Julius-Maximilians-Universität Würzburg
Institut für Informatik
Lehrstuhl für Kommunikationsnetze
Prof. Dr. Tobias Hoßfeld

Monitoring the Quality of Streaming and Internet of Things Applications

Frank Loh

Würzburger Beiträge zur
Leistungsbewertung verteilter Systeme

Bericht 01/24

Würzburger Beiträge zur Leistungsbewertung verteilter Systeme

Herausgeber

Prof. Dr. Tobias Hoßfeld
Universität Würzburg
Institut für Informatik
Lehrstuhl für Kommunikationsnetze
Am Hubland
D-97074 Würzburg

Tel.: +49-931-31-86631
Fax.: +49-931-31-86632
email: tobias.hossfeld@uni-wuerzburg.de

in memoriam: Prof. Dr.-Ing. Phuoc Tran-Gia †2023

Satz

Reproduktionsfähige Vorlage des Autors.
Gesetzt in LaTeX Linux Libertine 10pt.

ISSN 1432-8801

Monitoring the Quality of Streaming and Internet of Things Applications

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Julius–Maximilians–Universität Würzburg

vorgelegt von

Frank Loh

geboren in

Gunzenhausen

Würzburg 2024

Eingereicht am: 09.10.2023
bei der Fakultät für Mathematik und Informatik
1. Gutachter: Prof. Dr. Tobias Hoßfeld
2. Gutachter: Prof. Dr. Franco Davoli
3. Gutachter: Prof. Dr. Filip De Turck
Tag der mündlichen Prüfung: 04.12.2023

Danksagung

Mit dem Ende meiner Doktorarbeit möchte ich die Gelegenheit nutzen, einigen Menschen, die mich während dieser Zeit und damit während eines großen Abschnitts meines Lebens begleitet und unterstützt haben, meinen herzlichen Dank auszusprechen.

Zuallererst möchte ich mich bei Prof. Dr. Tobias Hoßfeld und bei Prof. Dr.-Ing. Phuoc Tran-Gia dafür bedanken, dass sie mir die Möglichkeit gegeben haben bei ihnen am Lehrstuhl für Kommunikationsnetze zu promovieren. Prof. Phuoc Tran-Gia hat mich bereits zu Studentenzeiten als Hilfswissenschaftler in den Lehrstuhl integriert und trotz einer zunächst ungewissen Zukunft nach seiner Emeritierung als Doktorand eingestellt. Danach hat mich Prof. Tobias Hoßfeld übernommen und ist mir stets mit Rat und Tat zur Seite gestanden. Ganz besonders habe ich das sehr gute fachliche, aber auch persönliche Miteinander am Lehrstuhl geschätzt. Während der Arbeit an der Doktorarbeit, an Forschungsprojekten oder in der Lehre hatte ich immer genügend Freiheiten, um eigene Ideen außerhalb der Projekte zu entwickeln und zu diskutieren. Dafür möchte ich mich hiermit ganz herzlich bedanken.

Als Nächstes sind die Gutachter meiner Dissertation und hierbei zunächst mein Doktorvater und Erstgutachter Prof. Dr. Tobias Hoßfeld zu nennen. Ganz besonders möchte ich ihm für die ausgezeichnete Betreuung meiner Arbeit und die wertvollen Kommentare, Hinweise und Ergänzungen bedanken. Weiterer Dank gilt meinem Zweitgutachter Prof. Dr. Franco Davoli, den ich während einer Workshoporganisation kennenlernen durfte. Die wertvollen Kommentare zu meiner Arbeit und Diskussionen auch darüber hinaus haben mich sehr unterstützt. In diesem Zuge möchte ich auch meinem dritten Gutachter, Prof. Dr.

Filip De Turck dafür danken, dass er sich bereit erklärt hat, meine Dissertation zu begutachten. Ich habe stets die fachlichen Diskussionen auf verschiedenen Netzwerk-Monitoring Konferenzen sehr geschätzt. Außerdem möchte ich hiermit noch Prof. Dr.-Ing. Samuel Kounev und Prof. Dr. Guido Dietl dafür danken, dass sie sich als Prüfer für meine Disputation zur Verfügung gestellt haben.

In meiner Zeit als Doktorand und während verschiedener Projekte haben mich stets verschiedene Post-Docs unterstützt. Hiermit möchte ich zunächst Dr. Florian Wamser herzlich danken, der mich als mein erster Gruppenleiter bei allen fachlichen Problemen stets unterstützt hat. Vor allem durch sein ausgezeichnetes technisches Verständnis verschiedenster Systeme konnte er mir immer weiter helfen. Zudem war er stets auch außerhalb des Lehrstuhls immer für ein Bier, wie zum Beispiel beim Skifahren, zu haben und ich bin dankbar dafür, dass er dadurch auch ein guter Freund wurde. Weiterhin ist hier Dr. Florian Metzger zu nennen, dem ich für die Projektleitung in mehreren IoT Projekten und die ausgezeichnete fachliche Unterstützung im Allgemeinen, aber auch gerade im IoT Bereich danken möchte. Für die Endphase meiner Doktorarbeit, und gerade bei der Unterstützung und dem detaillierten Feedback bei einigen Publikationen und für die umfassenden Korrekturen meiner Arbeit möchte ich Dr. Stefan Geißler sehr herzlich danken. Er hatte nicht nur als Gruppenleiter, sondern auch für wildeste Forschungs- und Publikationsideen immer eine offene Bürotür für Diskussionen.

Außerdem möchte ich meinen aktuellen und ehemaligen Kollegen Dr. Kathrin Borchert, Dr. Valentin Burger, Katharina Dietz, Nicholas Gray, Alexej Grigorjew, Prof. Dr. Matthias Hirth, Dr. Christopher Metter, Dr. Christian Moldovan, Kien Nguyen, Dr. Anh Nguyen-Ngoc, Fabian Poignée, Simon Raffeck, David Raunecker, Dr. Susanna Schwarzmann, Anika Seufert, Prof. Dr. Michael Seufert, Viktoria Vomhoff und Prof. Dr. Thomas Zinner für die gemeinsame Zeit, die unzähligen Diskussionen und Kaffeepausen, aber auch für alle Aktivitäten außerhalb des Universitätsalltags danken. Ganz besonders möchte ich Dr. Lam Dinh-Xuan und Nikolas Wehner danken, mit denen ich lange Zeit ein Büro teilen durfte. Durch das stets gute Klima am Lehrstuhl haben sich viele Freundschaften gebil-

det.

Ein Leben am Lehrstuhl ohne unsere gute Seele Alison Wichmann hätte man sich nicht vorstellen können. Dafür möchte ich ihr hiermit sehr herzlich danken. Seien es Dienstreiseanträge oder -abrechnungen, diverse Bestellungen, Fragen zur Verwaltung oder zu Projektdokumenten. Sie hatte stehts ein offenes Ohr, um bei der Lösung von Problemen tatkräftig zu helfen. Aber auch abseits der Arbeit war sie immer für einen Kaffee zu haben.

Nichts an einer Universität funktioniert ohne Studierende. Dies gilt auch für die Unterstützung bei der Lehre, bei Projekten und diversen Forschungsarbeiten. Somit möchte ich mich hiermit bei allen meinen Hiwis und Studierenden bedanken, sei es für eine Hiwitätigkeit am Lehrstuhl, eine Abschlussarbeit, ein Seminar, eine gemeinsame Publikation oder auch nur ein gemeinsames Bier oder einen Kaffee.

Zu guter Letzt möchte ich mich bei meinen Freunden und meiner Familie bedanken. Vielen Dank an meine Freunde für die Unterstützung vor allem abseits des Arbeitsalltags, sei es durch langjährige Freunde aus Ehingen und Umgebung, Freunde aus der Schulzeit, aus der Zeit des Studiums, aber auch Freunde, die ich während der Promotion kennengelernt habe. Der gute Ausgleich zur Arbeit hat immer wieder zu neuen Ideen und Problemlösungsansätzen geführt. Insbesondere gilt mein unendlicher Dank meinen Eltern Erna und Hartmut, die mir die Ausbildung und das Studium erst ermöglicht haben. Sie haben mich stehts gefordert und gefördert und mich bei allen meinen Entscheidungen immer mit voller Kraft unterstützt und lieber selbst zurückgesteckt, um mir den nächsten Schritt zu ermöglichen. Während der Doktorarbeit habe ich auch meine Freundin Viktoria kennengelernt. Ihr möchte ich herzlich für die Unterstützung, vor allem in den letzten Monaten meiner Doktorarbeit, danken, aber auch für die schöne Zeit abseits der Arbeit, für die guten Diskussionen, und dafür, dass sie jeden Tag an meiner Seite ist.

Contents

1 Introduction

The foundation of our globally connected world's success lies in continuous advancements in industries, agriculture, services, work, and leisure activities. Smart sensors continuously monitor various aspects and transmit vast amounts of data to processing centers to optimize production. Industry 4.0 facilities utilize cameras for real-time fault detection, generating significant streaming data. Smartphone apps track our daily activities, diet, and sleep to enhance health and time management. This rapid automation affects a significant portion of the global population. Additionally, online activities, multimedia consumption, and social media engagement of everyone leads to a substantial increase in mainly mobile-based traffic. This growth is evident as the number of global Internet users surpassed five billion between 2022 and 2023 [37, 38]. Internet traffic has exceeded 100 billion gigabytes in 2022 [39], with video streaming contributing significantly to this volume, accounting for 65 % of general and over 67 % of global mobile traffic in 2022 [40]. Among this landscape, the influence of the Internet of Things (IoT) is gaining momentum, projected to reach up to 18 % of traffic share by 2026 [41]. However, this increasing traffic comes with resource and energy consumption challenges across end devices, networks, and processing entities. More complex network traffic by a growing variety of flows necessitates more sophisticated monitoring and processing systems. Furthermore, according to Cisco, the number of devices connected to Internet Protocol (IP) networks is set to surpass the global population by over threefold in 2023, reaching nearly 30 billion devices [42]. Notably, a significant proportion of these devices are dedicated to Machine-To-Machine (M2M) connections, functioning without human interaction [42] contributing to the overall resource requirements

with technologies including IP networks, Bluetooth, and Low Power Wide Area Networks (LPWANs), having annual growth rates between 16 % and 18 % [43]. Yet, among these demands and development, there is a mandatory need for energy consumption reduction, a more efficient resource usage and streamlined processes without losing service quality to address the challenges posed by climate change. While energy requirements in the information and communication technology sector continue to rise, there is a simultaneous demand for less resource-intensive and more efficient solutions to manage data transmission, monitor facilities, and simplify processes. This ongoing and evolving usage of networks, involving personal end devices and the substantial dissemination of IoT applications in large-scale sensor networks, presents two critical challenges that require attention. Firstly, there is the task of effectively managing the vast and continually expanding data traffic. Secondly, there is the need to address the substantial number of end devices resulting from the rapid adoption of the IoT.

For that reason, ensuring a robust network quality that aligns with all Service Level Agreements (SLA) is crucial. Consequently, it becomes imperative to identify significant factors that contribute to network quality degradation within existing deployments. This knowledge can be leveraged to inform the implementation of new deployments, taking advantage on insights gained from ongoing monitoring of established deployments. To attain comprehensive and accurate network measurements, a thorough comprehension of Key Quality Indicators (KQIs) is essential. This understanding extends to identify the root causes behind quality issues specific to individual applications or application areas. Such an initial step is beneficial for the development of tailored network monitoring instances that align with the unique requirements of applications, effectively mitigating the risk of excessive costs or resource allocation. However, as the landscape evolves with the roll-out of the IoT, a notable gap exists in comprehending emerging network access technologies and the distinct demands of novel applications. Additionally, there remains a degree of uncertainty surrounding network behavior under varying load conditions, necessitating in-depth investigation. This research is essential for the deployment of

cutting-edge networks to uphold high quality standards and remain scalable to accommodate the anticipated surge in load caused by future applications.

Currently, an extensive analysis of application quality, particularly those generating a substantial portion of global Internet traffic, is commonplace. For instance, widely investigated applications like video streaming employ various quality quantification methods. These methods carefully examine each transmitted packet in both uplink and downlink directions, utilizing resource-intensive Machine Learning (ML) solutions to predict playback quality (e.g. [44–46]). However, with the increasing number of video stream viewers consuming content at higher quality levels, the volume of data requiring analysis is growing. Consequently, more resources are demanded to monitor and process data, necessitating high-powered hardware and resulting in escalated energy consumption. Hence, a more resource-efficient and scalable approach is imperative, one that reduces computational complexity while simultaneously addressing energy consumption. Such an approach should be capable of accommodating a substantial number of parallel streaming sessions generated by multimedia applications consumed by individuals but also monitoring entities in the IoT context. However, the landscape in the rapidly expanding domain of IoT is much more diverse and goes beyond the transmission of data in existing networks. There, many networks are still in the planning phase and require a comprehensive grasp of critical quality factors regarding novel application domains. Beyond the current predominant focus on simple coverage in emerging IoT access network technologies, there lies a fundamental need to thoroughly comprehend these technologies. This understanding should encompass complex and sophisticated details of network expansion, data transmission, resource and energy consumption, and factors contributing to interference, message collisions, and data loss. Such foundational knowledge is integral to ensure a robust, future-proof IoT network plan and to effectively address application quality considerations.

The challenges posed by both total network traffic volume and the distribution of numerous devices are central to current and prospective network monitoring and management efforts. With this in mind, we focus our attention on

video streaming as a critical application area in our private life and in industry that significantly contributes to the substantial data flow across the Internet on a daily basis. In this context, we introduce a streamlined approach that exclusively relies on uplink data. Our objective is to assess the primary causes of quality degradation, which profoundly impact end users' perceived experience. This lightweight technique is especially effective to evaluate quality factors. Additionally, we delve into the deployment process and operational aspects of an LPWAN, thereby deriving insights into the key factors leading to network quality impairments within a prominent IoT access network solution. By a performance analysis of a novel gateway placement method based on graph metrics, we gain valuable insights for enhancing the most crucial factors that contribute to optimal quality within a future-proof LPWAN. We also explore the current channel access approach and develop an innovative time-scheduled mechanism compliant with existing regulations. Finally, we perform a comprehensive performance evaluation encompassing robustness against interference and cross-traffic, energy consumption, and overall energy efficiency. Through these analyses, we address a range of research questions, enhancing our understanding of the intricate dynamics at play in network management and quality assurance.

- How can we efficiently predict significant factors that lead to quality degradation in video streaming while employing lightweight and resource-friendly methodologies?
- How can we strategically devise an effective and future-proof gateway placement strategy for an IoT access network technology, one that enhances overall quality within a LPWAN context?
- How can we formulate an innovative channel access methodology for an IoT network technology that adheres to existing regulations while concurrently enhancing quality?
- Can we establish a general method to quantify the energy consumption and efficiency of data transmissions within an IoT access network?

1.1 Scientific Contribution

The subsequent paragraphs outline the scientific contributions of each domain covered within this thesis. A more detailed description of the identified research questions, along with the major contributions, is summarized at the beginning of each chapter. Note, each chapter concludes with a summary of lessons learned. A general overview with selected topics addressed in the course of this thesis is visualized in Figure 1.1. The y-axis categorizes the conducted research based on the methodologies employed in measurement, modeling, and simulation, indicated by the green, yellow, and red background colors, respectively. Along the x-axis, the conducted research is categorized by general themes. Noteworthy topics addressed within this thesis are illustrated as colored boxes. The respective references are depicted in bold font, accompanied by the corresponding chapter numbers in the bottom right of each box. The discussion on streaming, as explored in Chapter 2, is positioned towards the left, depicted by the red-colored boxes. The topics belonging to the IoT sector, investigated in Chapter 3 and Chapter 4 respectively, are situated to the right. These are visually delineated by the blue and the green boxes in Figure 1.1, respectively.

Streaming Monitoring and Processing Effort Assessment: To comprehend and quantify the network traffic involved in video streaming, an exhaustive dataset is measured and concisely summarized in Chapter 2. Our primary aim is to develop an exceptionally lightweight approach to monitor and process streaming traffic and determine the quality during video playback. We achieve this by comparing the total uplink traffic required to request video content with the corresponding downlink, which contains the actual video. Therefore, we undertake a thorough examination of the effort involved, considering scenarios utilizing only the uplink, downlink, or the complete packet trace within a monitoring or processing framework and model this assessment by a queuing model. To substantiate this modeling, we leverage our extensive dataset alongside a specially generated streaming dataset. Through this approach, we establish the applicability of our effort assessment beyond our dataset.

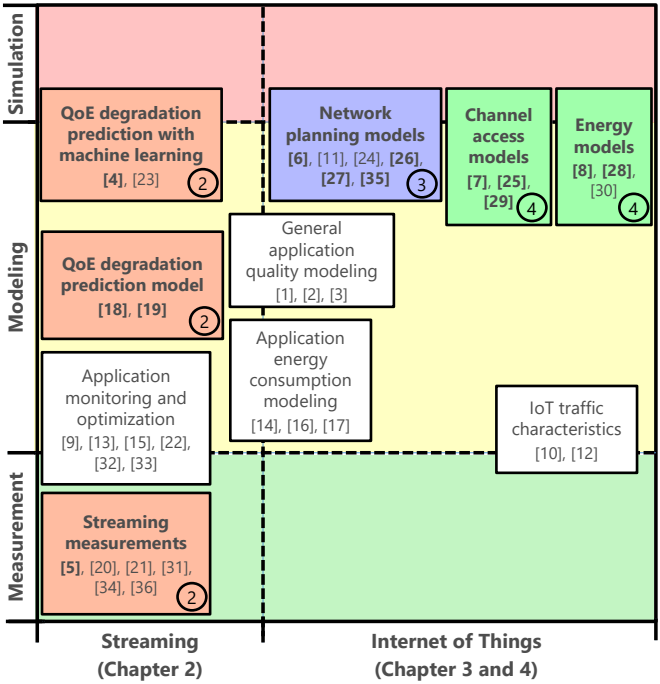

Figure 1.1: Categorization of research work conducted by the author. Topics and references highlighted in bold text are covered in this manuscript. The chapter that covers the respective publications is annotated in the circles at the bottom right of the respective topic box.

Streaming Quality Estimation: Subsequently, we aim at the development of two streamlined real-time methodologies to predict KQIs within video streaming. Our specific focus is on the anticipation of the initial video playback start, playback quality changes, general playback quality, and video re-buffering. Remarkably, both models exclusively leverage uplink data, utilizing a minimal amount of the data typically employed for such predictive analyses. Our mod-

eling approach, detailed in Section 2.4, supports real-time predictions and offers the flexibility of accommodating expert insights or functions independently with minimal resource overhead. Additionally, our random forest based methodology, explained in Section 2.5, identifies quality degradation factors with remarkable precision, employing merely up to ten features. Thus, these approaches can be effortlessly deployed on widely available and affordable hardware, thereby serving as invaluable tools for providers to enhance their networks and improve Quality of Experience (QoE) when streaming video content.

Gateway Placement: In Chapter 3, we design a gateway placement mechanism for an LPWAN, aiming to enhance the quality of the underlying network. Our approach employs graph metrics as the foundation, facilitating the strategic placement of gateways. This placement adheres to a target constraint on coverage, allowing for optimization with respect to the minimal count of required gateways and collision probability. Leveraging a comprehensive large-scale simulation study, we determine the essential factors influencing an optimal placement. These insights guide the development of a comprehensive scenario analysis, delivering invaluable understanding to LPWAN providers regarding the formulation of a forward-looking future-proof network design. Additionally, we investigate the impact of diverse sensor deployments on our gateway placement. This effort contributes to a broad performance evaluation of our proposed placement strategy. To overcome computational constraints, we adopt a divide and conquer approach, segmenting the complete network into discrete instances. This systematic strategy enables us to achieve valid placements for comprehensive deployments. Consequently, our gateway placement approach is equally applicable to both small-scale and large-scale LPWAN configurations, thereby extending its utility to network providers across various contexts.

Network Access Planning: Chapter 4 extends our focus beyond gateway placement. There, the goal is to improve data transmission in an LPWAN IoT network. To this end, we introduce a novel approach designed to enhance chan-

nel access within an LPWAN, mitigating interference from other concurrent messages from the same or from another network. Through theoretical exploration, we ascertain the maximum number of devices that can efficiently transmit within such a network using our innovative approach. This investigation spans a wide array of network configurations, and the findings are outlined in Section 4.2. Consequently, this analysis provides network providers with essential insights to optimize channel access, depending on the projected number of end devices. To validate the effectiveness of our approach, we subject it to a rigorous large-scale simulation study. This assessment includes performance evaluations in compliance with network-specific regulations studying the resilience against interference from cross-traffic as we elaborate in Section 4.3.

Energy Consumption and Energy Efficiency Assessment: Finally, different channel access approaches are investigated by means of the energy consumption and the energy efficiency in Chapter 4. To this end, we introduce a comprehensive methodology to quantify the energy requirements to transmit data in an LPWAN with various channel access approaches (Section 4.4). The methodology is validated, and the findings are tested in a comprehensive simulation study for different ways to access channels and transmit data (Section 4.5). The simulation results provide valuable information about the best suiting channel access methodology dependent on the load in the network, device capabilities, and the expected energy efficiency to transmit data. This is key for the deployment of future LPWANs, as little energy consumption and high energy efficiency are unique selling arguments for these networks and their devices.

1.2 Thesis Outline

The structure of this thesis, along with its key points, is concisely depicted in the schematic overview presented in Figure 1.2. Positioned centrally within the figure is the network itself, featuring both streaming and IoT end devices. These distinct device types actively participate in generating or receiving data traffic within the network, all while maintaining a connection to an application server. Despite the divergent nature of these devices and their specific data traffic characteristics – streaming end devices are anticipated to transmit a substantial volume of data, whereas network load from IoT devices stems from their sheer abundance – both categories critically rely on optimal network quality. The lateral arrows signify the points within this thesis where we extract data traffic or relevant information. The chapter boxes on either side of the central depiction mirror the color scheme introduced in Figure 1.1, corresponding to the respective chapters. Furthermore, the individual boxes within each chapter mirror the research methodology adopted: green for measurement, yellow for modeling, and red for simulation. The internal arrows within each box symbolize the sequential progression of topics addressed within each individual chapter.

The following Chapter 2 deals with the measurement, processing, and analysis of video streams towards the identification and prediction of crucial quality degradation factors. This exploration covers the application domain that significantly contributes to the overall traffic in today's Internet. Our study underscores the viability of precise quality degradation factor prediction to improve the perceived quality when streaming video, based solely on uplink traffic data.

As the other big contributor to today's network load, we study a Low Power Wide Area IoT network in Chapter 3. Here, our focus centers on gateway placement decisions through an innovative graph-based approach. We comprehensively analyze various facets, encompassing the required number of gateways, projected collision probabilities within the network, algorithm runtime, and the resultant placement quality. Our study reveals that optimal placements are achievable when the maximum distance between sensors and gateways remains

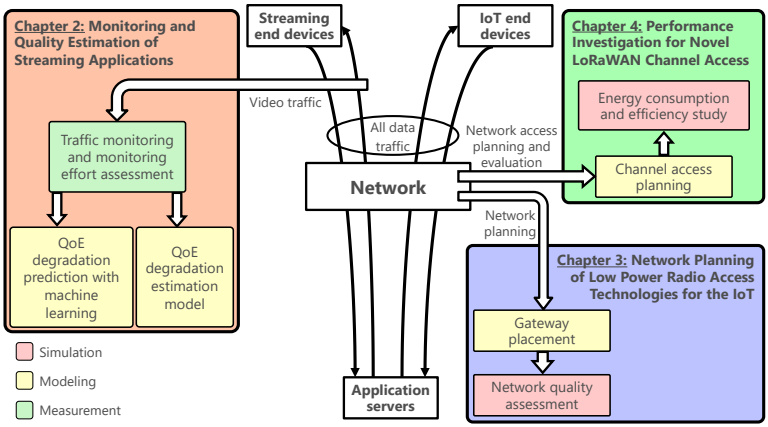

Figure 1.2: Organization and schematic overview of this thesis.

limited. This configuration allows individual sensors to transmit data to proximate gateways, thereby minimizing interference from other devices.

Finally, in Chapter 4, we analyze the access of available channels to transmit data by devices in an LPWAN deployment. We investigate a fully time-scheduled approach, wherein we ascertain the maximal number of sensors that can transmit their messages while complying with network regulations. Moreover, we conduct a comparative study of our time-scheduled channel access against listen before talk and the presently employed ALOHA-like random channel access methodology. This comparison is thoroughly conducted by an extensive large-scale simulation study, where we evaluate performance outcomes under varied cross-traffic scenarios. Furthermore, our investigation extends to an energy consumption and an energy efficiency study for different LPWAN channel access approaches. By a novel energy efficiency metric, we draw insightful conclusions concerning the most energy-efficient channel access approach across diverse load situations within the studied LPWAN deployment context.

2 Monitoring and Quality Estimation of Streaming Applications

A ”Tsunami of Video” is one of the top five trends that will accelerate over the next decade [47]. But video streaming is already responsible for about 65 % of the global Internet traffic in the first half of 2022 [40], with Netflix and YouTube being the most popular video apps for general traffic and YouTube for mobile traffic [40]. For that reason, an important quality indicator for current and future wired and mobile networks is their performance in providing videos with best possible quality to their customers. And although mobile Internet connectivity is steadily evolving [48], the available bandwidth can vary largely because of, among others, moving end devices, different access network generations, urbanization, geography, user's travel speed, or dead spots in mobile network coverage [48]. From the perspective of an Internet Service Provider (ISP), however, it is important to satisfy their customers with satisfactory network speed but also to operate economically. This leads to an increasing relevance of an intelligent and predictive service and network monitoring and management concept to provide good network quality, meet user demands, and save resources.

However, make fast and intelligent management decisions in growing networks with an increasing traffic volume is a resource intense and complicated task. One reason for this vast amount of new traffic is the steadily growing number of streamed videos in continuously increasing video resolutions. Since the goal for an ISP is to cope with this development, guarantee high-quality net-

works, and ensure a good service for all end-user applications, more efficient network monitoring is required. Otherwise, the network quality for end-users is impaired, leading to customer churn that is one major cause of revenue loss [49]. In the past, this so-called network layer Quality of Service (QoS) was estimated by measuring in-network parameters like bandwidth, packet loss, or jitter, and the expected application quality was calculated. However, details about the analyzed application, the behavior of the application when these in-network parameter change, and the perceived quality by the end-user were often neglected. Furthermore, application requirements and main influencing factors leading to a worse quality are often not clear and beyond simple uncorrelated QoS.

To overcome this limitation, the concept of Quality of Experience (QoE) has been introduced as "the degree of delight or annoyance of the user of an application or service" [50]. Thus, network parameters are no longer the only measure for good application quality and in return, good networks. To quantify the QoE, degradation factors for the perceived quality by the end-user have been researched and outworked for different applications. These QoE degradation factors impact video streaming negatively and are determined as initial video playback delay, video playback quality, quality changes, and video re-buffering events [51]. In the past, they were directly measured in the network (e.g. by [52]). However, most network traffic is encrypted nowadays [53] and in-depth monitoring to determine crucial streaming related quality information directly from the packet payload is no longer possible for an ISP. It is therefore necessary to rely on alternative techniques to predict QoE degradation factors, using for example packet header information or different traffic patterns only.

Though, the analysis of video streams at packet level is complex and does not scale. Alternatives are required, given the enormous number of parallel global streaming sessions and the associated massive data exchange in petabyte scale [54]. One idea is to enable decentralized monitoring in the last mile towards the customer. There are usually only limited resources available and lightweight solutions without unmanageable processing overhead are required to use available resources in the most efficient way or save resources for other tasks. This

requires a scalable in-network monitoring of selected data at decentralized entities, serving as input for a proper QoE degradation factor prediction model.

For that reason, this chapter of the monograph studies whether the resource demand by means of monitored data, data aggregation, and complexity of a monitoring and management approach can be limited, while still successfully predicting important QoE degradation factors for video streaming. These factors are essential to derive the QoE, using QoE models. Therefore, a comprehensive dataset of more than 1,500 h of total playtime is measured from the YouTube mobile application as largest mobile traffic generator on the Internet [47]. All measured data are aggregated to their uplink requests and used by a lightweight uplink based model developed in this chapter to estimate drops in the video playback buffer. These drops are the first indicator leading to quality degradation events influencing the QoE negatively. Furthermore, to predict all relevant QoE degradation factors in a complete video session, an uplink data based Machine Learning (ML) model is developed, tested towards different parameter combinations, and evaluated in terms of prediction quality. With these approaches, the following research questions are identified that are covered in this chapter.

RQ2.1: What is the possible monitoring and processing effort benefit if only uplink data is considered to predict relevant QoE degradation factors including initial delay, quality changes, playback quality, and stalling?

RQ2.2: Which relevant QoE degradation factors are predictable with a simple uplink based model without relying on a ML solution, and what is the performance compared to state-of-the-art literature?

RQ2.3: Is uplink data sufficient to predict the initial video playback delay, quality changes, playback quality, and video re-buffering events as main QoE degradation factors with a lightweight ML model, and what are benefits, drawbacks, and performance differences in comparison to other solutions with and without ML?

In the following chapter, Section 2.1 provides fundamental background to understand video streaming. Furthermore, the concepts of traffic patterns and streaming phases emerging during video streaming are introduced and the main QoE degradation factors are discussed. At the end of the section, related literature is summarized and compared to the approaches discussed in this chapter. Afterwards, Section 2.3 presents a summary of the used dataset and answers research question RQ2.1 by quantifying the monitoring and processing effort of a full packet trace analysis scheme versus the uplink only based monitoring and processing, as discussed in this chapter. Then, Section 2.4 describes and evaluates an uplink based model to predict QoE degradation factors without relying on ML approaches to answer research question RQ2.2. Afterwards, research question RQ2.3 is answered in Section 2.5 by introducing and evaluating a random forest based ML model to predict QoE degradation factors in a lightweight manner on uplink data. Finally, Section 2.6 concludes this chapter and provides a discussion of lessons learned regarding the scientific contribution. The main contributions provided in the following chapter can be summarized as follows.

C2.1: A comparison of the required effort to monitor and process a video stream in the network based on only uplink data or a full packet trace. Therefore, the general amount of data is investigated and a Markov model is presented to determine both the expected system response time and the possible number of streaming flows that can be processed in parallel using uplink only or uplink and downlink traffic.

C2.2: A lightweight model that is able to predict video resolution changes and stalling events in real time with only uplink traffic and without the requirement of complex ML techniques. The system can learn important parameters during runtime and works independently of the resolution of the streamed content, the streaming platform, or any expert knowledge.

