and sometimes it is not even possible. Especially when a great amount of data is being transferred, a disproportionate load could arise considering the limited computing resources of fog gateways. To solve the problem, a remote cloud server is presented in this architecture as an addition to fog gateways. In the entire LoRaWAN network, only one cloud server is available for all fog gateways. This cloud server provides a web-based graphical interface and APIs to manage fog gateways, devices and applications. A diagram of Architecture A is presented in Figure 11.

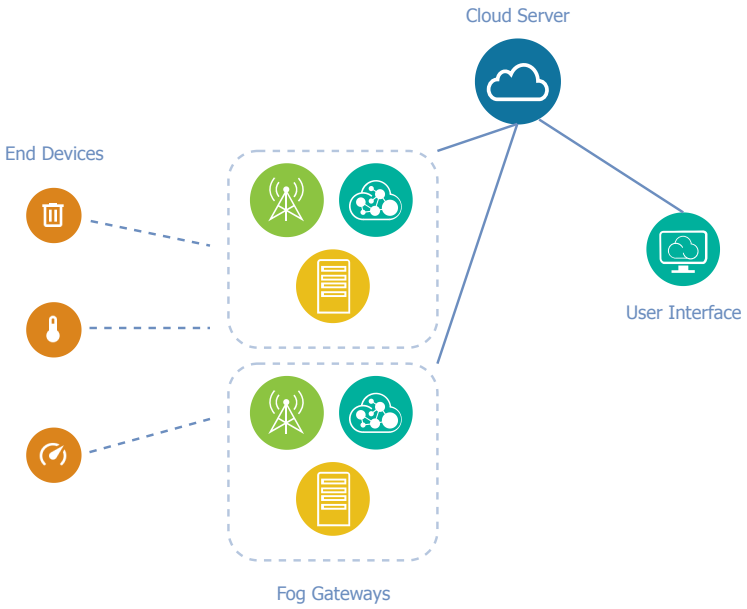

Figure 11: Diagram of network Architecture A.

5.3.2 Architecture B

Architecture B applies a slightly different view on optimisation. Unlike Architecture A, the proposed Architecture B does not require integration of the network and application servers into each fog gateway, but it is integrated into only one gateway in the network. The remainder of the network's gateways function identically to the gateways in a standard LoRaWAN network and can only receive, convert and relay packets. For the correct operation of this architecture, it

is necessary to involve a certain communication model. Because a single complex node controls multiple nodes with limited logic, the master/slave model comes into play.

Master/slave is a communications model in which a single node (master) controls multiple slave nodes and mediates communication between them. These master and slave nodes perform as gateways, in this case. The slave gateway operates as a standard LoRaWAN gateway and transfers all traffic to the master gateway. The master gateway integrates the network and application servers to decrypt the payload and control packet relaying by the slave gateways.

Due to the limited computational and storage resources of a fog gateway, a remote cloud server forms a part of the architecture. This cloud server is connected to the master gateway via the Internet and is used to store long-term data. A diagram of Architecture B is presented in Figure 12.

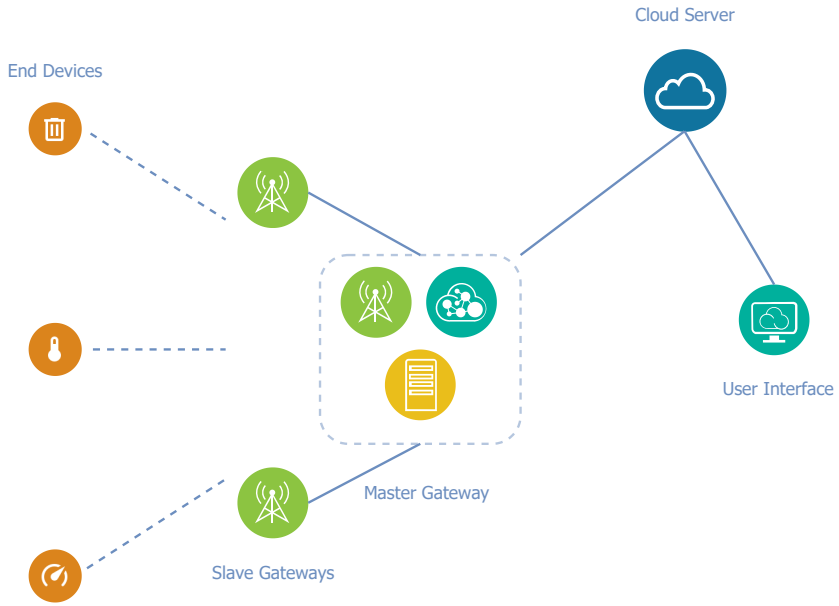

Figure 12: Diagram of network Architecture B.

The main advantage of this architecture is the elimination of the need for an internet connection for each gateway. An internet connection is necessary while applying the standard LoRaWAN architecture because the network and application servers are situated in the cloud. Only the master gateway is connected to the Internet in Architecture B since it communicates with the remote cloud server.

5.3.3 Architecture C

Architecture C applies an entirely different principle to the previous architecture optimisations. Instead of reallocating individual components of the standard LoRaWAN architecture, the main difference here is the communication between a gateway and the application server. In this

manner, Architecture C does not differ from the standard LoRaWAN architecture in terms of structure.

As mentioned above, due to end-to-end encryption, a LoRa gateway itself does not know the session keys and cannot access the decrypted payload since it is the application server which performs decryption. The core concept behind this optimisation is negotiation between a specific fog gateway and the application server to obtain and subsequently store the necessary keys for the individual end devices in the secured database located on the fog gateway. In this manner, a fog gateway sends requests to the application server only at first communication with a specific end device or in the case of need. Interaction with the application server is thereby reduced to a minimum. A diagram of Architecture C is presented in Figure 13.

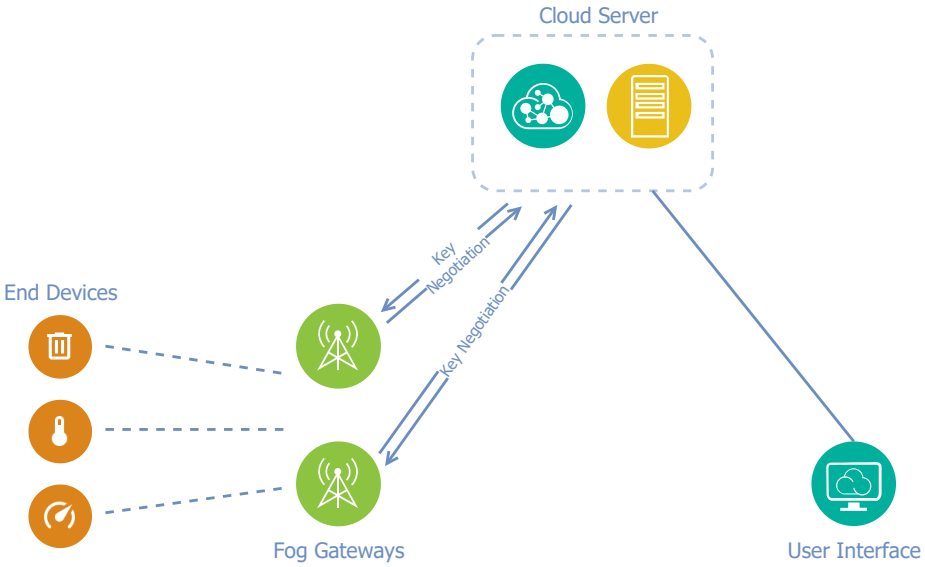

Figure 13: Diagram of network Architecture C.

Communication occurs as follows:

1. After a successful process of personalisation and activation of a specific end device, this end device can send a message via the LoRaWAN network

2. When a fog gateway in this LoRaWAN network receives the message, it checks the DevAddr in the message header and compares it to the content in the local database.

3. Suppose the specified DevAddr is present in the local database. In this case, the end device has already communicated with this fog gateway, and the session keys can therefore be loaded to decrypt the message payload. If the specified DevAddr is not present in the local database, this represents the first communication with the specific end device. In this case, the fog gateway must negotiate the session keys with the application server and then transfer and store them in the private local database for further use.

4. When the session keys are loaded, the fog gateway can decrypt and process the message payload.

5.4 Comparison of the Proposed Architectures

A comparison of the proposed architectures is presented in this section. To obtain fair results, comparisons both in terms of execution time [Jal21] and functional properties were performed.

5.4.1 Comparison in Terms of Queuing Theory (Execution Time)

Since an IoT network can be viewed as a particular queuing system, it is possible to use queuing theory and compare the proposed architectures in terms of service time. Individual messages from end devices can be seen as requests to be served. The LoRa gateway, network server and application server are queuing nodes. Assuming that the gateway can receive up to eight LoRa messages simultaneously with different SFs on different channels [82], the queuing node representing the gateway has eight servers which can serve requests concurrently. To establish the same conditions for each of the proposed architectures, I assumed that all end devices fall into Class A, use the ABP activation method and send messages without the need for confirmation. The downlink response is therefore unnecessary.

Since the considered physical layer (LoRa) is the same for all architectures, the simulation is mainly focused on the processing of data by individual elements of the architecture. For correct simulation of data processing, it is necessary to model the behaviour of an individual end device while sending messages in a certain manner. It is often the case that IoT devices such as sensors, trackers and other similar devices send their data periodically in specific time intervals. Aggregated traffic from a large number of the mentioned IoT devices can be seen as a superposition of deterministic point processes. If these individual point processes are considered independent and the individual devices generate their messages independently, this operation can be modelled according to the Poisson process. We can describe aggregated traffic using interarrival times, which are the intervals between end device messages from a LoRa gateway perspective. Because of the presence of this periodicity during message transmission by particular end devices, an error element is involved in the Poisson approximation. The authors of publication [83] assumed that in the case of lower loads with an average use of 0.55 in a comparison of $nD/D/1$ and $M/D/1$ systems, the difference between them was negligible. However, if we consider the real environment of an IoT network, messages and their payloads have varying sizes. Service time is thus not deterministic but variable and corresponds to an exponential distribution. The individual queuing nodes in this dissertation are therefore modelled according to the M/M/n model, where n is selected according to the specific simulated component of the network.

The problem at this point is that generally valid values for the required service times do not exist, but they can be obtained through experimental measurements on real hardware. Of

course, the results depend on the computational performance of these components, but assuming identical hardware for each of the proposed architectures, the simulation results should not possess any distortions.

Experimental Measurement

To obtain the service times of the components of each architecture, a series of experimental measurements were conducted for this study. As hardware for the LoRa gateway, a Raspberry Pi 4 Model B with 2 GB of RAM and Raspbian OS was used. I selected a virtual machine with 2 GB RAM and OS Debian, which offered sufficient computational power to accomplish the function of the network and application servers. Because message propagation time in the radio environment was not essential for these experimental measurements, execution time was measured from the moment the gateway received the message.

For the proposed architectures A and B, the gateway (packet forwarder) execution time (T_{pktfwd}) was measured until the message was forwarded to the network server, which continued with message processing. The network server's execution time was measured from the moment when this server received the message until the moment when the message was processed and forwarded to the application server. Execution time by the application server was measured from the moment when the message was received until the moment when the decrypted payload was added to the database. The experimental measurements were conducted for 1000 messages. From these measurements, I calculated the average times. The results of these experimental measurements are shown in Table 1.

Table 1: Execution times of the components.

Component	Average Execution Time (ms)
Packet forwarder (gateway)	0.16
Network Server	194.62
Application Server	15.16

Proposed Architecture C applies a different method since the packet forwarder is specifically modified to perform fog computing operations. In this case, the packet forwarder was designed to forward data to a network server and additionally decode the messages and decrypt the payload. It is therefore necessary to add the execution times of individual fog computing functions performed by the gateway, i.e., message decoding (T_{decode}), payload decryption ($T_{decrypt}$) and subsequent storage of the payload to the database ($T_{database}$), to the experimental measurement result of the execution time by the packet forwarder.

$$T_{fog} = T_{decode} + T_{decrypt} + T_{database}$$
$$T_{total} = T_{pktfwd} + T_{fog}$$

The above formulas can be used to obtain the total message execution time necessary to perform fog computing functions related to the proposed Architecture C. As in the previous case, a Raspberry Pi 4 Model B was used as the gateway hardware for the experimental measurements. One thousand measurements of the individually mentioned functions were gradually performed. The algorithm generated a pseudo-random string of at least 20 characters for each decoding and decryption process measurement. The decryption function is described in Section 7.2. Storage of metadata and payload to the database was subsequently measured to obtain the final time component. The results of these experimental measurements are shown in Table 2.

Table 2: Execution times of the fog gateway.

Time component	**Average Execution Time (ms)**
T_{decode}	1.95
$T_{decrypt}$	1.19
$T_{database}$	7.05
T_{fog}	10.19
T_{total}	10.35

Simulation Preface

In the previous step, I obtained the required execution times for each component of the architecture, and it was possible to proceed to the simulation itself. For the simulation, I used the graphical programming environment Simulink, which is an extension of the software tool Matlab. The Time-Based Entity Generator block generated individual service requests for each of the simulated architectures. The integration time corresponded to the Poisson distribution, where the mean, for the sake of the simulation, was set to 0.1. I selected this value to keep use of the packet forwarder low (around 20%), and therefore the resulting time was not affected by blocking requests in the queue. To reduce the complexity of the simulation, I used only the data privacy value as an input to decide whether to select fog computing or cloud computing. In the case of a private message payload, the message was processed by the fog gateway. In the case of a public message payload, the message was transferred to the public cloud for further processing. All simulations were performed for 10,000 messages.

Architecture A Simulation

For proposed Architecture A, the n-server block representing the packet forwarder on the gateway

side was the first block included in the simulation. Here, the average service time was set to 0.16 ms and the number of lines (n) to 8. The request was then passed to the infinite server ($n{=}\infty$) block, which represented a network server with an average service time of 194.62 ms, followed by a chained infinite server block, which represented an application server with an average service time of 15.16 ms. After processing by these chained queuing systems, the application server evaluated whether the data was private. If true, the current processing time was stored, and the simulation cycle ended. If it was not private, the application server forwarded the message to a remote cloud server. A delay on the line of 0.5 ms thus contributed to the total execution time. Because the fog gateway forwarded the data to the cloud server after processing by the network and application servers, the cloud server only inserted the data into the database. Its average execution time was therefore brief and set to 7 ms, according to the results of the experimental measurements. When the decrypted payload was stored in the database, the total execution time was then saved, and the simulation cycle ended. A diagram of the simulation model is shown in Figure 14.