C2.3: A simple but comprehensive ML model to predict the main QoE degradation factors, initial delay, quality changes, playback quality, and stalling with only uplink data as input. The benefit of this approach is the little

requirement of monitored data and as a result, less monitoring and processing overhead. For that reason, more video sessions can be monitored and processed in parallel, or the model can be distributed across multiple edge nodes closer to the end-user.

The contributions have already been published in the past and are summarized for this monograph based on the following scientific publications.

- Loh, F., Wamser, F., Moldovan, C., Zeidler, B., Tsilimantos, D., Valentin, S., Hoßfeld, T.: "Is the Uplink Enough? Estimating Video Stalls from Encrypted Network Traffic", in IEEE/IFIP Network Operations and Management Symposium (NOMS), 2020. [11]

- Loh, F., Poignée, F., Wamser, F., Leidinger, F., Hoßfeld, T.: "Uplink vs. Downlink: Machine Learning-Based Quality Prediction for HTTP Adaptive Video Streaming", in Sensors, 2021. [4]

- Loh, F., Wamser, F., Poignée, Geißler, S., Hoßfeld, T.: "YouTube Dataset on Mobile Streaming for Internet Traffic Modeling and Streaming Analysis", in Nature, Scientific Data, 2022. [5]

- Loh, F., Pimpinella, A., Geißler, S., Hoßfeld, T.: "Uplink-based Live Session Model for Stalling Prediction in Video Streaming", in IEEE/IFIP Network Operations and Management Symposium (NOMS), 2023 [12].

2.1 Background

This section contains important definitions and essential background information on video streaming. The process of streaming a video is described step by step from content request to playback. At the end of this section, different factors influencing the overall streaming quality are introduced.

2.1.1 Notion of Video Streaming

Video streaming is the process of simultaneously playing back and efficiently downloading audio and video content from the streaming server. In the following, this process is explained in detail and important terminology is given.

Adaptive Streaming

In the past, when a user requested a video on a streaming platform, it was downloaded completely in a single quality in the best effort manner. However, if the average download rate between the client and a content server is insufficient or deteriorates, this commitment to a single quality led to a long initial prebuffering phase or playback interruptions. In contrast, if the downlink rate increased during the streaming session, the user received a worse average video quality than it would have been possible. To compensate such network throughput fluctuations and to enable the possibility of changing the played out video quality during a video stream, an adaptation was introduced. The goal of this adaptation is the selection of the playback quality, and more precisely the video bitrate, towards the currently available network throughput. Thus, the goal of this HTTP Adaptive Streaming (HAS) approach is to play out the best possible video quality with minimized initial delay and without playback interruptions, dependent on the currently available bandwidth and end-user's player settings, even if the underlying network conditions are changing. To enable this dynamic quality adaptation, the complete video is split in independent segments on application layer that can be requested individually and in different quality representations. We refer to Seufert et al. [51] for further details.

Requesting and Download: As soon as a player requests a new video, an initial Transport Layer Security (TLS) encrypted uplink request is sent to open the connection to the streaming content server. For YouTube streaming, a connection to Google servers, symbolized as Content Delivery Network (CDN) in Figure 2.1, is required. There, audio content is usually available in one constant

bitrate, shown by *AQ1* in Figure 2.1 and the blue color. In contrast, video content can be available in different quality representation to enable quality changes. In this example, three qualities are available: *VQ1* is the worst quality with an average bitrate of 500 kbit/s in red, *VQ2* has medium quality with 1,000 kbit/s in yellow and *VQ3* has 2,000 kbit/s and is indicated by the green color. If the initial request arrived successfully at the server, it replies with a Hypertext Transfer Protocol (HTTP) response. Based on the content requesting and buffering strategy, and the player settings, it is determined which quality is requested next. Then, the browser uses the HTTP and opens or reuses existing Transmission Control Protocol (TCP) or QUIC connections to request content. The requested data is also TLS encrypted at the server and downloaded to the end-user's device, as indicated in Figure 2.1. The network layer portion of the requested data in the downlink is a video or audio *chunk* or chunk download, as we highlight in Figure 2.1. Back to the client, there is no parallel download of chunks of the same type and quality [11, 55]. Although video and audio chunks can be downloaded in parallel [12, 55], or YouTube can download chunks of the same quality in parallel in case of varying network conditions [56], to trigger a quality change.

Buffer: Each chunk arriving at the end-user's device is added to the associated, either audio or video buffer. Therefore, it is split into its segments, the application layer portion of the requested video data. Regarding Figure 2.1, video chunk 1 contains two segments, *V1* and *V2* in medium quality. Both of them are added to the video buffer, but played out one after the other. Audio segment *A1* is added to the audio buffer. As soon as the video and the audio buffer contain enough data, normally at least one complete segment each, the playback at the user's end-device can start. To analyze the buffer and the buffering behavior, the video buffer, as main buffer [11] is taken into consideration in the following.

If the buffer contains sufficient data, it can compensate small bandwidth fluctuations or a small variance in the video bitrate of the same quality by storing several seconds of playback in advance. However, the dimensioning of a buffer is no simple task. It is vital to avoid the buffer running empty after an initial filling,

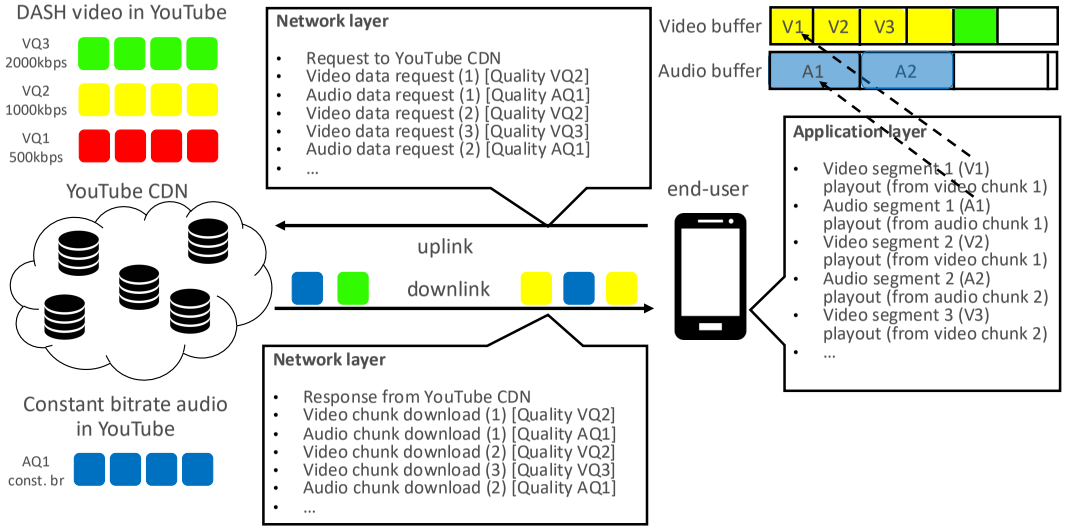

Figure 2.1: YouTube streaming: network and application view.

leading to playback interruptions. But it is also not advisable to pre-buffer the complete video to prevent unnecessary video data download if video playback is aborted by users before the video is played out completely. Then, all remaining video data in the buffer is discarded, and network resources are wasted. This situation is further analyzed in [57]. To this end, the appropriate buffer size must be chosen according to three main aspects: First, the tolerable amount of data that can be discarded when a user aborts defines the boundaries for the buffer so that phases with reduced bandwidth or complete outages in the network connection can be outlasted. Second, the buffer must be at least large enough to compensate for a variation in the end-to-end network download rate if the playback quality is not changed. Third, it must be at least large enough to compensate variations in the video bitrate when data is taken from the buffer. Therefore, a general assessment of a good buffer size is required, and provided in [58]. Besides the general buffer dimensioning, also the filling process of the buffer is essential to provide an end-user with a good quality streaming without interruptions. In general, Sani et al. identifies three major ways to estimate and request the correct and required data to fill the buffer [59], a throughput based, a buffer based, and a power based Adaptive Bitrate (ABR) selection. More details about the ABR selection are available in [59]. For streaming with the YouTube mobile app, we identified a form of buffer based ABR selection [13] with additional information about the current bandwidth estimation. However, this bandwidth estimation is not included in the available application information from the YouTube app anymore, and is thus not considered in the following. In [13], we describe an initial buffer filling after playback start in the best effort manner until a pre-defined maximal buffer limit is reached. Then, the player constantly tries to keep the buffer close to this limit. If the buffer level decreases because of any changes in the network or the video bitrate, and the buffer drops below a specific threshold, a lower quality is requested to adapt the video bitrate against the changing available bandwidth. In this way, the player tries to prevent video playback interruption during streaming by running the buffer completely empty.

Traffic Patterns

In the past, when network traffic was mainly not encrypted, important information about the current streaming session like the currently played out video quality or buffer filling level could be easily extracted from the packet payload. However, since nowadays most network traffic is encrypted in the Internet [53], alternative ways are required to determine if enough data is downloaded and buffered to avoid downwards quality changes or video re-buffering events. One approach is to analyze the complete network traffic and look for patterns. A pattern in video streaming traffic is a sequence of network packets corresponding to the requested network chunks. Thus, different traffic patterns can be observed when mapping the application level video behavior to network traffic. This is possible since different sequences of network chunks are requested in different playback situations, dependent on the buffer state, the available throughput, or the amount of required data by the application. As a consequence, the application state can be estimated from such traffic patterns. The understanding of these patterns forms the basis to detect the current buffer level, the playback quality, and thus, the overall player state. One possibility to determine a traffic pattern is the analysis of the time between content requests. If this inter-request time is larger than the number of video seconds requested with each chunk, the buffer depletes. If the inter-request time is shorter, the buffer is filled up. Consequently, it is possible to draw conclusions from the uplink traffic request pattern to buffer changes. With regard to the overall application layer streaming behavior, different streaming phases can be defined. Based on the traffic patterns in these phases, the streaming behavior estimation process is done in this chapter.

Streaming Phase Definition

Streaming phases are states of the internal streaming process. At the user's end, only the effects and consequences of these phases on the streaming state are noticeable. That manifests to the user as smooth streaming, quality changes, or interruptions in video playback. An application view on possible buffer fill-

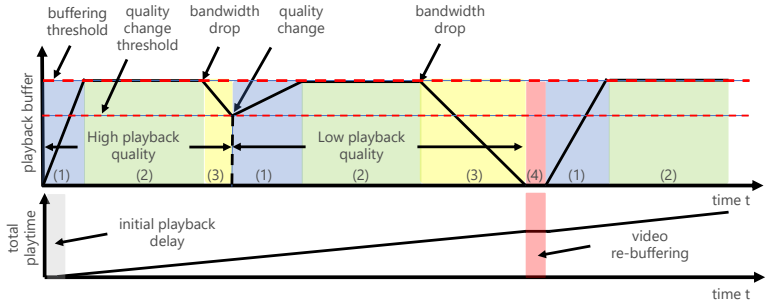

Figure 2.2: Buffering and playback procedure during video streaming.

ing states of a video player to visualize streaming phases is given in the top part of Figure 2.2. The y-axis shows the playback buffer level and the x-axis the time t. The solid black line is the example playback buffer available for the streaming player. In the past, first, two different phases were defined for video streaming in [60]: An initial buffering phase and a periodic buffer refill phase. In the initial buffering phase (1), visualized by the blue color in the figure, video data is requested and delivered in a best effort manner until the target buffering threshold, shown by the dashed red line, is reached. Then, the player switches to the periodic buffer refill phase (2), marked in green. This phase corresponds to the normal streaming behavior, where a similar amount of data is downloaded as extracted from the buffer for playback. The goal here is to keep the buffer slightly above a target threshold. However, this definition only covers a streaming without any buffer depletion through, for example, bandwidth degradation. The authors of [61] overcome this limitation with the definition of three phases. The initial buffering is called buffer filling and the periodic buffer refill is the steady-state phase. Additionally, a buffer depletion phase (3) is defined, where the buffer level decreases, for example after a bandwidth drop, shown by the yellow color in Figure 2.2. Lastly, we introduce a stalling phase (4) in [4]. The video playback is interrupted in this phase, shown by the red color. Further-

more, after any depletion and stalling phase, additional buffer filling phases can occur, as shown in the figure. To analyze the quality, and in particular the QoE of a video stream, the goal is to detect patterns in the uplink traffic during different streaming phases. These patterns are then used to draw conclusions on the video player's playback behavior or to estimate impairments during playback.

2.1.2 Influencing Factors on Quality of Service and Quality of Experience

The QoS and the QoE in video streaming can not be directly measured or derived from a single parameter. For that reason, we introduce influencing factors in the following. These influencing factors can be measured or predicted and help to determine the QoS or the QoE during streaming, respectively. However, for the QoE, we name them degradation factors as they impact the QoE negatively. A classification of both QoS influencing, and QoE degradation factors is given in Figure 2.3. Note, the QoE perceived by an end-user is, in addition to the degradation factors presented in this thesis, also influenced by the subjective opinion of the user, environmental, or other context factors. This influence is represented by the dashed boxes in the figure, as an in-depth analysis is not the focus of this work. We refer to literature for further details [50].

Quality of Service Influencing Factors

When a YouTube video is streamed by the end-user's YouTube app, data is requested from a YouTube CDN and video data is downloaded to the user's end device, as visualized in the bottom part of Figure 2.3. These data, exchanged between the end device and a server, can be monitored directly in the network and used to derive QoS influencing factors. These factors include, among others, the end-to-end delay, the jitter, the throughput, the number of transmitted packets, the inter-packet time, or the inter-request time. Such influencing factors can be used to calculate the QoS. However, no information about the user and the user's perceived quality is available relying on QoS only.

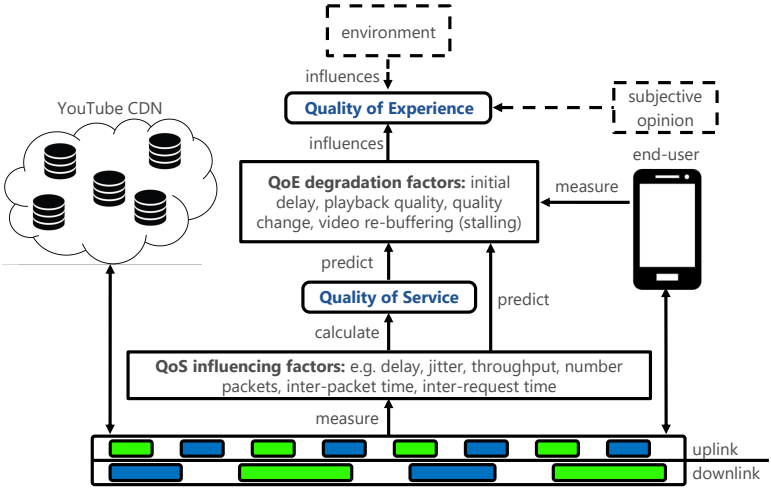

Figure 2.3: Classification of QoE degradation factors in QoS and QoE context.

Quality of Experience Degradation Factors

For that reason, the goal in this thesis is to determine factors that impact the QoE negatively. According to [51], these QoE degradation factors include the initial playback delay, the playback quality, playback quality changes, and stalling. These QoE degradation factors can be directly determined from the application of the end-user's device. However, from a network operator's or network provider's point of view, this information is not directly available. For them, it is only possible to predict these QoE degradation factors out of network data or out of QoS influencing factors. Nevertheless, it is essential to determine these QoE degradation factors, as we use them as input for video QoE models to derive the QoE perceived by the end-user when streaming a video. A comparison of different video QoE models using a crowdsourced data set is presented by Seufert et al. [62]. For that reason, a sole investigation of the buffer filling level,

shown in the top part of Figure 2.2, is not sufficient from the application's point of view. It is also essential to investigate the general playback behavior leading to QoE degradation factors, as shown in the bottom part of the figure. Therefore, we describe each individual QoE degradation factor in the following.

Initial Delay: Video playback does not start immediately after the initial data download but when a specific amount of seconds is available in the buffer, as shown in the bottom part of Figure 2.2. This initial time is called initial delay and describes the waiting time between the first video request and the video playback start. According to [14], the initial delay can be divided into two parts, the time to load the streaming application's page and the time delay between the initial video request and the playback start. The initial page load time is not only influenced by the currently available network conditions, the server behavior, and the requested playback quality, but also by, among others, the end device type, quality, and memory usage. Thus, the delay from initial video request to playback start is only investigated as initial delay in this chapter.

Streaming Quality and Quality Changes: Next, the currently played out quality and the number and frequency of quality changes during streaming influences the QoE of an end-user. Since, in HAS, the goal is to adapt the playback bitrate to the currently available bandwidth, frequent quality changes are possible, especially if the bandwidth variability is high. Thus, the trade-off is to minimize the number of quality changes and maximize the average playback quality to achieve the maximal possible average video bitrate. For that reason, mechanisms like quality change thresholds, as shown by the lower dashed red line in Figure 2.2 are implemented in most streaming players. Additional adaptation algorithms define rules to trigger quality changes. We refer to Seufert et al. [51] and Sani et al. [59] for further details.

Stalling: The interruption of the video playback is the last degradation factor on the QoE, which is discussed in this chapter. These so-called video rebuffering or stalling events have the highest negative influence on the overall QoE [51]. Frequent stalling events have a more severe impact on the QoE than fewer but longer stalling [63]. Thus, detailed QoE quantification or estimation in video streaming is mainly based on the possibility to detect or estimate such stalling events. However, according to Figure 2.2, stalling can be directly derived from the streaming phase definition. Thus, stalling can be detected if the stalling phase is detectable when streaming a video.

2.2 Related Work

Service quality measurement, flow identification, and application behavior estimation is essential in modern network management to draw correct conclusions on network performance and applications working correctly. However, raw performance measures like bandwidth or end-to-end latency leading to high QoS are not the only focus anymore. The ability to analyze network traffic and detect issues that might impair user experience is important for many applications and can directly lead to user satisfaction with the analyzed service. For video streaming, this is done in an early comprehensive survey by Seufert et al. [51], studying degradation factors for QoE. Besides knowing these factors, another important goal is to measure all relevant information to determine streaming quality. One of the first works tackling streaming measurements, calculating the application QoS, and estimating the QoE based on a Mean Opinion Score (MOS) scale was published in 2011 by Mok et al. [52]. In the same year, Hoßfeld et al. postulates a square relationship between the MOS and the standard deviation of opinion scores [64]. The authors show that the MOS is not enough in extreme situations. However, with the widespread adoption of traffic encryption on the Internet, also used for streaming video, quality estimation approaches became more complex. Simple Deep Packet Inspection (DPI) based techniques, where streaming relevant information like the buffer filling status is directly read from

the packet payload became largely obsolete. Nowadays, several challenges must be resolved for an in-depth quality estimation in video streaming.

In-network measurements for streaming platforms are conducted for, among others, YouTube [5, 13, 65], Netflix and Hulu [66], and Twitch.tv [12, 67]. They are a basis for video streaming analysis but contain not only streaming traffic. For that reason, relevant flows must be detected and separated from cross-traffic. In literature, this is already done using for example classification models [61, 68–70] or ML [67, 71–74]. Other works also separate audio and video flows by examining the request size or the size of the data volume downloaded by the client, respectively [55, 67]. This procedure can help to understand the streaming behavior better, adding complexity and potential errors if data are misclassified.

When the correct video flow is determined, the approach to estimate QoE degradation factors is defined. In modern streaming, two approach types are well adapted: (1) session reconstruction and (2) the use of ML models. An overview of selected, especially relevant, related work with regard to QoE estimation in video streaming is summarized in Table 2.1. The main characteristics which differentiate approaches, as reflected in the table, are six: real time applicability, the target platform, the approach itself, the prediction goal, the data focus, and the target granularity from a time window (window) based approach to a complete session investigation. If QoE is listed as prediction goal, all major QoE degradation factors, namely initial delay, playback resolution, resolution changes, and stalling are investigated. Furthermore, session models can either reconstruct a complete video streaming session or only parts of the streaming session, as we do in this work, by analyzing only phases in more detail, where we assume a quality degradation.

Session modeling approaches show that it is possible to predict all relevant QoE degradation factors. Whereas in [11], only stalling is predicted with a session model on a few different videos. Mangla et al. show its utility for modeling other QoE degradation factors and compare the performance to ML approaches [80]. In particular, the task of stalling estimation is tackled by Schatz et al. [81] and Dimopoulos et al. [82] by packet trace inspection of HTTP logs.

Table 2.1: Overview of select related work.

Reference		Real time	Target platform	Approach	Prediction goal	Data focus	Granularity
Mazhar'18	[75]	✔	YouTube desktop	tree-based	QoE	packet	window
Wassermann'20	[44]	✔	YouTube mobile, desktop	random forest	QoE	packet	window
Orsolic'20	[76]	✔	YouTube desktop	tree-based	QoE/KPI	packet	session
Bronzino'19	[77]	✘	YouTube, Netflix, Twitch desktop	random forest	initial delay, resolution	packet	window
Dimopoulos'16	[78]	✘	YouTube desktop	random forest	stalling, quality, quality change	packet	session
Shen'20	[45]	✔	YouTube, Bilibili desktop	CNN	initial delay, resolution, stalling	packet	window
Lopez'18	[79]	✔	–	CNN, RNN	QoE	packet	window
Mangla'18	[80]	✘	–	session model	QoE	packet, request	session
Gutterman'20	[55]	✔	YouTube mobile, desktop	random forest	buffer warning, video state, video quality	request	window
Madanapalli'21	[67]	✔	YouTube, Twitch	random forest	QoE	request	window
This chapter	–	✔	YouTube mobile	random forest, session model	QoE	request	window, session

In recent years, several ML approaches have been published to estimate QoE degradation factors in video streaming, e.g. [83, 84]. Overall, real time capable approaches with focus on analyzing all network packets in a YouTube stream are available by, among others, Mazhar [75], Wassermann [44], and Orsolic [76], summarized in Table 2.1. For all works, an in-depth analysis of network data is done and more than 100 features are extracted each. While Mazhar uses the YouTube desktop version in the study, the mobile version is also considered by Wassermann. Wassermann shows that a stalling prediction of up to 1 s granularity is possible with a time-window approach, with accuracies of more than 95 % and recall values of 65 % for their bagging approach, and 55 % for the random forest approach. The precision is 87 % and 88 %, respectively. Without real time capability, Dimopoulos et al. study QoE degradation factors streaming YouTube with up to 70 features [78] and Bronzino et al. train an initial delay and resolution prediction model for Amazon, Twitch.tv, and YouTube streaming [77]. The authors calculate information like throughput, packet count, or byte count.

Considering live-streaming, Jimenez investigates the YouTube live-streaming behavior on full packet trace information [85]. Recently, the usage of a random forest model trained for on-demand streaming has been tested on Twitch.tv live-streaming data to determine whether new QoE degradation factor estimation models are required or known models from on-demand streaming can be reused [15]. The presented model adapts well for stalling and playback resolution but achieves bad results for quality changes. In addition, Madanapalli et al. examined on-demand streaming with YouTube and live-streaming with Twitch.tv based on a large-scale dataset of about 23,000 video streams measured in synthetic network conditions [67]. The authors developed a Long Short Term Memory (LSTM) model to distinguish live from on-demand streams and a random forest model to predict resolution changes. Furthermore, they use a statistical model to estimate re-buffering events. The authors use different uplink request based features as input. Similarly, Gutterman et al. selected a video chunk-based approach for YouTube on demand streaming with four states: increasing buffer, decreasing buffer, stalling, and steady buffer in [55]. The approach is similar to

the ML approach of this chapter, while the data resolution on application layer is higher in the approach from Gutterman. In Gutterman's dataset, the phases with dropping buffer are even more underrepresented. Furthermore, Gutterman uses sliding windows up to a size of 20 requests, including historic information of up to 200 s. This leads to challenges for short videos that are very typical streaming YouTube videos. Overall, similar results are achieved with the ML model. However, in the buffer decay phase (depletion in this monograph), the results in this thesis outperform Gutterman by more than 6 % in the precision score and 50 % in the recall score. This is in particular important, since the depletion phase is the first indicator for network quality impairment and thus, quality changes or stalling and the recall score indicates the probability to detect these impairments. Most related ML approaches use at least one tree-based model. This popularity is a result of multiple factors, including ease of development and lower computational resource demand. Moreover, the tree-based algorithms, e.g., random forest, perform on a similar level or better than others [44, 55, 77], when multiple algorithms are compared within one work. Gutterman [55] investigates the performance of a Neural Network (NN), with similar results as a tree-based approach. Shen [45] uses a Convolutional Neural Network (CNN) to infer initial delay, video resolution, and stalling. Their real-time approach uses video data from YouTube and Bilibili and windows of 10 s length for prediction. Among the tested ML algorithms, the CNN achieves the best results. The potential of deep learning to predict QoE degradation factors is emphasized further in the work of Lopez [79] by combining CNN and Recurrent Neural Network (RNN).

In contrast to related work, this chapter of this thesis covers both a simple request-based session modeling approach and a random forest based ML approach. The benefit of the session model is its simplicity without the requirement of any complex ML pipelines. For stalling and quality change estimation, it achieves similar accuracies as related work. In addition, with recall values for stalling detection of up to 95 %, it clearly outperforms related, often much more complex, mechanisms from literature. The model uses only uplink data as input, reducing complexity and resource demand for both monitoring and the predic-

tion itself. This reduced complexity without ML model training increases the first reaction time to a change of the video player's behavior and allows a timely and simple adoption to modified circumstances. The drawback of the simplicity is no possibility to quantify the initial delay or the playback quality. This limitation is overcome in an additional uplink only based ML approach presented afterwards in this chapter. The random forest model achieves macro average recall values of more than 80 % for stalling estimation and a comparable precision. In addition, other QoE degradation factors like initial delay and playback resolution can be estimated with high accuracy. Thus, in contrast to full packet trace analysis from related work, where thousands of encrypted packets must be considered in the network, on average only one single packet every 5 s - 10 s are used as input for the model, depending on the video chunk size and independently of the played resolution. Thus, the amount of data to be monitored and processed can be reduced massively while keeping the prediction performance at a comparable level or even outperform related literature.

2.3 Dataset Summary and Monitoring Effort Assessment

This statement about reduced complexity and monitoring effort using only uplink data to predict QoE degradation factors is discussed in detail in the following. Therefore, a brief summary about the conducted dataset used in this chapter of this thesis is introduced first. A description of the streaming measurement procedure is presented in Appendix A.1 and the data post-processing of the measured data is available in Appendix A.2. After details about the dataset, this section targets preliminary considerations on the monitoring and processing effort of the presented uplink based QoE degradation factor prediction in contrast to using a full packet trace. Therefore, we model the monitoring and processing effort for our dataset and a general dataset. To emulate a general dataset, we artificially generate video content with models from literature.

2.3.1 YouTube Streaming Dataset Overview

The final measured dataset after post-processing covers 13,759 valid video runs and more than 65 days of total playback time. This sums up to more than 1.1 M uplink requests. Parts of the dataset with different purposes are already published for the research community [4, 5, 13, 14]. In total, 7,407 videos, or 53.83 %, have at least one quality change summing up to 21,917 quality changes and 28,773 requests labeled as quality change. This difference is a result of parallel audio and video flows or of different qualities requested in parallel in advance of a quality change. In these cases, multiple requests are labeled as the same quality change. Note, in the following paragraphs, the terms resolution change and quality change are used interchangeably. In total, nearly 700,000 requests are in the filling phase, about 350,000 in the steady phase, and more than 100,000 in the depletion phase. Furthermore, 2,936 videos or 21.33 % have at least one stalling event, accounting for a total of 5,934 stalling instances and 20,070 requests labeled as stalling. This sums up to more than 32 h of total stalling time. In about 50 % of the stalling videos, one stalling event is detected, and a second event is detected in another 25 % of the runs. Only in less than 3 % of all stalling videos are more than 5 stalling events measured. Furthermore, in 40 % of all videos where stalling is detected, two or more different qualities are played out when stalling occurred. This means that, although the quality is adapted to the currently available bandwidth, a stalling event occurred. The stalling duration varies from less than one second to 117 s for all videos in the dataset. Short stalling of less than 5 s occurs in 20 % of all stalling cases. The median of the stalling duration is 10.5 s and the mean is 19.87 s. Since stalling prediction with historic information at the beginning of a video is not possible, and to avoid conflicts with the initial playback delay, stalling within the first 5 s of video playback time is not considered. In total, the dataset contains videos streamed with TCP and QUIC. Among all runs, 22.9 % are streamed with TCP, and the remainder with QUIC. Since no video specific information like video ID or resolution is available from encrypted network traffic, this is not studied in this work. Furthermore, the available downlink bandwidth to stream the videos is

not analyzed, since it is not directly detectable from the traffic trace. Note that there is no explicit analysis of the amount of totally downloaded and uploaded bytes for specific QoE degradation factors, since this is not the main focus of the approaches in this chapter, and it is highly dependent on the video resolution. The focus in this chapter is on measurable network data having an impact on the estimation result for the main QoE degradation factors initial delay, playback quality, quality changes, and stalling. Further details about these main QoE degradation factors and the streaming phase are summarized in Appendix A.3.

2.3.2 Artificially Generated Streaming Dataset

As a next step, the goal is to identify whether our dataset can be used exemplary for general video streaming, reflecting the required monitoring and processing effort to determine major QoE degradation factors. Therefore, we aim to generate an additional dataset of random videos and 'stream' them in a streaming simulation. In this way, it is examined how much data is transmitted in the network during video streaming of arbitrary videos and how much data needs to be monitored and processed to quantify the main QoE degradation factors.

Data Generation: The traffic models for H264/MPEG-like encoded variable bitrate video in literature can be broadly categorized into (1) data-rate models and (2) frame-size models [86]. While frame size models focus on generating single frames for different video content in order to gain the ability to create arbitrary, yet realistic videos [87, 88], we use data rate models, as they generate only the data arrival rate [86]. For that reason, they are sufficient to quantify the occurring load in the network when streaming a video, and thus, also the required monitoring and processing effort.

Therefore, we generate 9,518 random artificial videos with a duration of 5 min - 15 min in different qualities according to models from literature [86–88]. This results in a variable bitrate per second video content for each quality. Quality, bitrate, and general video size information of the videos is summarized

Table 2.2: General information about the artificially generated videos.

Quality	Bitrate [kbps]		Size [MB]
	min/∅/max	Q10/Q90	min/∅/max
144p	32/114/637	70.2/167.6	2/8/20
244p	175/256/749	212.3/310.1	7/19/33
360p	489/570/1,092	526.5/624.2	19/41/69
480p	974/1,057/1,546	1,013.5/1,111.1	38/79/124
720p	2,091/2,172/2,661	2,128.9/2,225.2	79/160/249
1080p	4,575/4,657/5,183	4,613.3/4,710.2	173/338/529

in Table 2.2. With these generated videos, we aim at analyzing a very general streaming session. The created videos are then streamed and mapped to requests at network layer in order to determine the amount of data while streaming. An adaption of three quality switches is applied with a duration of one third of the total video length. Further details about the video generation and the streaming simulation together with the scripts are available in [4]. Finally, this can be used to calculate how many bytes must be streamed, processed, or monitored if (1) the entire video has to be considered or (2) only uplink or downlink traffic is required for streaming monitoring or streamed data processing. Note that this is meant to be a pre-study with focus on arbitrary video content. The aim is to show how much data and effort is required when only partial data, like only uplink requests, is used for QoE degradation factor prediction in comparison to a full packet trace with uplink and downlink data. This is especially important in terms of predicting video streaming parameters in communication networks on a large, distributed scale. An efficient, low effort, approach is usually preferred here, which can be adopted on the last mile close to customers, where little compute power is available.

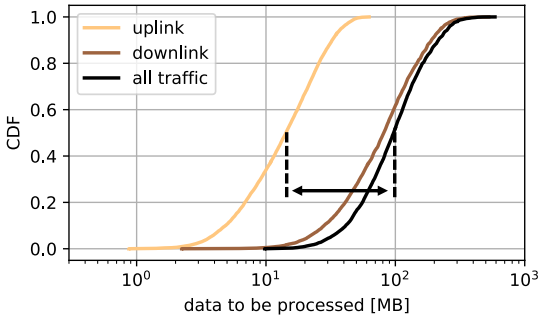

Figure 2.4: Uplink and downlink streaming data to be processed for monitoring or prediction.

Required Data for Monitoring and Processing: To quantify the expected monitoring or processing effort to stream our generated videos, we analyze the monitoring of only uplink, only downlink, or both, uplink and downlink traffic. The result is shown in Figure 2.4 as Cumulative Distribution Function (CDF) with the required data to be processed on the x-axis. The individual lines reflect the effort required to consider uplink traffic, downlink traffic, and the entire corresponding traffic. Overall, the 50 % quantile of the data to be processed results in a traffic reduction of around 86 % if only the uplink has to be considered compared to uplink and downlink traffic. This shows the improvement potential of approaches that only require uplink data for monitoring and prediction.

2.3.3 Considerations on Monitoring and Processing Effort

To further compare a full packet trace and an uplink only based approach for QoE degradation factor prediction, a simple queuing model is set up, studying the load of a monitoring, processing, or prediction entity if only uplink traffic is used compared to a full packet trace consisting of uplink and downlink traffic.

Markov Model

During traffic flow monitoring, all packets of a video stream arrive at the monitoring instance, are potentially buffered and need to wait until the monitoring entity is free. Then, they are subsequently processed in first in first out (FIFO) order. This is described by a Markov model with a packet arrival process with inter-arrival time A, an unlimited queue for arriving packets, and a processing unit with service time B. The arrival process can be described by a Poisson process with rate λ when a sufficient number of flows are monitored in parallel [89]. The processing unit B processes the packets in the system with rate μ. For a stable system, $\lambda \leq \mu$ is given which means the arrival rate must not exceed the processing rate to not overload the system. Furthermore, for real-time monitoring or to satisfy specific network management Service Level Agreements (SLA), a target average processing time $E[B]$ to process a single packet by the monitoring entity is studied. For reasons of simplicity and since the model is only used as an illustration, we assume an exponential processing time distribution. Thus, the monitoring entity is modeled as an $M/M/1-\infty$ system. Note, we could also extend the system to $M/GI/1-\infty$ or more advanced models. However, to analyze the processing effort, the $M/M/1-\infty$ model is sufficient to illustrate and quantify the gain of our uplink only approach that is a magnitude better than a full packet trace solution.

Results: To determine the monitoring and processing effort of an approach based on uplink requests to predict QoE degradation factors, and compare it to a full packet trace approach, two different questions are answered: (1) What is the average response time $E[T]$ of the system, monitoring or processing video streaming data to predict QoE degradation factors, dependent on the expected processing time $E[B]$ of single packets?; and (2) how many video streams, or streaming flows, can be monitored in parallel considering a full packet trace or considering only uplink data, without exceeding a target response time t of the system with a probability q? Both questions are answered, based on the arrival rates for uplink traffic ($\lambda_{\text{ulgen}} = 0.32/s$) and the complete packet trace contain-

ing uplink and downlink traffic ($\lambda_{\text{allgen}} = 108.22/s$) achieved by streaming our generated videos from above. Furthermore, the uplink traffic rate ($\lambda_{\text{ul}} = 0.22/s$) and the complete uplink and downlink traffic rate ($\lambda_{\text{all}} = 79.65/s$) of the measured dataset described in Appendix A are used.

To answer the first question, Little's Law [90] is used to receive the average response time by $E[X] = \lambda \cdot E[T]$, with $E[X] = \frac{\rho}{1-\rho}$ as long-term average number of customers in a stationary $M/M/1-\infty$ system, $\rho = \frac{\lambda}{\mu}$ as utilization of the processing unit, and $E[X]$ as mean number of customers in the system. With Little's Law [90], $E[T]$ is received as $E[T] = \frac{E[B]}{1-\rho}$ with $E[B] = \frac{1}{\mu}$ as average response time of the system. As processing time $E[B]$, a value range between 1 μs and 1 s is studied in Figure 2.5 on the x-axis. The y-axis shows the resulting response time $E[T]$ of the system. The yellow line shows the result when monitoring uplink and downlink data, considering the generated videos. The orange line is the result considering the full packet trace of the measured dataset. The results are shown in brown when monitoring only uplink traffic for the generated videos, and in black for the measured dataset. It is evident that the average response time of the system is similar for both traffic types up to a packet processing time of 10 ms. For slower systems, the processing entity of the monitoring system is not able to analyze all packets of the stream anymore. Thus, the response time increases drastically, and the system is in an overload condition. When the uplink data is monitored only, the system can process all data up to a processing time of more than 1 s per packet. Thus, slower and cheaper hardware could be used.

To answer the second question of how many streaming flows can be supported in parallel, we consider a target response time t and a desired probability q of packets that must be analyzed faster than t to guarantee specific SLA. The system is analyzed for the packet arrival rates $\lambda_{\text{ulgen}} = 0.32/s$ and $\lambda_{\text{ul}} = 0.22/s$ for only uplink traffic of the generated videos and the measured dataset respectively, and for $\lambda_{\text{allgen}} = 108.22/s$ and $\lambda_{\text{all}} = 79.65/s$ for the full packet trace of the generated videos and the measured dataset respectively. The distribution function for the response time is $F(t) = 1 - e^{-(\mu-\lambda)t}$ for $t \geq 0$.

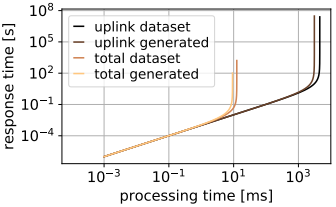

Figure 2.5: System response time comparison dependent on processing time of monitoring entity for uplink and all traffic, respectively.

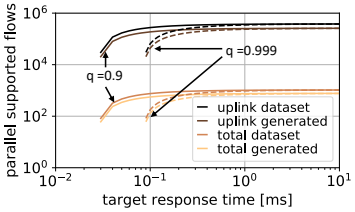

Figure 2.6: Possible number of parallel supported flows dependent on a target delay of a monitoring entity for uplink and all traffic, respectively.

Solving this, the supported positive arrival rate λ is dependent on the target probability q according to $0 < \lambda \leq \frac{\ln(1-q)+\mu t}{t}$.

To compare the expected behavior of a monitoring system when monitoring the full packet trace or only uplink data, we assume our monitoring system is capable of processing 1 Gbps. We assume that all arriving packets always fill the maximal transmission unit on the Internet of 1,500 B. This means our example monitoring system can process up to 83,333.33 packets per second. The target probability q is set to 90 % and 99.9 %, respectively. Figure 2.6 shows the number of flows that can be supported in parallel by our monitoring entity when only uplink traffic (brown and black line) or all uplink and downlink packets (orange and yellow line) are monitored based on a target response time t in milliseconds. Note that the analysis of a system with other capabilities works analogously and the main outcome of this investigation is to showcase the potential performance difference between a full packet trace and an uplink based monitoring approach. The figure shows that focusing on only uplink traffic monitoring allows 100 to 1,000 times more flows compared to a full packet trace monitoring using our example processing unit. Furthermore, we see a similar behavior for our measured dataset and the artificially generated random videos with models from literature. Moreover, since there is an active trend towards increasing data rates with higher resolutions and qualities, uplink-based monitoring is a

valuable methodology to deal with current and future video streams while saving hardware cost or distribute monitoring entities. However, given the varying degrees of complexity and required data that need to be monitored, there is a trade-off between prediction accuracy and computational effort when only uplink data or a full packet trace must be used. Thus, it is to be determined whether simple, lightweight, and resource friendly approaches, using only uplink data achieve reasonable results in predicting major QoE degradation factors.

However, based on these considerations, we can answer our first research question RQ2.1 as follows: *When we consider only uplink streaming data in comparison to a full packet trace to predict QoE degradation factors, we can reduce the monitored or processed data by around 86 %. Thus, we could use less powerful hardware to monitor or process video streaming sessions in the network or increase the possible number of streaming flows processed in parallel with the same hardware by a factor of 100 to 1,000.*

2.4 Quality of Experience Degradation Factor Estimation Model

Streaming a video is a continuous download and playback of video data. As soon as the buffer is filled to a target buffer level, as visualized in Figure 2.2, the same video time is played out as downloaded and the buffer fluctuates around a constant filling level. Thus, if the downlink throughput and the playback time downloaded by one chunk does not change, we assume a regular requesting and data downloading pattern, as long as the buffer does not deplete. We use this assumption in the following for a simple uplink based model detecting outliers during the streaming chunk requesting process to estimate QoE degradation factors. Therefore, we define different detection mechanisms and conduct a numerical evaluation for all mechanisms at the end of this section.

2.4.1 Methodology

The raw YouTube streaming dataset described in Section 2.3.1 serves well for an in-depth streaming study or issue prediction with ML techniques. However, to adapt the data to the scopes of an uplink based modeling approach, additional cleaning and processing steps are required.

Data Removal and Re-Labeling: The model in this chapter works with a sliding window of up to 10 requests. Thus, in addition all those measurement runs capturing less than 20 requests are excluded from the analysis. This leaves 12,253 valid video runs and nearly 1 M uplink requests from the dataset. In the comprehensive dataset, also very short buffer drops are labeled as buffer depletion events. However, from application data perspective, very short buffer depletion events, where the buffer is nearly full or is again filled immediately, are not important for issue detection or prediction.

Therefore, three cases where the buffer depletion phase is re-labeled are recognized: (1) If the buffer is filled for more than 100 s and there is no buffer health drop between request start and the next request, the request is marked with *steady*; (2) if more than 100 s of playback is available in the buffer, and the next request after a depletion request is again a filling or steady request, the phase is set to *steady*; (3) since a full video buffer is achieved with more than 120 s of playback time in YouTube mobile streaming [11], all requests with more than 120 s buffered playback time are marked as *steady*. This is consistent since the goal is not to detect small buffer drops of several seconds that are immediately filled again afterwards with this model, but rather to detect real playback issues that might lead to quality changes or stalling. After re-labeling, 46.01 % requests are associated to the filling phase, 49.75 % to the steady phase, 2.27 % to the depletion phase, and 1.97 % to a video stalling event. This sums up to 22,009 requests labeled as depletion phase and 19,112 requests labeled as stalling.

2.4.2 Quality of Experience Degradation Factor Estimation

In a running stream, beyond the initial delay, stalling and playback quality degradation are the most important QoE degradation factors [51]. The goal of this section is to detect stalling or buffering issues that lead to downwards quality changes by only uplink request data in a very lightweight manner.

Stalling Estimation Approach Idea

A video streaming session is characterized by a continuous download and playback of downloaded video data. If playback is slower than the download, the buffer level increases, whereas it decreases vice-versa. If we assume that the server delivers x seconds of video playback time as reply to each uplink request, then, the client buffer keeps at an almost constant filling status if the client issues one request per x seconds and the video playback is not paused. Should the inter-request time become smaller, the buffer increases in length and vice-versa, as already described in literature [11]. Thus, smaller inter-request times in the buffer filling phase and larger inter-request times in the steady phase are assumed. The largest intervals occur during the buffer depletion and stalling phases. To study these aspects, the CDF of the inter-request times in seconds observed during the four defined streaming phases for each video in the dataset is plotted in Figure 2.7. As one can see, 15 % of the inter-request times are similar and close to 0 s, regardless of the streaming phase. The smallest inter-request times are detected during the buffer filling phases, represented by the black line, with 80 % of all inter-request intervals being shorter than 5 s and nearly none of them larger than 10 s. Larger inter-request times are observed during depletion (orange line) and stalling (yellow line) phases, when 60 % and 50 % of them are larger than 10 s, respectively. Interestingly, a clear regular pattern of similar inter-request times during the steady phase (brown line) is not observed, where rather a nearly linear increase is visible from 0 s to 10 s. This is due to a parallel requesting process of audio and video content, as described in [11, 55]. To cap-

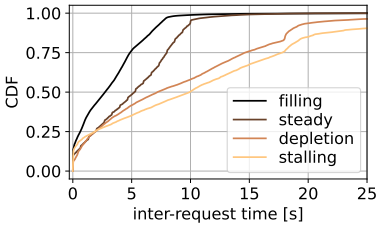

Figure 2.7: Inter-request time in streaming phases.

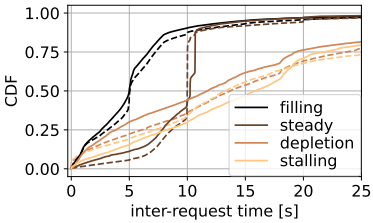

Figure 2.8: Inter-request time by flow separation (solid: video, dashed: audio).

ture this aspect, a simple video and audio flow separation algorithm is designed in the following. The main challenge here is that audio and video is sometimes transmitted in different flows and sometimes multiplexed to a single one. Differently from literature, where information in the downlink direction of the traffic, such as the amount of downloaded data or the number of downloaded packets after each request is used for separation, the focus is on uplink data only which makes the approach much more lightweight.

Flow Separation

The audio and video flow separation is based on two characteristics detected in the data. First, some streams use different ports for consecutive packets. Second, different sizes of the uplink request packets in bytes are observed for the two types of content. Interestingly, both characteristics are directly observable from uplink data. Note, no downlink following the uplink requests is considered here, only the single uplink packets and their packet sizes. Furthermore, it is assumed that an alternating pattern of uplink requests to video and audio content is sent to the server, similar to [11]. With this in mind, a simple, live, and uplink-request based only approach to separate audio and video flows is presented.

Separation Approach: The combination of network port number and uplink request packet size information is used to separate audio and video flows. Pre-tests show that poor separation accuracy is achieved when a port number only based approach is used. Regarding uplink packet size, two characteristics are visible in the data. First, alternating requests show different request sizes in a large percentage of the measured data. In other words, even order requests are as large as x while odd order request size equals $x + k$, as already presented in [55]. This is reasonable, as generally, additional information about the requested quality is sent to the server when video content is requested by the client, while audio content is usually available in a single constant bitrate. However, it can be shown that a separation process only based on uplink request sizes does not generalize to all videos [67]. This is because uplink request sizes increase from playback start to playback end. In other words, as playback time increases, the size of even and odd order requests increases to $x + \epsilon$ and $x + k + \epsilon$, respectively. Again, it is assumed that later in the video, additional information like the segment number is included in the client request.

Based on this observation, Algorithm 1 is used to separate audio and video flows. For a given video session, the algorithm takes the set $\mathcal{R}$ of uplink requests $R(t_{\mathrm{irt}}, s, p)$ as input, where a request R is structured as a 3-tuple of inter-request times t_{irt}, request packet size in Bytes s, and port number p. A sliding window approach is then implemented to separate streaming traffic into two flows, where the window size is chosen according to results of pre-test evaluations and fixed to $r = 10$ requests (second line of input in Algorithm 1). According to this approach, the algorithm determines whether the ten requests in the window have different request sizes first (line 1). If this is the case, and an even split is possible, i.e., a split of the ten requests with ratios of either 5 - 5 or 4 - 6 requests, the algorithm splits the requests accordingly. We applied this 'even' split as we assume a similar number of audio and video requests per video streaming session and both content types being downloaded in a continuous manner to keep the respective buffer sufficiently filled. Conversely, if more than 60 % of all requests in the window are same sized, an even split is not possible. In such a case,

Algorithm 1 Flow separation algorithm

Input: All uplink requests $R(t_{\mathrm{irt}}, s, p)$
Sliding window with $r = 10$ requests

1: **if** even split with $R.s$ **then**
2: label all requests R with larger $R.s$ as video and smaller $R.s$ as audio
3: **else if** different ports $R.p$ AND even split with $R.p$ **then**
4: split requests using $R.s$ and $R.p$
5: **else**
6: split requests alternating audio and video labels
7: **end if**

the algorithm checks whether an even split using the port number $R.p$ of the requests in the window is possible (line 3). Note that in both cases, the algorithm labels smaller requests as audio, while larger requests are labeled as video. As third option, if an even split using both, request sizes $R.s$ and port numbers $R.p$ is also not possible, requests are split as audio and video in an alternating fashion (50 % - 50 %), since parallel requesting of audio and video contents is assumed (line 6 of Algorithm 1). After the algorithm is performed, the inter-request time for audio and video flows is studied again, as displayed in Figure 2.8. Fixing the color code to the one used in Figure 2.7, solid lines are used for video and dashed lines for audio inter-request times. Two main differences compared to Figure 2.7 can be observed. First, 50 % and 60 % of all inter-request times are equal to 10 s for audio and video flows, respectively. Second, the percentage of requests in the depletion and stalling phase with an inter-request time of more than 10 s increases from 50 % to an average of 70 % after the audio and video separation. This confirms that the detection of playback issues during a streaming session goes in hand with the detection of very large inter-request times.

Quality of Audio and Video Separation: To determine the quality of the audio and video separation approach, now also the downloaded data is analyzed, as for the audio and video separation approach, only the uplink request packets and its sizes in Bytes have been used. It is considered that audio requests

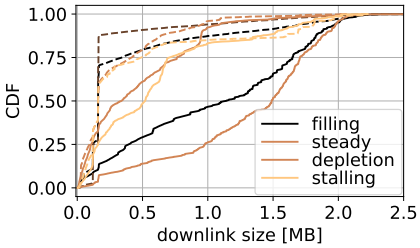

Figure 2.9: Downlink size after separation (solid: video, dashed: audio).

are followed by less downloaded data than video requests [29]. Indeed, this is true for larger video qualities, as the bitrates associated to 144p, 240p, 360p, and 480p video resolutions (up to 500 kbps [91]) are comparable to the largest bitrate used for audio contents (256 kbps [92]). Therefore, considering that the largest number of uplink requests in the dataset is observed when requesting 720p and the corresponding bitrate is on average between 1.5 Mbps and 4.0 Mbps, the 720p resolution is used first to analyze the accuracy of the presented separation algorithm as the difference is better visible. Remarks on the performance when using all data are given at the end of this paragraph. Note that the proposed quality determination is based only on the information available from the encrypted traffic flows rather than on unencrypted ground truth application information.

In contrast to the separation approach, where we only use the uplink requests, we also consider the downloaded data here to determine whether a sufficient audio and video separation has been done. Figure 2.9 plots the CDF of the downlink size (in Megabytes) of the server responses to 720p resolution uplink requests, for each defined streaming phase. Again, dashed lines are associated to audio requests and solid lines to video requests. Considering the steady and filling phases, a clear differentiation in the downlink size is visible between audio and video flows. In particular, more than 90 % of the responses to video requests are larger than 90 % of the ones corresponding to audio requests in the steady phase and larger than 75 % for the ones associated with the filling phase respectively.

Furthermore, it is observed that the audio response size has a multi-modal distribution during steady and filling phases, with 25 % of the responses being as large as 0.123 KB and 65 % of them reaching a size of 0.395 KB. Larger sizes are observed for the remaining responses. This confirms the assumption that the separation works rather good with constant bitrate audio flows for nearly 90 % and 70 % of the requests in the steady and filling phases, respectively.

A less remarked but still evident differentiation is shown during the depletion and stalling phases. In such cases, it is underlined that factors such as drops in the download bandwidth or unfinished request downloads can lead to noisy flow separation. This is especially true for the largest 40 % of the requests, where for both phases the requests labeled as audio are larger than the average request size of audio requests observed during the steady phase. Note that the separation algorithm works remarkably good also when the player returns to the steady phase after a depletion phase, i.e., also multiple buffer depletion or stalling phases can be considered for detection.

For completeness, similar analyses are performed for 360p, 480p, 1080p resolutions, and on all data. Regarding 360p resolution videos, results are similar to 720p, while when 480p is requested, a less evident distinction between audio and video is observed during the buffer filling phase. Similar trends are observed for 1080p videos, e.g. due to the weaker representation of this request classes in the considered dataset. Using all monitored data, the trend for a bimodal distribution for audio data and again, larger video requests is also visible. More than 80 % of all audio requests and nearly 90 % of all audio requests in the steady phase are smaller than 0.5 MB. In contrast, it is only 60 % of all video requests and 35 % of video requests in the steady phase. Especially in the filling and steady phase, differences are again visible clearly.

As final remarks, it is observed from the inspection of the results that a high accuracy is reached when splitting is based on request size information, while port-based and alternation-based separation are much more error-prone. In such cases, errors reduce if the port numbers previously identified as audio or video are used as additional information for separation. However, this exten-

sion comes with some additional minor data storage overhead per monitored video session, especially if ports change frequently. Thus, it is not adopted in the following. Also, it is reminded that the goal of this approach is to perform a very simple and lightweight separation as first step to detect buffering issues, rather than achieving top performance using more accurate but more-complex full packet trace based ML models. This is achieved in the ML approach shown in the next section. In fact, minor classification errors are either irrelevant if depletion phases are already detected or can anyway be compensated by cross-validation of the audio and video inter-request times, as shown later. Furthermore, the stalling estimation approach introduced in the following does not require knowing exactly which of the traffic flow carries video and audio content, but just the separation of the flows.

Quality Degradation Estimation Approach

According to Figure 2.8, a large fraction of inter-request times are between 10 s and 11 s during the steady phase after the separation of audio and video flows. Thus, it is assumed that issues during playback are more likely when the time between two consecutive uplink requests of the same type (i.e., either audio or video) is larger, due to the fact that fewer data is requested, received, and added to the buffer than played out. For that reason, two quality degradation estimation approaches are proposed: i) a *Fixed Threshold* based approach, that uses a fixed threshold value for the inter-request time to detect issues during content buffering, and ii) a *Moving Average* based approach, that adapts the threshold value according to recent history. Note that the latter approach works independently of the streaming platform as no expert knowledge is required to preselect inter-request time threshold values. Differently from literature, with this approach, the detection of issues in content buffering at exact positions is not the main target. Instead, the goal is to detect a buffer depletion event as early as possible, to provide a network operator or service provider with enough time to react to buffer drops and avoid downwards quality changes or stalling via smart network management techniques.

Fixed Threshold: With the fixed threshold approach, all requests exceeding a pre-defined inter-request time threshold η are marked as *issue expected*, for both audio and video flows independently. In other words, both audio and video flows are processed from the first request to the last request, exactly as this approach can be applied on the fly at a monitoring point of presence. Note that in this approach the first and last five requests are excluded from the detection phase to avoid unwanted behaviors related to the beginning of the filling phase or to the end of the video session.

Moving Average: The drawback of the fixed threshold is the requirement of expert knowledge to determine the value of the inter-request time threshold, that in general depends on, among others, the streaming platform and on the behavior of the streaming adaptation algorithm. This limitation is overcome by introducing a moving average based approach, where the best threshold value is learned online during the streaming session. This is effective, considering that a change of the streaming phase is followed by a drastic change in the inter-request time. Based on this idea, the *issue expected* labeling is adjusted from the *fixed threshold* approach as it follows. First, the threshold value is set as equal to the mean of the inter-request times over the last $m - 1$ requests. Second, the inter-request time of the current request is compared to the last updated threshold value: if the former exceeds the latter, the last request is labeled as *issue expected*. Finally, the value of the threshold is updated, including the last measured inter-request time in the moving average window. Note that in this case, the first m requests in each video are not considered, applying exactly the same moving average procedure for all requests. In addition, as done with the fixed threshold based approach, the last five requests are also not considered.

2.4.3 Parameter Study Definition

By means of the presented quality degradation estimation approaches, the different mechanisms are studied for different parameters. For that reason, the following four studies are defined to detect the defined playback issues, namely buffer depletion phases, quality changes to a lower resolution, and stalling.

Study S 1: In the first study, the performance of the fixed threshold approach when using different inter-request time threshold values η is compared. The goal is to determine the impact of expert knowledge on the choice of the threshold value, for both audio and video traffic flows.

Study S 2: In the second study, the moving average approach, where the algorithm dynamically adapts the threshold value for audio and video flows independently is used. Thus, no expert knowledge is required in this case.

Study S 3: The detection criterion is modified in this study, labeling a request as issue if the last two consecutive inter-request times exceed the threshold value, according to the selected approach. The rationale behind this study is that issues leading to quality changes or stalling are usually longer than a single increased inter-request time.

Study S 4: In the fourth study, a request is marked for a content type (e.g., audio or video) as issue only if the successive request for the other content type (i.e., video or audio) is detected as issue as well, regardless of the used detection approach. The rationale behind this idea is to reduce wrong estimates by potential misclassification during audio and video flow separation and improve the overall estimation performance.

On the one hand, the performance of all studies using the fixed threshold approach is evaluated with threshold values η in the set $\{12\,s : 0.5\,s : 30\,s\}$. On the other hand, considering the study adopting a moving average to tune the threshold value, after a trial and error procedure the value of m is fixed to 10. Moreover, for each moving window size, a request is labeled as issue if the current inter-request time exceeds the updated threshold value by $p\,\%$, with p varying in $\{0.1, 0.25, 0.5, 0.75, 1.0, 1.5, 2.0, 2.5, 3.0\}$.

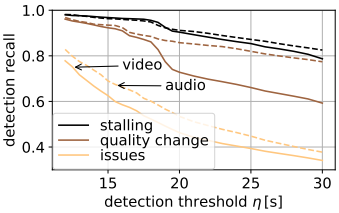

Figure 2.10: S 1: detected stallings and issues in streaming.

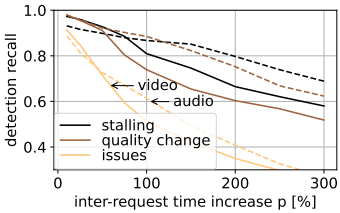

Figure 2.11: S 2: detected stallings and issues: moving average $n = 10$.

2.4.4 Numerical Evaluation

The evaluation discusses the results for the best parameter settings for the four defined studies above. As for performance evaluation, a request detected as expected issue is labeled as correct detection if any of the n successive requests is associated to a ground truth issue. The value of n represents the length of the detection horizon: in this work n is fixed to $n = 4$, as a trade-off between short-term (less time for providers to react to impairments) and long-term look-ahead capabilities. Note that when an issue is correctly detected, the full depletion phase including the detected issue is also considered as correctly detected, as well as the stalling events or quality changes occurring during the same phase, independently of its duration. This approach is valid, since either a depletion phase is instantly detected from its beginning or missed completely.

Study S 1: The performance of the proposed quality estimation strategy when the fixed threshold approach is used versus the different values of the threshold η is plotted in Figure 2.10. The performance is expressed as recall values, i.e., the ratio of correctly detected stallings (black lines), quality changes (brown lines), and general issues (yellow lines) with respect to all occurrences when the player drops into the depletion phase. Again, solid lines refer to the performance using video flows and dashed lines using audio flows. As one can see, high detection

performance is reached when stalling or quality changes are targeted by the algorithm. In particular, when $\eta = 12\,s$, the recall for stalling and quality change detection equals 98 % and 97 %, respectively, while it decreases to 83 % when the identification of the whole depletion phase is targeted. Also, it is observed that fewer issues are detected for larger threshold values: as an example, 90 % of all stallings are still detected regardless of the flow type when $\eta = 20\,s$, while the performance drops to 74 % (video) and 85 % (audio) in the case of quality change estimation and to less than 50 % when the detection of the depletion phase is targeted. On average, 26 % of the issues are detected before they happen when $\eta < 15\,s$, i.e., the corresponding request succeeds the request used to perform estimation. This is important, as a network provider could use the output of the system to implement smart network management strategies to allocate more resources and eventually avoid the estimated issue. Note that the drawback of this high detection percentage is the number of false positives, corresponding to a precision value between 0.3 and 0.6. Nevertheless, many false positives can be avoided by considering larger threshold values that consequently lead to missing very short issues during playback. Lastly, it is worth to compare the time duration of the detected issues versus the missed ones. In particular, the mean number of requests included in detected issues is between 4.15 and 4.36 for video flows, while it is between 3.91 and 4.32 for audio flows. Interestingly, the mean duration of missed issue phases lays between 1.24 and 1.73 requests for both audio and video. As a consequence, the algorithm privileges the detection of longer issues, that most likely lead to quality changes or stalling.

Study S 2: In S 2, stalling, quality changes, and general issues are predicted using a moving average approach with a moving window of $m = 10$ requests. Figure 2.11 plots the performance of this approach versus fixed values for the parameter p. Note that the algorithm expects an issue if the inter-request time of the current request increases by more than p of the mean inter-request time computed over the last $m - 1$ requests. As one can see, when $p = 10\,\%$ stalling is detected based on the video or the audio flow with a recall of 97 % and 93 %,

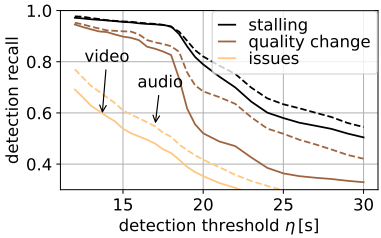

Figure 2.12: S 4: fixed threshold, validation with other flow.

respectively. A recall of 97 % and 90 % is achieved when quality changes or general issues are targeted, for both flow types. Lower performance is achieved for larger values of p. Similarly to study S 1, the precision is the drawback of this approach with results between 0.16 and 0.41. In the following, two alternative solutions to counteract the occurrence of many false positives are described.

Study S 3: In this study, the performance of the system is evaluated when more than one request is used to perform quality degradation detection, as a way to reduce false positives. In particular, an issue is detected by the algorithm only if two consecutive requests exceed the thresholds η (if the fixed threshold approach is adopted) or p (if a moving-average is used by the system). Results show that this solution slightly improves the precision of the algorithm at the price of small decreases of the recall. In particular, recall values between 92 % and 86 % are achieved between $\eta = 12s$ and $\eta = 23s$ for stalling events when the fixed threshold approach is adopted. The minimum precision increases to 0.31, but the maximal precision stays at 0.59. From the inspection of the results, the conclusion is that this solution does not significantly improve the detection.