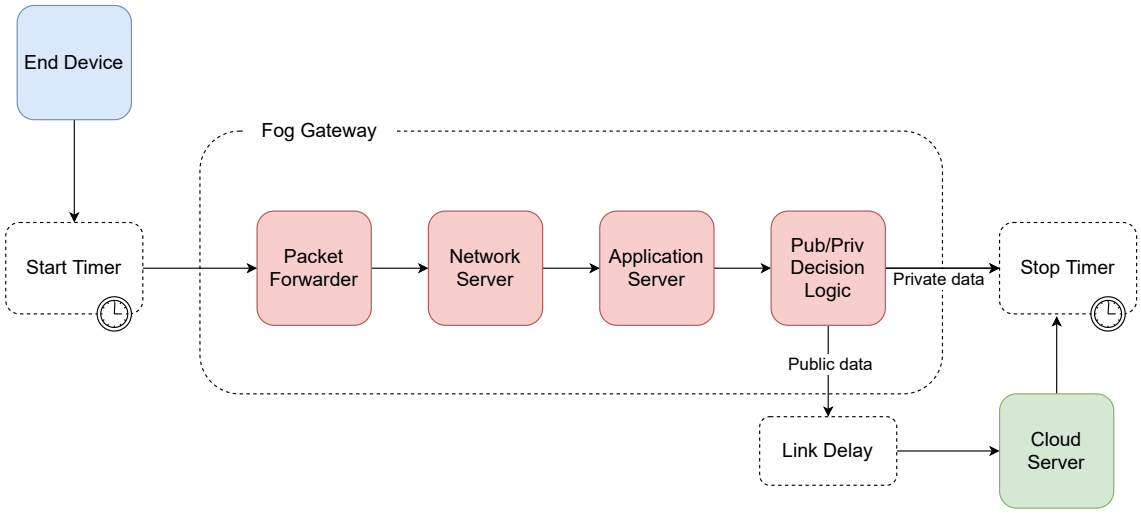

Figure 14: Diagram of the simulation model of Architecture A.

Since Architecture B is only a variation of Architecture A in which the network and application servers were not implemented at each gateway but only at a master gateway, it was not necessary to perform a simulation for this architecture. The resulting execution times differed only in a line delay of an expected 0.5 ms during transmission between the slave and master gateways, which was not needed in Architecture A.

Architecture C Simulation

For proposed Architecture C, the n-server block representing the packet forwarder was the first block included in the simulation. The average service time of this block was set to 0.16 ms and the number of lines to 8. The data was then evaluated whether it was private or public. In the

case of private data, the processing request was routed to the chained infinite server blocks for decoding, decryption and insertion into the local database. These blocks used average service times T_{decode}, $T_{decrypt}$ and $T_{database}$, respectively. This step ended message processing in the case of private data. The total execution time was then saved, and the simulation cycle ended. In the case of transferring public data, the message was forwarded to a cloud with a link delay of 0.5 ms. The cloud consisted of network and application servers with average service times of 194.62 and 15.16 ms, respectively. When processing on these queuing systems was completed, the total execution time was then saved, and the simulation cycle ended. A diagram of the simulation model for proposed Architecture C is shown in Figure 15.

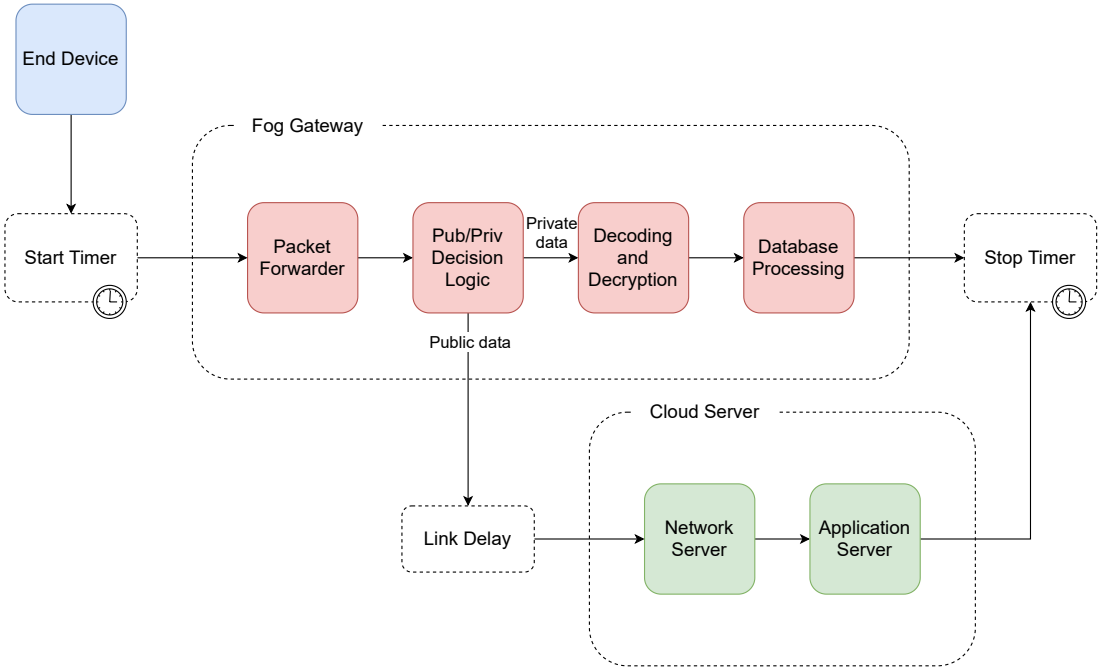

Figure 15: Diagram of the simulation model of Architecture C.

Results

A comparison of the total service times resulting from the simulation, with a 5% probability of public messages, is shown in Figure 16. The chart shows that Architecture C has a much shorter total message execution time, the average service time being 19.44 ms compared to 209.96 ms in Architecture A. That, of course, applied only when private data predominated and primary data processing was done by the fog gateway, which is the principle of fog computing and the type of solution proposed in this study. Otherwise, the execution times for architectures A and C were almost identical. We can observe significant jumps in execution time in Architecture C, illustrating the transmission of messages to a public cloud server, where messages are then

further processed. If the simulation is limited only to fog gateway processing, these jumps do not occur.

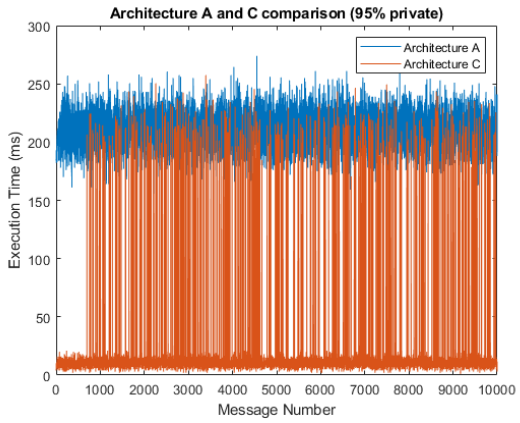

Figure 16: Comparison of architectures A and C (95% private messages).

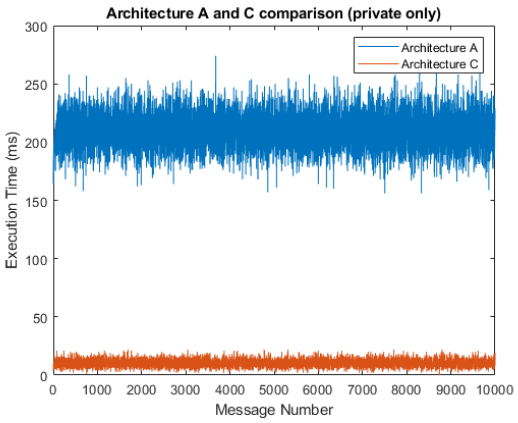

Figure 17: Comparison of architectures A and C (100% private messages).

A simulation without transmission to the cloud server is charted in Figure 17. In this case, the average service time was 10.28 ms in Architecture C compared to 209.81 ms in Architecture A. This is the best-case execution time (BCET), as the discussion concerns the shortest time for any possible combination of inputs. The chart shows that the BCET for Architecture C was lower because of the absence of significant jumps and because the data did not need to be processed by the network and application servers. In Architecture A, the BCET was similar to the 5% probability of public messages. The similarity is a result of the presence of the network and application servers at each fog gateway. In this manner, data processing was mainly performed by the fog gateway, and the cloud server only performed insertions into the local database. The worst-case execution time (WCET) occurred when all data was set to public, and in this manner, the fog computing operations remained idle. In this case, the WCET of Architecture C was 210.41 ms, a result similar to the WCET of Architecture A with 217.78 ms.

Figures 18 and 19 display the respective histograms for Architecture A and Architecture C and a 95% probability of private messages. A comparison of these histograms reveals that while the total service time values for Architecture A were not scattered significantly, the histogram for Architecture C was dispersed, and the variance of these values was much higher as a result of the already mentioned difference in private and public processing. Figures 20 and 21 display the respective histograms for the simulations of Architecture A and Architecture C and a 100% probability of private messages. These histograms suggest that while the spread of the histogram for Architecture A was almost the same in both cases, the variance of values presented in the histogram for Architecture C and a 95% probability of private messages was much higher than the case of private only processing.

To approximate the resulting service times for any hardware solution, we can select a time base T corresponding to the BCET of Architecture C (10 ms). Then, the resulting times can be

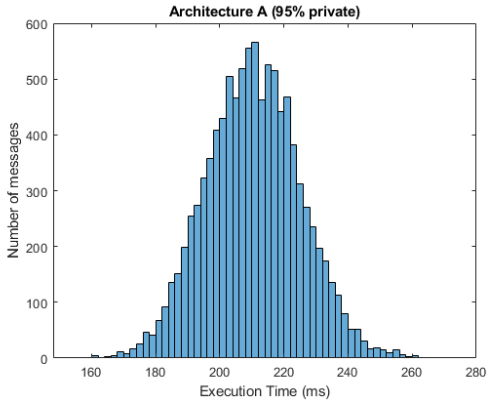

Figure 18: Histogram for Architecture A (95% private messages).

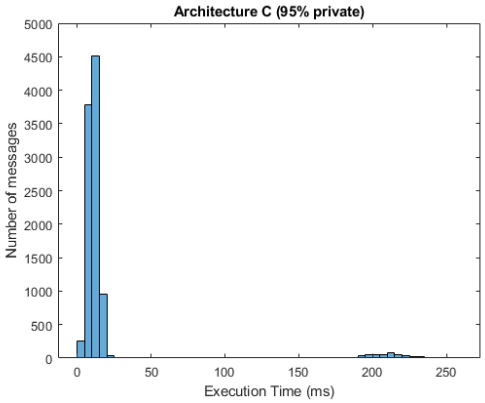

Figure 19: Histogram for Architecture C (95% private messages).

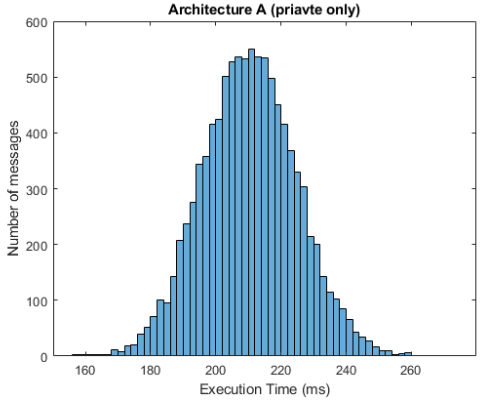

Figure 20: Histogram for Architecture A (100% private messages).

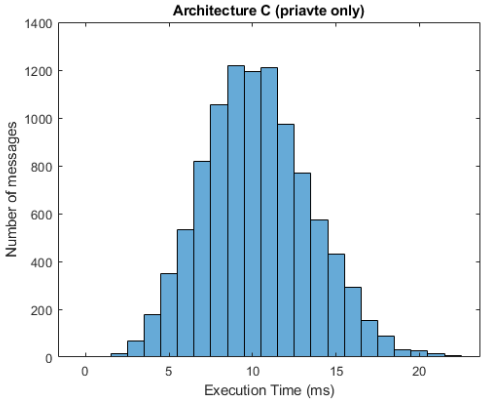

Figure 21: Histogram for Architecture C (100% private messages).

expressed as a multiplication of the time base. Table 3 provides a summary of the approximated resulting times obtained with the simulations. The constant T_{link} is the link delay during transmission between the slave and master gateways. As we can see, proposed Architecture C achieved the best results in all cases. However, the table also indicates that the gap between the BCET and WCET for this architecture is wide. The simple rule is that the more public data transferred to the network, the closer the total execution time approaches the WCET.

Table 3: Comparison of results from the simulation.

Feature	**Architecture A**	**Architecture B**	**Architecture C**
Average execution time	$21 \cdot T$	$21 \cdot T + T_{link}$	$2 \cdot T$
BCET	$21 \cdot T$	$21 \cdot T + T_{link}$	T
WCET	$22 \cdot T$	$22 \cdot T + T_{link}$	$21 \cdot T$

5.4.2 Comparison in Terms of Functional Properties

Since each of the proposed architectures attempts to solve the optimisation problem differently, it is essential to compare the architectures in terms of functional features. These features affect the potential use cases of the architectures and the possibility of integrating the fog gateway into existing infrastructures.

Architecture A is notable for integrating network and application servers into each fog gateway. This approach eliminates the need to forward each message from an end device to the remote network and application servers. The problem of unavailable connections can be prevented in this manner. Since not every message needs to be forwarded to a public cloud server, the privacy of the transmitted data is maintained, which slightly reduces message processing time. However, integrating the network and application servers into the fog gateway presents several issues. The first problem is a significant increase in computational load on the fog gateway, placing a greater demand on its performance, as each time a message is received from any end device, the fog gateway performs tasks related to the network and application servers in addition to standard LoRa gateway operations. The other problem is that several gateways in a multi-gateway LoRaWAN can receive an identical message. Hence, a network server performs redundant (duplicate) message filtering. In a standard LoRaWAN architecture, only one shared network server is available to the gateways in a single network. Architecture A, however, contains a private network server for each fog gateway, and thus the network server cannot filter redundant messages. This leads to the necessity of some form of message synchronisation between the fog gateways in Architecture A. If this type of synchronisation is not present, non-overlapping coverage must be provided by deploying only one fog gateway to cover a given area or by ensuring sufficient distance between individual fog gateways.

Architecture B applies a master/slave approach. The slave gateways operate under a standard LoRa gateway mode and only forward data to the single master gateway integrating the network and application servers. This approach is suitable if an internet connection cannot be supplied for each gateway. A gateway which does not have an internet connection can only communicate with a locally available master gateway. The master gateway analyses whether an end device message is private or public, and according to this information, it either stores the message payload in its local database or forwards the message to the cloud server. This architecture is simply a modification of Architecture A, with the advantage that the network and application servers need not be implemented into each gateway. Providing synchronisation between the individual gateways is not required, but it is necessary to guarantee connectivity to the master gateway for all slave gateways.