Study S 4: As last experiment, the detection output for the audio (video) traffic flow is used to validate the detection performed on the video (audio) traffic flow. The results of this study are plotted in Figure 2.12, if the fixed thresh-

old approach is adopted. Recalls for small threshold values are similar to S 1 for both, quality change and stalling detection, while less general issues are detected, as shown by the yellow line. However, in contrast to the other studies, much higher precision values are achieved in this case. In particular, the precision equals 43 % and 45 % for video and audio respectively when $\eta = 12s$, and it increases to more than 60 % when $\eta = 18.5s$, where also more than 90 % of all stallings are detected regardless of the flow type. A further increase in the precision (more than 70 %) is observed for $\eta = 21s$, even though the stalling detection recall drops to 75 % in this case. This result highlights a clear trade-off between precision and recall performance, and thus, detection of any quality degradation versus the number of false positives. Note that the observed precision levels are higher than the corresponding performance achieved in related literature using ML-based approaches [44, 45, 55] for similar recall values. As far as the moving average approach is considered, a higher percentage of issues are detected, similarly to S 2. Nevertheless, no significant improvement in the precision is observed when a moving average approach is adopted. The reason is a large difference in the inter-request times when the player switches from the filling to the steady phase. This leads to additional false positives with the moving average based prediction.

Based on these findings, the second research question (RQ2.2) of this chapter can be answered as follows: *With a simple uplink based model, it is possible to estimate two of the four major target QoE degradation factors, namely, quality changes to a lower resolution and stalling. The recall and precision of the approach is similar or better compared to state-of-the-art literature, mainly using full packet traces or resource consuming ML-solutions. The prediction quality is improved with expert knowledge by means of the expected inter-request time of uplink requests or very accurate audio and video flow separation. However, also the cross-validation of video flow based prediction with audio and vice versa, as shown in study S 4, achieves good results. The drawback is that neither the initial delay nor the playback quality can be estimated because of the requirement of historic information or the downlink size of the requests, respectively.*

2.5 Quality of Experience Degradation Factor Prediction with Machine Learning

The session model has the following limitations: (1) it does not work for initial delay estimation since according to the initial delay study in the dataset presented in Appendix A.3.1, the initial delay can not be quantified by requests only. (2) The requesting process is independent of the playback quality and thus, the quality as important QoE degradation factor can not be estimated. (3) A general differentiation between stalling and not stalling is not possible at exact positions during playback but only in advance, dependent on the buffer filling status and a requesting pattern change. For that reason, a ML model is presented in this section to overcome these limitations and extend the prediction possibilities. To be more precise, the prediction of each QoE degradation factor is discussed in the following. Therefore, a simple random forest model is used, since according to literature [93], good results are achieved with streaming and QoE related challenges when training with random forest based approaches. Thus, the *sklearn* library implemented in Python is used for model setup in this work. Further implementation details for the model are available at Zenodo [36]. In the following, first the general methodology and the model setup is discussed, followed by the prediction results for each QoE degradation factor.

2.5.1 Feature Set Assessment and Data Preparation

The feature set assessment of this section covers an overview of the features for all target QoE degradation factors first. Afterwards, initial delay and quality prediction is discussed in detail, since the estimation is done without additional information from the streaming behavior. Then, the approach to predict quality changes and stalling is introduced by presenting the idea with video phase detection, prediction, and the correlation to specific uplink request patterns.

Table 2.3: General feature overview.

Index	Feature	Explanation
f1	request start	relative request timestamp
f2	inter-request time	time between two requests
f3	downlink duration	duration: request to last downlink packet
f4	request size [byte]	size of request packet
f5	downlink [byte]	total downlink volume for request
f6	downlink packets	number packets downloaded for request
f7	uplink [byte]	total uplink volume for request
f8	uplink packets	number packets uploaded for request
f9	port	server port of video flow
f10	protocol	used network layer protocol

Feature Set Overview

To receive satisfactory and correct results by a ML model, a thorough feature selection is required. The goal is to keep the volume of network layer data and thus, processing effort for the analysis as low as possible, without losing prediction accuracy. Therefore, only uplink traffic and aggregated downlink traffic from all network data after post-processing is used instead of analyzing each downlink packet. A general overview of these data is summarized in Table A.1 in the Appendix. Table 2.3 gives an overview of all features used for our QoE degradation factor prediction with an explanation.

The request start timestamp *f1* is the UNIX timestamp of each uplink request to the server relative to the initial request. Thus, the initial request is always set to 0 s. The *request size* feature *f4* is the packet size in Byte of each single uplink request. Note that a more detailed description of the other non-trivial features is given in Figure A.3 in the Appendix. For a correct ML model, all categorical features are *one-hot-encoded* to avoid overfitting, in particular by the protocol TCP or QUIC. With all features from Table 2.3, several feature sets are established for each target QoE degradation factor, summarized in Table 2.4.

Table 2.4: Overview of all feature sets.

QoE degradation factor	Full	Selected	Uplink
initial delay	f1 – f10	f1–f8	f1, f2, f4, f7
quality	f2 – f8, f10	f2 – f8	f2–f4, f7, f8, f10
streaming phase	f2 – f8, f10	f2, f3, f5	f2, f4, f7, f8, f10
quality change	f2 – f8, f10	f2, f3, f5	f2, f4, f7, f8, f10
stalling	f2 – f8, f10	f2, f3, f5	f2, f4, f7, f8, f10

Full Feature Set: The *full* feature set contains all relevant features for each target QoE degradation factor prediction. These are all available features for the initial delay prediction. For the other QoE degradation factors, the duration is more important than the actual timestamps of the request starts. Furthermore, the port is usually randomly selected and the model may overfit with it. While port information could be important to detect server changes, our pre-studies show little influence on the performance and a very little feature importance score is achieved using *SelectKBest* [94]. Thus, the request start and the ports are excluded for quality, quality change, streaming phase and stalling prediction.

Selected Feature Set: For the *selected* feature set, *SelectKBest* [94] is used to achieve a feature importance score with the *full* feature set as input. The result of the *SelectKBest* algorithm for all predicted QoE degradation factors is presented in Table 2.5. The goal is to exclude less relevant features, i.e., features with very small scores. The *mutual_info_regression* score function is used for the regression based initial delay prediction. For all classification based predictions, the *mutual_info_classif* score function is used. To determine the selected features, the process is as follows: if there is a clear differentiation possible between features with high scores and small scores, only all features with high scores are used for the selected feature set. This is possible for the streaming phase, quality changes, and stalling, as shown in Table 2.5. For the initial delay and the quality prediction, no clear best features can be identified. Thus, only the features with

Table 2.5: Feature importance (bold: high scores for selected feature sets).

QoE deg. factor	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
initial delay	**0.55**	**0.44**	**0.36**	**0.40**	**0.45**	**0.43**	**0.35**	**0.34**	0.05	0.13
quality	–	**0.15**	**0.19**	**0.28**	**0.64**	**0.36**	**0.33**	**0.25**	–	0.11
streaming phase	–	**0.93**	**0.89**	0.21	**0.76**	0.19	0.28	0.12	–	0.15
quality change	–	**0.59**	**0.59**	0.06	**0.49**	0.09	0.17	0.05	–	0.09
stalling	–	**0.60**	**0.60**	0.24	**0.55**	0.16	0.26	0.14	–	0.23

very little score are not considered. Therefore, the port (f9) and the protocol (f10) is not included for initial delay prediction. For quality prediction, we consider all features except the port, as test quality predictions without the inter-request time (f2) and the downlink duration (f3) lead to bad prediction results. At the end, all features with the bold scores are used in the selected feature set.

Uplink Feature Set: To answer research question RQ2.3, whether the initial delay, quality changes, playback quality, and stalling as the most important QoE degradation factor are predictable with a simple ML-based approach and only uplink information, an uplink feature set is developed for all QoE degradation factors. Therefore, pre-studies are conducted to determine for which uplink feature combination an in-depth model training is meaningful. The resulting uplink feature sets for all target degradation factors are summarized in the right column of Table 2.4. For initial delay prediction, promising initial results are achieved with uplink only data. In contrast, pre-studies show much worse results for the other QoE degradation factors. Especially for quality prediction, it was not possible to achieve satisfactory results with accuracies of less than 65% when only uplink features are used. Thus, the downlink duration (f3) is added to the uplink feature set as a trade-off between prediction accuracy and data usage. Only the timestamp of the last downlink packet per chunk download is required to determine the downlink duration. For that reason, only one additional downlink packet for each request must be kept for prediction.

Machine Learning Data Preparation

For an accurate prediction, the data is prepared as input for a ML model in several steps using bootstrapping, shuffling, and normalization. Furthermore, the data set is split in a test- and training-set, hyperparameters are optimized, and three-fold-cross-validation is used to achieve the best configuration and results. More details on these data preparation steps are available in Appendix A.4.

2.5.2 Machine Learning-Based Prediction Results

This section summarizes results for predicting the most important QoE degradation factors: initial delay, video quality, quality change, and stalling. Furthermore, the video phase prediction results as basis for quality change and stalling prediction are presented.

Initial Delay

In this section, we present the estimator performance, starting with the usage of the *full* feature set and model variations. Since it is assumed that only a specific time period at the beginning of the video download is relevant to predict the initial delay accurately, only the initial 20 requests are considered for the prediction. In the following, also the influence of a different number is tested.

Model Variation: To study the performance of the random forest model, three different initial settings can be varied: the videos in the test- and training-sets, the tree generation of the random forest algorithm, and the features themselves. For that reason, the influence of video split and tree generation is investigated first. Using a fixed seed of a specific value for both, random video split and tree generation, the run produces repeatable, reproducible results. By using random seeds for the video split, the videos used for training and testing are varied and the behavior and possibly the model performance change. Furthermore, since the problem of learning an optimal decision tree is known to be NP-complete [95], different tree generations can be tested to determine

the performance variation. This is done by varying the sample of the features to consider, when looking for the best split at each node of the tree. The four scenarios of fixed and random video and tree seed (fix seed of 42) are studied with 15 repetitions each. The goal is to see differences in the performance estimating the initial delay. From a detailed examination of all runs, no significant differences are achieved. The mean absolute error (MAE) for the prediction is between 0.65 s and 0.68 s for all scenarios, the 75 % percentile is between 0.79 s and 0.81 s, and the 90 % percentile is between 1.77 s and 1.83 s. Thus, there is no benefit in using random seeds in the model creation and the fixed value of 42 for both, the video and the tree seed is used for better and active reproducibility. Note that with other seed values slightly better results may be possible. However, studying this is not the main focus in this chapter and the improvement of another seed could be a drawback using a different dataset.

Next, the number of requests used as input for the model is varied between 5, 10, 15, and 20 requests. Similar results and a MAE of 0.65 s - 0.70 s is achieved with 10, 15, and 20 requests and a significant performance loss with 5 requests and a MAE of 0.96 s is determined. For that reason, the following models are trained with a fixed video and tree seed and 10 requests as baseline scenario, using a minimum number of requests while keeping similar performance.

Estimation Performance: In this section, the baseline scenario with 10 requests is compared to a manual split in test- and training-videos. The test-set with the manual split contains no videos the model is learned on. Thus, the model is tested on completely unknown videos and the study shows valuable results for generalizability. The performance is presented in Figure 2.13. The *unknown video* scenario, represented by the brown line, shows the manual split with completely different videos used for the test- and the training-sets. This is compared to the baseline scenario in black. To quantify the quality of the results, the *mean error* line in orange is added, describing the estimation error when always estimating the initial delay with the real mean initial delay of 2.64 s as benchmark. The *mean error unknown videos* line in yellow is the same

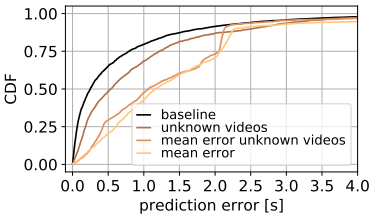

Figure 2.13: Prediction comparison using known or unknown videos in the test-set.

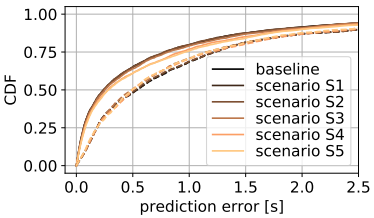

Figure 2.14: Feature set comparison (solid: known videos, dashed: unknown videos).

benchmark, but for the prediction with only unknown videos in the test-set. The prediction always performs better than simply estimating the mean initial delay. The baseline scenario shows a MAE of 0.68 s, and the *mean error* scenario has a MAE of 1.62 s. The model can predict the initial delay with a prediction error of less than 1.0 s even with unknown videos in more than 65 %. The MAE for predicting the initial delay based on unknown videos is 1.00 s, and the median prediction error is 1.06 s. The difference in the prediction error range of 2.0 s and more is a result of different splits in test- and training-sets. Nevertheless, similar results are achieved if the videos in the test-set are exchanged. This approach outperforms always predicting the overall mean initial delay, in more than 85 % of the cases. Hence, the approach is also valuable for videos that were not measured in this work and are not included in the dataset.

Scenarios for Different Feature Sets: Last, the estimator performance for the initial delay prediction is studied when using other feature sets. In addition to the *full* feature set, discussed in the previous section, the *selected* feature set described as scenario S1, and the *uplink* feature set, (scenario S3 of Table 2.5), two other feature sets are studied. This is done to determine the potential of this approach in greater detail and the influence of reducing the number of data any further, exemplary with the initial delay. This leads in total to five reduced feature sets that are summarized in Table 2.6. In scenario S2, all packet-based in-

Table 2.6: Feature overview.

Scenario	Indices	Explanation
S1	f1 – f8	selected feature set
S2	f1 – f5, f7	excluded packet information
S3	f1, f2, f4, f7	uplink feature set
S4	f1, f2, f4	only request based features
S5	f1, f4	only request start and size

formation is excluded in contrast to the selected feature set S1, but the downlink information is retained. In scenario S4, only the *request start*, the *inter-request time*, and the *request size* are used. Lastly, S5 uses only the *request start* and the *request size* as the highest ranked uplink features directly determined from each uplink request packet without the requirement of further calculations.

Modified Feature Set Investigation: The study's result is shown in Figure 2.14 as CDF. The x-axis shows the prediction error in seconds, limited to 2.0 s to observe the feature set differences. The trend for larger prediction errors is comparable to the studies above, and is not investigated again. The solid lines show the random test- and training-set split and the dashed lines reflect the separation with only unknown videos in the test-set. The different lines describe the scenarios as shown in Table 2.6. The baseline scenario presents the results for the full feature set. The scenarios with a reduced feature set perform similarly to the baseline scenario. The overall best performance when videos are randomly assigned to test- and training-set is achieved by S2 with a MAE of 0.67 s, a 75 % percentile of 0.77 s, and a 90 % percentile of 1.79 s. The highest MAE is achieved by S5, with a MAE of 0.78 s. The performance impairment when different videos are in the test- and training-set is similar to the scenario above. However, in general, it is evident that the reduced feature set is sufficient to predict the initial delay accurately. Especially scenarios S4 and S5, that contain only information from the uplink requests, offer good results for initial delay prediction.

Quality Estimation

To estimate the playback quality and avoid errors having quality changes or stalling events, no requests with a maximal buffer filling level smaller than 10 s are considered. Before the prediction starts, a standard scaling [96] is done to avoid unexpected or unwanted behavior. This is done, as variables measured at different scales do not contribute equally to the learned model and the model fitting function. This can lead to an unwanted bias. The standard scaler transforms data in a form such that its distribution has a mean value of zero and a standard deviation of one. The prediction results are presented in the following for the *full*-, the *selected*-, and the *uplink*-feature sets from Table 2.4.

Full Feature Set: The correct quality for the full feature set is predicted with 84.3 %. The full confusion matrix is available in Table 2.7. The best prediction is achieved for 720p quality with prediction accuracy, precision, recall, and F1 scores of 92 %. The worst performance is achieved for 240p, with an accuracy of 75 %, a precision of 75 %, a recall of 62 %, and an F1 score of 68 %. The reason for the large prediction differences is the nature of the different qualities. The qualities are better distinguishable by the downlink size for larger qualities. Furthermore, especially between 720p and 1080p, a clear difference by the inter-request time is determined. More details about this is available in the Appendix with a plot in Figure A.4.

The different number of requests for each resolution influence the overall prediction quality slightly. The macro average precision is 0.81, compared to 0.84 for the weighted average precision. The macro average recall is 0.72 and the macro average F1 score is 0.74. In comparison, the weighted average precision and F1 score are 0.84. An overall prediction accuracy of 84.68 % is achieved with hyperparameter optimization. The macro and weighted average values for precision, recall, and the F1 score are similar to the prediction without bootstrapping. Furthermore, it is evident that for most mispredictions, the adjacent resolution is predicted. The error for non-adjacent resolutions is less than 10 %. In addition, no statistically significant differences in the prediction results are

Table 2.7: Confusion matrix for quality prediction result: full feature set.

	144p	**240p**	**360p**	**480p**	**720p**	**1080p**
144p	**0.7949**	0.1095	0.0231	0.0268	0.0350	0.0108
240p	0.0935	**0.7524**	0.0618	0.0465	0.0316	0.0141
360p	0.0236	0.0451	**0.7785**	0.0971	0.0433	0.0124
480p	0.0148	0.0234	0.0732	**0.7703**	0.0860	0.0323
720p	0.0079	0.0066	0.0118	0.0341	**0.9181**	0.0215
1080p	0.0100	0.0088	0.0100	0.0365	0.1060	**0.8287**

Table 2.8: Confusion matrix for quality prediction result: uplink feature set.

	144p	**240p**	**360p**	**480p**	**720p**	**1080p**
144p	**0.8443**	0.0466	0.0213	0.0319	0.0483	0.0076
240p	0.2423	**0.5305**	0.0611	0.0804	0.0737	0.0120
360p	0.0507	0.0439	**0.6554**	0.1510	0.0861	0.0128
480p	0.0370	0.0248	0.0615	**0.7244**	0.1280	0.0243
720p	0.0206	0.0075	0.0127	0.0443	**0.8932**	0.0217
1080p	0.0320	0.0124	0.0178	0.0738	0.1220	**0.7420**

achieved when using only unknown videos, thus videos the model is not trained on, in the test-set. It is assumed that the video information has only minor impact on the prediction performance.

Selected Feature Set: For the *selected* feature set, a prediction accuracy of 83.11 % is achieved. The weighted average precision, recall, and F1 score are at 0.83, while the macro average precision is 0.79 and the macro average recall and F1 score is 0.78. This shows a minor class imbalance due to the higher percentage of 720p-resolution requests available in the dataset. Compared to the estimation with the full feature set, though, the macro average values are improved. Thus, it is assumed that removing less relevant features reduces overfitting without losing overall prediction accuracy.

Uplink Feature Set: For the uplink feature set, results regarding precision, recall, and the F1 score for all resolutions are shown in Table 2.8 to present the influence of the reduced feature set on the overall prediction quality. It is evident that for 720p, the prediction still works rather well, whereas for 240p and 360p, the prediction is much worse compared to the full feature set. For 240p, in 24.23 % of all instances the 144p resolution is falsely predicted. We see a similar result for the macro average precision of 0.75, recall of 0.73, and F1 score of 0.74. The weighted average values are 0.79 each. Thus, it becomes clear that the uplink feature set is not enough to predict each resolution accurately. For low resolutions in particular, this does not work, although for high resolutions, especially 720p in this case, the results are good. For 720p, accuracy, recall, and the F1 score are at 0.89 and precision is at 0.88.

Video Phase Estimation

Video phase information can be used as additional input to predict quality changes or stallings. As already shown with the uplink model in the previous section (e.g. Figure 2.7), the inter-request time includes valuable information for this prediction task. However, in the following, the usage of all feature sets is evaluated. The results are summarized in Table 2.9. The top of the table shows the feature sets and the performance metrics as P (Precision), R (Recall), and the F1 score. The column at the left shows the prediction, in this case either the phase, the macro average score, or the weighted average score.

Prediction Accuracy: The results show large differences, depending on the predicted phase. The filling and steady phase is predicted with good results and F1 scores of 0.94 and 0.90, respectively, using the full feature set. The result is different for the depletion phase and the stalling phase, with F1 scores below 0.80. In the depletion phase, most prediction errors predict the steady phase. For the stalling phase, most mispredictions are for the depletion phase or the filling phase. The difference in the macro and weighted average scores is a result of the class imbalance in the nature of streamed videos, with much more requests

Table 2.9: Phase prediction result for full, selected, and uplink feature set (P: Precision, R: Recall, F1: F1 score).

	Full			Selected			Uplink		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
depletion	0.76	0.81	0.79	0.74	0.77	0.75	0.74	0.78	0.76
filling	0.95	0.93	0.94	0.94	0.92	0.93	0.95	0.93	0.94
stalling	0.77	0.75	0.76	0.72	0.69	0.71	0.77	0.74	0.75
steady	0.89	0.91	0.90	0.86	0.89	0.88	0.87	0.90	0.89
macro avg	0.84	0.85	0.85	0.82	0.82	0.82	0.83	0.84	0.83
weighted avg	0.91	0.91	0.91	0.89	0.89	0.89	0.90	0.90	0.90

in the steady and filling phases compared to the stalling and depletion phases. Furthermore, the table shows a similar performance for different feature sets for the streaming phase prediction task. The selected feature set shows slightly worse results than the full feature set, and the uplink feature set improves the F1 score compared to the selected feature set. Thus, in contrast to the video quality prediction, the uplink-based feature set is convenient to predict the current video phase. When using only unknown videos in the test-set, where the model has not been trained on, the results are similar to the other phase prediction experiment for both approaches, with some limitations; predicting the depletion and stalling phases achieves precision, recall, and F1 score values of 0.5 or less. One reason is the class imbalance in the dataset again. Furthermore, if the buffer is slowly depleting over a longer timespan, requests are labeled with *depletion* but behave very similarly to requests in the steady phase.

Quality Change Estimation

The best results to predict quality changes are summarized in Table 2.10 and are achieved with a random test- and training-set split and the correct video phase as additional input feature. The table shows that the prediction of no quality change works very well, with F1 scores of 0.97 and higher for all feature sets.

Table 2.10: Quality change prediction result for full, selected, and uplink feature set (P: Precision, R: Recall, F1: F1 score).

	Full			Selected			Uplink		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
no quality change	0.99	1.00	0.99	0.99	1.00	0.99	0.99	1.00	0.99
quality change	0.84	0.56	0.67	0.82	0.54	0.65	0.83	0.55	0.66
macro avg	0.91	0.78	0.83	0.90	0.77	0.82	0.91	0.78	0.83
weighted avg	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

Note that the results listed with a score of 1.00 show high 99 % results. Since the negligible differences in the high 99 % range have no influence on the overall statement, no more digits are provided.

In contrast, the prediction performance for requests with quality changes is worse. While the precision of about 0.80 is rather good, the recall is only slightly higher than 0.50. This is explainable by the common streaming behavior, where not many quality changes are performed compared to the total playtime. Out of more than 1.1 million requests, less than 30,000 requests are labeled as quality change. Second, a downward shift to a lower quality is usually triggered in the middle of a depletion phase or after stalling. However, there are also changes to a higher quality during filling phases. This variety of situations where quality changes can be triggered makes prediction difficult. However, the high precision shows a high true positive rate, and thus most predicted quality changes could be identified as such. The drawback is that many quality changes are missed. Two additional experiments are done to predict quality changes: using only unknown videos in the test-set where the model has not been trained on and quality change prediction without knowing the video phase. The results show that knowing the video phase before predicting quality changes increases the prediction result by up to 3 % for both approaches. Using different videos for training and testing has a significant negative influence on the recall result.

Stalling Estimation

Video stalls are estimated in this subsection, as the last but most important QoE degradation factor. The prediction results, with a random test- and training-set split, and without the video phase as input, are summarized in Table 2.11. The table structure is retained as earlier. The *no stalling* line is the prediction result for all requests that did not stall, and the *stalling* line for all requests with stalling. Again, precision, recall, and the F1 score for the *no stalling* case are higher compared to the *stalling* case. This, is again a result of the higher number of *no stalling* requests. However, it is also obvious that for *stalling* prediction, both approaches perform similarly for the *full*- and the *selected*-feature sets. For the *uplink*-feature set, the result is even slightly better. Additionally, similar F1 scores are achieved for all feature sets. Thus, it is possible to predict stallings at network layer with uplink requests only with an F1 score of 0.89. In addition, for 60 % of all false positives, a video buffer of less than 9 s is detected, independent on the used feature set. Other experiments show that the current video phase is also a valuable input to increase prediction performance. However, since the *stalling phase* is defined as one of the streaming phases, the input of the phases is meaningless. Nevertheless, the exclusion of buffering and steady phases, which are detected quite accurately, offers high potential to improve the performance. Having only unknown videos in the test-set, where the model has not been trained on, the macro average F1 score drops to 0.67 or less and lower precision and recall values for the stalling case for all experiments are detected. Here, improvement potential is also visible using the streaming phase.

As a consequence and to conclude, the third research question RQ2.3 of this chapter can be answered as follows: *The random forest based ML model can estimate the initial delay, quality changes, and stalling with a good performance when using uplink network traffic only. To predict the playback quality with satisfying accuracy, only the usage of uplink data is not sufficient. As trade-off between accuracy and complexity, the downlink duration is added as additional information. In summary and in contrast to literature, either the estimation performance is similar, but complexity is reduced by far or the estimation performs better.*

Table 2.11: Stalling prediction result for full, selected, and uplink feature set (P: Precision, R: Recall, F1: F1 score).

	Full			Selected			Uplink		
	P	**R**	**F1**	**P**	**R**	**F1**	**P**	**R**	**F1**
no stalling	0.99	1.00	1.00	0.99	1.00	0.99	0.99	1.00	1.00
stalling	0.78	0.70	0.74	0.73	0.66	0.70	0.77	0.71	0.74
macro avg	0.89	0.88	0.87	0.86	0.83	0.85	0.89	0.85	0.87
weighted avg	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99

2.6 Lessons Learned

The generated traffic volume by streaming videos is one major challenge for the network and its service providers nowadays. In addition, a steady growth in the number of users, streamed videos, and playback quality is expected in the future. Since video streaming is used by a large percentage of the global population and is responsible for a significant portion of global traffic, good network performance can be identified by a high network quality when streaming videos towards the end-user. Therefore, it is one key goal to provide all customers with a high service quality for their streaming services to avoid user churn. This leads to the need for a constant development in streaming quality quantification.

However, the sole QoS can not determine the perceived quality of end-users. For that reason, the QoE has been defined, including also the subjective opinion about the quality of a service. Thus, the identification of key factors that impair this subjective quality is essential. For video streaming, four main QoE degradation factors, namely, initial playback delay, playback quality, quality changes, and stalling [51] have been determined influencing the QoE in the most negative way. In the past, when video streaming data were transmitted without encryption, the current playback quality and the buffer filling status could be directly extracted from the packet payload by DPI. This was sufficient to determine the main QoE degradation factors. But because of the comprehensive encryption of

video traffic in today's Internet, this is no longer possible and other methodologies are required for network operators or providers to determine the streaming quality in their network. Nowadays, the estimation of QoE degradation factors is often tackled by comprehensive packet monitoring, processing of the monitored data, and ML solutions. This requires an analysis of the arrival time and the header of each single packet in the network and the calculation of a multitude of features as input for the ML. Based on the gathered information, models to predict QoE degradation factors are derived. The output of such prediction models can help network providers to implement intelligent and proactive traffic management strategies to avoid issues and improve the overall QoE for users when streaming video. However, since the number of different streams and the resolution of videos distributed on the Internet is constantly growing, considering each single packet in the network for such QoE degradation factor estimation solutions can consume many resources or is not feasible without very specific hardware. Consequently, the question arises, if the full packet trace of uplink and downlink data is required to predict QoE degradation factors or if partial information like only uplink traffic is sufficient.

Based on these challenges, the following three research questions have been identified and investigated in this chapter.

RQ2.1: What is the possible monitoring and processing effort benefit if only uplink data is considered to predict relevant QoE degradation factors including initial delay, quality changes, playback quality, and stalling?

RQ2.2: Which relevant QoE degradation factors are predictable with a simple uplink based model without relying on a ML solution, and what is the performance compared to state-of-the-art literature?

RQ2.3: Is uplink data sufficient to predict the initial video playback delay, quality changes, playback quality, and video re-buffering events as main QoE degradation factors with a lightweight ML model, and what are benefits, drawbacks, and performance differences in comparison to other solutions with and without ML?

To address the research questions, a large dataset summing up to more than 1,500 h of total playtime has been measured from the native mobile YouTube app. The data is aggregated to uplink requests, relevant uplink based parameters are extracted, and the traffic volume in uplink and downlink direction is quantified both for the measured dataset and an artificially generated dataset describing general videos. Our results show that monitoring only uplink data saves 86 % of the traffic volume in bytes compared to the full packet trace of uplink and downlink data. When we study the required monitoring or processing effort of only uplink data against a full packet trace to estimate QoE degradation factors, the analysis with a Markov model shows the potential of processing 1,000 times more flows in parallel with the same hardware using only uplink traffic (RQ2.1).

The value of an uplink based QoE degradation factor estimation model is shown in this chapter. We propose a very simple and lightweight approach to estimate quality changes to a lower resolution and stalling by leveraging data only in the uplink direction of the traffic and without resorting to ML algorithms. Our fast and efficient audio and video traffic separation is the major improvement to achieve good estimation results. Furthermore, we can use the approach without any expert knowledge by a moving average of recent inter-request times. The system is able to estimate nearly all stallings, up to 90 % of all buffer drops, and more than 95 % of all quality changes to a smaller resolution in the best case. Thus, the estimation performance is similar or better than related literature, mainly using full packet trace approaches (RQ2.2).

Since the uplink-based model requires historic information and does not rely on any downlink, the initial delay and the played quality can not be estimated. In contrast, the simple random forest based prediction approach proposed in this chapter is able to predict all relevant QoE degradation factors with high accuracy. The initial delay estimation works with a mean error of less than 1 s and the quality estimation with F1-scores of more than 0.8. The estimation of high video qualities performs even better with 90 % correct classifications. Nevertheless, for a decent quality prediction, an extension of the uplink only features with the downlink duration of each request increases the prediction accuracy

significantly and is recommended. This small adjustment increases the complexity of the approach only slightly but has a large benefit on the prediction accuracy. Furthermore, in comparison to literature, both models perform better than other approaches with similar complexity and similar or even better than more complex approaches with feature spaces of more than 100 features (RQ2.3).

For a real setup, usable by a network provider, it is recommended to use the simple uplink-data based model for early estimates of a severe quality degradation with minimal overhead. Either, the approach with expert knowledge input can be used, where results show recall values of more than 90 % and a precision higher than 60 % for stalling estimation. Otherwise, the dynamically adapting moving average approach is suitable to detect issues with high performance without requiring input of expert knowledge. The moving average solution makes the approach independent of the streaming player and the behavior of the streaming adaptation algorithm. For practical implementation, the inter-request time thresholds can be learned over time by logging the mean inter-request time if the player is in the steady phase and the variance of the inter-request time is small. This additional information can reduce the false positive rate of the model with little processing and storage overhead and requires no additional expert knowledge. For a more detailed QoE degradation factor estimation, the use of the presented ML approach to make more fine-grained predictions for each single video request is suggested. The findings of the uplink-data based model can serve as additional input, leading to a long term monitoring and management entity. It can be deployed on network nodes where sufficient memory and storage resources are available for a simple ML solution but not enough for full packet trace monitoring and evaluation. With these steps, monitoring, management, but also decision-making can be distributed on hardware with less available resources and thus, move closer to the end-user.

In summary, the studies show that all major QoE degradation factors can be estimated with uplink data. Thus, uplink data only monitoring, processing, and the usage of the presented models can provide network operators enough information to improve their network by improving video streaming QoE.

3 Network Planning of Low Power Radio Access Technologies for the Internet of Things

The increasing requirement for automation and data driven use cases fosters the development of Internet of Things (IoT) solutions. Besides the ever-increasing traffic on the Internet, mainly fostered by high traffic applications like video streaming, the number of connected devices is raising drastically. The IoT will be applicable in, among others, healthcare, edge computing, security, sustainability, wearables, and digital twins [97], and is thus, already or in the close future relevant in every part of our daily life. Although the number of active IoT device connections already surpassed non-IoT connections in 2019 [98], growth rates further increased in the enterprise IoT sector recently [97]. This leads to total expected revenues in the IoT sector of more than one trillion dollar by 2024 [97].

However, IoT is not only about connecting a huge quantity of new sensors using existing Wi-Fi or mobile communication solutions or by the sole use of current 5G networks or future 5G and 6G deployments. The goal is to provide end devices for different use cases with the best suitable access network technology to increase quality including, among others, accessibility, packet delivery rate, resource consumption, and cost. In recent years, many alternatives to traditional mobile access network technologies arose with different characteristics and advantages, but also challenges compared to already available solutions.

In particular, one interesting technology is the family of Low Power Wide

Area Networks (LPWANs) with Long Range Wide Area Network (LoRaWAN) as one of their most prominent and fastest growing representatives. LoRaWAN promises low energy requirements, long possible transmission distances, and a communication that is in general robust against interference due to its exceptionally robust physical layer [99]. It provides comparably cheap devices with little management and maintenance cost, because of battery lifetimes of up to 15 years. Thus, it is the preferable solution over, for example, Narrow-Band (NB)-IoT if cost efficiency is a key metric [100]. For that reason, LoRaWAN is the ideal technology for simple monitoring tasks in the smart city, smart home, metering, and agriculture area, where a vast number of new devices will be deployed in future networks and a long battery lifetime and cheap deployment and maintenance is more important than large data rates.

Nevertheless, LoRaWAN has one major drawback with regard to data transmission quality against traditional mobile communication: no reliable transmission in the network and thus, occasionally message loss. The reason is a large potential for message collisions, leading to data loss due to the random frequency channel access for transmissions. And to operate sensors several years without battery changes and minimize the power consumption in LoRaWAN, frequent re-transmissions, message acknowledgments, or additional communication possibilities between sensors and gateways is limited.

However, from the perspective of a network operator, addressing these limitations and guaranteeing specific network quality through SLAs for their customers is essential. For that reason, different ideas to plan a LoRaWAN from scratch or redesign and improve already deployed networks are crucial. The primary goal is to identify mechanisms that effectively handle message collisions and minimize losses while maintaining, or even improving, energy efficiency. In particular, one idea is to use the expected collision probability within the complete LoRaWAN as one key quality metric during the network planning phase. While existing literature explores the potential gains from utilizing fewer gateways to cover all devices within a LoRaWAN (e.g., [101]), strategic decisions regarding gateway placement have demonstrated their effectiveness to reduce

the achieved collision probability [16].

For that reason, this chapter investigates a novel gateway placement approach for LoRaWAN based on a graph construction solution to provide full sensor coverage and tackle the applicability to arbitrarily large problem instances. Efficient network setups are studied by a simulation to predict collision probabilities as an important quality metric during network planning and the gateway placement phase. We further discuss several placement constraints and their impact on the load in the network and identify the maximal distance between gateways and sensors as major influencing factor leading to message collisions. By comparing the expected collision probability achieved with our placement to related literature, we showcase the improvement against the state-of-the art. Furthermore, different scenarios and transmission patterns are studied by a large scale simulation to cover heterogeneous situations in LoRaWAN and, in particular, analyze the influence of varying transmission rates and message sizes. Finally, the limitation of a high runtime for larger problem instances with different placement algorithms from literature is tackled. To overcome this challenge, pre-clustering of the complete network into smaller problem instances is employed, and clusters are solved individually with the presented algorithm. This step reduces the required processing time significantly, while only a marginal overhead with respect to the number of placed gateways for both artificial and synthetically generated real-world scenarios is achieved.

Based on these considerations, the following research questions are defined and covered in this chapter.

RQ3.1) Is it possible to perform efficient gateway placement for a LoRaWAN based on an abstract, graph-based view of a set of geographically distributed Long Range (LoRa) nodes? Which graph metrics are important and which constraints during graph creation influence the overall collision probability as important quality metric in the network most?

RQ3.2) Is the graph-based approach generalizable for a multitude of different networks, how can such networks be synthetically generated, and is it

possible to compete with state-of-the-art literature?

RQ3.3) How can a graph-based approach be applied to arbitrary large problem instances, and what are limits of scalability?

In the following chapter, Section 3.1 provides fundamental background for LoRa and LoRaWAN with focus on message transmission and quality metrics and summarizes related literature. Afterwards, the general placement idea is presented and the graph based gateway placement is introduced by a five-step approach in Section 3.2. A description of the used simulation and a summary of all evaluation scenarios that are defined to answer the research questions follows in Section 3.3. The evaluation of the scenarios is presented in Section 3.4, starting with an in-depth study of the presented gateway placement, a comparison to related literature, and an investigation of placement solutions for arbitrary large problem instances. At the end of this chapter, Section 3.5 concludes and provides a discussion regarding the scientific contribution and lessons learned. In summary, this chapter contains the following contributions.

C3.1) A simple and efficient graph based gateway placement for initial network planning with focus on collision probability and a reduction of the number of gateways, while providing complete network coverage.

C3.2) A methodology to generate synthetic sensor deployments based on urban building data and a quality improvement for a LoRaWAN compared to the state-of-the-art literature by means of collision probability and the number of required gateways using our graph-based gateway placement.

C3.3) A clustering-based solution for arbitrary large problem instances, dividing the LoRaWAN gateway placement problem into multiple subproblems that can be solved individually without large overhead by means of required number of gateways or collision probability.

The contributions have already been published in the past and are summarized in this monograph based on the following scientific publications.

- Loh, F., Mehling, N., Metzger, F., Hoßfeld, T., Hock, D.: "LoRaPlan: A Software to Evaluate Gateway Placement in LoRaWAN", in International Conference on Network and Service Management (CNSM), 2021 [35].

- Loh, F., Mehling, N., Geißler, S., Hoßfeld, T.: "Graph-Based Gateway Placement for Better Performance in LoRaWAN Deployments", in Mediterranean Communication and Computer Networking Conference (MedComNet), 2022 [17].

- Loh, F., Mehling, N., Geißler, S., Hoßfeld, T.: "Efficient Graph-Based Gateway Placement for Large-Scale LoRaWAN Deployments", in Computer Communications Journal, 2022 [6].

- Loh, F., Baur, C., Geißler, S., ElBakoury, H., Hoßfeld, T.: "Collision and Energy Efficiency Assessment of LoRaWANs with Cluster-based Gateway Placement", in International Conference on Communications (ICC) Workshop on Green and Sustainable Networking, 2023 [30].

3.1 Background and Related Work

This section provides fundamental background information to understand current LoRaWANs. Details on a message transmission with LoRa in a LoRaWAN, key LoRaWAN quality metrics, and influencing factors showing optimization potential for network operators in current LoRaWAN are described. At the end of this section, related work is summarized.

3.1.1 General LoRaWAN Background

The LPWAN modulation technique LoRa uses Chirp Spread Spectrum (CSS) and was developed by Cycleo, later acquired by Semtech in 2012 [102]. The key characteristics of LoRa include long transmission ranges of 2 km – 15 km, a battery

lifetime of up to 10 years for sensor nodes, and low data rates [103]. This physical layer transmission technique is used by the LoRaWAN medium access control protocol. It operates at the license free 868 MHz frequencies in Europe that includes nine distinct transmission channels used for uplink and an additional channel for downlink transmission purpose [104]. Since the focus of this monograph is on optimizing a LoRaWAN in general, and network access in particular, we omit details regarding physical layer interference and frequency hopping during message transmission of different sensors. Thus, in the following, a single channel is considered for the investigation while the possible improvement using multiple frequencies is highlighted very briefly at the end of this chapter.

Although the simple and license free usage of LoRaWAN has many advantages, one major drawback is a high potential for unplanned sensor deployment and channel access. Since currently, a random channel access scheme is used in LoRaWAN [105], there is a constant risk of interference among transmissions. This can lead to collisions and potential data loss. Such an interference of two or more transmitted messages is named message collision in the remainder of this thesis. It occurs, if more than one sensor transmits its data on the same frequency channel at the same time, while being located close enough. If each message collision leads to data loss, the maximal utilization in such an ALOHA-like random channel access environment is 18.4 % [106]. However, the actual utilization is higher due to the robust physical layer of LoRa [99].

Besides relying on the robust physical layer, there is still much optimization potential to decrease message collision, loss, and thus, improve network goodput in a LoRaWAN. In general, the goodput can be increased by a higher message sending rate per sensor or a smaller collision probability leading to less message loss. However, according to [107], the message sending rate is limited by the maximal channel throughput of ALOHA. Furthermore, the transmitted data per sensor is often dependent on its application area or specific tasks. For that reason, the study in this thesis is focused on decreasing the collision probability of LoRa messages. In the following, the most relevant parameters of a LoRa message and its transmission are covered.

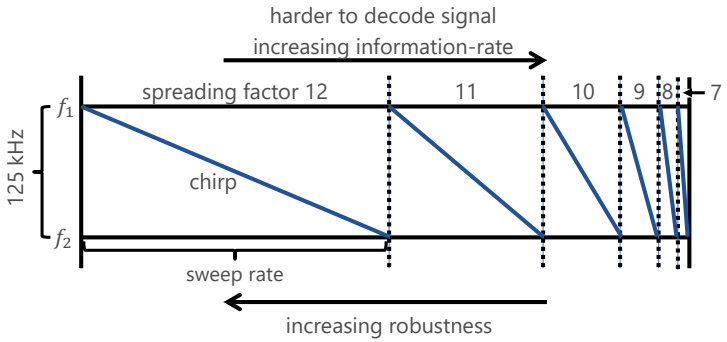

Figure 3.1: LoRa modulation visualization.

Time on Air of LoRa Messages

The most important parameter when studying channel utilization of LoRa transmissions is the Time on Air (ToA). It describes the required duration to transmit a single message, and thus, also the duration one message blocks a channel during transmission. It is dependent on three parameters: the available bandwidth, the spreading factor, and the total LoRa message length. These parameters are described in the following paragraphs in detail.

Bandwidth: Available bandwidths in LoRaWAN are 125 kHz, 250 kHz, and 500 kHz. In this work, the 868 MHz frequency band (EU868) is chosen, as typically used in Europe. It supports 125 kHz and 250 kHz bandwidth, while except for a single LoRaWAN channel, transmission is only possible with 125 kHz [104]. For that reason, we use the 868 MHz frequency band with 125 kHz bandwidth in this thesis, exemplary visualized in Figure 3.1, having 125 kHz bandwidth between frequencies f_1 and f_2. One approach to transmit data is the usage of different frequencies, e.g. using f_1 is equal to 1 and $f_2 = 0$, named frequency shift keying. Instead of only switching between two frequencies, LoRaWAN uses a more robust modulation technique for communication named CSS.

Chirp Spread Spectrum: In CSS, information is transmitted in chirps, shown in Figure 3.1. Up-chirps refer to a chirp from a lower to a higher frequency and down-chirps vise versa. To encode data, chirps are cyclically shifted and their frequency jumps (e.g. from f_2 back to f_1 in this example). The start frequency and thus, the jump point determines the encoded information. Furthermore, the sweep rate is determined by the spreading factor and influences the possible data transmission rate. A higher sweep rate leads to shorter chirps and the possibility to transmit more information in shorter time, as indicated by Figure 3.1.

Spreading Factor and Path Loss: The spreading factor determines the chirp duration. Larger spreading factors lead to longer chirps and an increasing robustness against interference at the cost of longer channel occupancy per transmission. Furthermore, this increasing robustness also increases the possible transmission distance when a larger spreading factor is used. With a larger spreading factor, signals can still be decoded if they arrive with a weaker Received Signal Strength Indication (RSSI) at the receiver. The RSSI is a measure in telecommunication for the power present in a received signal. The signal strength at the receiver can be achieved knowing the sending strength of the device and the path loss between sender and receiver. This path loss describes the reduction of signal power when it propagates from sender to receiver. It is often modeled with a path loss model like the free-space-path loss if a line of sight transmission is possible without any obstacles, or the Hata model [108] for urban areas, as used in this thesis. The path loss is influenced by the distance between the transmitter and the receiver, but also by, among others, environment, vegetation, or urbanization. In contrast, shorter chirps increase the information rate but make signal decoding harder, as visualized in Figure 3.1. In LoRa, information is transmitted in symbols and one chirp maps to one symbol. The different spreading factors determine the number of raw bits a single symbol can carry. For example, for spreading factor 7 (SF 7), one symbol maps to 7 bits. The symbol duration T_s to transmit a single symbol can then be calculated with

Figure 3.2: LoRa message structure.

$$T_s = \frac{2^{\mathsf{SF}}}{\mathsf{BW}}, \tag{3.1}$$

where $\mathsf{SF} \in \{7, 8, 9, 10, 11, 12\}$ is the used spreading factor and BW is the bandwidth for message transmission.

LoRa Message: In LoRaWAN, different message types are available. Besides uplink and downlink data messages, Medium Access Control (MAC) messages, or messages to exchange keys are possible [109]. In this monograph, we focus on uplink messages as main form of data communication in LoRaWAN. Uplink data messages can be confirmed or unconfirmed. While confirmed data messages must be acknowledged by the receiver, unconfirmed data do not need to be answered. Downlink messages, like message acknowledgments, are sent by the network server from a single gateway. For these messages, a separate LoRaWAN channel is available [104]. To calculate the number of symbols that must be transmitted for a single LoRa data message, the structure of a message is introduced in Figure 3.2. A LoRa message consists of a preamble, and four fields for the actual message: an optional header (PHDR), the optional header's Cyclic Redundancy Check (CRC) (PHDR CRC), the actual LoRa payload of a message and a CRC for the complete message [110]. The preamble length n_{preamble} must consist of 8 symbols for the EU868 band, used in this monograph. The radio transmitter adds 4.25 symbols for synchronization summing up to 12.25 symbols [110]. The optional header contains information about the payload size and the message CRC. The header CRC (PHDR CRC) is an optional field only meant to correct or detect errors in the header. In general, the total number of required symbols $n_{\mathrm{sy},m}$ for the LoRa message without preamble can

Table 3.1: LoRa message parameter overview.

Parameters	Variable	Possible range
bandwidth	BW	125 kHz, 250 kHz, or 500 kHz
spreading factor	SF	7 – 12
coding rate	CR	4/5 – 4/8
payload	PL	up to 222 B for SF 7 – SF 8, up to 115 B for SF 9 and up to 51 B for SF 10 – SF 12
cyclic redundancy check	CRC	0 or 1
enabled or disabled header	H	0 or 1
low datarate optimize	DE	0 or 1
preamble length	n_{preamble}	8 symbols

be calculated with

$$n_{\text{sy},m} = 8 + \max\left(\left\lceil \frac{8\mathsf{PL} - 4\mathsf{SF} + 28 + 16\mathsf{CRC} - 20\mathsf{H}}{4(\mathsf{SF} - 2\mathsf{DE})} \right\rceil \cdot (\mathsf{CR} + 4), 0\right) \quad (3.2)$$

Besides header and CRC, each message contains a payload PL and an optional Low Datarate Optimize (DE) for small spreading factors to improve transmission robustness. Furthermore, a forward error correction process adds a different number of redundant bits to transmitted LoRa messages to resolve corrupted data due to interference. These correction bits are used by the receiver to restore corrupted information. Therefore, coding rates CR of 4/5, 4/6, 4/7, and 4/8 are possible in LoRaWAN [111]. For example, for every four bits of useful information, a total of eight bits of data is generated if a coding rate of 4/8 is used. More details about all relevant parameters are given by the Semtech data sheet [112]. Further information on LoRa messages are available from the Things Network [109]. The different LoRa-related parameters are summarized

in Table 3.1 together with its possible ranges. Note that we keep the notation similar to major LoRaWAN literature, e.g. [105, 113, 114], to avoid confusion. The total number of required symbols for a single message together with the preamble and synchronization is then

$$n_{\text{complete}} = n_{\text{preamble}} + 4.25 + n_{\text{sy},m}. \tag{3.3}$$

Thus, with

$$\mathsf{ToA} = T_s \cdot n_{\text{complete}} = \frac{2^{\mathsf{SF}}}{\mathsf{BW}} \cdot n_{\text{complete}}, \tag{3.4}$$

the ToA of a single LoRa message is calculated. As a consequence, the payload and the spreading factor influence the total transmission time if the bandwidth and LoRa packet parameter are kept. This behavior is visualized in Figure 3.3. This example figure shows the required ToA on the y-axis to transmit different payloads on the x-axis with a bandwidth of 125 kHz, a coding rate of 4/8, enabled cyclic redundancy check and header, and disabled low datarate optimization to showcase the influence of the spreading factor and the payload on the transmission ToA. The different spreading factors from spreading factor 7 to spreading factor 12 are represented in different colored and dashed lines. The payload is limited to 1 B – 51 B as maximal range transmittable with all spreading factors. It is visible that a larger payload requires a longer ToA. However, the spreading factor dominates the payload by far with regard to the ToA. For that reason, it is essential to reduce the required spreading factor of LoRa messages to reduce the channel occupancy time. Since transmissions with larger spreading factors are more robust against interference and, thus, allow transmission across longer distances, one option in network and gateway placement planning is to place gateways closer to the sensors. This reduces the required spreading factor, and as a consequence, the network load per transmission. Thus, the interference potential of different devices in the network is reduced and the overall transmission quality in a LoRaWAN is improved.

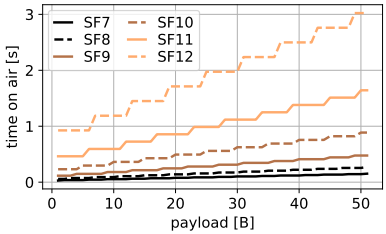

Figure 3.3: Payload and spreading factor (SF) to time on air mapping.

3.1.2 LoRaWAN Quality Metrics and Influencing Factors

In this chapter, we focus on transmission quality improvement in LoRaWAN by intelligent gateway placement decisions. Thus, we aim at deploying gateways in suitable locations dependent on the available end device locations to improve the quality in a LoRaWAN. Therefore, different quality metrics can be inferred that must be taken into consideration for the gateway placement idea, introduced in this chapter. The goal is to optimize these metrics. Naturally, some goals may oppose each other, e.g. coverage and the required number of gateways. In this case, it is important to identify a suitable trade-off that enables reliable system operation. Note, details about energy consumption and energy efficiency are not covered here and are tackled in the next chapter.

Coverage: First and most important, coverage must be guaranteed for all devices in the network to provide high QoS. There are two major ways to increase coverage in a LoRaWAN: increase the possible transmission distance between sensors and gateways, or place more gateways. The possible transmission distance of LoRa sensors is influenced by the sending strength and the used spreading factor at the sensor side, the interference during transmission, and the antenna gain by the receiver. Higher sending strength allows transmission across longer distances and vice versa. The sending strength is highly correlated with sensor quality, battery lifetime, and other factors that can not be adjusted

by a provider. In this work, we assume that sensor locations are immutable in the network. However, the presented approach also works for mobile sensors, considering all possible locations during mobility. Since the interference during transmission is different for different locations and existing development, only the spreading factor can be actively adjusted and is, thus, of major interest. Here, the general idea is that larger spreading factors are more robust against interference and allow transmission across longer distances with the drawback of longer transmission airtime, as already discussed. This leads to a trade-off between transmission distance and ToA.

Number of Gateways: Next, the number of placed gateways and the quality of placement decisions is important. In [16], it is shown that increasing the number of gateways in combination with a good gateway placement can increase the potential number of sensors by a factor of five with only 2.5 times more gateways and a similar collision probability. Thus, besides the costs for additional gateways, the collision probability in growing networks must be taken into account. Furthermore, a denser gateway placement makes LoRaWAN more robust against future load increase [16] if networks scale.

Collision Probability: Finally, the collision probability is a crucial quality factor in LoRaWAN. In this thesis we define each interference between two or more sensors as a collision. It occurs, if multiple devices that are in transmission range of each other access a channel at the same time. Details about the transmission range calculation follow in the methodology section. The collision probability in LoRaWAN depends on the number of sensors in a specific area that can potentially interfere each other by, for example, transmitting to the same gateway, and the sensor's transmission behavior. The transmission behavior includes the message sending rate, the required ToA for messages, and the transmission distance that is equal to the interference distance. However, the number of sensors in a target area can often not be adjusted by the network operator and is thus, not particular focus of this investigation.

LoRaWAN Gateway Setup: Since the transmission behavior of a single sensor is dependent on, among others, the specific sensor type, the use case, and configuration, it can not be modified in many situations. Thus, the collision probability can only be affected if LoRaWAN gateways are placed in an intelligent way to cover fewer sensors. For that reason, the number of gateways has a direct influence on the collision probability. In addition, if more gateways are placed and sensors are located closer to the next gateway, another positive effect is visible. The average distance between gateways and sensors is reduced, which leads to a smaller average spreading factor required for transmission and the average message in a LoRaWAN requires a smaller ToA. This directly reduces the collision probability. On the other hand, the placement of too many unnecessary gateways increase placement cost without an improvement in the network. For that reason, the goal is to find a good trade-off between the initial number of required gateways during placement and the expected collision probability when gateways are placed. Furthermore, a good placement is robust against future load increase and is scalable with the expected future load. However, a good future-proof placement requires a good initial placement and offers extension only where needed [16].

3.1.3 Related Work

Network planing, gateway placement, and other improvement potential in a LoRaWAN is also covered in literature. Since network quality by means of collision probability is studied by a simulation in this chapter of this monograph, this section summarizes related literature with focus on collision probability studies and gateway placement leading to an improved network quality. As related literature also studies gateway placement with different optimization goals, for different networks sizes, and with other general approaches, most relevant and recent literature and the difference to this chapter is summarized in Table 3.2.

An early collision study for LoRa messages was done in 2017 [99]. The authors show that the robust physical layer of LoRa can reduce the collision prob-

ability using random channel access compared to pure ALOHA. Other works study collisions and packet loss in more detail [115–117], develop mechanisms to decode multiple LoRa messages transmitted in parallel [118–120], study inter-spreading factor interference and the influence on message collisions or overall throughput [121, 122], or investigate different spreading factor investigation schemes [123]. To improve the network by reducing message loss, several loss reduction techniques by adding redundancy are introduced [18]. Furthermore, the real time applicability of LoRaWAN is studied in different works [124–126]

One LoRa specific simulator to study the behavior when adjusting the spreading factor based on the current conditions is FLoRa [127]. In addition, theoretical works are available, e.g., [128]. The authors optimize a LoRaWAN and reduce collisions by the assignment of radio frequency parameters through a mixed-Integer Linear Program (ILP) formulation. A theoretical model to calculate packet loss with a uniform spreading factor in relation to the ToA is available in [115]. Another possibility to study collisions in LoRa is an orthogonality study of different spreading factors. This is done by a theoretical examination in [129]. The authors show that it is possible to transmit with two spreading factors simultaneously if the receive-power is comparable. These approaches can perform well if the number of sensors transmitting to a single gateway is limited. This can be regulated by an appropriate gateway placement that limits the geographical area one gateway has to handle, as we will discuss in this chapter.

The idea of demand based gateway placement dates back to mobile network studies in 1998 [138]. Since then, many researchers have studied placement approaches based on network load (e.g., [139, 140]). However, these approaches could only steer available load in a network with static or mobile devices. In contrast, the idea discussed in this chapter of this monograph directly influences the network load by adjustments on the average distance between sensors and gateways and thus, the network setup. These distance changes directly influence the spreading factor in LoRaWAN and the transmission duration of LoRa messages. Besides load steering and network quality improvement by gateway placement decisions, the solution in this chapter goes one step further. It im-

Table 3.2: Overview of select related work.

Reference		Optimization goal	Max. network size	Approach type	Approach details
Mikhaylov'20	[130]	real-life multi-gateway network operation	$>$ 20 gateways, 231 test locations	measurement	distance and radio quality study
Cruz'22	[131]	coverage, number gateways	2 devices, 957 locations	measurement and model	propagation model and genetic algorithm
Ousat'19	[132]	PDR, number gateways, energy efficiency	small networks	model	mixed-integer non-linear optimization problem
Citoni'21	[133]	PDR	2 gateways, 5,000 sensors	simulation	discrete-event simulation with ns-3
Matni'20	[101]	PDR, cost	25 gateways, 1,000 sensors	simulation	k-means and fuzzy c-means clustering
Jin'20	[134]	packet reception rate	3,000 sensors	simulation	constrained clustering
Mnguni'21	[135]	number gateways, PDR	2 gateways, 700 sensors	simulation	transmission simulation with FLoRa
Da Silva'22	[136]	RSSI, signal-to-noise ratio, delay, distance, cost	1,000 sensors, 100 km^2	simulation	fuzzy c-means, Gustafson-Kessel, k-means
Abakar'22	[137]	packet length, number gateways, PDR	5 gateways, 1,000 sensors	simulation	network simulation with ns-3
Loh'21	[16]	number gateways, collision probability	28,000 sensors	model and simulation	local search and ILP
This chapter	–	number gateways, collision probability	20,000 sensors	model and simulation	clustering, graph centrality metric calculation

proves the network quality and reduces the message collision potential as a result of network load reduction by transmission airtime reduction of messages. This is also proven as viable solution to increase PDR by Citoni et al. in [133]. The authors present an ns-3 based simulation for a much smaller setup of two gateways studying QoS for LoRaWAN. Their advice is to avoid placing gateways too close to each other, but also suggests avoiding spreading factor 11 and 12, as PDR drops by 7 % and 10 % for both spreading factors respectively.

In contrast, available LoRaWAN gateway placement approaches in literature follow in general the goal of gateway reduction for a given area or network. Matni et al. use k-means and c-means approaches [101] while other clustering approaches are studied in [134]. Mnguni et al. study LoRaWAN gateway placement by simulating a network with the FLoRa simulator [135]. They show a coverage potential of two gateways for a 10 km dense urban area. Cruz et al. investigate gateway placement with the target metrics coverage area and the number of gateways with Monte Carlo simulation methods [131]. A comprehensive survey of this research area is conducted by Mnguni et al. in [141]. Another approach is the usage of mixed-integer non-linear optimization by Ousat et al. [132]. However, their approach only works for small networks. A comprehensive measurement study with an existent placement and 20 gateways at 231 test locations is conducted in [130], and cell capacity is already studied in [142]. To achieve results about the optimal placement strategy based on different quality indicators like signal strength, delay, distance, and cost, da Silva et al. compare fuzzy c-means, Gustafson-Kessel, and k-means algorithms in [136]. Although they investigate the distance between sensors and gateways, they do not analyze this influence on the network quality and in particular, the collision probability. Correia et al., in contrast, target energy consumption as main investigation focus in their work [143]. The authors investigate different clustering algorithms used to place gateway and cover an agricultural region. Furthermore, a general tool to evaluate existing gateway placement decisions for LoRaWAN is presented in [35]. Abakar et al. studies the usage of multiple gateways in LoRaWAN, and its influence on PDR when different payload lengths are used [137].

They study the influence of confirmed and not confirmed traffic. They conclude that PDR decreases for more gateways when larger payloads are used in high traffic situations. To achieve their results, they use a simulation and vary the number of gateways between one up to five gateways at fixed locations.

The drawback of most LoRaWAN gateway placements in literature is the focus on minimizing the number of gateways. But without taking the collision probability into consideration, a general improvement of the transmission quality can not be guaranteed. This limitation is tackled in [16], where the number of sensors per gateway and the maximal possible distance between sensors and gateways are used as constraints. Thus, the collision probability in the network can be reduced significantly. However, their gateway placement approach is not optimal. There is still potential for placing a gateway at the edge of the network or a different number of gateways for different runs on the same network. Furthermore, long runtimes for the suggested ILP are not feasible for large networks and the ILP only provides optimal results based on the input constraints.

We solve these issues in this chapter by combining the benefits of a clustering and a graph-based approach, where gateways are placed in network centers, with high sensor density and further limit the placement with additional constraints to keep the expected collision probability in the network low. Our placement can plan a LoRaWAN from scratch but also optimize existing placement if future networks grow and more sensors need to be covered.

3.2 LoRaWAN Gateway Placement

In this section, the general methodology to set up a LoRaWAN is discussed, and the gateway placement approach is described by introducing the gateway placement idea first. Afterwards, the process of describing a LoRaWAN as a graph as basis for the placement is introduced. Then, the usage of different constraints is highlighted and the actual placement approach is presented.

3.2.1 Placement Idea

Currently, each sensor is transmitting data using random channel access in a LoRaWAN. Thus, the number of sensors in one cell and the channel occupancy time for each sensor are the most dominant factors influencing the collision probability, and thus, the transmission quality. Network improvement by adjusting the number of sensors in specific areas or the transmitted payload per sensor is often only dependent on the application and sensor distribution. Then, one possibility to reduce the collision probability is to manage the channel occupancy time and the number of messages transmitted per cell via, for example, intelligent gateway placement. In traditional mobile networks, in current 4G networks, and also in current and future 5G deployments, a single transmission with the same payload from the same sensor located in a specific cell requires in general the same channel occupancy time. Specifically, this is independent of the distance to the gateway if the propagation delay is neglected. Thus, the number of sensors that can be covered by a single gateway is limited by available frequencies and the transmission behavior and rate of sensors.