Unlike previous architectures, proposed Architecture C does not require constant connectivity to the network and application servers for proper functioning. If the fog gateway has all available end device addresses and their session keys stored in its local database, it can process all messages locally. The fog gateway can therefore operate entirely without cooperation with

the network and application servers. Another significant advantage is easy integration of the fog gateway into the standard LoRaWAN architecture. This solution is especially beneficial if only one fog gateway is sufficient for proper coverage. If one fog gateway receives a private message, it immediately initiates processing. Let us suppose that multiple fog gateways receive an identical private message from a single end device. In this case, each gateway performs decoding, decryption and subsequent storage of the decrypted payload in the local database. It is clear that this is not an ideal solution. It is therefore necessary to avoid overlapping areas of coverage or implementing a communications mechanism between the fog gateways.

The mechanism could be based on the idea that the fog gateway, which receives a message first, sends a notification via the MQTT protocol to all fog gateways in the coverage area. Notified gateways do not process the message; only the fog gateway which sent this notification proceeds with processing. When this method is used, the decrypted payload of a single end device is fragmented between the local databases of the various fog gateways because each message can be processed by any fog gateway in the range.

Another method of preventing the redundant processing problem is to reserve an individual fog gateway for processing only messages from specific end devices. If a fog gateway receives a message from the end device managed by this fog gateway, it can decode, decrypt and store the payload in the local database. In the other case, the fog gateway immediately terminates further processing of the message. In this manner, the payload from one end device is stored only in the local database of the given fog gateway which manages this end device. This method solves the fragmentation of a decrypted payload from a single end device into the local databases of many fog gateways.

5.4.3 Summary of Results from the Comparison

The above comparisons aided in selecting the optimal architecture from the the three proposed architectures. This section summarises the results and then selects one architecture for implementation.

The average execution time in the case of 95% probability of private messages achieved its lowest value in proposed Architecture C, at 19.44 ms. This time was achieved by omitting the network and application servers in the case of fog processing. The execution time was even less in the case of the BCET, reaching only 10.28 ms. Proposed architectures A and B integrate the network and application servers in each fog gateway, and therefore their processing times were many times higher.

Because evaluation of data privacy in proposed Architecture C occurs before the payload is decrypted, the fog gateway can forward the received message to the public cloud unchanged. This fact ensures that the fog gateway is compatible with a standard LoRaWAN infrastructure. Architecture A and B implementations are different. Because each fog gateway integrates the network and application servers, the message always passes through each processing stage. In this manner, the fog gateway forwards already decrypted payloads to the cloud server, and there-

fore it is not possible to deploy this type of fog gateway into the existing standard LoRaWAN infrastructure.

However, the fog gateways in all the proposed architectures allow autonomous operation if the connection to the cloud server fails. It is also evident that integrating the network and application servers into a fog gateway increases the demand for computing power. Table 4 summarises the features of the individual proposed architectures. The table shows that Architecture C attained the best results. I therefore selected proposed Architecture C for further research.

Table 4: Summary of features of the proposed architectures.

Feature	**Architecture A**	**Architecture B**	**Architecture C**
Average execution time	209.96 ms	Similar to arch. A	19.44 ms
BCET	209.81 ms	Similar to arch. A	10.28 ms
WCET	217.78 ms	Similar to arch. A	210.41 ms
Deployment to existing LoRaWAN	No	No	Yes
Autonomous function	Yes	Yes	Yes
Computational perf. requirements	Higher	Higher	Low

6 Machine Learning Methods to Control Data Transfer Between Fog and Cloud Layers

The next step after defining the network architecture for the first aim was to research a reliable method or a combination of methods for controlling data transfer between the fog and cloud layers. The proposed methods use information predefined by the end device owner and information resulting from a machine learning algorithm.

6.1 Controlled Data Transfer

The basic terms of ML were described in Chapter 3. This section discusses a solution to the defined problem. Many reasons exist for transferring computation or storage between the fog and cloud layers, but the most usual are:

- **Data privacy** – if the messages of a specific end device are marked as private, they should not be transferred to the public cloud.
- **Quicker response** – the lower latency of fog computing allows a faster response to the message received from a specific end device.
- **Unavailability** – this could be due to a lack of computing power or memory in the fog layer. Another possible reason is loss of connection to the cloud.
- **Bandwidth saving** – because messages are not necessarily transferred to the cloud layer for processing or storage, bandwidth can be saved.
- **Data complexity** – according to the historical data from a specific end device, complex data processing can be expected. If required, complex computations can be transferred to the cloud.

As the number of connected end devices grows, several end devices may transmit in short intervals. Because fog gateways have limited computing power, it is not possible to process every received message. It is therefore necessary to monitor the actual memory use of each fog gateway and subsequently transfer data to the cloud according to this parameter. It is important to reach a compromise between the limited computing power and bandwidth savings.

One of the essential features of the fog computing paradigm is the partial independence from the cloud layer. For this reason, fog gateways must implement a connection failure detection method. If a connection failure is detected, the fog gateway begins processing all received messages locally until the connection with the cloud layer has been re-established.

The methods based on unavailability, bandwidth saving or data complexity should be used in conjunction with the other described methods. A combination of several methods can help to reach higher efficiency.

To respect data privacy, a fog gateway must implement a specific mechanism to check whether a received message is private or public and handle the message correspondingly. For example, let us suppose that a company would like to use an IoT network with several sensors and collect specific information about their business in a production hall. They have a web application for observing individual information or control activities based on the obtained data. At this point, a problem could arise if the company does not want to transfer private data to the public cloud but want to store historical data there. Fog computing could be the solution. The end device owner can define data privacy, and this information would subsequently be transferred to a corresponding fog gateway. Once the fog gateway has this information available, private data can be processed and stored in a local database while the other data can be transferred to the cloud.

The lower latency in fog computing compared to cloud computing plays a key role in applications which require quicker responses. A mechanism to inform the gateways in the fog layer of the fact that the data requires low-latency processing is therefore essential. This mechanism may be similar to the processing of private data described above.

Because it is not always necessary to transfer all messages to the cloud server, filtering can be performed at the fog layer. At this point, the involvement of ML could be beneficial in automated selection between computation and storage by the fog or cloud layer. I tested different classifiers during this stage of the work to choose a suitable solution. The following section describes and validates several methods for classifying data privacy.

6.1.1 Dataset

Because the solution applies ML, it is first necessary to define a particular data structure to search common patterns in the data. The default dataset contained 321604 anonymised records, each record corresponding to a single received message from a specific IoT end device. These data were captured over a period of three years as part of an experimental academic IoT network deployment. In addition to the internal device identifier, message identifier and message payload, each record contains time stamps when a specific message was received, SNR and RSSI radio parameters, channel numbers, data rates and coding rates. An example of selected records from the dataset is given in Figure 22.

```
messageid,deviceid,time,snr,rssi,channel,data_rate,coding_rate,payload
230,31,2018-01-31 23:06:11.855,5.5,-108,7,SF12BW125,4/5,alarm: true
362,30,2018-02-02 18:19:57.932,-12.8,-119,3,SF12BW125,4/5,light_main: true
399,30,2018-02-03 04:40:36.559,-15.8,-119,4,SF12BW125,4/5,light_main: false
56393,61,2018-10-22 14:59:12.413,8.8,-93,1,SF7BW125,4/5,light_hall: false
56394,58,2018-10-22 14:59:16.452,4.5,-96,2,SF9BW125,4/5,garbage_container: 74
321002,46,2019-06-05 20:05:43.398,7,-65,6,SF12BW125,4/5,humidity: 49
```

Figure 22: Dataset records example.

The dataset contains messages from a total of 100 unique IoT end devices. While some end devices sent only dozens of messages, some sent thousands of messages. Table 5 lists basic statistics of the selected attributes from the dataset.

Table 5: Basic statistics of the dataset

Attribute	Min	Max	Mode
Time	2017-12-22 11:18:38.586	2020-03-23 23:01:08.219	-
SNR	-22.8	14.8	8.8
RSSI	-131	-11	-105
Channel	0	7	0
Data Rate	SF7BW125	SF12BW125	SF9BW125
Coding Rate	4/5	4/5	4/5

6.1.2 Payload Structure

It should be noted here that the application data payload does not have any format specified by the LoRaWAN network protocol, and it may contain data in various forms. This fact introduces complications into the solution based on ML, and it becomes necessary to define a uniform data structure. Assuming that fog gateways are deployed within a single production hall or an area under the management of a single provider, it should not be a major problem to follow a consistent data structure. For simpler decoding and readability, the selected payload format is as follows:

$$variable\_name: value$$

6.1.3 Classification of Data Privacy

As mentioned above, one of the properties for deciding whether data should be processed in the fog or cloud layer is data privacy. The classification of data privacy can be based entirely on the judgement of the end device owner or administrator. However, if we assume an IoT network with a large number of connected end devices, such manual classification would be very time-consuming. It is therefore prudent to partially automate the classification process. For proper classification, the learning model uses historical data and user-defined preferences during the training process. By default, the end device owner selects which data should be private or public for a small set of end devices. After proper training and testing, the ML algorithm can decide whether a message received by a newly registered end device should be processed by a fog gateway or forwarded to the network and application servers located in the cloud. Since one end device, i.e., a specific sensor, usually sends consistent data and only the value of the observed parameter changes, data privacy can be specified at the level of the particular end device. It is therefore not necessary to deal with each message separately. The dataset, however, does not

contain many attributes to specify data privacy. The following is an overview of the attributes which can be applied to specify data privacy.

Message Periodicity

Individual IoT end devices can send their messages to the network periodically at predefined time intervals. This is not the only option, as in some cases, such periodic messaging may be pointless. The other option is to send messages only when the observed value changes. For example, if we assume a temperature sensor, sending data periodically is logical, as air temperature changes often during the day. If we consider a motion sensor, periodically sending the status is irrelevant, as it could lead to sending redundant messages containing the same content. Here, transmitting a message when the sensor only detects some movement is logical.

After analysis of the message payloads from individual end devices recorded in the dataset, we can conclude that in most cases, data of an informative nature (temperature, humidity, etc.) are sent periodically. In this manner, message privacy could be classified according to periodicity. Since one of the attributes in the dataset is the message arrival time at the gateway, I was able to trace whether the messages were sent periodically.

The default dataset must be pre-processed to determine message periodicity and to build the training set. Since all messages are unconfirmed, some messages from the end devices may not reach a gateway and are therefore not present in the dataset. A temporary outage may also occur in an end device, for example, because of a low battery level. These types of error must be taken into account. First, I assumed that the entire date-time format of message receipt time was not essential, and I extracted only seconds from this component. Pre-processing therefore consists of extracting seconds from the *Time* attribute and then filtering only messages which belong to a certain end device. The number of identical derived seconds was then determined for these filtered records, and the total number of received messages for specific end devices and the maximum, minimum and average numbers of identical values of the seconds (*count*) were calculated. Analysing these records, I estimated which end devices sent their data periodically and which not. I then applied the following condition: if the minimum and a maximum number of matches are greater than 100 (statistically evaluated) or if the number of matches is predominant concerning the total number of end device messages, it can be concluded that the given end device sends its messages periodically. An example of records for the training set is shown in Figure 23.

```
deviceid,sum(messages),maximum(count),minimum(count),average(count),periodicity
11,13,13,13,13,true
30,324,16287,244,271.4,true
33,387,11,1,6.6,false
41,1891,44,19,31.5,false
58,111018,2006,1717,1850.3,true
```

Figure 23: Training set example - Periodicity

Based on an empirical comparison of the classifiers and individual configurable parameters, the following settings were selected:

- Decision tree: 5 maximum depth, pruning applied, 0.1 confidence.
- k-NN: 5 neighbours, numerical measures, Euclidean distance, weighted vote.
- SVM: dot kernel type, automatic kernel scale.

For FFNN, it is more complicated since it was necessary to find the optimal multi-layer network topology first. Based on the nature of the data, which unambiguously determines the number of neurons in the input and output layers, a basic set of topologies was empirically derived and subsequently assessed for applicability in solving the given problem. As indicated, the maximum number of neurons in a single hidden layer is 4. Additional neurons in the hidden layer improved the result so minimally that their use did not benefit in terms of higher demands on computing power and time. The following is a list of the examined FFNN topologies:

- 4 - 1 - 1 (4 input neurons, 1 hidden neuron, 1 output neurons)
- 4 - 2 - 1
- 4 - 3 - 1
- 4 - 4 - 1
- 4 - 3 - 2 - 1
- 4 - 4 - 2 - 1

The most commonly used activation function, the sigmoid, was applied as the activation function for the hidden and output layer neurons. The input layer used a linear activation function. In all the above-mentioned topologies, bias neurons were also used in every layer apart from the output layer. These bias neurons were inserted by default and therefore not included in the topology description. The learning rate was empirically set to 0.01 and the number of training cycles to 300.

Table 6 shows the results obtained in measurements where the only variable was the neural network's topology. All other settings remained the same. The table indicates that the 3-layered topologies achieved better results than either of the 4-layered topologies, which is due to significant shortcomings in the performance of some 4-layered topology networks. Taking into account the measured parameters, I selected the 4-4-1 topology for comparison with other classifiers.

Tables 7, 8, 9 and 10 compare the individual results obtained from different classification algorithms trained and tested on the dataset mentioned above. The performance of these classifiers was obtained using the k-fold cross-validation method, where, considering the size of the dataset, $k = 5$.

Network Topology	Accuracy	RMSE	Spearman Correlation
4 - 1 - 1	77%	0.402	0.518
4 - 2 - 1	77%	0.402	0.521
4 - 3 - 1	77%	0.404	0.524
4 - 4 - 1	78%	0.401	0.546
4 - 3 - 2 - 1	71%	0.441	0.343
4 - 4 - 2 - 1	72%	0.440	0.367

Table 6: Classification results for different FFNN topologies – periodicity.

The tables reveal that none of the classifiers achieved identical results for the number of TP, TN, FP and FN. For FFNN, the precision that a message is classified *periodic* is 100%, while the precision that a message is classified *non-periodic* is 74.12%. For SVM, the *periodic* class precision is 100%, while the *non-periodic* class precision is 72.41%. The decision tree classifier achieved the worst results of the selected classifiers in terms of the *periodic* class precision, while the *non-periodic* class precision achieved the best values. The best overall results were achieved by the k-NN classifier: 87.50% precision for the *non-periodic* class and 100% for the *periodic* class.

Predicted	**True Class**		**Class Precision**
Non-Periodic	62	8	88.57%
Periodic	1	29	96.67%
	Non-Periodic	Periodic	
Class Recall	98.41%	78.38%	

Table 7: Decision Tree confusion matrix – Periodicity.