However, the situation is fundamentally different in LoRaWAN. Despite the available frequencies and the sensors' transmission behavior, the spreading factor is influencing the required channel resources. Sensors transmitting with larger spreading factors can transmit across longer distances with the acceptance of a longer ToA for messages and longer channel occupancy per transmission. For that reason, it is possible to reduce the required channel resources by reducing the average spreading factor, and thus, the distance between sensors and gateways for transmission. For that reason, it is generally more efficient to place a gateway in the middle of a dense network part with many sensors and avoid splitting these parts in separate cells. This reduces the average distance between sensors and gateways in the network and the average required spreading factor. Therefore, the idea for an efficient gateway placement is the identification of dense parts in a LoRaWAN first, and place the gateways at suitable locations in these dense areas afterwards.

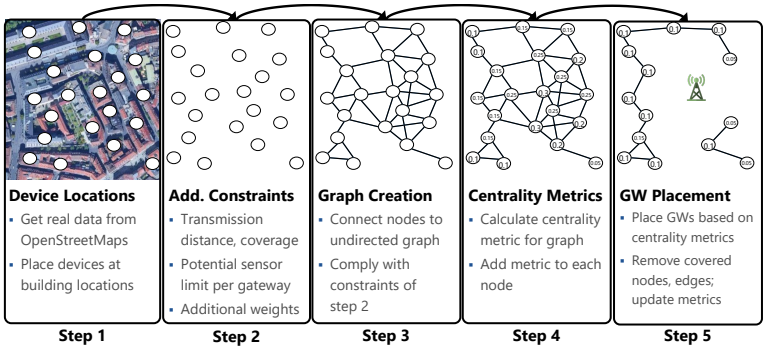

Figure 3.4: Overview of network setup: graph creation and gateway placement.

3.2.2 Network Setup Procedure

The network establishment and the gateway placement procedure in this work contains five steps. This procedure is introduced in the following, starting with the establishment of sensor locations and ending with the fully planned LoRaWAN, including gateway locations. A step by step visualization of this approach is presented in Figure 3.4 and in [17].

Step 1: Sensor Locations based on Urban Building Data: To represent a real world LoRaWAN, possible locations for sensors in the network must be determined. It is possible to use real sensor locations or synthetically generate locations for a network. In this thesis, we derive the centroids of buildings of different real cities from OpenStreetMap to generate the network. For each building, an x- and y-coordinate is obtained, augmented with an ID, and the resulting object is used to represent a single sensor. We assume that each sensor is a potential gateway during the following gateway placement to achieve optimal gateway locations for the underlying network of sensors. This approach has two benefits to display more realistic scenarios. First, more densely populated regions receive more sensors and second, the placement is not optimized for

synthetic, evenly distributed scenarios but for different real city deployments. However, please note that a mapping of sensors to other entities like trees, specific buildings, other infrastructure, or real data is also possible.

Based on the actual network size by means of geographic distribution and the number of sensors, and the available processing resources and acceptable runtime of the placement algorithm, optional pre-clustering can be performed. We have chosen k-means clustering because of the simplicity, the good performance, and the good results. In contrast, density based clustering solutions like DBSCAN perform worse in very heterogeneous networks, where in specific areas sensor locations are very dense and in others very sparse. Furthermore, the only goal is to divide the network into smaller parts in this pre-clustering step. Then, the same placement approach can be applied individually to all clusters according the divide and conquer scheme.

During pre-clustering, the value k defines the number of independent clusters, and thus independent LoRaWAN gateway placements. At the end, clusters are re-combined into a single network. With this divide and conquer approach, the following benefits are achieved: the algorithm is much more flexible and can be adjusted towards different network sizes, available processing resources, and acceptable processing times. Since intelligent pre-clustering already uses an available sensor distribution to create clusters, only little overhead is expected.

In particular, additional gateways do not create additional traffic, and in the best case reduce the required spreading factors of single sensors. This further reduces the average airtime of transmitted messages and in return, reduces the collision probability. Please note that the gateways in this example do not create traffic since no messages are acknowledged. In reality, the gateways could use one specific channel for message acknowledgments or the dedicated downlink channel to not interfere with any sensor. Since gateways are always online and powered systems, they can coordinate between other gateways to avoid collisions of their downlink messages.

Step 2: Placement Constraints: Placement constraints are important to first, achieve a valid placement and second, improve the achieved result. Therefore, three constraint classes are defined in the following: class 1 must be satisfied. The only class 1 constraint in this work is coverage of all sensors. Class 2 constraints are variable parameters in this chapter with the goal of reducing the collision probability in the LoRaWAN with the minimum possible number of gateways. Class 2 constraints are the maximal distance between gateways and sensors and the number of sensors a gateway can process. Last, class 3 constraints could be additional information in the network like transmission characteristics of specific sensors, sensor importance, or message importance. These constraints can be added by additional weights during graph creation but are omitted in the following, since all sensors are expected to have the same importance. Furthermore, sensor transmission patterns are not controllable by network operators and are often not available in the planning phase. Nevertheless, differences in the transmission behavior can have an influence on network load and should, thus, be considered when planning a LoRaWAN. However, with the following placement approach, different load can be modeled with more sensors at specific locations and vice versa.

Besides network load, adjusted with a different number of sensors, the ToA of each single message has a direct influence on the load. It is directly influenced by the distance between a sensor and its gateway and thus, the used spreading factor. Since it is one of the most important constraints in this work, it is described in detail in the following. It is achieved by calculating the possible transmission distance of the sensors.

Possible Transmission Distance: In particular, the possible transmission distance is, due to the technical relation of transmission distance and spreading factor, a major contributor to the collision probability. In this work, the possible transmission distance d_{tr} from transmitter Tr to receiver Re is determined with the urban version of the Hata propagation model that is widely used in mobile communication [144, 145]. Hence, we compute the transmission distance as

$$d_{\mathrm{tr}} = 10^{\frac{(-(69.55+26.16\log_{10}(f)-13.82\cdot\log_{10}(h_{\mathsf{Re}})-a(h_{\mathsf{Tr}})-\mathsf{L})}{(44.9-6.55\cdot\log_{10}(h_{\mathsf{Re}}))}} \tag{3.5}$$

for the frequency $f = 868\,\mathrm{MHz}$ with

$$a(h_{\mathsf{Tr}}) = 3.2 \cdot (\log_{10}(11.75 \cdot h_{\mathsf{Tr}})^2) - 4.97 \tag{3.6}$$

and L as maximum tolerable path loss for the connection. Thus, since larger spreading factors tolerate higher path loss, or in particular require a lower RSSI limitation to decode signals according to Table 3.3, longer transmission distances are possible. The tolerable path loss is further dependent on the sensor's, and in particular its transceiver's sending strength and the antenna sensitivity. For our study, we use an SX1276 transceiver [112] as reference with an antenna gain of +8 dBm that can generally be achieved by many common gateways [146]. The transmitter height h_{Tr} is set to 15 m and the receiver height h_{Re} to 1 m. This is done to not cover the complete studied area with only a single receiver and show the potential of the presented approach. Since the height values are only used as input for the Hata path loss model, any other heights work accordingly. Other input would lead to different maximal transmission distances with different spreading factors and thus, a different number of required gateways. However, this does not influence the gateway placement procedure that places more gateways in denser locations or the general statement of this chapter of this monograph. Since the Hata path loss model is only an example model, the same holds true using other path loss models, as we investigate for example with our network planning tool in [35]. The requirement is the possibility to determine distances between sensors and gateways for any topology.

Step 3: LoRaWAN Graph Creation: In the graph creation procedure, the first step for each cluster, if multiple clusters are available, or the complete LoRaWAN, is to express all sensors and gateways as nodes n_i while the complete set of nodes in the LoRaWAN is expressed as N_{all}. Each node $n_i \in N_{\mathrm{all}}$ has

Table 3.3: RSSI and max. transmission distance for diff. spreading factors (SF).

SF	RSSI	Distance	SF	RSSI	Distance
SF 7	-131 dBm	971.08 m	SF 10	-140 dBm	1,695.16 m
SF 8	-134 dBm	1,169.24 m	SF 11	-141 dBm	1,803.41 m
SF 9	-137 dBm	1,407.85 m	SF 12	-144 dBm	2,171.44 m

an x- and y-coordinate to determine its location. Afterwards, for all pairs of nodes $(n_i, n_j) \in N_{\text{all}} \times N_{\text{all}}$, an edge e_{ij} connecting n_i and n_j is added if n_i is within reach of a LoRa transmission from n_j and vice versa. Thus, the LoRaWAN can be expressed as an undirected, unweighted or weighted graph. The requirement of an undirected graph is important for a real network to ensure that receivers (the gateways in the following) could also transmit to transmitters (the sensors in the following). This is guaranteed in this consideration since the possible transmission distance from transmitters to receivers is smaller than vice versa. The graph can be unweighted if each sensor has the same behavior, importance, or transmission characteristics, or weighted if additional characteristics are included in the gateway placement decisions. The result of the graph creation is not necessarily a connected graph, for example if at least one sensor that is out of range of all other sensors exists. However, the following approach can identify gateway locations both in a connected and a not connected graph.

Step 4: Graph Centrality Calculation: Next, it is important to determine good locations to place gateways. As discussed in Section 3.2.1, for LoRaWAN specifically, dense network parts should be preferred to place gateways if all other placement constraints can be met. Since both the degree centrality and the betweenness centrality show good results to describe node centrality in literature [147], they are considered as gateway placement metrics in the following. To identify the usability of both metrics, we elaborate in [17] its complexity and performance for the gateway placement problem. We see that placement with degree centrality as graph metric requires fewer gateways for the same or better

network quality by means of collision probability. Furthermore, the complexity to calculate betweenness centrality is $O(n^3)$ for dense networks [148] with n as the number of nodes in the network. In contrast, the complexity to calculate degree centrality is $O(n^2)$. For that reason, the degree centrality is chosen as graph metric in the following. Therefore, for all nodes $n_i \in N_{\text{all}}$, the degree centrality $C_D(n_i)$ is calculated according to [147] with

$$C_D(n_i) = \frac{\text{Deg}(n_i)}{|N_{\text{all}}| - 1}. \tag{3.7}$$

$\text{Deg}(n_i)$ is the node degree of node n_i, achieved by summing up the number of outgoing edges from node n_i. Consequently, the degree centrality $C_D(n_i)$ is a measure for the importance of a node in the network. It is an indicator for the number of other nodes in close distance and, in this chapter of this monograph, in possible transmission but also interference distance.

Step 5: LoRaWAN Gateway Placement: The input for the gateway placement is one graph or multiple graphs if several clusters exist that represent the sensor locations as defined in step 3 and the calculated graph metrics from step 4.

Gateway Selection: If a fresh placement without already existing gateways is done, the node with the highest degree centrality is selected as next gateway. If several nodes have the same degree centrality, the node with the lowest ID is chosen. However, it is observed that random selection does not statistically significantly change result quality. Furthermore, choosing the node with the lowest ID makes the algorithm repeatable and reproducible. If a placement in an already existing network is done, first all already placed gateways must be considered. Thus, it is iterated over all existing gateways first, and one of them is chosen in this step and the graph is updated, as described in the following. This is done to not consider already covered sensors by existing gateways when new gateways are placed. If no existing gateway is left, the remaining node with the highest degree centrality is selected.

Graph Update: In the graph update step, the selected gateway node and all sensor nodes that are covered by this new, or already existing, gateway based on the constraints defined in step 2, as well as all edges that connect to at least one of these nodes, are removed from the graph. Please note, if the distance between the selected gateway and any of the nodes having an edge to that gateway exceeds the maximal transmission distance of LoRa sensors, these nodes are kept for the next gateway selection step and are not removed from the graph. This occurs if the distance limit set in the graph creation is larger than the possible transmission distance of LoRa nodes. The special handling of these nodes guarantees coverage in the final gateway placement for the LoRaWAN.

Metric Update: The algorithm terminates with a valid placement when all covered nodes are removed and no more nodes are available. Otherwise, the graph metrics of the remaining graph are updated, and the placement algorithm continues with the next gateway selection. The quality of a placement is steered by the possible transmission distance, as maximal distance between two nodes that receive an edge, and the possible number of sensors a gateway can cover, as maximal number of edges per node in the initial graph establishment.

Cluster Combination: If the problem instance is initially split into multiple smaller deployments in the first step, all calculated clusters and placement results are combined at the end to one single network. The number of required gateways in each sub-problem is summed up to achieve the total number of required gateways for the complete network. Afterwards, the quality of the network is investigated by a simulation study.

3.3 Simulation Approach and Scenario Definition

This section describes the applied simulation approach to evaluate the presented graph-based gateway placement. Based on sensor and gateway locations, the collision probability simulation is described in detail in the following. In the scenario definition at the end of this section, different constraints are defined to optimize and study gateway placement decisions based on the presented graph creation. The quality of the achieved placement is determined by the expected collision probability in the network and the number of required gateways.

3.3.1 Methodology to Evaluate Gateway Placement

To study the performance of the described graph-based gateway placement, a lightweight discrete event simulation is programmed in Python. Each transmitted message start and end timestamp is an event in the simulation, changing the state of the system from *channel free* to *channel occupied* and vice versa. If the channel is occupied by multiple messages at the same time, and the transmitting devices are in range of each other, a message collision occurs. Details about sensor and gateway locations, transmission range, and message collision are described in the following.

Simulation Input: The input for the simulation is a set of sensors S and gateways G, achieved with the presented gateway placement approach. Each sensor and gateway has an x- and a y-coordinate to describe its location. The graph-based gateway placement provides one target gateway for each sensor. The goal for each sensor in a LoRaWAN is then to transmit to the closest gateway with the smallest possible spreading factor to save ToA.

Transmission Distance Calculation: Based on the gateway and sensor locations received from the graph algorithm, the distance to the closest gateway is calculated for each sensor first. With the possible maximal transmission distance from the Hata path loss model summarized in Table 3.3, each sensor receives a

spreading factor to transmit with. This means, a sensor transmitting with one specific spreading factor from the table can transmit to gateways or interfere transmissions of other sensors in that specific range.

LoRaWAN Parameter Setting: The calculated transmission distance describes the distance a sensor potentially interferes other messages. To calculate the collision probability in the network, the possible duration of each interference is also required. Therefore, we calculate the ToA of the transmitted LoRaWAN messages. The parameters used in the following are in accordance with default LoRaWAN from standardization. The possible ranges were already introduced in Table 3.1. To simulate the transmission of messages, the header is enabled, and low data rate optimization is disabled for all spreading factors to keep consistency. Furthermore, a coding rate of 4/5 is applied to limit the redundancy overhead. To obtain reproducible results and to isolate the impact of placement decisions on the observed collision probabilities, 16 B payload is used. Thus, for each message, the ToA can be calculated described in Equation 3.4.

Interference Detection: In the next step, it must be determined which sensors potentially interfere with each other. Therefore, an interference list is calculated for each sensor $s_i \in S$. We assume that each transmission is propagated circularly around the transmitting sensor. All other sensors $s_j \in S \backslash s_i$ in the interference radius of s_i are added with the respective ToA if at least one of three interference cases occur. All cases are visualized in Figure 3.5 for an example sensor s_1, interfered by s_2. The first interference case for a sensor s_i (case 1) includes all sensors, where s_i is in their direct transmission radius with their specific spreading factors according to Table 3.3. This is visualized by the first case in Figure 3.5, where the black transmission radius of s_2 overlaps s_1 and vice versa. Thus, both sensors interfere each other in the orange shaded interference area. The sensors do not necessarily need to transmit to the same gateway, as pointed out by the red arrows symbolizing their transmissions in opposite directions. In the second case, all sensors transmitting to the same gateway as s_i

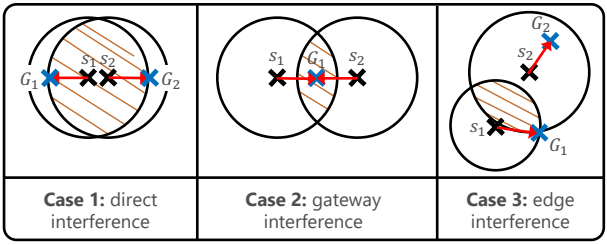

Figure 3.5: Transmission interference cases of two sensor nodes s_1, s_2 and gateways G_1, G_2.

interfere s_i. This is the case if the gateway is in the interference area of the sensors, as visualized by case 2 in the figure. There, the sensors do not necessarily need to hear each other but can interfere each other (hidden node problem). In case 3, all sensors, where the transmission radius is intersected by the direct line of sight transmission from s_i to the closest gateway from s_i interfere (exposed node problem). In this edge case, the transmission of s_1 to its gateway G_1 only intersects the orange shaded interference area. Note that if the network has been set up by pre-clustering, each sensor is transmitting to the closest gateway, even if it is not in the same cluster. The interference list calculation is always done for the complete network. In the following, we use this information to calculate the collision probability.

Collision Probability Calculation: The actual collision simulation is performed individually for each sensor $s_i \in S$ based on the sensors in each sensor's interference lists from the interference detection step above. This scales if networks are not extremely large since interference lists are comparably small in good network deployments to keep collisions probability low. Furthermore, since the collision computation with this approach is independent for each sensor, it can be parallelized to deal with larger problem instances. Two messages m_1 and m_2 collide in a LoRaWAN channel when the transmission interval I_1 of

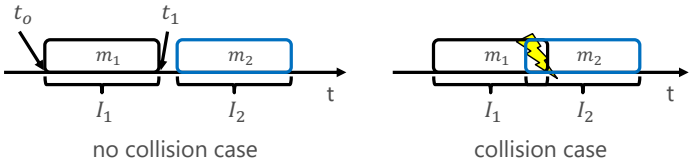

Figure 3.6: Visualization of the message collision case.

m_1 and I_2 of m_2, defined by the transmission start timestamp t_0 and the transmission end timestamp t_1 overlap. A visualization for no collision (left) and the collision case (right) is presented in Figure 3.6. For each sensor $s_i \in S$ in the network, we receive a number of sensors k_s with the respective ToA potentially interfering with s_i from the interference detection above. Based on the transmission rate of all sensors, and in particular of the sensors in the interference list of each sensor, we get the number of messages per time frame that must be simulated for each individual sensor.

We set the transmission rate to one message per investigation interval per sensor and set the investigation interval to one hour. Note that the following simulation works accordingly for different investigation intervals. Smaller investigation intervals are equal to a higher transmission rate per sensor and vice versa. Later, to emulate more sensors in the network, we adjust the number of messages transmitted from one sensor location per investigation interval while different messages from the same sensor location are independent of each other. This means, different messages from the same sensor location can interfere with each other. This scenario is equal to having more sensors in the network and emulates an increasing network in the future. Note, this approach is chosen to simulate a larger geographic area and study the impact of gateway placement decisions. For that reason, the focus is not on the study of small geographic networks with many sensors deployed in, for example, single Industry 4.0 production sheds or in Smart City deployments in skyscrapers. Thus, this is not covered in this work.

Since the collision probability simulation process is the same for each individual sensor $s_i \in S$, we describe the simulation for one sensor $s_1 \in S$ exemplary in the following. The input for the simulation is the ToA of sensor s_1 and all sensors in the interference list of s_1 achieved above. First, we calculate the transmission start timestamps for the transmissions of s_1 and all sensors in its interference list. As we assume random channel access, we assume the transmission start timestamps as uniform random numbers between 0 s and 3600 s to simulate an investigation interval of one hour and thus, one transmission per sensor per hour. This is valid for a sufficiently large number of sensors according to Metzger [149]. With the same approach, we can also cover non-periodic transmissions of single sensors if the total number of messages per time interval is not changing. Adding the ToA of each sensor to the transmission start timestamp, we receive the transmission end timestamps and the transmission intervals for all sensors. This is visualized in Figure 3.6 for example messages m_1 and m_2 with transmission start timestamp t_0 for m_1, transmission end timestamp t_1, and transmission interval I_1 for m_1. If this transmission interval of sensor s_1 overlaps in time with any transmission interval of any sensor in the interference list of s_1, s_1 collides. We repeat this process for each sensor $s_i \in S \backslash s_1$ with its respective interference list to achieve the collision probability in the LoRaWAN.

Since we only simulate transmissions and calculate the collision probability, we do not manipulate any transmission after it is started or any other behavior. For that reason, we can simulate a sensor with its complete interference list in one step by calculating transmission start and end timestamps for the complete simulation duration at once. This generates much less overhead in contrast to more general LoRa simulators like FLoRa [150] or different ns-3 [151, 152], or SimPy [153] based simulators. Because of the randomness of transmission start timestamps, we simulate 100 hours with each sensor transmitting randomly once per hour to study the collision probability in the network and achieve statistically significant results. Note that overlapping transmission intervals always lead to collisions, independent of the used spreading factor and whether the sensors are in the same cluster if pre-clustering is performed during

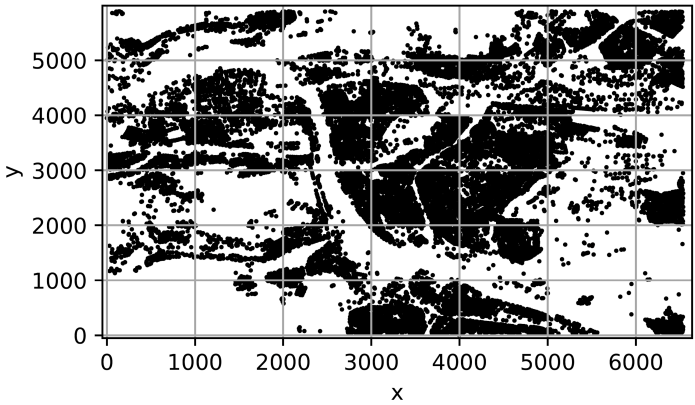

Figure 3.7: Example visualization for sensor locations for Würzburg, Germany (each dot represents one example sensor).

gateway placement. We did not include potential quasi-orthogonality of spreading factors as it is for example done by Caillouet et al. in [154]. Thus, this study can be seen as the worst case investigation.

3.3.2 Scenario Overview

To evaluate the performance of our graph-based gateway placement and answer the research questions, we study different scenarios with varying network load and sensor setups. Therefore, the sensor locations and the different scenarios are introduced in the following.

Sensor Locations: To study the performance of our gateway placement, different sensor location sets in different cities are obtained from OpenStreetMap. First, we obtain 29,113 sensor locations for Würzburg, Germany by mapping building data to sensor locations, as presented in [16]. An example overview of the sensor locations normalized to x- and y-coordinates is visible in Figure 3.7

Table 3.4: Overview: number sensors for different cities.

City	Country	District	Number sensors	Nodes per km^2
Würzburg	Germany	complete city	10,000	114.0
London	UK	City of London	1,959	412.0
Munich	Germany	Schwabing-West	3,094	489.0
Shanghai	China	Pudong	17,210	5.8
Sydney	Australia	City of Sydney	1,058	193.1
Bangkok	Thailand	complete city	14,443	4.7
New York City	USA	Manhattan	11,521	46.2
San Francisco	USA	complete city	20,048	19.2

(edges are cut in this example for better visibility). To study the performance of the graph-based placement and different parameters, but also limit computation time without excluding specific areas of the city or modifying the general sensor distribution, we select 10,000 sensors randomly from the sensor set. By using different randomly selected sets of 10,000 sensors from the full dataset, it is observed, similar to [16], that the randomness of sensor selection has no statistically significant influence on the result. Thus, it is not presented in detail hereafter. Furthermore, since reducing the number of sensors is equal to reducing the overall load in the network, only a decrease in general collision probability is expected but no difference for the general result. This is validated by increasing the load later in the evaluation (scenario S 4).

In addition, seven sensor sets for different global cities are studied to analyze the usability of the approach in other, differently urbanized areas. An overview of all cities is given in Table 3.4. Furthermore, different city sizes, districts, and urbanization is chosen during city selection. Note that for Bangkok and Manhattan, only 50 % of the available buildings are randomly used as sensor locations and for San Francisco, only 33 % of all buildings are chosen to limit the size of the networks and study differently dense and sized large deployments.

Scenario Definition: Based on these considerations, five scenarios are defined to place gateways in a LoRaWAN, summarized in Table 3.5 with different research goals. For each scenario, the maximal distance between any sensor and a gateway, and the maximal number of sensors per gateway must be defined for the graph creation and the placement. Afterwards, a collision probability simulation is performed for each scenario, as described above. Only placements are considered without any pre-placed gateways. The gateway placement process of additional gateways in already existing networks is similar but shows never better results than a completely new placement, as already discussed in [16]. Scenarios S 1 - S 4 are used to evaluate our gateway placement approach without pre-clustering to study its general performance. Then, Scenario S 5 evaluates whether a network pre-clustering can overcome computational limits without significant overhead in the number of gateways or the collision probability if network sizes increase in the future. Furthermore, the outcome and performance of the pre-clustering on artificially generated networks is investigated. Each scenario is designed to answer one of the following questions.

Scenario S 1: What is the optimal sensor to gateway distance limit used as a constraint in a LoRaWAN gateway placement?

Scenario S 2: What is the ideal maximum number of sensors covered by a gateway during the graph creation process, and how does this limitation impact the collision probability?

Scenario S 3: How does the presented approach perform in different environments and in comparison to the state-of-the-art?

Scenario S 4: What is the impact of network load changes as a result of different payloads or transmission rates on the collision probability, and how does the collision probability change if networks grow in the future and additional gateways must be placed?

Scenario S 5: Is it possible to overcome limits in computational scalability by splitting the network into multiple smaller instances and perform independent gateway placement? What is the resulting overhead by independently solving the placement problems?

Table 3.5: Graph based gateway placement scenario overview.

Sc.	Max. distance sensor to gateway	Number sensors	Research goal
S 1	300 m – 2,600 m step size 50 m	1,000	Achieve optimal sensor to gateway distance limit
S 2	2,171.44 m	300 – 3,000, step size 50	Achieve optimal sensor per gateway limit
S 3	according to the best performance	750	Study approach for different locations and compare to related work
S 4	different settings	1,000	Study different transmission patterns for load increase
S 5	2,171.44 m	1,000	Placement for large deployments

3.4 Numerical Evaluation

This section summarizes the results conducted during the gateway placement scenario study summarized in Table 3.5. Each scenario is tackled in detail in the following, starting with scenario S 1.

3.4.1 Scenario S 1: Distance between Sensors and Gateways

The first scenario S 1 varies the distance between sensors and gateways in the graph creation and keeps the maximal number of sensors at a fix number of 1,000 to exclude further influence of this parameter. This value showed to be large enough to not influence placement decisions but also avoid unnecessarily many sensors transmitting to a single gateway in very dense network parts leading to high collision probability anyway. The goal of this study is to find an

optimal sensor to gateway distance for our graph creation, as larger sensor to gateway distances require larger spreading factors. This leads to a higher collision probability because of a larger ToA for each message.

Spreading Factor: The first investigation is the spreading factor usage based on the set distance limit to the next gateway. Figure 3.8 shows the result as the percentage of sensors using a specific spreading factor on the y-axis based on the distance limit between sensors and gateways on the x-axis. The colors indicate the spreading factors. The vertical dashed white lines show the distance limits for the usage of different spreading factors with spreading factor 7 on the lower end up to 971.08 m and spreading factor 12 to the right with up to 2,171.44 m. Table 3.3 shows the exact values for all spreading factors. Note that the study is conducted up to 2,600 m, which is outside the reach of a sensor transmitting with spreading factor 12. This setting is chosen to study larger maximal distances during graph creation and can influence the graph centrality metrics and the final placement of gateways. However, as described in the graph setup, the final gateway placement guarantees a maximal distance from every sensor to the next gateway of 2,171.44 m, so that it can transmit data with at least spreading factor 12. Thus, the coverage constraint is still fulfilled. The figure shows more large spreading factors shortly before each dashed line and more small spreading factors again afterwards. This is especially visible before the third and the last dashed line. As we aim on using less large spreading factors, these results show that it is not advisable to perform a gateway planning with the exact distance limits of a spreading factor, especially visible for spreading factor 12 and the last dashed line. There, a distance limitation of 2,200 m performs much better than 2,150 m, although the distance limit to transmit data with spreading factor 12 is 2,171.44 m. In this case, the *coverage*-constraint that guarantees a gateway for each sensor in a maximal distance of 2,171.44 m in combination with a distance limitation of 2,200 m gives the placement more flexibility than the hard distance limit of 2,150 m alone. In this case, either an additional gateway is placed or gateways are placed better without that strict spreading factor limit.

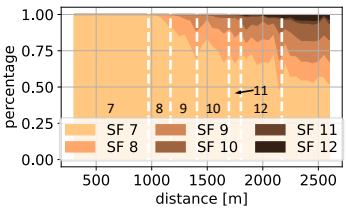

Figure 3.8: Spreading factor (SF) distribution for different distance to gateway settings.

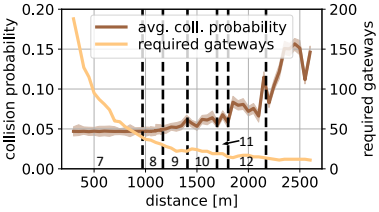

Figure 3.9: Collision probability and required number gateways for different possible distance.

Collision Probability: Since the spreading factor distribution can only draw conclusions on the distance distribution between sensors and gateways in a LoRaWAN, it is not possible to describe the performance of the gateway placement with any network quality metric by using the spreading factor only. Therefore, the collision probability is achieved as described in Section 3.3.1. However, the average collision probability presented in Figure 3.9 shows similar results as the spreading factor before. The brown line shows the collision probability on the left y-axis of the figure, based on the maximal allowed distance between sensors and gateways on the x-axis. The shaded area around the line indicates the minimal and maximal collision probability observed for all 100 runs. This visualization is chosen since even the 99 % confidence interval is very narrow and hence, not visible. Based on this, we conclude that visible collision probability differences presented by the brown line are statistically significant with a 99 % confidence. The yellow line presents the number of required gateways on the right y-axis for different distances. The dashed black lines again indicate the spreading factor distance limits.

The brown line shows an average collision probability of less than 5 % when the distance is below 1,150 m. This is equal to a maximal spreading factor of 7 or 8. An increase in collision probability is visible for larger distances, and in particular larger required spreading factors. However, shortly before the maximal distance is equal to the next spreading factor distance limit, an increase in the

collision probability is visible and shortly after it, a decrease is detected. This is explained by the spreading factor distribution already discussed in Figure 3.8. Furthermore, we see a drop in the collision probability for distances larger than 2,500 m. However, distance limits larger than 2,100 m are not advisable because of a large increase in the collision probability. The best results are detected with a maximal distance smaller than 1,150 m, which is equal to the usage of spreading factor 7 or spreading factor 8 as maximum. Note, there is an upper bound for LoRaWAN optimization by gateway placement. The collision probability by intelligent gateway placement can only be reduced, as long as sensors are transmitting messages with spreading factors larger than 7. If all sensors have one gateway close to their location and are able to transmit with spreading factor 7, no further improvement can be made with more gateways. However, it is possible to react on occurring collisions as discussed in [18].