Predicted	**True Class**		**Class Precision**
Non-Periodic	63	24	72.41%
Periodic	0	13	100.00%
	Non-Periodic	Periodic	
Class Recall	100.00%	35.14%	

Table 8: SVM confusion matrix – Periodicity.

Predicted	**True Class**		**Class Precision**
Non-Periodic	63	9	87.50%
Periodic	0	28	100.00%
	Non-Periodic	Periodic	
Class Recall	100.00%	75.68%	

Table 9: k-NN confusion matrix – Periodicity

Predicted	**True Class**		**Class Precision**
Non-Periodic	63	22	74.12%
Periodic	0	15	100.00%
	Non-Periodic	Periodic	
Class Recall	100.00%	40.54%	

Table 10: Feed-Forward NN confusion matrix – Periodicity

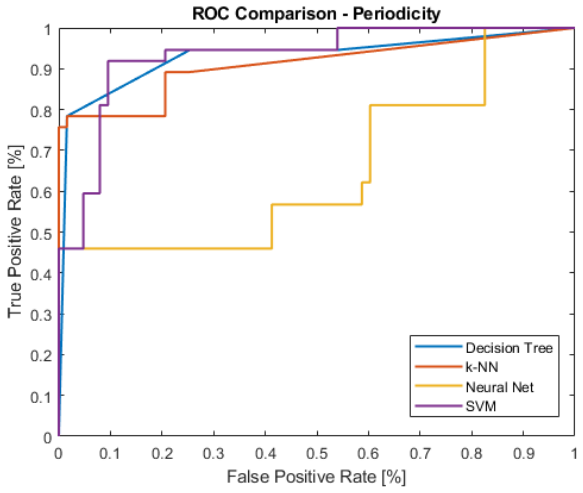

Figure 24: Comparison of ROC curves for different classification algorithms.

Figure 24 charts the ROC curves for the four mentioned classifiers. The classifier with the worst performance, with an AUC value of 0.65, was FFNN. k-NN was third in performance, with 0.91. The decision tree classifier had an AUC value of 0.93. The the SVM classifier had the highest AUC value.

Payload Content

Another method of distinguishing privacy for received messages is according to the individual message payload. A standard LoRaWAN gateway does not have access to the decrypted payload, whereas a fog gateway permits searching for specific common patterns in the decrypted payloads

of received messages from individual end devices. Another method is to decide according to the data type of the payload variable.

In the first approach, if the precise structure of the payload has been defined, it should not be too difficult to observe certain patterns of behaviour in the dataset. For example, the value of temperature or humidity might not be critical from a privacy perspective, although the status of a door or an alarm may be more critical, and the end device owner may want to keep them private. To create the training set, it is necessary to pre-process messages from the default dataset. It is not necessary to evaluate each message separately, as this would be an expensive task computationally or with respect to time, but it is possible to unify the individual messages according to the end device identifier and subsequently classify only individual end devices. An example of a modified dataset used for training and testing is given in Figure 25.

```
deviceid,variable,privacy
7,smoke,true
9,temperature,false
11,humidity,false
12,illumination,false
13,light,true
14,motion,true
```

Figure 25: Training set example – Payload.

The difference from the previous method based on checking periodicity is that the payload checking method contains categorical attributes in the input data. To use classification with FFNN or SVM, one-hot encoding can be applied to transfer the categorical values into a form which can be provided to the ML algorithm. As the number of categories grows, the number of input neurons (in the case of FFNN) or dimensions (in the case of SVM) increases. Since the variable name is usually subjective, a large number of such categories can be defined. FFNN and SVM were therefore ineffective in solving this problem, and I used the naive Bayes and random forest classifiers as substitutes. These should be suitable for this type of classification.

Based on the comparison of individual configurable parameters, the following settings for individual classifiers were selected:

- Decision tree: 5 maximum depth, pruning applied, 0.1 confidence.
- k-NN: 5 neighbours, nominal measures, nominal distance (the distance between two values is 0 if both values are the same, or otherwise 1), weighted vote.
- Random forest: 100 trees, 5 maximum depth, pruning applied, 0.1 confidence.
- Naive Bayes: Laplace smoothing applied.

Tables 11, 12, 13 and 14 compare the individual results obtained from different classification algorithms trained and tested on the dataset mentioned above. Regarding the precision of

the classification, the Random Forest and Naive Bayes algorithms achieved the best results, with 95.52% of the *private* class and 100% of the *public* class. Despite the accuracy of the random forest classifier (97%) being the same as the accuracy of the naive Bayes classifier, the computational complexity of random forest is much greater. The naive Bayes classifier was therefore a logical choice for solving the payload classification problem.

Predicted	**True Class**		**Class Precision**
Private	64	36	64.00%
Public	0	0	0.00%
	Private	Public	
Class Recall	100.00%	0.00%	

Table 11: Decision tree confusion matrix – Payload.

Predicted	**True Class**		**Class Precision**
Private	64	3	95.52%
Public	0	33	100.00%
	Private	Public	
Class Recall	100.00%	91.67%	

Table 12: Naive Bayes confusion matrix – Payload.

Predicted	**True Class**		**Class Precision**
Private	64	8	88.89%
Public	0	28	100.00%
	Private	Public	
Class Recall	100.00%	77.78%	

Table 13: k-NN confusion matrix – Payload.

Predicted	**True Class**		**Class Precision**
Private	64	3	95.52%
Public	0	33	100.00%
	Private	Public	
Class Recall	100.00%	91.67%	

Table 14: Random forest confusion matrix – Payload.

Figure 26 shows the ROC curves for the four mentioned classifiers. In this case, the classifier with the worst performance, with an AUC value of 0.42, was the Decision Tree algorithm. The other classifiers attained AUC values over 0.90. The best results were achieved with the naive

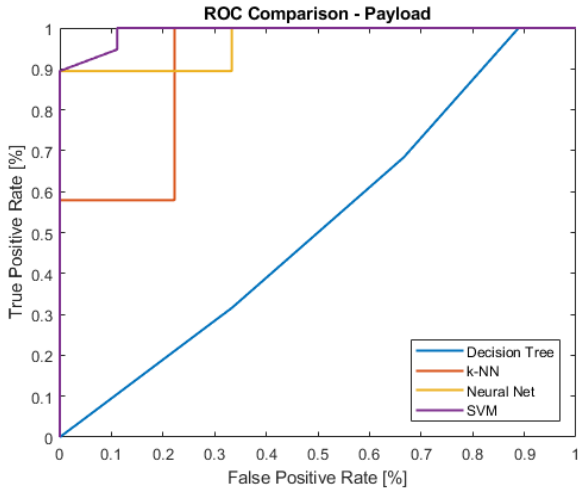

Figure 26: Comparison of ROC curves for different classification algorithms – Payload.

Bayes and random forest classifiers, each with an AUC value of 0.99. The performance of the k-NN classifier was 0.96.

However, the results are affected significantly by the number of unique payload variables. Proof of this behaviour is in the three FP errors which occurred when the naive Bayes classifier was applied (shown in Table 12). Examining these errors, they appeared only in cases where the *variable* name was completely unique and occurred only once in the dataset. To verify this behaviour, I altered the dataset to contain multiple unique non-recurring *variable* names (specifically, 21 new unique names). I also removed 10 end devices with a categorical payload type and added 10 end devices with a numerical payload type to the dataset. A greater number of numerical *variable* names created a more balanced dataset given the ratio of private and public end devices. Table 15 presents the results of the naive Bayes classifier trained and tested on the modified dataset. Figure 27 shows the ROC curves for the four classifiers trained and tested on the modified dataset. The results show that all of the classifiers performed worse and were clearly affected by the greater number of unique *variable* names.

Predicted	**True Class**		**Class Precision**
Private	54	25	68.35%
Public	0	21	100.00%
	Private	Public	
Class Recall	100.00%	45.65%	

Table 15: Naive Bayes confusion matrix – Payload (modified dataset).

The second approach in data privacy classification based on payloads assumes that numerical variables often do not have sensitive content, while the privacy of nominal values must be maintained. An example is temperature or humidity, these values, in most cases, being sent

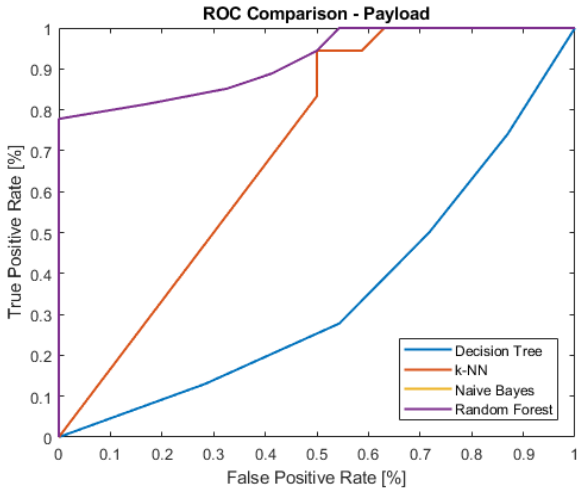

Figure 27: Comparison of ROC curves for different classification algorithms – Payload (modified dataset).

in numerical form, whereas alarm or door states are sent as nominal values. For this type of classification, the solution does not need to implement ML, but privacy can be determined from a simple condition.

A combination of the first and second approaches, however, improved precision, and the classification achieved significant accuracy, even with a modified dataset. The ROC curves of the classifications for this combination of approaches, trained and tested on the modified dataset, are presented in Figure 28. The AUC value of all classifiers was over 0.95.

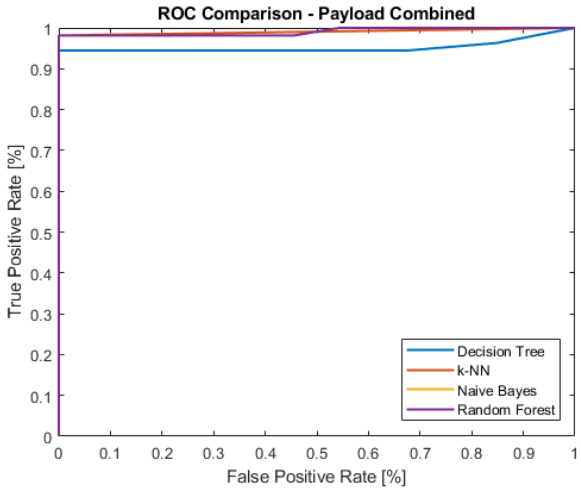

Figure 28: Comparison of ROC curves for different classification algorithms – Payload Combined (modified dataset).

6.2 Discussion of the Results

This section presents a summary of the results attained in this phase of the study. The simplest method for determining whether data is processed in the fog layer or the cloud layer is based on statically predefined options for individual end devices. The disadvantage of this method is the necessity to manually define the processing method for each new end device.

Since a fog gateway is able to decode and decrypt the message payload, it can also distinguish the payload data type, and the decision whether the payload contains a categorical or numerical variable is a trivial problem. To solve the problem, the fog gateway does not need to employ ML, and moreover, it is wasteful to decrypt each message separately to determine the payload data type. Because individual end devices usually do not change the payload structure during their operation, it is possible to determine whether the payload contains nominal or numerical values from the first message sent by a specific end device and then store this information. Once a fog gateway has the information, it can make a decision for subsequent messages from the given end device.

Another option is to use ML. If the classification is based on the name of the variable, the naive Bayes classifier achieved an accuracy of 97%. However, the result is affected significantly by the payload variability. Let us suppose that the fog gateway is deployed to collect data from a particular production hall where the message payload can respect a defined data structure. In this example, payload variability can be kept low because the payload follows a certain format and contains only specific prefixes or suffixes describing the purpose of the data message. In this situation, the method may be considered reliable, but otherwise, the method might have a significant error rate. This type of behaviour was demonstrated after modification of the dataset. After modification, the accuracy of the naive Bayes classifier fell to 75%. However, adding another attribute to the feature vector reduced the dependence on the number of unique variable names.

The other option mentioned in this dissertation is classification based on the periodicity of messages. In this case, the k-NN classifier accuracy reached 91%. The main problem with this method is the need for a large amount of historical data to correctly evaluate the periodicity of messages. Thus, the periodicity can only be assessed for end devices that have been communicating with the network for a longer period, providing a large number of messages to build the training set. In addition, since LoRaWAN does not guarantee reliable delivery of messages, the periodicity may be violated because of a non-received message or low battery level. From that point of view, of the three methods presented, this method is the most prone to message delivery errors.

All the methods, however, are based on the assumption that a specific link between the payload or the periodicity of the messages and the data processing in the fog layer or cloud layer exists. Since data privacy depends mainly on the the end device owner's decision, it is possible that no such link may exist. These methods therefore cannot be applied generally, but it is

always the better choice to apply adjustments according to the specific application. Another problem associated with applying ML to a fog gateway is its limited computational power.

An alternative is to use a semi-automated mode and implement the option of using ML in a web interface hosted by a cloud server. When a user selects this option, the ML algorithm automatically selects data privacy for each end device according to the historical data and user preferences. The user may then confirm, modify or reset the automatically generated settings. A machine learning algorithm can thus improve its accuracy over time.

7 Implementation of Key Components of the Architecture for Verification

Proposed architecture C was implemented to test its effectiveness and functionality. This chapter presents and discusses the details of implementation.

7.1 Chirpstack

ChirpStack (previously known as LoRaServer) is an open-source solution of the LoRaWAN network stack and provides components for the LoRaWAN network to form a solution which is ready to use. The solution is also very flexible and allows a private LoRaWAN network to be built without dependence on shared infrastructure. ChirpStack forms the basic LoRaWAN infrastructure for the purposes of this dissertation.

ChirpStack provides the following components:

- **ChirpStack Gateway Bridge** – is a service placed between the packet forwarder of a LoRaWAN gateway and MQTT server. When a LoRaWAN gateway receives a message, the gateway bridge converts the LoRa packet forwarder protocols into the data format supported by the ChirpStack network server (JSON in this case). It then publishes the message with relevant metadata to the specific topic via the MQTT protocol.
- **ChirpStack Network Server** – is a component of the stack. Because the network server can receive identical messages from different gateways, its primary responsibility is to eliminate redundant received LoRaWAN frames. Among other tasks, the network server handles authentication for the received frames, communication with the application server and scheduling of downlink frames. As the name suggests, it performs a function of the LoRaWAN network server.
- **ChirpStack Application Server** – is a component of the stack. Its main responsibility is management of devices and encryption/decryption of application payloads. Part of the ChirpStack application server is a web-based application dedicated to the management of applications and devices. This management is also available through the API for external services. As the name suggests, this component performs a function of the LoRaWAN application server.

The other component of the ChirpStack is Gateway OS, an open-source operating system based on Linux. This embedded operating system contains selected ChirpStack components installed by default [84]. However, Gateway OS was not used for the purposes of this dissertation.

7.2 Implementation of the Architecture

Since the proposed architecture adopts basic architectural features of the standard LoRaWAN architecture, it is necessary to apply modifications, mainly to the fog gateway itself, while

maintaining the standard function of the network and application servers. The advantage of this approach is seamless integration of the fog gateway into the standard LoRaWAN network.

As already mentioned, the main problem in decryption of the application payload on the gateway is the end-to-end encryption method. However, a necessary condition for implementation of the proposed fog computing optimisation is the ability to decrypt the message payload at the gateway, which is conditional on the knowledge of the relevant session keys and methods. Once the gateway knows the session keys for a particular end device, its software can be modified to decrypt the application payload and continue further processing.

It is therefore necessary to design a support mechanism. The fog gateway first determines the short device address (DevAddr) and then queries the application server for the session keys. However, this process is executed only when the fog gateway does not have the session keys stored in its local database from the previous communication. It is necessary to know the LoRaWAN message format to decode the DevAddr and other required information relating to the current message received by the fog gateway.

Each LoRa uplink or downlink message transmits the payload (PHYPayload). This payload starts with the MAC header (MHDR) and ends with the message integrity code (MIC). The structure of the LoRaWAN message is shown in Figure 29.

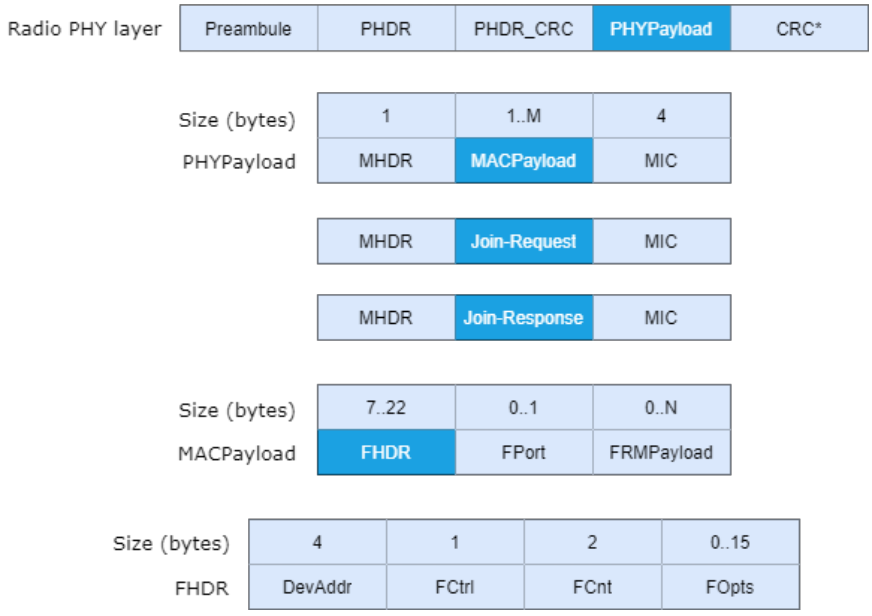

Figure 29: LoRaWAN message structure.

The MHDR contains important information specifying the type of message (MType). Table 16 shows eight different MAC message types. The Unconfirmed/Confirmed Data Up types which transmit application data are important for this dissertation. As the names suggest, the difference between unconfirmed and confirmed messages is that the unconfirmed messages do not require acknowledgement from the recipient. By contrast, confirmed messages must be ac-

knowledged. Data messages can also be proprietary, where a non-standard message format can be implemented, but the message format must be readable to the end devices [46], [85].

Table 16: MAC message types.

MType	Description
000	Join Request
001	Join Accept
010	Unconfirmed Data Up
011	Unconfirmed Data Down
100	Confirmed Data Up
101	Confirmed Data Down
110	Reserved for Future Use
111	Proprietary

The FHDR field is important in assigning a message to a specific end device, as it contains the DevAddr, which can identify the message sender. The DevAddr identifier is a 32-bit device address unique to a particular network served by a single network server. To identify the end devices globally, LoRaWAN applies a 64-bit globally unique device address (DevEUI). If an end device uses the ABP method of activation, it can be identified by the DevAddr, which is pre-configured in the end device program. With the OTAA method of activation, it is more complicated because the DevAddr is dynamically assigned during a join procedure with the network. In this manner, an end device activated with OTAA can be uniquely identified at any time only with DevEUI, which is not a part of a data message. In the case of the OTAA method, more frequent communication is required between the application server and the fog gateway to determine the current DevAddr belonging to the given DevEUI.

Once a data MAC message contains a payload, it is necessary to encrypt the FRMPayload field before calculating the MIC. The calculation is performed over all fields in the message (MHDR + MACPayload). LoRaWAN security uses the Advanced Encryption Standard (AES) symmetric block cipher with a key length of 128 bits. Whether NwkSKey or AppSKey is used for encryption/decryption depends on the value of the FPort field (shown in Table 17) in the message [46].

Table 17: Relationship between FPort and session keys.

FPort	Key
0	NwkSKey
1..255	AppSKey

The cipher algorithm mentioned above defines a sequence of blocks A_i for $i = 1 \dots k$ with $k = ceil(length(FRMPayload)/16)$ for each data message. The structure of these blocks is

illustrated in Figure 30.

Size (bytes)	0x01	4 x 0x00	Dir	DevAddr	FCntUp or FCntDown	0x00	i
A_i	1	4	1	4	4	1	1

Figure 30: Block structure.

The Dir field is 0 or 1 depending on whether the message is sent in the uplink or the downlink direction (uplink - 0, downlink - 1).

Blocks A_i are encrypted to obtain a sequence S of blocks S_i.

$$S_i = aes128\_encrypt(Key, A_i)\ for\ i = 1 \dots k$$
$$S = S_1|S_2| \dots |S_k$$

Truncation to the first $length(FRMPayload)$ octets is then applied to encrypt and decrypt the payload [46].

$$(\text{FRMPayload}|pad_{16})\ xor\ S$$

In this manner, it is possible to decode the individual information necessary to decrypt the payload and then perform the decryption itself. However, a part of the gateway must be defined where it is possible to access each received LoRaWAN message and decompose it to obtain information from individual fields. After a full analysis, I modified the packet forwarder. Semtech packet forwarder [86] is a program which runs on LoRa gateway. It forwards RF packets with associated metadata (RSSI, timestamp, etc.) received by the concentrator to the server through an IP/UDP link. In the opposite direction, the packet forwarder transmits RF packets for messages received from the server. Packets can be forwarded over an internet connection, local network or localhost, depending on the server location. In the solution presented in this dissertation, public messages are forwarded to the localhost, where they are captured by the gateway bridge and delivered via the MQTT protocol to the associated network server.

During standard operation, the packet forwarder encodes the received binary data into Base64 format. This is then combined with the metadata, wrapped into JSON, and sent to the network server. It was therefore necessary to modify the packet forwarder program for the correct implementation of the architecture. In addition to forwarding RF packets, the packet forwarder performs end device identification and subsequent payload decryption. Algorithm 1 presents the pseudocode of the proposed algorithm applied in the packet forwarder program.

If the Base64 form of the received data is converted into a sequence of hexadecimal bytes, it is possible to systematically decompose this sequence and obtain the DevAddr to identify the source of the message. Once the DevAddr has been obtained, it can be determined whether it is an end device whose messages should be processed privately/faster. In the case of a public message, decryption of the payload can be omitted and the standard method can be applied

Algorithm 1 Packet forwarder fog computing algorithm

Input: Message in Base64 format
Result: Forward message to cloud server or store *decyptedPayload* in local database

```
for Each received message do
    dataHex ← Decode Base64 encoded message to hexadecimal format
    mhdr ← getMhdr(dataHex)
    if mhdr = Unconfirmed Data Up then
        macPayload ← getMacPayload(dataHex)
        fhdr ← getFhdr(macPayload)
        devAddr ← getDevAddr(fhdr)
        if devAddr is not in local database then
            Contact application server to negotiate nwkSKey and appSKey
        else
            if devAddr belongs to current LoRaWAN network then
                if devAddr belongs to private device then
                    nwkSKey ← loadNwkSKey(devAddr)
                    appSKey ← loadAppSKey(devAddr)
                    frmPayload ← getFrmPayload(macPayload)
                    fPort ← getFPort(macPayload)
                    fCnt ← getFCnt(fhdr)
                    dir ← 1
                    if fPort = 0 then
                        key ← nwkSKey
                    else
                        key ← appSKey
                    end if
                    decryptedPayload ← decryptPayload(frmPayload,devAddr,dir,fCnt,key)
                    Store decyptedPayload in local database
                end if
            end if
        end if
    end if
    if devAddr does not belong to private device then
        Forward message to cloud server
    end if
end for
```

by sending a JSON to the network server. For a private message, it is necessary to use the NwkSKey or AppSKey key (according to the value of the FPort field) for the given end device and then build the blocks for encryption according to the fields obtained from the sequence of hexadecimal bytes representing the message. Encryption of these blocks and truncation of the PHYPayload can then proceed. This process results in a decrypted payload which can be stored in a local database of the fog gateway.

However, as already mentioned, the standard LoRaWAN gateway does not know the session keys for the individual end devices. It was therefore necessary to implement a mechanism into

the fog gateway dedicated to contacting the application/network server with a request to obtain the session keys for a specific end device. The ChirpStack application server has a gRPC API interface for obtaining/editing/creating applications, end devices and users. The gRPC is an open-source remote procedure call (RPC) framework. The fog gateway can use this API to obtain current information about the defined end device. Authentication by providing per-RPC credentials is mandatory to prevent unauthorised access to data through the gRPC API methods [87]. The pseudocode in Algorithm 2 defines the negotiation procedure between the fog gateway and the application server.

Algorithm 2 Negotiation with the application server

```
   Input: devAddr
   Result: Store nwkSKey and appSKey in local database
   devEui ← Contact the application server via API to receive DeviceEUI
 2: if Given devEui belongs to current LoRaWAN network then
       metadata ← Contact application server via API to receive metadata for given devEui
 4:    nwkSKey ← getNwkSKeyFromMetadata(metadata)
       appSKey ← getAppSKeyFromMetadata(metadata)
 6:    Store nwkSKey and appSKey in local database
   else
 8:    nwkSKey ← 00000000000000000000000000000000
       appSKey ← 00000000000000000000000000000000
10:    Store nwkSKey and appSKey in local database
   end if
```

In this dissertation, a script for negotiation with the application server through the gRPC API was implemented. This script requires the DevAddr identifier as a parameter. Based on the parameter, the fog gateway sends a request to obtain the DevEUI, which is then used to obtain the session keys. The session keys are subsequently stored in the fog gateway's local database together with information about the application payload's privacy. By default, the payload of individual end devices is considered non-private. If the DevAddr obtained from the current message is already in the local database, the session keys are read immediately, and the script is therefore not required since negotiation with the application server is unnecessary.

However, because standard LoRaWAN gateways receive messages from all end devices in the range and a redundancy check occurs only on the network server, the DevAddr in the currently received message may not belong to the given LoRaWAN network. In this case, the fog gateway can ignore such a message. To avoid repetitive requests to the application server due to an end device which periodically sends messages but does not belong to the given LoRaWAN network, the DevAddr of this end device is added to the local database, and its AppSKey and NwKSKey are set to zero. The fog gateway can therefore immediately recognise that it is an end device registered in another LoRaWAN network and terminate the processing. The message is not sent to the network server, and it does not have to be unnecessarily burdened.

Each fog gateway also hosts a web application designed to manage and display available end devices. The web application offers an important option which allows the user to control the computing method (fog or cloud) for the individual end devices.

8 Testbed and Verification of Results

This chapter describes the testbed located on the campus of VSB - Technical University of Ostrava and contains a presentation and verification of the results. In this chapter, the optimised network architecture is compared with both the standard LoRaWAN network architecture already deployed at the campus and a LoRaWAN network solution based on ChirpStack.

8.1 Current Solution at the VSB-TUO Campus

The network architecture contains end devices, gateways, The Things Network (TTN) server, a KoIoT server, and a web application for end-users. LoRaWAN gateway serves as a bridge between the wireless and wired components of the network. Each gateway is connected via an Ethernet interface to the Internet to provide a connection to the TTN backend system, which is one of the main parts of the solution and responsible for routing IoT data between end devices and applications. The TTN provides a LoRaWAN server and functionality related to applications. The KoIoT server is another main part of the network infrastructure. It is fully managed by the Department of Telecommunications and is designed to store data and host a backend web interface for end-users. Figure 31 provides a schematic representation of the architecture.

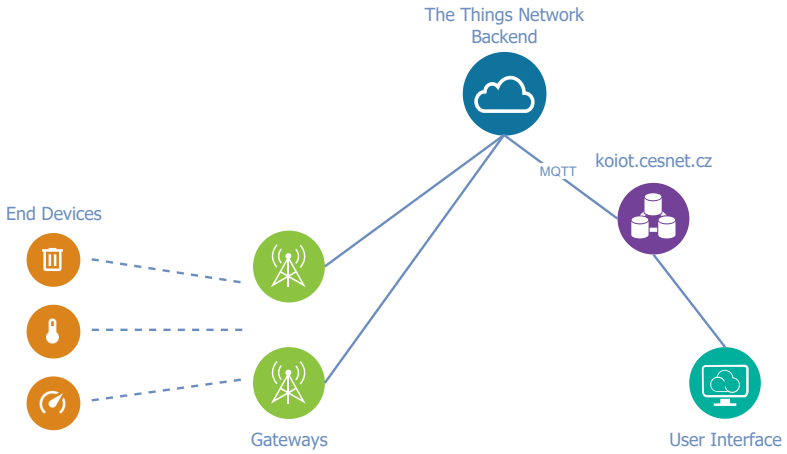

Figure 31: Diagram of the network infrastructure at the VSB-TUO campus.

The LoRaWAN gateway at the campus solution is based on the fourth generation monolithic microcomputer Raspberry Pi 4 model B and a fully compatible LoRaWAN 868 MHz iC880A concentrator. This concentrator is connected to Raspberry Pi via a serial peripheral interface (SPI). The iC880A concentrator is a multi-channel high-power transmitter/receiver radio module which enables up to eight LoRaWAN packets to be received simultaneously from different end devices, with different SFs and on different channels [82]. The gateways are placed primarily on the rooftops of large buildings. All electronic components are connected with a single twisted

pair Ethernet cable to provide both an internet connection and electric power. This method is generally referred to as a power over Ethernet (PoE) system, with 230V to 12 V/1 A or 24 V/1 A source adapters. All parts of the gateway are housed inside a metal wall-mounted electronic box (SKYBOX MB2520D140) with an IP65 rating [Jal19b]. These hardware components are identical in each of the compared solutions.