Number of Gateways: The number of required gateways for different distances is shown by the yellow line and the right y-axis of Figure 3.9. As expected, the number of required gateways is decreasing with an increasing distance between sensors and gateways. However, it is visible that the gradient of the yellow line is decreasing with larger distances. Especially, only a small gradient is visible for more than 1,300 m. Between 1,400 m and 1,500 m, it is increasing again, and we see a good trade-off between the number of gateways and the collision probability between 1,000 m and 1,400 m. This is equal to a used spreading factor of 8 or 9 as maximum.

Additional Performance Metrics: In addition, other performance metrics like the load distribution and thus, the number of sensors single gateways must cover can be evaluated with respect to network quality. For scenario S 1 and 300 m or 350 m distance between sensors and gateways, the results show that no gateway covers more than 4.3 % of all available sensors. The average number of sensors per gateway is for 850 m to 1,250 m distance still good to handle with 200 to 400 sensors. For larger distances of 1,800 m or more, the variance in

the number of sensors per gateway is increasing. Some gateways need to cover many sensors in dense areas and some gateway need to cover only a few sensors. Thus, also from this perspective, it is not advisable to increase the maximal sensor to gateway distance too much. Another performance metric is the number of redundant gateways that cover single sensors. Although double coverage makes a network more robust against gateway failure, the drawback of redundant coverage is additional interference. For small distances below 400 m, each sensor is covered on average by more than four gateways. This number decreases, as expected, with larger distances. Up to 900 m, the average sensor is covered by more than three gateways while for the remaining distances, the average sensor is covered by two or three gateways. Furthermore, this number is not decreasing anymore for distances larger than 1,100 m (2.89 gateways on average). For comparison only, a distance limit of 2,200 m shows 2.85 gateways on average and a distance limit of 2,500 m shows 2.86 gateways on average. As a result, we see that dense areas are covered by more gateways while less dense areas are often only covered by a single gateway, like intended in a good placement. While both, the load distribution in the network and the number of redundant gateways is highly dependent on the location of sensors in a real placement, we see a general tendency. Reducing the maximal distance between sensors and gateways with our graph-based gateway placement reduces the general load per gateway and thus, the collision probability. Especially the usage of the degree centrality as placement metric helps to place LoRaWAN gateways in dense parts of the network, reducing the average spreading factor and the collision probability.

3.4.2 Scenario S 2: Number of Sensors per Gateway

The goal of scenario S 2 is to identify the influence of sensor limits per gateway. However, the simulation results show a fluctuating collision probability without any clear statement about a good sensor number per gateway with focus on the collision probability. Only the number of required gateways is decreasing with more sensors per gateway, as expected.

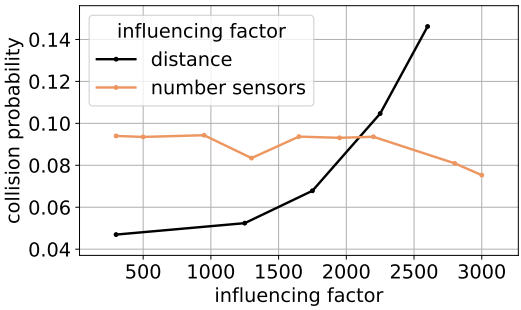

Figure 3.10: Influence of distance and the number of sensors on the collision probability.

When the spreading factor distribution for scenario S 2 is studied, several sensors need to transmit with spreading factor 12 because of the large distance to the next gateway, even for a small sensor limit of 500. This increases the average ToA and the collision probability. When the number of sensors per gateway is used as input, the average collision probability is never below 5 %. In contrast, it is below 5 % for all maximal sensor to gateway distances below 1,150 m in Figure 3.9. Thus, in general, the maximal number of sensors per gateway has a small influence on the overall collision probability.

However, the distance between sensors and their gateway dominates the sensor limit, as visualized by the interaction plot in Figure 3.10. The y-axis of the plot shows the collision probability and the x-axis the influencing factor on the collision probability. The influencing factors *distance* between sensors and gateways and the *number of sensors* are presented by the black and orange line, respectively. While the distance shows a clear tendency towards a larger collision probability for longer distances, no clear tendency for the number of sensors is observed. For that reason, we omit further investigation of this scenario and refer to a more detailed study in [17].

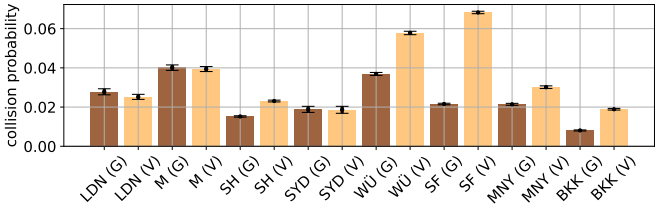

Figure 3.11: Comparison Graph (G) to Voronoi (V) for different cities; Bangkok (BKK), London (LDN), Manhattan, New York (MNY), Munich (M), San Francisco (SF), Shanghai (SH), Sydney (SYD), Würzburg (WÜ).

With these results, we can answer the first research question with *yes, gateway placement in a LoRaWAN can be performed by a graph-based approach with the degree centrality as graph metric. The most important constraint influencing placement decisions and the collision probability is the distance between sensors and gateways. Since this distance directly influences the spreading factor used for transmissions, the network load and in particular, the collision probability can be steered effectively by distance between sensors and gateways.*

3.4.3 Scenario S 3: Comparison to the State-of-the-Art

The state-of-the-art approach from literature studying the collision probability based on different gateway placements for a LoRaWAN is, to the best of our knowledge, the Voronoi-Cover approach [16]. For that reason, we compare our results with the Voronoi approach for different cities by means of collision probability and the number of required gateways. Therefore, we generated different datasets for different urban areas as introduced in Section 3.2.2. The different generated datasets with location, number of generated sensors, and network density are summarized in Table 3.4.

Collision Probability: To compare the placement approaches, Figure 3.11 shows the average collision probability as a bar plot on the y-axis and the different cities pair-wise for our Graph (G) approach and the Voronoi (V) approach on the x-axis. The error bars indicate 95 % confidence intervals. Our approach achieves in the worst case a similar collision probability as we see no statistically significant difference to the Voronoi approach with the 95 % confidence interval for small networks in London (LDN), Munich (M), and Sydney (SYD). However, the graph approach performs better in all other deployments. For San Francisco (SF), much better results are achieved, reducing the collision probability of about 70 %. There, the graph approach shows an average collision probability of 2.16 % against 6.83 % with the Voronoi approach. Thus, we conclude that the graph approach performs in the worst case similar to state-of-the-art from literature and can reduce the collision probability by up to 70 % in the best case, especially in large networks with many sensors or in placements in a large geographic area.

Number of Gateways: Another placement quality indicator is the number of placed gateways. It is summarized in Table 3.6 for the different cities of Figure 3.11 for the Graph (G) approach and the Voronoi-Cover (V). The Graph approach requires fewer gateways in all cities except Munich. A large difference in the number of required gateways is visible for Bangkok, San Francisco, Würzburg, and Shanghai. The largest improvement is visible for large deployments, especially if the number of sensors per km^2 is small. But also for small deployments with fewer sensors like in London, significant improvements with only about 2/3 of the required gateways compared to related work is possible.

Thus, we can answer our second research question as follows. *The presented graph-based gateway placement approach performs well, independent of geographic network size, number of sensors, and for different network layouts, as represented by the synthetically generated sensor deployments based on different urban areas. Furthermore, it performs similar in the worst case and better compared to state-of-the-art literature for all scenarios by means of the number of required gateways and the collision probability in the resulting network.*

Table 3.6: Comparison: required gateways Graph and Voronoi.

City	Graph	Voronoi	City	Graph	Voronoi
Bangkok	287	329	**London**	4	6
Manhattan	41	52	**Munich**	7	7
San Francisco	71	93	**Shanghai**	242	355
Sydney	4	5	**Würzburg**	30	50

3.4.4 Scenario S 4: Different Transmission Patterns

To study the performance of the placement for different transmission patterns, a random message payload is assigned and studied first. Afterwards, the total number of sensors in the networks is increased to emulate an increase in transmission load. An overview of all studied sub-scenarios is given in Table 3.7. The collision probability results are summarized in Figure 3.12 with the 95 % confidence interval. Details for all sub-scenarios of scenario S 4 are given in the following. For all scenarios, the test dataset from Würzburg, Germany is used.

Random Payload Assignment: A random payload scenario R is created to study a variable message payload between 1 B and 51 B for each transmission. This is the maximum possible payload with spreading factor 11 and spreading factor 12 in LoRa. In contrast, scenario B_1 is configured as best performing scenario from the tests above with a fixed payload of 16 B and serves as *baseline* from the previous results. The maximal distance between sensors and gateways is set to 1,150 m, so that each sensor transmits with spreading factor 7 or spreading factor 8. Comparing the average payload, it is 16 B in scenario B_1 and 26 B in scenario R. The remaining placement, simulation, and gateway placement is kept the same. The goal is to study the collision probability only. The results show a mean collision probability of 5.79 % for scenario R compared to 3.70 % in the baseline scenario B_1 with 16 B payload. Thus, the increase in collision probability is with 56.49 %, a little smaller than the proportional increase in payload

Table 3.7: Sub-scenarios for scenario S 4.

Abbr.	Payload	Max. dist. sensor to gateway	Number sensors	Explanation
R	1 B – 51 B	1,150 m	10,000	random payload
$\mathbf{B}_1$	16 B	1,150 m	10,000	baseline SF 8
$\mathbf{B}_2$	16 B	1,150 m	20,000	baseline SF 8 double load
$\mathbf{D}_1$	16 B	2,000 m	10,000	increased distance sensor - gateway
$\mathbf{D}_2$	16 B	2,000 m	20,000	increased distance sensor - gateway; double load
$\mathbf{E}_1$	16 B	950 m	10,000	extended placement SF 7
$\mathbf{E}_2$	16 B	950 m	20,000	extended placement SF 7 double load

(62.5 %) because of header and preamble overhead of LoRa messages. However, we see in the results that the approach is not limited to a single payload but also usable for random payload assignments.

Increasing Transmission Rate: Another parameter is the network load achieved by increasing the number of messages in the network. To study this, the number of sensors of the baseline B_1 is doubled in sub-scenario B_2. Compared to the collision probability of 3.70 % from the baseline B_1, the collision probability increases to 7.11 %, as shown in Figure 3.12. Furthermore, additional studies have shown that linearly increasing the number of sensors nearly linearly increases the collision probability. One possibility to deal with increasing traffic is the placement of additional gateways in an already deployed LoRaWAN instance. This can show the future robustness of the placement algorithm if networks scale and load increases. Therefore, we set up a distance scenario D_1, where we set the maximal distance between sensors and gateways to 2,000 m with 10,000 sensors. If our graph based approach is used for an initial placement, the complete network is covered by 17 gateways. A mean collision probability of 7.3 % is

achieved, shown by sub-scenario D_1 in Figure 3.12. However, deployments are expected to grow in size and the number of sensors in the future. If the number of sensors is doubled in the future, the mean collision probability is increasing to 13.7 % shown by sub-scenario D_2 where 20,000 sensors are in the network, but the same number of gateways is used as in D_1.

To tackle this issue, the graph based gateway placement algorithm is performed on a network with already placed gateways. Therefore, we create a graph of all sensors and the already placed gateways but reduce the maximal distance between the sensors and the gateways. This reduces the number of sensors covered by each existing gateway. Sensors that are no longer covered by one of the existing gateways will need to be processed during iterative placement. Due to the graph-based nature of our approach, this iterative placement is achievable by simply constructing the graph on now uncovered sensors. This is done in scenario E_1 and in scenario E_2 that iterate the placements used in D_1 with 10,000 sensors and in D_2 with 20,000 sensors, respectively. For both scenarios, the initial maximal distance between sensors and gateways was 2,000 m, for the initial placement and is adjusted to 950 m. Both adjustments can reduce the collision probability by roughly 50 %. However, the extended placement requires 46 gateways for E_1 and E_2. For that reason, a good initial placement is preferred to constant re-placement on demand. Nevertheless, this study shows the robustness against future load increase and that the approach can deal with scaling networks in the future.

3.4.5 Scenario S 5: Placement for Large Deployments

Gateway placement for LoRaWAN in large networks is still challenging [16] or not possible with approaches from literature [132]. However, this is important for growing networks in the future. Since our approach places gateways at dense network parts, we can apply clustering to split large problems into several sub-problems. We attempt this for the data from Würzburg by applying k-means clustering before computing placements on the resulting clusters of de-

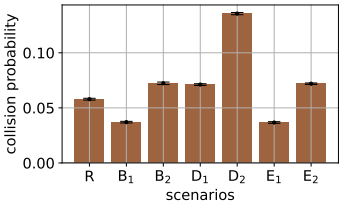

Figure 3.12: Scenario S 4, transmission pattern sub-scenarios according to Table 3.7.

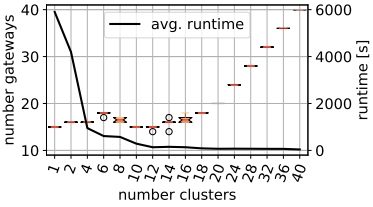

Figure 3.13: Number of gateways and runtime for different number clusters.

vices. As a baseline, we use a maximal distance between sensors and gateways of 2,171.44 m, for which we need 15 gateways to cover all sensors. Afterwards, we divide the network into two, four, five, ten, 15, and 20 clusters and perform individual placement on each cluster. We require 16, 16, 17, 16, 17, and 20 gateways, while the maximal segmentation using 20 clusters only requires one gateway per cluster. Based on these promising results, we extend the investigation as follows. In Scenario S 5.1, we conduct an in-depth performance study of the pre-clustering idea by means of the Würzburg dataset with a maximal distance between sensor and gateway of 2,171.44 m, and thus the spreading factor 12 limit to guarantee coverage. In Scenario S 5.2, we extend the complete network generation idea towards artificially generated networks to investigate (1) the performance on synthetic data and (2) to study the influence of pre-clustering in more detail, and in particular investigate the number of required gateways and the required runtime for network creation. Last, in Scenario S 5.3, we select network data obtained from other cities, as summarized in Table 3.4 to draw conclusions about the performance in real-world cities. Therefore, we compare the performance of the graph based gateway placement algorithm without pre-clustering with the achieved results when the network is pre-clustered. We again use a maximal distance between sensor and gateway of 2,171.44 m and 1,000 sensors per gateway as maximum. We need to change the input parameters here com-

pared to Scenario S 3, where placement for the same cities has been discussed. This is required since when we set the maximal distance between sensors and gateways to the best performing one, almost every sensor is transmitting with spreading factor 7, and thus with a small ToA. This would lead to a little collision probability and to similar collision probabilities for slightly worse placements. For that reason, we increase the maximal distance between sensor and gateway to 2,171.44 m which in turn will lead to much worse collision probabilities for worse placements, emphasizing the need for good placement decisions.

S 5.1: Pre-Clustering Performance Study

First, the number of required gateways is studied to evaluate the performance of a gateway placement with our graph based placement approach with pre-clustering. The result for a different number of clusters is shown in Figure 3.13. The left y-axis shows the number of gateways against the number of clusters along the x-axis. One cluster is presented as reference for a placement without pre-clustering. It can be seen that the number of required gateways has little to no variance for different numbers of clusters and is only increasing slightly up to 18 clusters. While the run without clustering required 15 gateways, also only 15 gateways are required for ten and 12 clusters. However, the approach with clustering shows some variance in the results with the need of only 14 gateways for a single run with 12 and 14 clusters, respectively. With more clusters, however, the number of gateways is increasing since cells become smaller than needed, resulting in each cluster being assigned exactly one gateway.

The goal of the pre-clustering mechanism is a faster, more resource friendly gateway placement calculation and thus, the possibility to process larger networks. For that reason, the right y-axis of Figure 3.13 presents the runtime for the gateway placement in seconds for the different number of clusters. The data shows that pre-clustering can reduce the processing time drastically. While the approach without pre-clustering required nearly 6,000 s until termination, the runtime could be reduced to approximately 1,000 s for four clusters. The minimal runtime of 40.67 s is achieved with 40 clusters. A good trade-off is achieved

between 12 and 18 clusters. There, the runtimes are on average between 70 s and 150 s without drastically increasing the number of required gateways. Based on these observations, we can conclude that pre-clustering large problem instances achieves good trade-offs between the number of required additional gateways and runtime. This improvement in the processing time is achieved since it is no longer required during graph creation for each n sensors to determine whether an edge to all other $n-1$ sensors is required, thus $n \cdot (n-1)$ comparisons. Instead, in the best case, each of c clusters includes n/c sensors. This lead to only $c \cdot (n/c) \cdot ((n-1)/c)$ operations and an improvement by factor c. Furthermore, it is shown in [30] that pre-clustering can help for problem instances of several million devices covering complete states.

In Figure 3.14, the collision probability is studied for different numbers of clusters. The y-axis represents the collision probability, the x-axis the number of clusters. The solid line highlights the mean collision probability while the shaded area presents all collision probability values for 100 re-runs for each number of clusters. The 100 re-runs are achieved by ten re-runs for each number of clusters and ten further re-runs for each of the created networks for each cluster. The collision probability calculation is similar to the one in the previous scenarios. The baseline scenario without clustering achieves a mean collision probability of about 4.5 %. In contrast, 4, 6, 8, 12, and more clusters achieve a better collision probability while two and ten clusters achieve a worse one. In general, more gateways increase the number of sensors that are able to transmit with smaller spreading factors and thus, reduce the collision probability. However, in contrast, the pre-clustering might cluster sensors together, that would not connect to the same gateway in the pure graph based approach. Thus, small variation in the collision probability in both, positive and negative direction are possible. But since the maximal average increase in collision probability is 1 % for ten clusters the approach is very valuable. For 14 – 18 clusters, the average collision probability decreased compared to no pre-clustering and a good trade-off by means of the number of required gateways is shown.

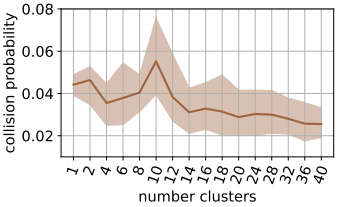

Figure 3.14: Collision probability for different number clusters.

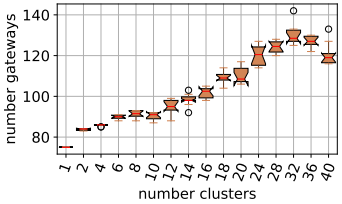

Figure 3.15: Number gateways for clustering artificial network.

S 5.2: Study for Artificially Generated Networks

To study the clustering performance on other artificially created networks, an area of 504 km^2 around the city of Würzburg is selected. The area is 23.4 km long and 21.4 km wide and can symbolize a medium-sized city with some suburbs. In this area, we placed 2,000 sensors by means of four different placement strategies and gateways, based on sensor locations. First, we uniformly place sensors to cover the complete area with similar density, and second randomly in the complete area. The third method is a random placement in a circle around the middle while the circle has a radius of 10.7 km. Thus, its edge is hitting the edge of the area selected for placement. This placement strategy results in two possible distributions: (1) a random distribution with the same probability for a sensor at each position in the circle, and (2) in a particularly dense area in the middle, with more spread out sensors along the edges, emulating city center and suburb behavior. The results of this study show that the pre-clustering has more negative impact on the number of required gateways for artificially generated network, shown as an example in Figure 3.15 for complete random sensor assignment. The graph based placement approach without clustering places 77 gateways. When the network is pre-clustered, always more than 80 gateways are required. The value is increasing to about 130 gateways for 32 clusters and decreasing again afterwards. However, we see no increase in the overall collision

Table 3.8: Number of required gateways after pre-clustering for different cities (Bangkok (BKK), Manhattan - NY (MNY), San Francisco (SF), Shanghai (SH)).

Number clusters	BKK	MNY	SF	SH	Number clusters	BKK	MNY	SF	SH
0	129	22	34	115	**20**	140	21	35	130
4	133	26	41	116	**25**	148	25	35	134
8	132	22	46	119	**30**	147	30	35	136
12	138	21	37	117	**35**	154	35	38	128
16	140	19	38	120	**40**	148	40	41	128

probability when the network is pre-clustered. For no clustering and all clustering representations, a collision probability between 1 % and 2 % is achieved. In addition, as already seen in Scenario S 5.1, the processing time for the algorithm decreases significantly when the network is pre-clustered. Without clustering, a runtime of more than 800 s is required, for two clusters less than 200 s is required on average, and it further decreases for more clusters, similar to Scenario S 5.1. Furthermore, we see the same behavior for the number of gateways, collision probability, and processing time also for the other synthetic networks generated by the other placement mechanisms described above.

Based on these observations, we can conclude that pre-clustering helps to reduce the processing time with only minor to no influence on the collision probability. However, the number of required gateways is increasing. This means the pre-clustering can deal less efficiently with randomly or evenly distributed sensors. For that reason, areas where sensors are deployed in a rather even distribution should not be split into several problem instances in practice but processed as a whole with a single graph to place the gateways. However, such deployments are very uncommon for a real network.

S 5.3: Study for Different Real-World Cities

To determine whether the pre-clustering can improve the general placement for artificially generated networks on an underlying real city, it is also evaluated on the cities studied in Scenario S 3. The goal is mainly to determine whether the number of required gateways increases if the network is pre-clustered. We see only little changes for London, Munich, and Sydney. However, these networks are rather small with 4 – 7 placed gateways and can be calculated efficiently without pre-clustering. The more interesting study can be conducted on the datasets for Bangkok, Manhattan, San Francisco, and Shanghai. The sensor density and the number of sensors per dataset is visible in Table 3.4. Without clustering, the placement for Bangkok requires 129 gateways, Manhattan requires 22 gateways, 34 are required for San Francisco, and 115 for Shanghai. First, the results show, that updating the maximal transmission distance between sensors and gateways from 1,150 m, as used for Scenario S 3 to 2,171 m significantly reduces the number of required gateways, as is expected. All cities require only about half of the gateways or even less.

Based on this baseline number of gateways, we perform pre-clustering before computing gateway placements. A summary is given in Table 3.8. The data shows the number of required gateways with the graph-based gateway placement algorithm dependent on the number of clusters, where zero clusters show the baseline without pre-clustering. First, we see that for large, less dense networks more gateways are required when pre-clustered. For both, Bangkok and Shanghai, no clustering achieves the smallest number of required gateways. However, for both networks, no significant increase is visible with 4 – 8 clusters. Nevertheless, this small number of clusters still reduces the required processing time drastically and is a viable solution to achieve fast results while maintaining a consistent quality for the resulting placement. From a collision probability point of view, we see no significant increase for both cities when clustered, since a similar number or more gateways must be placed. The situation is different for Manhattan and San Francisco. It is comparable to the behavior for Würzburg presented in Scenario S 5.1. For Manhattan, the baseline does not achieve the

best result by means of the required number of gateways, for San Francisco it is only slightly better. While the least number of gateways for Manhattan is achieved for 16 clusters, it is achieved - except for no clustering here - with 20, 25, and 30 clusters for San Francisco. However, this minor reduction in the number of gateways leads to a slight increase in the collision probability for Manhattan and a similar one for San Francisco. Again, we conclude that pre-clustering is a viable solution to reduce the processing time. Nevertheless, it is essential to select a good number of clusters, which is highly dependent on the network size, density, and structure in general.

With these results, we can answer the third research question as follows. *Larger deployments show computational limits, in particular for the graph creation and gateway selection process. However, a pre-clustering can handle arbitrarily large network instances with only minimal overhead for real deployments and artificially generated networks based on real cities but has limits on completely artificially generated networks. It achieves the best results from a required number of gateways point of view on smaller but denser networks and requires little overhead in the number of gateways for less dense but larger networks. However, a reduction in the number of gateways always showed an increase of the collision probability while slightly more gateways kept the collision probability similar. In addition, as expected, pre-clustering reduces the required processing time significantly.*

3.5 Lessons Learned

The trend towards the deployment of massive swarms of IoT devices is expected to increase further in the coming years. Application areas include environmental monitoring with wireless sensor networks, home and city automation by smart city solutions, or various use cases in the Industry 4.0. However, not every connected sensor requires low delays and transmits a large amount of data. Specifically, low energy consumption for long battery life times combined with transmissions across large geographic distances for fair prices are key features for many use cases such as simple temperature or weather sensors. Dealing with

challenges like cheap, simple, and energy efficient transmission across long distances is a unique selling point for LPWAN with LoRaWAN as one of the most prominent representatives providing these features.

In order to be able to design, but also deploy and manage LoRaWAN roll-outs, mechanisms to plan configurations and predict their expected performance are crucial. For a comprehensive deployment, the general performance in the network must be known and solutions to assess the impact of network growth or infrastructure changes are required. Thus, the currently often unplanned gateway deployment leaves much room for improvement.

In the gateway placement procedure for LoRaWAN, the following three research questions have been identified and investigated in this chapter.

RQ3.1) Is it possible to perform efficient gateway placement for a LoRaWAN based on an abstract, graph-based view of a set of geographically distributed LoRa nodes? Which graph metrics are important and which constraints during graph creation influence the overall collision probability as important quality metric in the network most?

RQ3.2) Is the graph-based approach generalizable for a multitude of different networks, how can such networks be synthetically generated, and is it possible to compete with state-of-the-art literature?

RQ3.3) How can a graph-based approach be applied to arbitrary large problem instances, and what are limits of scalability?

To address these questions, we provide a novel gateway placement for LoRaWAN by transforming the network into a graph and identify additional constraints and characteristics that have significant impact on the quality of computed placements. By means of different graph creation constraints and by using the degree centrality as key metric to set up the graph, it is possible to influence the resulting placement and thus, the overall collision probability. We see a general trade-off between the number of required gateways and the average collision probability in the network. However, we see the best results when the

maximal distance between sensors and gateways is limited in a way, that each sensor can use spreading factor 7 or spreading factor 8 for transmissions (RQ3.1).

Furthermore, we synthetically generate networks based on underlying real cities by using real building location of different cities around the globe. With these networks, we compared the graph based gateway placement with the Voronoi-Cover approach as state-of-the-art literature [16]. Our placement requires in the worst case the same number of gateways and achieves similar collision probability in the network as the Voronoi approach but can reduce the number of gateways by 40 % in combination with a reduction in collision probability by up to 70 % (RQ3.2).

Finally, our approach is independent of the underlying network, already preplaced gateways, number of sensors, or the network size. We specifically investigate the applicability of our approach to increasingly large deployments. By exploiting k-means clustering, we show that large problem instances can be split into several smaller problems, which can subsequently be solved individually. This divide and conquer approach can be used to drastically reduce the size of individual problems, which in return reduces the runtime to compute placements by several orders of magnitude. At the same time, the division into sub-problems results in minor or even negligible overhead, both with respect to the number of deployed gateways and the expected collision probability for real-world scenarios. When it comes to synthetic problem instances, we have observed a slight degradation in placement quality when clustering sensors that exhibit uniform placement distribution. However, this characteristic is unlikely encountered in the real-world, due to the layout of cities and their surrounding suburbs, which allows us to conclude that our proposed placement mechanism can be employed to increase the reliability and performance, while in addition reduce the cost for LoRaWAN deployments efficiently (RQ3.3).

For a realistic setup, this graph based gateway placement can assist network providers during network planning of new and already deployed LoRaWANs. In addition to the distance between sensors and gateways, additional parameters like geography, data importance, or population density can be added to

customize the placement for different purposes. However, the approach is only applicable if it is possible to determine all sensor locations or achieve a comprehensive view on potentially interfering sensors. This is essential during the graph creation process. While the general graph for gateway placement is unweighted, additional edge weights can be added to improve the importance of specific nodes, increase the weight of different edges because of bad geography, or influence the node centrality, and thus the general gateway placement. Our results show that in future approaches, additional weights based on the time on air show most potential as they influence the collision probability most. However, when all sensors transmit with spreading factor seven in a LoRaWAN, improvement potential with gateway placement is exhausted. Then, additional loss reduction techniques can be implemented [18]. However, these approaches should be limited to important messages to not overload the system with acknowledgments or re-transmissions. For such important or time critical messages, it is also possible to reserve single channels to increase successful transmission probability. But, as a consequence, the load and hence, the collision probability of all other applications is increasing in all other channels when random channel access is used. Thus, it is suggested that all channels are randomly selected for all transmissions or that further studies investigate the fairness in a LoRaWAN if specific traffic is prioritized.

4 Performance Investigation for Novel LoRaWAN Channel Access

The introduction of smart solutions in most applications of the everyday life is one of the fastest growing and most dynamic technology trends nowadays. Although requirements for, among others, Industry 4.0, traffic management systems in Smart Cities, or simple weather forecasts are completely different from a technology perspective, they have several things in common: first and basically, the requirement for any form of data acquisition and communication.

One solution to create large wireless sensor networks are widespread 5G networks. The drawback is costly end devices for applications, where sometimes only few data is transmitted. Thus, the possibilities 5G networks provide are often not required. Another possibility is the usage of specific energy efficient IoT access network technologies like Long-Term Evolution Machine Type Communication (LTE-M) or NB-IoT, developed and standardized by the 3rd Generation Partnership Project (3GPP). Using the LoRaWAN protocol, managed in an open and non-profit way can be a different approach. It promises economically priced infrastructure and end devices, long battery life times for sensors because of little device energy consumption, and large transmission distances. These benefits induced Amazon to use LoRaWAN as one of their access network technologies for the Amazon Sidewalk project, recently opened to the public [155]. Their Amazon Echo and Ring devices function as gateways and provide cloud connectivity to many sensors [156]. This leads to easy LoRaWAN access for more than 90 % of the US population [156, 157] and is the next logical step in LoRaWAN development, with a market size valued at $2 billion in 2022 [158].

Because of the cheap and easy usage possibility and long transmission distances, LoRaWAN is the ideal network for various IoT use cases, from simple weather or temperature monitoring, to metering in Smart City environments, or environmental and agriculture monitoring tasks. Nevertheless, the benefits of the simple-to-use network with energy efficient transmissions also come with one major drawback: the inherently unreliable transmission due to little protection against interference. With the currently applied random channel access, sensors can use the available channels simultaneously. This leads, in the worst case, to severe message collisions and data loss. As we discuss in Chapter 3, one solution to cope with message collisions is a smart network planning and gateway placement. But additional gateways are costly, and changing the location of already deployed gateways is difficult in practice. For that reason, several research activities focus on the development of alternative channel access solutions for LoRaWAN, with listen before talk and time scheduled access as the most promising approaches [159, 7, 160]. Though, additional required functionality leads to more complex data transmission cycles, more complex hardware, and influences the total energy consumption of end devices. Different literature already quantifies the energy demand [161, 162] of very specific hardware components, or analyze the LoRa transceiver. But a generic model which can be parameterized for arbitrary hardware to investigate LoRa transmissions is not available in literature yet. With this knowledge, an in-depth analysis of channel access approaches and the influence on current transmission regulation can provide guidelines for the best utilization of a LoRaWAN. This is required to cope with the rapid deployment speed of LoRaWAN.