As already mentioned, the current solution at the VSB-TUO campus uses the open global TTN backend, each gateway being connected to a single shared TTN router. The router communicates with one or more TTN brokers. While the broker is a central component of the TTN, the router performs gateway status management and transmission scheduling. A TTN handler is then connected to each broker to handle data for a group of associated end devices.

To provide more architectures for comparison, I modified the solution deployed on campus to employ ChirpStack components instead of the TTN. The basic software platform is therefore the same as the platform used in the proposed solution, which should lead to more objective results. The ChirpStack is described in Section 7.1. The modified network infrastructure is depicted in Figure 32.

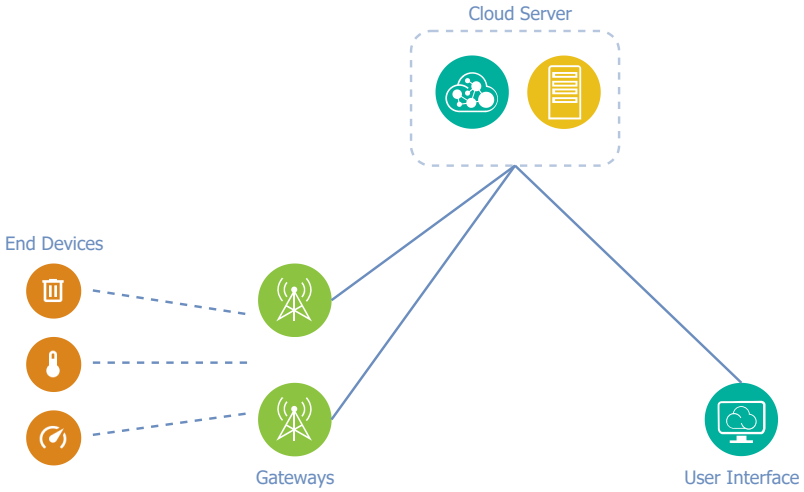

Figure 32: Diagram of the modified network infrastructure.

8.2 Verification of Results

The following section presents a series of tests designed to verify the proper functioning of the optimised architecture and a comparison of this architecture with the VSB-TUO campus network and the ChirpStack network.

8.2.1 Comparison of Execution Times

This test is designed to compare the service times of the solutions. To obtain the results for comparison, it was necessary to measure the time of the overall message processing from arrival

to the gateway until storage of the decrypted payload into the database. In the optimised architecture, the payload can already be decrypted and stored on the fog gateway.

The test was performed with two end devices which sent messages every ten seconds. One end device was set up for fog processing. With a semi-automated solution, the user can choose the processing mode (fog, cloud, automated) in the web user interface hosted by the fog gateway. One hundred messages sent from each end device were compared.

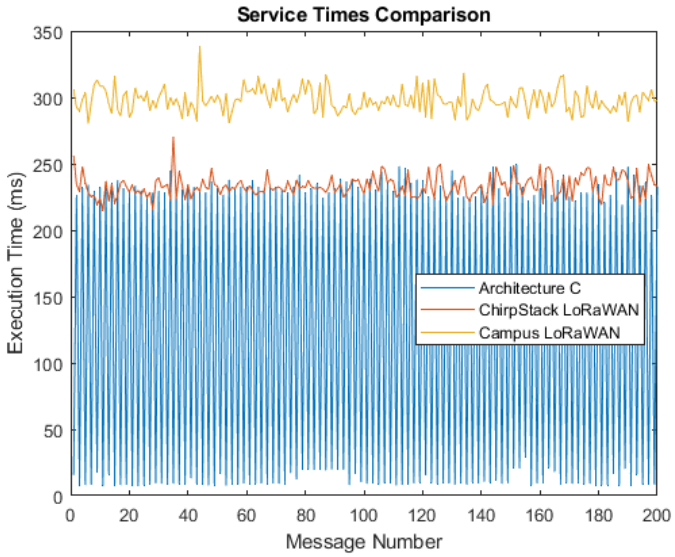

Figure 33: Results of service times comparison.

The results of the comparison are graphed in Figure 33. The figure reveals a vast difference between the fog gateway and the cloud execution times. This fact was also highlighted by the results of the simulations presented in Section 5.4. The low service time values for the optimised architecture correspond to fog gateway processing of messages from the first end device. Fog computing support was switched off for the second end device, and payloads were therefore processed in the cloud. This state is represented by the individual increases in execution time in the optimised architecture. For this test, the average total service time of the optimised architecture was 121.74 ms. If the fog gateway processed the majority of messages, the average total time required was lower. Exclusive processing of all messages by the fog gateway required an average total service time of 10.44 ms. Exclusive processing of all messages in the cloud layer required an average service time similar to the ChirpStack based network since the basic software platform is the same. The variability of the measured values for the campus (TTN) and ChirpStack networks is not significant since these networks did not employ fog computing; all processing occurred in the cloud. The average total service time of the ChirpStack-based network was 231.92 ms. The solution for the campus IoT network, based on TTN, required an average total service time of 297.93 ms. If 50% of the messages are processed and stored in the

fog gateway, the optimisation proposed in this dissertation thesis reduces the service time by more than twice compared to the current campus IoT network solution.

In many cases, the fog gateways have limited computing power. It is therefore necessary to monitor the actual CPU and memory usage, which increases with the number of end devices served at the same time. In the case of unavailable resources, the computing is transferred to the cloud.

8.2.2 Data Privacy Test

The purpose of this test was to examine the reliability of the data privacy protection. If the end device owner marks the end device messages as private, the fog gateway processes the messages and saves them in its local database. If the messages are not private, the fog gateway does not need to process the messages and they are forwarded to the public cloud.

During this test, two end devices periodically sent messages. Messages from one end device were set as private, while messages from the other end device were public. Each end device sent 100 messages. I then compared the presence of the decrypted payload in the databases. If the proposed mechanism functions correctly, all the messages from the first end device are stored in the local database of the fog gateway, while messages from the second end device are stored in the cloud.

Table 18: Privacy test results.

DB location	Private end device	Public end device
Fog gateway	99	0
Cloud	1	100
Total	100	100

The test results presented in Table 18 show that with the exception of one message, all messages marked as private were decrypted by the fog gateway and stored in its local database. If the private end device communicates with the fog gateway for the first time, the first message is processed in the cloud. This first message is the catalyst for the fog gateway to obtain the session keys from the application server and store them locally. Subsequent messages are then processed privately. The results demonstrate the correct functioning of the proposed mechanism implemented in the fog gateway.

For the campus and ChirpStack LoRaWAN networks, all messages were processed and stored in the cloud since these architectures are based on the cloud computing paradigm.

8.2.3 Offline Processing Test

One of the most important features of the fog computing paradigm is partial independence from the cloud. This allows the fog node can process messages in an offline state. Since the optimised

architecture employs the fog computing paradigm, this functionality is implemented in the fog gateway; it consists in checking the network connection status with each incoming message, and when a connection failure is detected, the fog gateway automatically switches to offline mode. In this mode, the fog gateway processes all messages locally and stores them in its database. Public messages are therefore not required to be sent to the public cloud, ensuring that messages are preserved even if connection to the cloud is interrupted.

This experimental testing was based on repeated intentional disconnection of the network connection between the fog gateway and the cloud. During this test, the end device periodically sent its messages to the fog gateway. The fog gateway was set to forward each message to the cloud. When the connection between the fog gateway and the cloud was established, the gateway forwarded messages to the cloud, but when connection failure was detected, all processing was transferred to the fog gateway.

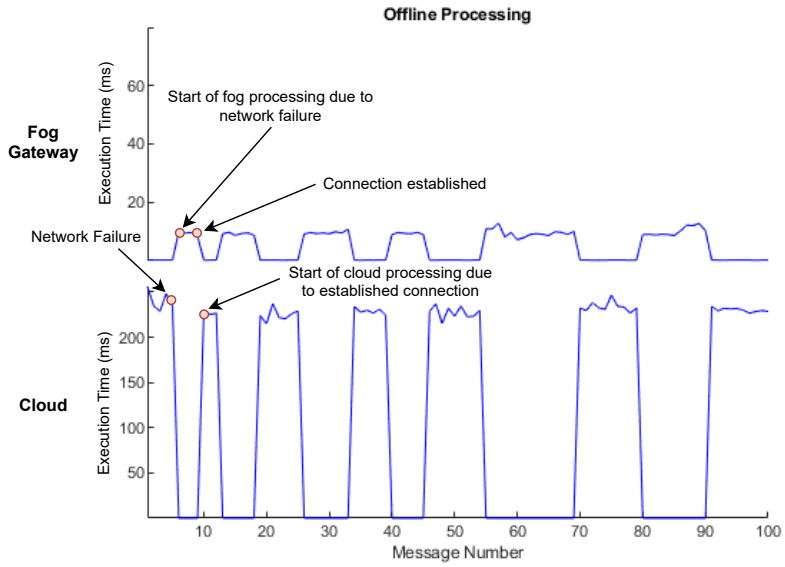

Figure 34: Offline processing test results.

Figure 34 displays the results of offline testing in a stacked chart. The upper chart represents message processing by the fog gateway. Cloud computing is shown in the lower chart. The first five messages were processed in the cloud, after which the connection failure occurred, which caused processing to be transferred to the fog gateway. Without an online connection, the fog gateway functioned in offline mode, during which all incoming messages were processed and stored locally. Once the connection to the cloud was re-established, computation was transferred back to the cloud. Connection failure detection is not immediate, with a delay of 0.5 seconds. However, since the end devices communicate only at certain times and not continuously, this delay is negligible.

8.3 Discussion of the Results

In the previous section, three validation tests were described to verify the results obtained in this dissertation. This section discusses the results of the tests.

The first test aimed to compare the implemented optimised architecture and the campus LoRaWAN infrastructure. However, the campus network includes TTN services, which are shared. I therefore modified the campus network with ChirpStack to obtain more objective test results. These results demonstrated that the optimised architecture achieves significantly lower latency in message processing since the fog gateway is able to process messages from specified end devices locally. The test results therefore confirmed the simulation results and the hypothesis that fog computing will significantly reduce service time.

The second test verified compliance with messages privacy. If the end device message is marked as private, it should be processed by the fog gateway and then stored in its local database. The gateway may not forward private messages to the public cloud. The test confirmed the suitability of the mechanism to evaluate predefined message privacy.

The final test verified the offline capabilities of the fog gateway. The fog computing paradigm allows for partial independence from cloud services: if the fog gateway has the necessary session keys available in the local database, it is able to process incoming messages without communicating with the cloud. The charted test results verified that when the connection to the cloud service failed, messages from all known end devices were automatically processed by the fog gateway.

9 Conclusions and Expected Contributions

Three main aims were established in this dissertation. Chapter 5 presented a solution to pursue Aim 1 and determine which of three fog computing architecture proposals was the optimal solution. Chapter 6 addressed Aim 2 and investigated various machine learning algorithms and controlled data transfer methods to determine the method with the greatest accuracy. Chapter 7 and Chapter 8 focused on Aim 3 and discussed implementation of the optimised architecture and testbed. The proposed optimal architecture was additionally verified in a series of tests especially designed to demonstrate correct functioning of the architecture. The obtained results were compared with a standard LoRaWAN network architecture located at the campus of the VSB-Technical University of Ostrava.

All three aims of the dissertation defined in Chapter 4 were fulfilled. The core of this dissertation was published in [Jal21]. Other results related to this dissertation can be found in [Jal19a, Jal19b]. The following section summarises the content and the results of the dissertation's experiment.

Because the resulting optimised architecture employs the fog computing paradigm, it was necessary to describe the general fog computing architecture, which consists of the IoT devices and the fog and cloud layers. Standard LoRaWAN architecture was subsequently introduced because the optimised network architecture applied LoRaWAN infrastructure components. The standard LoRaWAN architecture corresponds to the cloud computing paradigm scheme. The fog computing paradigm provides computation nearer to a data source while maintaining cooperation with cloud services.

As a part of this dissertation, I designed three network architectures and subsequently tested which one was an optimal solution. Architecture A integrates the network and application servers in the fog gateway and uses the cloud service for data storage and management. Architecture B is derived from Architecture A, but it adopts a master/slave architecture to eliminate the need for integration of the network and application servers into each gateway. These servers are implemented only in the master gateway, and the other gateways operate in slave mode. The slave gateways perform the same functions as the standard LoRa gateways. Architecture C applies an entirely different principle. Instead of moving the network and application servers, Architecture C optimisation is based on communication between the fog gateways and the application server. In this manner, the gateway can negotiate the necessary information regarding the end device and related session keys. When the gateway obtains all the information, the message payload can be decrypted and subsequently stored in the local database without forwarding the messages to the cloud server.

To determine which of the three proposed architectures is optimal, I conducted various comparisons. First, I compared the proposed architectures in simulations, where queuing theory was applied. The simulation results showed that Architecture C achieves shorter service times than the other proposed architectures. The resulting service times for Architecture C were

strongly affected by whether the processing occurred on the gateway or in the cloud. The larger number of messages processed in the cloud increased the service time. Without fog processing, Architecture C achieves similar results as the other proposed architectures. Second, I compared the functional properties, where the architectures were compared in terms of deployment to the existing LoRaWAN infrastructure, autonomous functions and computational performance requirements. Architecture C achieved the best overall results in both comparisons.

A specific mechanism is required for the decision whether to process a message in the fog gateway or in the cloud. Methods which applied ML were also investigated in detail. These methods assumed that messages from different end devices can be divided into private and public classes. While the private messages cannot be published and therefore require local processing by the fog gateways, public messages can be sent and processed in the public cloud. The first method assumed a specific link between the periodicity of messages and their privacy. It was assumed that the informative payload is usually sent periodically and that the status payload is sent when the observed parameter changes. According to the proposed method, periodic messages are processed in the cloud, while non-periodic messages are processed in the fog gateway. The best results were achieved by the k-NN classifier, whose accuracy reached 97%. The second method examined the application payload. With a defined payload structure, it can be assumed whether the message is informative or status. Based on this information, the classifier can determine the privacy of the message. In this case, the best results were achieved with the naive Bayes classifier. However, the accuracy decreased with the number of unique variable names. The problem of all these methods was that the decision privacy mechanism for messages assumes a link between the message privacy and its content or periodicity. In real IoT data traffic, this link may not exist since privacy can be fully managed by the end device owner. It is therefore necessary to train the classifier according to the specific deployment of the fog gateway.

The individual components of the selected optimal Architecture C were then implemented. I selected the open-source solution Chirpstack to implement the network and application servers, but made major adjustments to the gateway, where I applied an algorithm to decode each message on the fog gateway to obtain the DevAddr identifier, encrypted payload and other necessary data. Because the standard LoRa gateway does not have access to the session keys, I implemented a provisioning script to negotiate the keys between the fog gateway and the application server. A payload decryption algorithm was subsequently introduced into the fog gateway. In this manner, the fog gateway can obtain the application payload, which can be further processed. The fog gateway does not process all received messages, only messages marked for fog processing.

Once all the components of the optimised architecture had been introduced, it was then possible to proceed with verification of results; three tests were designed for this purpose. The first test compared the optimised architecture with the campus LoRaWAN infrastructure in terms of total execution time. The test results showed that the optimised architecture achieves

significantly lower latency in message processing due to the omission of the network and application servers during fog layer computing. The average service times were 121.71 ms in the optimised architecture, 297.93 ms for the campus LoRaWAN network (TTN based), and 231.92 ms for the ChirpStack based LoRaWAN network. It can be stated that in the case of 50% occurrence of private messages, the proposed optimisations reduce the average service time by more than twice compared to the current campus IoT network solution. A simple rule follows from the optimised architecture: if the number of messages processed in the cloud is higher, the average processing time increases. The second test verified compliance with message privacy. If the fog gateway receives a private message, it processes the message, otherwise, the fog gateway forwards the message to the public cloud. The results showed that the first message received from a specific end device serves as a catalyst for the fog gateway to negotiate the associated session keys with the application server. The first message is therefore always processed in the cloud independently of the privacy settings. For subsequent messages, the session keys have already been stored in the local database of the fog gateway. The final test verified the offline functions of the fog gateway. If a connection to the cloud service fails, the fog gateway should temporarily operate autonomously and process all received messages from known end devices. The results of this test demonstrated the correct functioning of this offline process. The results of all performed tests verified the suitability of the optimisations applied to the architecture.

The designed and deployed optimised architecture will serve as a basis for future studies which explore the fog computing paradigm. The main contributions of this dissertation are the following:

1. Optimisation of the current standard LoRaWAN architecture for applications which require quicker responses and message privacy handling.

2. Research of ML-based methods to control the transfer of data computation and storage between the fog and cloud layers.

3. Computation and data storage better adapted to end device requirements and the demands of the end device owner through location awareness.

4. Seamless deployment of fog gateways into existing LoRaWAN infrastructure.

5. In the case of locally available session keys and device identifiers, the fog gateway can be deployed without the network and application servers.

6. Received messages are not lost if a connection between the fog gateway and the cloud layer fails.

References

[1] A. Ikpehai, B. Adebisi, K. M. Rabie, K. Anoh, R. E. Ande, M. Hammoudeh, H. Gacanin, and U. M. Mbanaso, "Low-power wide area network technologies for internet-of-things: A comparative review", *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2225–2240, 2019. DOI: 10.1109/jiot.2018.2883728.

[2] M. Ballerini, T. Polonelli, D. Brunelli, M. Magno, and L. Benini, "NB-IoT versus LoRaWAN: An experimental evaluation for industrial applications", *IEEE Transactions on Industrial Informatics*, vol. 16, no. 12, pp. 7802–7811, 2020. DOI: 10.1109/tii.2020.2987423.

[3] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview", *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 855–873, 2017. DOI: 10.1109/comst.2017.2652320.

[4] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "Overview of cellular LPWAN technologies for IoT deployment: Sigfox, LoRaWAN, and NB-IoT", in *2018 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, IEEE, 2018. DOI: 10.1109/percomw.2018.8480255.

[5] T. Dillon, C. Wu, and E. Chang, "Cloud computing: Issues and challenges", in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, IEEE, 2010. DOI: 10.1109/aina.2010.187.

[6] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges", *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016. DOI: 10.1109/jiot.2016.2579198.

[7] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing", *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009. DOI: 10.1109/mprv.2009.64.

[8] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications", in *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, IEEE, 2015. DOI: 10.1109/hotweb.2015.22.

[9] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues", in *Proceedings of the 2015 Workshop on Mobile Big Data - Mobidata '15*, ACM Press, 2015. DOI: 10.1145/2757384.2757397.

[10] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges", *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, 2018. DOI: 10.1109/comst.2017.2771153.

[11] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions", *IEEE Access*, vol. 6, pp. 47 980–48 009, 2018. DOI: 10.1109/access.2018.2866491.

[12] OpenFog Consortium, *IEEE standard for adoption of OpenFog reference architecture for fog computing*. DOI: 10.1109/ieeestd.2018.8423800.

[13] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things", in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*, ACM Press, 2012. DOI: 10.1145/2342509.2342513.

[14] M. Chiang and T. Zhang, "Fog and IoT: An overview of research opportunities", *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016. DOI: 10.1109/jiot.2016.2584538.

[15] T. N. Gia, M. Jiang, A.-M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fog computing in healthcare internet of things: A case study on ECG feature extraction", in *2015 IEEE International Conference on Computer and Information Technology;Ubiquitous Computing and Communications;Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, IEEE, 2015. DOI: 10.1109/cit/iucc/dasc/picom.2015.51.

[16] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges", *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018. DOI: 10.1016/j.future.2016.11.009.

[17] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog computing for the internet of things: Security and privacy issues", *IEEE Internet Computing*, vol. 21, no. 2, pp. 34–42, 2017. DOI: 10.1109/mic.2017.37.

[18] P. Fraga-Lamas, M. Celaya-Echarri, P. Lopez-Iturri, L. Castedo, L. Azpilicueta, E. Aguirre, M. Suárez-Albela, F. Falcone, and T. M. Fernández-Caramés, "Design and experimental validation of a LoRaWAN fog computing based architecture for IoT enabled smart campus applications", *Sensors*, vol. 19, no. 15, p. 3287, 2019. DOI: 10.3390/s19153287.

[19] P. Barro, M. Zennaro, J. Degila, and E. Pietrosemoli, "A smart cities LoRaWAN network based on autonomous base stations (BS) for some countries with limited internet access", *Future Internet*, vol. 11, no. 4, p. 93, 2019. DOI: 10.3390/fi11040093.

[20] S. Sobhi, M. A. Ali, and M. F. Abdelkader, "Combining fog computing and lorawan technologies for smart cities applications", ser. The 2nd Europe – Middle East – North African Regional Conference of the International Telecommunications Society: "Leveraging Technologies For Growth", International Telecommunications Society (ITS), 2019.

[21] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer, "A comparative study of LPWAN technologies for large-scale IoT deployment", *ICT Express*, vol. 5, no. 1, pp. 1–7, 2019. DOI: 10.1016/j.icte.2017.12.005.

[22] Y. Oh, J. Lee, and C.-K. Kim, "TRILO: A traffic indication-based downlink communication protocol for LoRaWAN", *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–14, Sep. 2018. DOI: 10.1155/2018/6463097.

[23] F. V. den Abeele, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Scalability analysis of large-scale LoRaWAN networks in ns-3", *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2186–2198, 2017. DOI: 10.1109/jiot.2017.2768498.

[24] J.-T. Lim and Y. Han, "Spreading factor allocation for massive connectivity in LoRa systems", *IEEE Communications Letters*, vol. 22, no. 4, pp. 800–803, 2018. DOI: 10.1109/lcomm.2018.2797274.

[25] A. Tiurlikova, N. Stepanov, and K. Mikhaylov, "Method of assigning spreading factor to improve the scalability of the LoRaWan wide area network", in *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, IEEE, 2018. DOI: 10.1109/icumt.2018.8631273.

[26] E. Sallum, N. Pereira, M. Alves, and M. Santos, "Improving quality-of-service in LoRa low-power wide-area networks through optimized radio resource management", *Journal of Sensor and Actuator Networks*, vol. 9, no. 1, p. 10, 2020. DOI: 10.3390/jsan9010010.

[27] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin, "Improving reliability and scalability of LoRaWANs through lightweight scheduling", *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1830–1842, 2018. DOI: 10.1109/jiot.2018.2815150.

[28] A. Lavric, A. I. Petrariu, E. Coca, and V. Popa, "LoRa traffic generator based on software defined radio technology for LoRa modulation orthogonality analysis: Empirical and experimental evaluation", *Sensors*, vol. 20, no. 15, p. 4123, 2020. DOI: 10.3390/s20154123.

[29] P. G. V. Naranjo, Z. Pooranian, M. Shojafar, M. Conti, and R. Buyya, "FOCAN: A fog-supported smart city network architecture for management of applications in the internet of everything environments", *Journal of Parallel and Distributed Computing*, vol. 132, pp. 274–283, Oct. 2019. DOI: 10.1016/j.jpdc.2018.07.003.

[30] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments", *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017. DOI: 10.1002/spe.2509.

[31] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of experience (QoE)-aware placement of applications in fog computing environments", *Journal of Parallel and Distributed Computing*, vol. 132, pp. 190–203, Oct. 2019. DOI: 10.1016/j.jpdc.2018.03.004.

[32] S. Z. Khan, H. Malik, J. L. R. Sarmiento, M. M. Alam, and Y. L. Moullec, "DORM: Narrowband IoT development platform and indoor deployment coverage analysis", *Procedia Computer Science*, vol. 151, pp. 1084–1091, 2019. DOI: 10.1016/j.procs.2019.04.154.

[33] S. S. Basu, A. K. Sultania, J. Famaey, and J. Hoebeke, "Experimental performance evaluation of NB-IoT", in *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, IEEE, 2019. DOI: 10.1109/wimob.2019.8923221.

[34] A. P. Matz, J.-A. Fernandez-Prieto, J. Cañada-Bago, and U. Birkel, "A systematic analysis of narrowband IoT quality of service", *Sensors*, vol. 20, no. 6, p. 1636, 2020. DOI: 10.3390/s20061636.

[35] L. Oliveira, J. Rodrigues, S. Kozlov, R. Rabêlo, and V. Albuquerque, "MAC layer protocols for internet of things: A survey", *Future Internet*, vol. 11, no. 1, p. 16, 2019. DOI: 10.3390/fi11010016.

[37] R. S. Sinha, Y. Wei, and S.-H. Hwang, "A survey on LPWA technology: LoRa and NB-IoT", *ICT Express*, vol. 3, no. 1, pp. 14–21, 2017. DOI: 10.1016/j.icte.2017.03.004.

[38] R. Ratasuk, B. Vejlgaard, N. Mangalvedhe, and A. Ghosh, "NB-IoT system for m2m communication", in *2016 IEEE Wireless Communications and Networking Conference*, IEEE, 2016. DOI: 10.1109/wcncw.2016.7552737.

[39] W. Ayoub, A. E. Samhat, F. Nouvel, M. Mroue, and J.-C. Prevotet, "Internet of mobile things: Overview of LoRaWAN, DASH7, and NB-IoT in LPWANs standards and supported mobility", *IEEE Communications Surveys & Tutorials*, vol. 21, no. 2, pp. 1561–1581, 2019. DOI: 10.1109/comst.2018.2877382.

[40] C. B. Mwakwata, H. Malik, M. M. Alam, Y. L. Moullec, S. Parand, and S. Mumtaz, "Narrowband internet of things (NB-IoT): From physical (PHY) and media access control (MAC) layers perspectives", *Sensors*, vol. 19, no. 11, p. 2613, 2019. DOI: 10.3390/s19112613.

[41] M. Chen, Y. Miao, Y. Hao, and K. Hwang, "Narrow band internet of things", *IEEE Access*, vol. 5, pp. 20 557–20 577, 2017. DOI: 10.1109/access.2017.2751586.

[42] Sigfox, *Sigfox Technology*, Available online, https://www.sigfox.com/, (Accessed: 12 December 2019).

[43] B. Vejlgaard, M. Lauridsen, H. Nguyen, I. Z. Kovacs, P. Mogensen, and M. Sorensen, "Coverage and capacity analysis of sigfox, LoRa, GPRS, and NB-IoT", in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*, IEEE, 2017. DOI: 10.1109/vtcspring.2017.8108666.

[46] LoRa Alliance, *LoRaWAN 1.0.4 Specification*, Available online, https://lora-alliance.org/resource_hub/lorawan-104-specification-package, (Accessed: 1 February 2021).

[47] M. Bor and U. Roedig, "LoRa transmission parameter selection", in *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, IEEE, 2017. DOI: 10.1109/dcoss.2017.10.

[48] R. Kufakunesu, G. P. Hancke, and A. M. Abu-Mahfouz, "A survey on adaptive data rate optimization in LoRaWAN: Recent solutions and major challenges", *Sensors*, vol. 20, no. 18, p. 5044, Sep. 2020. DOI: 10.3390/s20185044.

[49] D. Croce, M. Gucciardo, I. Tinnirello, D. Garlisi, and S. Mangione, "Impact of spreading factor imperfect orthogonality in lora communications", in *Digital Communication. Towards a Smart and Secure Future Internet*, Springer International Publishing, 2017, pp. 165–179. DOI: 10.1007/978-3-319-67639-5_13.

[50] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, and I. Tinnirello, "Impact of lora imperfect orthogonality: Analysis of link-level performance", *IEEE Communications Letters*, vol. 22, no. 4, pp. 796–799, 2018. DOI: 10.1109/lcomm.2018.2797057.

[51] A. Waret, M. Kaneko, A. Guitton, and N. El Rachkidy, "Lora throughput analysis with imperfect spreading factor orthogonality", *IEEE Wireless Communications Letters*, vol. 8, no. 2, pp. 408–411, 2019. DOI: 10.1109/lwc.2018.2873705.

[52] *Duty Cycle for LoRaWAN Devices*, Available online, https://www.thethingsnetwork.org/docs/lorawan/duty-cycle.html, (Accessed: 18 March 2021).

[53] M. Saelens, J. Hoebeke, A. Shahid, and E. D. Poorter, "Impact of EU duty cycle and transmission power limitations for sub-GHz LPWAN SRDs: An overview and future challenges", *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, 2019. DOI: 10.1186/s13638-019-1502-5.

[54] Semtech Corporation, *LoRa and LoRaWAN: A Technical Overview*, Available online, https://lora-developers.semtech.com/uploads/documents/files/LoRa_and_LoRaWAN-A_Tech_Overview-Downloadable.pdf, (Accessed: 1 March 2021).

[55] Semtech Corporation, *LoRa Modulation Basics AN1200.22*, Available online, https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf, (Accessed: 1 March 2021).

[56] LoRa Alliance, *Understanding the LoRa Adaptive Data Rate*, Available online, https://lora-developers.semtech.com/uploads/documents/files/Understanding_LoRa_Adaptive_Data_Rate_Downloadable.pdf, (Accessed: 25 February 2021).

[57] J. Toussaint, N. E. Rachkidy, and A. Guitton, "Performance analysis of the on-the-air activation in LoRaWAN", in *2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, IEEE, 2016. DOI: 10.1109/iemcon.2016.7746082.

[58] J. Kim and J. Song, "A dual key-based activation scheme for secure LoRaWAN", *Wireless Communications and Mobile Computing*, vol. 2017, pp. 1–12, 2017. DOI: 10.1155/2017/6590713.

[59] A. Sunyaev, *Cloud Computing*, ser. Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies. Springer International Publishing, 2020, pp. 195–236. DOI: 10.1007/978-3-030-34957-8_7.

[60] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: A survey", *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019. DOI: 10.1016/j.future.2019.02.050.

[61] R. F. de Mello and M. A. Ponti, *Machine learning : A practical approach on the statistical learning theory*. Cham, Switzerland: Springer International Publishing, 2018. DOI: 10.1007/978-3-319-94989-5.

[62] S. Gollapudi, *Practical Machine Learning*, ser. Community experience distilled. Packt Publishing, 2016, ISBN: 9781784399689.

[63] S. Marsland, *Machine Learning: An Algorithmic Perspective, Second Edition*, 2nd. Chapman & Hall/CRC, 2014, ISBN: 1466583282.

[64] U. Kamath and K. Choppella, *Mastering Java Machine Learning: A Java Developer's Guide to Implementing Machine Learning and Big Data Architectures*. Packt Publishing, 2017, ISBN: 1785880519.

[65] L. Rokach and O. Maimon, *Decision Trees*, ser. Data Mining and Knowledge Discovery Handbook. Boston, MA: Springer, 2005, pp. 165–192. DOI: 10.1007/0-387-25465-x_9.

[66] R. García Leiva, A. Fernández Anta, V. Mancuso, and P. Casari, "A novel hyperparameter-free approach to decision tree construction that avoids overfitting by design", *IEEE Access*, vol. 7, pp. 99 978–99 987, 2019. DOI: 10.1109/access.2019.2930235.

[67] M. Schonlau and R. Y. Zou, "The random forest algorithm for statistical learning", *The Stata Journal: Promoting communications on statistics and Stata*, vol. 20, no. 1, pp. 3–29, Mar. 2020. DOI: 10.1177/1536867x20909688.

[68] M. Bicego and M. Loog, "Weighted k-nearest neighbor revisited", in *2016 23rd International Conference on Pattern Recognition (ICPR)*, IEEE, 2016. DOI: 10.1109/icpr.2016.7899872.

[69] M. N. Murty and R. Raghava, *Linear Support Vector Machines*, ser. Support Vector Machines and Perceptrons: Learning, Optimization, Classification, and Application to Social Networks. Springer International Publishing, 2016, pp. 41–56. DOI: 10.1007/978-3-319-41063-0_4.

[70] Murty, M. N. and Raghava, Rashmi, *Kernel-Based SVM*, ser. Support Vector Machines and Perceptrons: Learning, Optimization, Classification, and Application to Social Networks. Springer International Publishing, 2016, pp. 57–67. DOI: 10.1007/978-3-319-41063-0_5.

[71] I. D. Dinov, *Probabilistic Learning: Classification Using Naive Bayes*, ser. Data Science and Predictive Analytics: Biomedical and Health Applications using R. Springer International Publishing, 2018, pp. 289–305. DOI: 10.1007/978-3-319-72347-1_8.

[72] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015, Available online, http://neuralnetworksanddeeplearning.com/.

[73] I. Vondrak, *Umela inteligence a neuronove site*. Ostrava: VSB - Technical University of Ostrava, 1994, ISBN: 80-7078-259-5.

[74] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey", *Heliyon*, vol. 4, no. 11, 2018. DOI: 10.1016/j.heliyon.2018.e00938.

[75] T. Fine, *Feedforward neural network methodology*. New York: Springer, 1999, ISBN: 978-0387226491.

[76] A. Zheng, *Evaluating machine learning models : A beginner's guide to key concepts and pitfalls*. Sebastopol, Calif: O'Reilly Media, 2015, ISBN: 978-1-491-93246-9.

[77] H. M and S. M.N, "A review on evaluation metrics for data classification evaluations", *International Journal of Data Mining & Knowledge Management Process*, vol. 5, no. 2, pp. 01–11, 2015. DOI: 10.5121/ijdkp.2015.5201.

[78] D. Berrar, "Cross-validation", in *Encyclopedia of Bioinformatics and Computational Biology*, Elsevier, 2019, pp. 542–545. DOI: 10.1016/b978-0-12-809633-8.20349-x.

[79] S. Yadav and S. Shukla, "Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification", in *2016 IEEE 6th International Conference on Advanced Computing (IACC)*, IEEE, 2016. DOI: 10.1109/iacc.2016.25.

[80] LoRa Alliance, *LoRaWAN Backend Interfaces*, Available online, https://lora-alliance.org/wp-content/uploads/2020/11/TS002-1.1.0_LoRaWAN_Backend_Interfaces.pdf, (Accessed: 20 February 2021).

[82] IMST GmbH, *WiMOD iC880A Datasheet*, Available online, https://www.wireless-solutions.de/downloads/Radio-Modules/iC880A/iC880A_Datasheet_V1_0.pdf, (Accessed: 15 March 2019).

[83] F. Metzger, T. Hobfeld, A. Bauer, S. Kounev, and P. E. Heegaard, "Modeling of aggregated IoT traffic and its application to an IoT cloud", *Proceedings of the IEEE*, vol. 107, no. 4, pp. 679–694, 2019. DOI: 10.1109/jproc.2019.2901578.

[84] *ChirpStack*, Available online, https://www.chirpstack.io, (Accessed: 2 December 2020).

[85] O. Pospisil, R. Fujdiak, K. Mikhaylov, H. Ruotsalainen, and J. Misurec, "Testbed for LoRaWAN security: Design and validation through man-in-the-middle attacks study", *Applied Sciences*, vol. 11, no. 16, p. 7642, 2021. DOI: 10.3390/app11167642.

[86] *Lora network packet forwarder project*, Available online, https://www.github.com, (Accessed: 8 December 2020).

[87] *ChirpStack API*, Available online, https://www.chirpstack.io/application-server/api/, (Accessed: 10 March 2021).

Candidate's research cited in this work

[Vas18] D. Vasicek, J. Jalowiczor, L. Sevcik, and M. Voznak, "IoT smart home concept", in *2018 26th Telecommunications Forum (TELFOR)*, IEEE, Belgrade, Nov. 2018. DOI: 10.1109/telfor.2018.8612078.

[Jal19a] J. Jalowiczor and M. Voznak, "Proposal and implementation of probe for sigfox technology", in *Lecture Notes in Electrical Engineering*, Springer International Publishing, Apr. 2019, pp. 420–428. DOI: 10.1007/978-3-030-14907-9_41.

[Gre19] E. Gresak, J. Jalowiczor, J. Rozhon, F. Rezac, and J. Safarik, "Detection of changes in the qualitative parameters for LoRaWAN and SigFox network", in *Disruptive Technologies in Information Sciences II*, M. Blowers, R. D. Hall, and V. R. Dasari, Eds., SPIE, May 2019. DOI: 10.1117/12.2518853.

[Jal21] J. Jalowiczor, J. Rozhon, and M. Voznak, "Study of the efficiency of fog computing in an optimized LoRaWAN cloud architecture", *Sensors*, vol. 21, no. 9, p. 3159, May 2021. DOI: 10.3390/s21093159, **IF: 3.576, Q2** (2020).

[Jal19b] J. Jalowiczor, E. Gresak, F. Rezac, J. Rozhon, and J. Safarik, "Development and deployment of the main parts of LoRaWAN private network", in *Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2019*, SPIE, May 2019. DOI: 10.1117/12.2518225.

10 List of Candidate's Research Results and Activities

10.1 Participation in Research Projects

- Grant agreement number 761349 – **TETRAMAX, Technology Transfer via Multinational Application Experiments**, The European Union's Horizon 2020 research and innovation programme, 2017 – 2021, PI at VSB-TUO: prof. Voznak, my position: junior researcher (6/2020 – 12/2020).
- Reg. No. EG19_262/0020122 – **Development of external modules TAMAS II**, Ministry of Industry and Trade within programme OP PIK, 2020 – 2022, PI at VSB-TUO: Dr. Tovarek, my position: junior researcher (3/2021 – still active).
- Reg. No. HS7851901 – **Modular Open Source Secure Mobile Communication**, contract research for The National Cyber and Information Security Authority (NUKIB), 2019 – 2021, PI at VSB-TUO: prof. Voznak, my position: junior researcher (3/2020 -– 3/2021).
- Reg. No. VI20172020079 – **Secure Gateway for the Internet of Things (SIoT)**, Ministry of the Interior within programme Security research for needs of the Czech Republic, 2017 - 2020, PI at VSB-TUO: prof. Voznak, my position: junior researcher (10/2017 – 3/2020).
- Reg. No. CZ.01.1.02/0.0/0.0/16_084/0009815 – **Development of Industrial Sensors and Extension of IoT Services Portfolio**, Ministry of Industry and Trade within programme APLIKACE III., 2017 - 2020, PI at VSB-TUO: prof. Voznak, my position: junior researcher (10/2019 – 3/2020).
- Reg. No. SP2021/25 – **Networks and Communication Technologies for Smart Cities IV**, Specific research SGS FEI VSB-TUO project, 2021, PI at VSB-TUO: Dr. Rezac, my position: junior researcher (still active).
- Reg. No. SP2020/65 – **Networks and Communication Technologies for Smart Cities III**, Specific research SGS FEI VSB-TUO project, 2020, PI at VSB-TUO: Dr. Rezac, my position: junior researcher (2020).
- Reg. No. SP2019/41 – **Networks and Communication Technologies for Smart Cities II**, Specific research SGS FEI VSB-TUO project, 2019, PI at VSB-TUO: Dr. Rezac, my position: junior researcher (2019).
- Reg. No. SP2018/59 – **Networks and Communication Technologies for Smart Cities**, Specific research SGS FEI VSB-TUO project, 2018, PI at VSB-TUO: Dr. Rezac, my position: junior researcher (2018).

- Reg. No. SP2017/174 – **Networks and their security, modeling, simulation, knowledge mining and communication technologies for smart cities**, Specific research SGS FEI VSB-TUO project, 2017, PI at VSB-TUO: prof. Voznak, my position: junior researcher (2017).
- Reg. No. HS4401512 – **Unattended test tools for mobile networks**, contract research for Huawei Technologies, 2015 - 2018, PI at VSB-TUO: prof. Voznak, my position: junior researcher (2016).

10.2 Results of Research Activities Achieved During My Ph.D. Study

ORCID: 0000-0002-0838-4024

- Results indexed in Elsevier Scopus: 17 (14 conference papers, 3 articles in journals)
- Results indexed in Web of Science: 15 (12 conference papers, 3 articles in journals)
- Registered software (2)

10.3 Other Results Achieved During My Ph.D. Study Indexed in Elsevier Scopus or Web of Science

[Others01] H. T. Van, Q. -N. Van, D. H. Le, H. -P. Van, J. Jalowiczor, H. -S. Nguyen, and M. Voznak, "Opportunistic DF-AF Selection Relaying in Hybrid Wireless and Power Line Communication for Indoor IoT Networks", Sensors, vol. 21, no. 16, 2021. DOI: 10.3390/s21165469, **IF: 3.576, Q2** (2020).

[Others02] J. Rozhon, F. Rezac, J. Jalowiczor, and L. Behan, "Augmenting Speech Quality Estimation in Software-Defined Networking Using Machine Learning Algorithms", Sensors, vol. 21, no. 10, 2021. DOI: 10.3390/s21103477, **IF: 3.576, Q2** (2020).

[Others03] H. S. Nguyen, N. X. H. Nguyen, Q. P. Ma, J. Jalowiczor, and M. Voznak, "Symbol Error Probability of Secondary User in Underlay Cognitive Radio Networks with Adaptive Transmit Power Constraint", in Multimedia Communications, Services and Security, Springer International Publishing, 2020, pp. 307-319. DOI: 10.1007/978-3-030-59000-0_23.

[Others04] P. Partila, J. Tovarek, J. Rozhon, J. Jalowiczor, "Human stress detection from the speech in danger situation", in Mobile Multimedia/Image Processing, Security, and Applications 2019, 2019, art. no 109930U. DOI: 10.1117/12.2521405.

[Others05] J. Rozhon, F. Rezac, J. Safarik, E. Gresak, J. Jalowiczor, "Measuring and monitoring the QoS and QoE in software defined networking environments", in Signal Processing, Sensor/Information Fusion, and Target Recognition XXVIII, 2019, art. no. 110181M. DOI: 10.1117/12.2518838.

[Others06] F. Rezac, J. Rozhon, J. Safarik, J. Jalowiczor, E. Gresak, "Using embedded operating system as a modular provisioning platform for IP telephony", in Mobile Multimedia/Image Processing, Security, and Applications 2019, 2019, art. no. 109930I. DOI: 10.1117/12.2517752.

[Others07] K. Witas, S. Zabka, J. Frnda, M. Novák, J. Jalowiczor, M. Stolarik, R. Jaros, "Analysis of the effect of long-time thermal load on the total losses of the selected fiber-optic couplers", in Integrated Optics: Design, Devices, Systems, and Applications V, 2019, art. no 110311D. DOI: 10.1117/12.2519836.

[Others08] K. Witas, M. Stolarik, M. Pinka, S. Zabka, J. Jalowiczor, M. Novak, R. Jaros, "Perimetric monitoring: A comparison of a classical seismic sensor and fiber-optic interferometric sensor", in Optical Sensors 2019, 2019, art. no 110282K. DOI: 10.1117/12.2522309.

[Others09] J. Safarik, J. Jalowiczor, E. Gresak, and J. Rozhon, "Genetic algorithm for automatic tuning of neural network hyperparameters", in Autonomous Systems: Sensors, Vehicles, Security, and the Internet of Everything, 2018, art. no 106430Q. DOI: 10.1117/12.2304955.

[Others10] J. Rozhon, E. Gresak, and J. Jalowiczor, "Using LSTM Cells for SIP Dialogs Mapping and Security Analysis", in 2018 26th Telecommunications Forum (TELFOR), 2018, pp. 1-4. DOI: 10.1109/TELFOR.2018.8612019.

[Others11] L. Behan, L. Kapicak, J. Jalowiczor, "Development and implementation of VoIP honeypots with wide range of analysis", in Cyber Sensing 2018, 2018, p. 28-. DOI: 10.1117/12.2304602.

[Others12] F. Rezac, J. Safarik, E. Gresak, J. Rozhon, J. Jalowiczor, "Automatic voice control system for UAV-based accessories", in Unmanned Systems Technology XX, 2018, p. 26-. DOI: 10.1117/12.2301242.