To this end, we propose a time scheduled channel access approach in this chapter that avoids message collisions in a LoRaWAN completely. A theoretical investigation of the time scheduled access is presented to cope with transmission regulation that is required to design future LoRaWAN deployments. In addition, the performance is evaluated in realistic conditions with unavoidable cross-traffic by a large-scale simulation study. We further propose an energy model for LoRaWAN, based on the required energy during an optimal LoRa

transmission. Since we only consider energy ratios between different processes during a data transmission, we can model the energy consumption in a very general way for different channel access approaches. Furthermore, we extend the energy consumption definition to energy efficiency and can compare channel access in different situations, for different parameter settings, and during different network and load situations. This allows us to draw conclusions about the most energy efficient transmission option in a LoRaWAN. This leads us to the following research questions for this chapter.

RQ4.1) Is it possible to schedule all messages in a LoRaWAN channel to avoid collisions completely, despite the device and LoRaWAN specific challenges such as device clock drifts, different duration to transmit messages, the gateway duty cycle, and cross-traffic?

RQ4.2) Is it possible to model a LoRaWAN with different channel access approaches and assess the resulting energy consumption and energy efficiency, usable for various underlying hardware?

RQ4.3) What is the best channel access approach for LoRaWAN from an energy efficiency point of view, dependent on the load in the network?

To answer these research questions, Section 4.1 introduces background information on transmission regulation and channel access in LoRaWAN and introduces the energy consumption of LoRa sensor nodes first. At the end of the section, related literature is summarized. Then, Section 4.2 discusses the general methodology for different channel access approaches and presents details on a theoretical investigation for a time scheduled approach. At the end of the section, a simulation is described, and simulation scenarios are introduced to evaluate the performance of different channel access approaches. These scenarios are evaluated in Section 4.3 with main focus on collision probability. Afterwards, a generic energy model for different LoRaWAN channel access approaches that is configurable for arbitrary hardware is presented together with a metric to quantify energy efficiency in Section 4.4. This is evaluated in Section 4.5, and

guidelines for the best channel access solution in different network situations is given. At the end, Section 4.6 concludes and discusses the results regarding scientific contributions and lessons learned. To summarize, this chapter contains the following major contributions.

C4.1) The design and evaluation of a time scheduled channel access approach for LoRaWAN with a general relationship between the theoretically possible number of messages that can be transmitted within a specific time frame to avoid collisions completely and still comply with LoRaWAN transmission regulations.

C4.2) The development and evaluation of a generic energy consumption model for different LoRaWAN channel access approaches and the definition of an energy efficiency metric with guidelines for the most energy efficient channel access approach in a LoRaWAN.

C4.3) An exhaustive simulation to study and quantify the performance of different channel access approaches by means of collision probability, energy consumption, and energy efficiency.

The contributions have already been published in the past and are summarized in this monograph based on the following scientific publications.

- Loh, F., Mehling, N., Hoßfeld, T.: "Towards LoRaWAN without Data Loss: Studying the Performance of Different Channel Access Approaches", in Sensors, 2022 [7].
- Loh, F., Raffeck, S., Geißler, S., Hoßfeld, T.: "Generic Model to Quantify Energy Consumption for Different LoRaWAN Channel Access Methods", in International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2022 [19].
- Loh, F., Mehling, N., Geißler, S., Hoßfeld, T.: "Simulative Performance Study of Slotted ALOHA for LoRaWAN Channel Access", in Network Operations and Management Symposium (NOMS), 2022 [20].

- Loh, F., Raffeck, S., Geißler, S., Hoßfeld, T.: "Plan the Access? Generic Hardware Independent Energy Consumption and Efficiency Model for Different LoRaWAN Channel Access Approaches", in IEEE Internet of Things Journal, 2024 [8].

- Loh, F., Raffeck, S., Metzger, F., Hoßfeld, T.: "Improving LoRaWAN's Successful Information Transmission Rate with Redundancy", in International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2021 [18].

4.1 Background and Related Work

The investigation in Chapter 3 shows that intelligent gateway placement can improve the collision probability in a LoRaWAN when random channel access is used. A different approach to improve channel utilization and decrease the collision probability is channel access management. However, in contrast to the placement of additional gateways, adjusting the channel access methodology is challenging in current LoRaWAN. Changing the way LoRa sensors access a channel requires specific device capabilities and the agreement of all parties in a LoRaWAN on a standardized way to access the channel. The ideas and approaches discussed in this chapter are target for future LoRaWAN deployments. Intelligent channel management can be achieved by alternative channel access mechanisms, since the currently used random access suffers from many collisions as already discussed in Section 3.1. Thus, this section introduces different alternatives to random channel access for a LoRaWAN and discusses background information on the working sequence of LoRa devices during message transmissions that is essential for the energy consumption studies in this chapter. Little energy consumption is a unique characteristic of LoRaWAN, and it is important to take this aspect into consideration when alternative channel access methodologies are studied. Finally, related literature with main focus on channel access solutions for LoRaWAN is summarized at the end of this section.

4.1.1 Transmission Duty Cycle in LoRaWAN

Details about data transmissions of all devices in a network is of major interest to evaluate channel access approaches and perform channel access management to steer available traffic. This is possible by scheduling all transmissions in the best way to the available channel resources and avoid message collisions. Thus, all messages transmitted by any sensor, but also signaling, possible acknowledgment, and synchronization traffic generated by the gateway need to be considered. Details about LoRa messages, the time to transmit these messages, and the impact of, for example, the ToA on the collision probability is discussed in Chapter 3. There, the main parameters to understand the structure of a message and to calculate the ToA are available in Table 3.1. However, the traffic from gateway to sensor is not considered yet in this monograph. This is important for a LoRaWAN, as all devices including the gateway need to follow the duty cycle regulations. The duty cycle is a transmission time limitation agreement by all devices in LoRaWAN channels, defining the maximal percentage of time a single device is allowed to occupy a channel [103]. The objective of this duty cycle regulation is to leave free channel time for all other devices in a network but also allow different networks to operate simultaneously and efficiently. Duty cycle limits are often regulated by the government and typically allow usage percentages of 0.1 %, 1.0 %, or 10 % occupancy time per hour per device, dependent on the used frequency band [163]. Since it is commonly set to 1 % [163], we use this in the thesis. The duty cycle strictly limits the allowed number of messages per device per hour, especially when using a large spreading factor.

4.1.2 LoRaWAN Channel Access

In general, channel access planning in LoRaWAN is a complex task. Challenging factors are, among others, strict requirements at the end devices to save battery and the duty cycle limiting the number of re-synchronization messages sent from the gateway to the end devices. The collision avoidance potential can be limited if sensors are transmitting to the same gateway without the ability to

communicate with each other directly, known as the hidden node problem of wireless access technologies. In addition, anyone can deploy an own LoRaWAN by setting up a gateway and own sensors. This often unplanned approach can lead to potential cross-traffic from other devices or another LoRaWAN. A solution to reduce the overall data loss as a result of collisions with cross-traffic is to acknowledge messages or use loss reduction techniques [18]. This requires more complex hardware or more energy and leads to additional messages and more collisions [18]. Furthermore, the available duty cycle of the gateway is charged.

Device Classes

Besides network load, device capabilities must be known to select a channel access approach best suited for a LoRaWAN. The LoRaWAN specifications define three devices classes, class A, class B, and class C. All devices must implement class A, whereas class B and class C are optional extensions [164]. A class A device can send a message at any time. After the transmission is completed, a first receive window to receive messages from the gateway is opened after a reception delay. Such messages are typically acknowledgments, updates, or re-synchronizations. If the end device does not receive any data during the first receive window, it opens a second one after another receive delay. Then, if nothing is received, it returns to normal operation, i.e., dependent on the specific configuration, back to sleep mode until it has new data to transmit. In contrast, device class B opens receive windows at pre-configured times and the gateway does not need to wait for any uplink message to communicate with any device. This reduces communication latency but in contrast to class A devices, class B devices spend more time in active mode and consume more energy. Last, class C devices keep their reception window always open, unless they transmit own data. This further reduces latency and increases energy consumption. Since class A devices in LoRaWAN have most limitations in availability for gateway communication and re-synchronization by returning frequently into sleep mode to save energy, we use these devices for the investigation in this chapter. However, the suggested approaches also work for class B and class C devices, respectively.

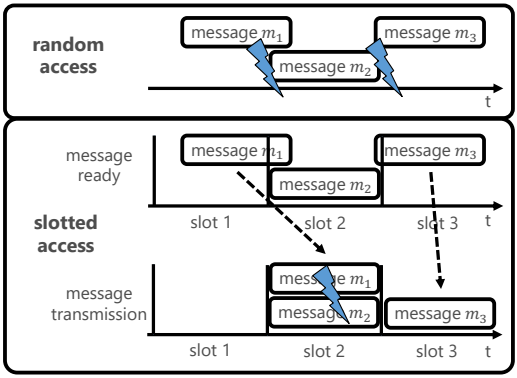

Figure 4.1: Possible types of channel access in LoRaWAN.

Channel Access Approaches

With the currently used random channel access approach, LoRaWAN uses a very simple but unplanned and uncoordinated channel access, visualized in Figure 4.1. Each sensor transmits its data directly when it is available, as shown in the top part of the figure. There is no planning or coordination, channel sensing, or collision avoidance mechanism and message collisions or data loss can happen. In the example figure, message m_1 is still transmitted when the transmission of m_2 starts. Both messages collide and get lost if recovery is not possible. The same holds true for message m_2 and message m_3 respectively.

Slotted ALOHA: One potential improvement is slotted ALOHA, shown by the slotted access approach in Figure 4.1. It divides available channels into time slots and allows channel access by slot allocation. End devices have to conform to these time slots and only initiate transmissions at the beginning of a slot. As a result, collisions can only occur if two or more devices transmit in the same time slot instead of being at a constant risk of interference from other devices,

as it is with random access. In the example figure, although message m_1 is ready for transmission earlier than message m_2, both are delayed to the next free slot (slot 2). However, since two messages are transmitted in this slot, they collide. Message m_3 is also delayed to the next slot (slot 3). This delay avoids a collision with message m_2 and results in a correct transmission of message m_3. This setup theoretically allows slotted ALOHA to reduce the number of collisions as well as the vulnerable time by 50 % when compared to random access [165]. Furthermore, the maximum channel utilization is increased to 36.8 % [106]. However, accurate timing information and re-synchronization is required to keep devices aligned to the time slots and clock inaccuracies must be taken into consideration. This is dealt with by using an appropriate slot length and additional guard times. Nevertheless, channel access with slotted ALOHA is only optimal if no slot space is left unused. That means, a message is transmitted in each available slot, each message has a slot with its individual length without wasting channel resources, and no message drifts out of its individual slot. In reality, several challenges prevent optimal slotted ALOHA for LoRaWAN. In particular, channel occupancy times for different messages vary between several milliseconds and seconds in real LoRaWAN deployments because of the large range of the ToA of LoRa messages (see Figure 3.3). Long slot lengths equal to the maximal ToA waste many resources when short messages are transmitted and small slot lengths can make it impossible to transmit longer messages. This aims for a reasonable slot length selection in a LoRaWAN, and an application with different payloads, spreading factors, and thus, different message ToA is challenging.

For that reason, it is only suggested to use slotted ALOHA if the message ToA is similar, for example if all sensors already transmit with spreading factor seven after an efficient network planning. Then, if slot lengths and guard times are well-chosen, slotted ALOHA can improve the overall collision probability in LoRaWAN, as elaborated in [20]. However, for a more heterogeneous network, where sensors can transmit with different spreading factors and a large possible range in the transmission airtime is expected, a different channel access approach is suggested. Since the goal in this chapter of this monograph

is to study a general channel access solutions for LoRaWAN, a more detailed investigation of slotted ALOHA is omitted, and we refer to [20] for details.

Time Scheduled: In contrast to slotted ALOHA, each device in a LoRaWAN must register for a transmission slot at the gateway in a time scheduled or scheduled approach. Only a single message would be assigned to each slot in Figure 4.1, instead of moving all messages to the next free slot. After a slot is assigned, a message is committed to use only this specific slot. With this idea, message collisions and data loss can theoretically be completely avoided if each device keeps to its slots. Since the time scheduled approach has an increased synchronization, management, and channel access planning overhead but several improvements against slotted ALOHA, it is expected to be a better alternative to random access. Random clock drifts or delays can occur in reality with cheap LoRa sensors [113]. This prevents a perfect time slotted channel access in real deployments. Such clock drifts occur due to the nature of many oscillator crystals used in lower cost devices. These oscillators produce an uncertainty of timing if running too slow or too fast. It can be expressed as a deviation from the nominal frequency in a parts per million (ppm) unit. According to literature, common drifts are in the range of 0.5 ppm – 100 ppm [166]. In total, a clock drift of 1 ppm is equal to a drift of 3.6 ms per hour. In addition, a non-linear drift is observed in reality due to, among others, different temperature [166]. Clocks with very little drift are comparably expensive and not applicable for a cheap and large scale LoRaWAN. For that reason, drifts smaller than 2 ppm are not taken into consideration in this work. In addition, too large drifts lead to large timing uncertainties per hour and to the necessity for large guard times, slot lengths, or frequent re-synchronization with additional messages. Frequent re-synchronization is not advisable since it limits or exceeds the gateway duty cycle. If messages are sent by devices with a large clock drift, additional self-calibration or correction approaches are suggested [167] or a time scheduled approach is not usable. We investigate drift ranges between 2 ppm and 150 ppm in more detail to analyze the impact of a broad range of drifts on our approaches.

Listen before Talk: A different possible channel access strategy is listen before talk. The devices listen on the channel before attempting a transmission and only transmit if the desired channel is free. No additional overhead for synchronization or channel access control is required when listen before talk is used. However, if a device is attempting to transmit data and the channel is occupied, the transmission is delayed by a specific form of back-off delay. According to the literature, there are different approaches to determine this back-off delay duration. One idea is to use back-off slots of a specific duration and delay the transmission for a specific number of such slots [168]. Furthermore, the back-off delay can be combined with the usage of another channel at a different frequency band named frequency hopping [169]. In total, the interference between devices in a network with many end devices trying to send at the same time can be reduced [159]. However, the possible performance of listen before talk is limited by the hidden node problem. The hidden node problem in LoRaWAN can occur, if obstacles are in the line of sight transmission of sensors, but also if the distance between sensors is large. When all nodes in a network are hidden from each other, the performance of listen before talk is reduced to the same level as pure ALOHA [159]. In reality, many nodes can be hidden from others due to the long range of LoRa transmissions [159].

4.1.3 Working Sequence of a LoRaWAN Sensor Device

Alternative channel access approaches can reduce the collision probability and potential data loss. The drawback of these alternatives is increased complexity due to, among others, channel sensing or synchronization. This can influence the energy consumption of the end devices as one key quality metric of a LoRaWAN. For that reason, and to understand the energy consumption of LoRa sensors, the working sequence of a sensor node is introduced in the following and visualized in Figure 4.2.

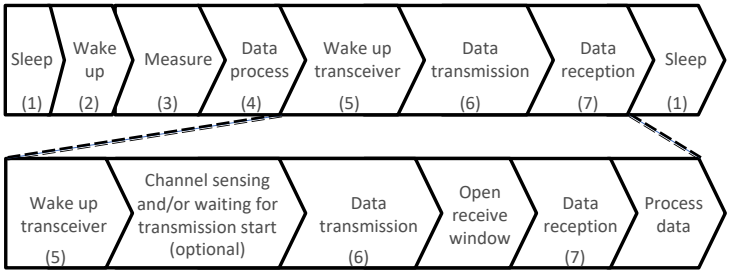

Figure 4.2: LoRaWAN transmission schedule workflow.

LoRaWAN Transmission Schedule Workflow

The typical working sequence of a LoRa sensor is already discussed in literature [161, 162] and highlighted in the top part of Figure 4.2. After a sensor wakes up from sleep (1,2), it measures (3) and processes (4) its data. The actual transmission starts after the transceiver wakes up (5,6). Afterwards, optional data is received (7) and the sensor returns to sleep (1). Note, we use the data reception (7) to receive potential re-synchronization messages for the time scheduled channel access approach in this work and do not include a separate process for it. The energy consumption of the individual parts of this working sequence is highly dependent on the duration of each part and the current consumption, and thus on the used hardware. Typical value ranges are summarized in [161], with durations of several milliseconds up to 3 s for data transmission and a current consumption of several μA up to more than 100 mA for the transmission.

The general data transmission process that is dependent on the channel access approach starts, when the transceiver wakes up and ends with an optional data reception, shown in the bottom part of Figure 4.2. The remaining working sequence is independent of the access approach and thus, not considered in the energy study in this chapter. Different access approaches can add channel sensing, a pre-defined waiting time, or optional receive windows that can be

opened after a transmission. If data is received in these windows, additional data processing can be required. At the end, the energy consumption of a sensor is determined by summing up the required power in each process multiplied with the time, a sensor is in each process.

4.1.4 Related Work

To improve channel access in a LoRaWAN, the goal is to use the available frequency channels more efficiently and reduce the collision probability without a large increase of the energy consumption. For that reason, different related approaches to improve channel access and decrease the collision probability in LoRaWAN are introduced and related energy consumption studies are summarized. Selected approaches from literature using random access, listen before talk, or a time scheduled approach for channel access, discuss the energy consumption when transmitting LoRa messages, and a comparison to this chapter of this monograph are summarized in Table 4.1.

One of the first works dealing with channel access in LoRaWAN is from Bankov et. al. [105]. The authors discuss limits of, among others, modulation and channel access. Especially scheduling approaches like slotted ALOHA and Carrier Sense Multiple Access (CSMA) are suggested as alternatives by Beltramelli [159]. Polonelli et al. propose slotted ALOHA on top of pure ALOHA for channel access in a LoRaWAN, study the approach in detail, and conduct measurements [113]. In addition, different back-off schemes to avoid collisions are presented in literature [170, 171], or aggregated acknowledgments are proposed to improve scalability and reliability for scheduled transmissions [172]. Garrido studies scheduling in LoRaWAN and presents a real-world implementation in [173]. CSMA [174–176] is also studied by many other works and valuable results are conducted with several limitations like the hidden node problem. Furthermore, CSMA-based approaches are discussed as listen before talk solutions [177–180]. In addition, sole listen before talk is studied and compared to pure ALOHA in literature [160, 169, 181], while explicitly the coexistence of lis-

ten before talk and pure ALOHA is studied in [168]. Recently, Raffeck et al. investigated an approach for sensors to self organize channel access to dissolve traffic bursts [182]. A general study for channel activity detection to better understand and improve channel access in LoRaWAN is done by Pham and Ehsan in [183]. The authors design guidelines for channel access and propose a lightweight collision avoidance mechanism. In their guidelines, they identify that an exchange of ready- and clear to send messages can not comply with the duty cycle regulations. Furthermore, they implemented the approach and achieved promising results, in particular for dense deployments. In contrast, Chinchilla et al. use different channels and the quasi-orthogonality of different spreading factors [114]. The main challenge of scheduled MAC or slotted ALOHA approaches, however, is the device synchronization. In literature, the created overhead, the duty cycle requirement of the gateway, and the hidden node problem is often neglected. This gap in literature is closed by this thesis and in contrast to major literature, summarized in Table 4.1, the performance of random access, listen before talk, and a time scheduled approach is evaluated in detail and the coexistence of different channel access approaches is studied.

LoRaWAN collision studies can be conducted by measurements, simulations, or analytical approaches. For example, large scale measurements are conducted by many works, e.g., [185–188]. In addition, more specific impact factors on the quality of a LoRaWAN, such as the performance in critical environments [189–191], temperature studies [192, 193], or performance comparison between indoor and outdoor deployments [194, 195] are tackled. Large measurement studies are time-consuming and expensive. For that reason, other authors deploy simulation-based approaches for, among others, scalability in urban environments [196–198] or the influence of the spreading factor on message collisions [122]. Other authors simulate LoRaWAN with common simulation tools such as ns-3, summarized by da Silva in a comprehensive survey [152], with Matlab [199], or other approaches [200, 201]. A general summary on simulation tools used for the LoRaWAN technology is presented in [202, 203]. The simulators from literature often generate much overhead and rely on complex

Table 4.1: Overview of select related work (RA: random access, LBT: listen before talk, SC: time scheduled).

Reference		RA	LBT	SC	Energy	Main analysis
Bankov'16	[105]	✔	✘	✘	✘	network capacity
Leonardi'20	[169]	✔	✔	✘	✘	packet loss ratio
Ortin'19	[160]	✔	✔	✘	✘	data extraction rate, delay
Leonardi'23	[181]	✘	✔	✘	✘	message loss ratio, delay
Yapar'19	[172]	✔	✘	✔	✘	scalability, success rate, amount downlink traffic
Garrido'21	[173]	✔	✘	✔	✘	PDR
Casals'17	[161]	✔	✘	✘	✔	bit error rate, collisions, energy performance
Maudet'21	[184]	✔	✘	✘	✔	number of nodes, collisions, energy consumption
Bouguera'18	[162]	✔	✘	✘	✔	energy consumption
This chapter	–	✔	✔	✔	✔	collisions, amount downlink traffic, energy efficiency

simulation environments like OMNeT++ [204]. Besides many opportunities for detailed simulations, large simulations can require long runtimes and create massive overhead with many parameter setting possibilities. In contrast, the simulation approach for the channel access simulation in this chapter is very lightweight. It only simulates the transmitted data and potential collisions, similar to the collision simulation in Chapter 3.

From an energy consumption point of view, literature already studied the energy performance of a LoRaWAN in general [161, 205] or of a LoRaWAN in the Industry 4.0 [206]. In [184], the authors propose a redefined energy model for LoRaWAN that is validated using empirical measurements and also taking the number of nodes and the resulting collision probability into account. The authors of [207] present an empirical measurement study and investigate the energy consumption of LoRaWAN. Using these results, they analyze the bat-

tery lifetime in general and in a real life use case. Lastly, an ns-3 simulation of a LoRaWAN is implemented in [208], using real life measurement results to investigate the power consumption of the different states during a LoRa transmission. With particular focus on the LoRaWAN communication protocol, Banti et al. conduct a comprehensive survey under energy efficiency perspectives [209]. Energy consumption values for different sensors transmitting with LoRa are already measured [161, 162]. In addition, there are energy models with specific hardware available for LoRaWAN [162, 19] and an energy model for IoT exist in general [31], but not for LoRaWAN in particular. We aim to eliminate the gap in literature of a missing energy consumption model for devices with arbitrary hardware and for different channel access approaches. Furthermore, we define a very general energy efficiency metric usable for LoRaWAN. Our metric is able to describe the energy efficiency of different LoRaWAN channel access approaches, different transmission patterns, and different device behaviors.

4.2 Methodology for a Time Scheduled Channel Access in LoRaWAN

This section covers a theoretical consideration on a time scheduled approach for channel access in LoRaWAN to avoid message collisions completely. At the end of this section, details for the channel access simulation with different channel access approaches are given and evaluation scenarios are defined. Important notation used for the theoretical investigation of a time scheduled channel access approach is summarized in Table 4.2.

4.2.1 Role of Time on Air for Channel Access

The variation of different LoRa-specific transmission parameters, the transmitted payload, and the spreading factor lead to a different transmission airtime of LoRa messages as already introduced in Section 3.1. This ToA leads to a specific channel occupancy duration that is relevant to obtain information about

Table 4.2: Summary of important channel access notations. Constants in lowercase, random variables in capital letters.

Variable	Explanation
$t_{t,m}$	maximal ToA for message transmission from any device
$t_{s,m}$	maximal ToA for re-synchronization message transmission
$t_{d,m}$	maximal time drift in seconds between two transmissions
l_{slot}	time for minimal slot length
c	constant for clock drift randomness
T_{sync}	ToA for re-synchronization message
p_{sync}	re-synchronization probability of a message
d	duty cycle limit
d_t	duty cycle time frame
n	number of devices transmitting once per duty cycle time frame
$t_{\text{drift},x}$	constant time drift in seconds of device x
t_l	clock drift limit before re-synchronization
k_{sync}	reciprocal of re-synchronization probability

interference between simultaneously transmitted messages within the same frequency channel. For that reason, the ToA is used as main influencing factor for the following studies. Since 51 B is the largest possible payload of messages transmitted with spreading factor 12 in LoRaWAN, this is set as payload limit for the ToA calculation. This leads to a total ToA range from 0.029 s for 1 B transmitted with spreading factor 7 up to 3.023 s for 51 B transmitted with spreading factor 12 if the parameters are set in accordance with the previous chapter.

4.2.2 Time Scheduled Access for LoRaWAN

The performance of a time scheduled approach is dependent on the available slots and the quality of slot lengths fitting to the transmitted messages. Thus, we elaborate on optimal slot lengths and guard times for LoRa messages, considering synchronization, device's clock drifts, and duty cycle regulations.

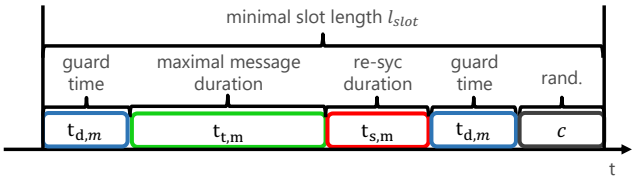

Figure 4.3: Visualization of the minimal slot length.

Slot Length

To avoid collisions in LoRaWAN, it must be guaranteed that no messages are transmitted at the same channel at the same time. The maximal possible number of messages that can be transmitted in a predefined time span without any overlap when a time scheduled channel access approach is used can be determined by the slot length and thus, the number of slots. Messages larger than their slots lead to systematic collisions with their slot neighbors. If no recovery mechanism is possible or available, these systematic collisions can prevent individual devices from being able to successfully transmit any message to the gateway. For that reason, this behavior can break the complete system and must be avoided. However, to set up a time scheduled channel access approach for LoRaWAN and avoid all collisions, a minimal slot length of l_{slot} with all slots having the same size can be determined by

$$l_{\text{slot}} = t_{t,m} + t_{s,m} + 2 \cdot t_{d,m} + c, \tag{4.1}$$

visualized in Figure 4.3. There, $t_{t,m}$ is the maximal ToA of any message in the network, $t_{s,m}$ is the maximal ToA for the re-synchronization messages, $t_{d,m}$ is the maximal constant clock drift of the devices in the LoRaWAN without included clock drift randomness, and c is a constant for additional clock drift randomness. Note, it is important to consider the maximal ToA, the maximal synchronization ToA, and the maximal clock drift to guarantee that no

device drifts out of its slot with equal sized slot lengths. We do not consider variable sized slots in the following to not additionally increase the synchronization overhead by the requirement of a detailed sensor to slot assignment based on the transmission ToA. The maximal ToA among all messages in the LoRaWAN is $t_{t,m} = 3.023\,\text{s}$ for 51 B payload transmitted with spreading factor 12 and the message parameter settings from the previous chapter. Furthermore, $t_{s,m} = 0.926\,\text{s}$ is the ToA for the largest re-synchronization message with spreading factor 12 and the smallest possible payload of 1 B up to 6 B. Note that messages with different payload sizes can have the same ToA dependent on the spreading factor. We assume that this small number of payload bytes is enough to re-synchronize an end device. To achieve the minimal slot length l_{slot}, it must be possible to receive re-synchronization messages at the end device immediately after the transmission is finished, as shown in Figure 4.3. Otherwise, an additional constant time must be added for the duration between message transmission end and reception window opening. Furthermore, additional transmission and processing delays are neglected in this example. Note that we assume that each device transmits once an hour in the following example slot length calculation. Other transmission rates for the devices are also practical, and it is only required to adjust the calculation of the clock drifts, and thus, the guard times of the slots accordingly. For an example sloth length calculation, we use a maximal constant clock drift $t_{d,m}$ of 100 ppm, equal to 360 ms per hour as reasonable value from literature [166]. Furthermore, it is not advisable to use the time scheduled channel access approach if clock drifts are very large and sensors regularly drift out of their assigned slots. To decrease the likeliness of collisions with the slot neighbors, the maximal clock drift $t_{d,m}$ is added before and after each slot as guard time. This also avoids collisions if the clocks of different devices drift in opposite directions which is very unlikely for similar climatic conditions [210]. Thus, this situation is not further investigated. As clock drifts are slightly variable for different temperatures [210], an additional clock drift randomness of $c = 10\,\%$ equal to 36 ms is added. In our example visualization in Figure 4.3, we place the additional randomness c after the second guard time

but placing it before the first one is also valid if the clock drifts in negative direction. By summing up all individual times that contribute to the required slot length, we achieve a minimal slot length $l_{\text{slot}} = 4.705\,\text{s}$ solving Equation 4.1. With this slot length, 765 devices can transmit one message per hour in 765 slots with perfect synchronization potential for every end device. However, a slot length of l_{slot} can require many re-synchronization messages if many messages are transmitted with the maximal ToA of $t_{t,m}$ and all sensors have large clock drifts. For that reason, we investigate the re-synchronization in more detail. Note, to simplify the following equations, we add a maximum clock drift randomness of $c = 10\,\%$ to the maximal clock drift $t_{d,m}$ hereafter. This adjustment does not influence the general calculation and can be changed according to real device clock drifts in a LoRaWAN.

Re-Synchronization Investigation

The duty cycle limits the total transmission time for all devices, and in particular also for all gateways to 0.1 %, 1.0 %, or 10 % per hour, dependent on the selected channel [163]. Thus, it also limits the number of messages a gateway can transmit in a specific time interval. If the number of devices transmitting messages is increasing or the clock drift of the devices is larger, more re-synchronization messages from the gateway are required. A general limit for the number of possible re-synchronization messages per duty cycle time frame d_t, in this work per hour ($d_t = 3600\,\text{s}$), based on the duty cycle d can be expressed by

$$n \cdot E[T_{\text{sync}}] \cdot p_{\text{sync}} \leq d \cdot d_t \tag{4.2}$$

with n transmitting devices, $E[T_{\text{sync}}]$ as expected message re-synchronization ToA, and p_{sync} as re-synchronization probability per transmitted message. The expected ToA for the re-synchronization message $E[T_{\text{sync}}]$ is dependent on the used payload and the spreading factor for the re-synchronization messages. The duty cycle is defined by the LoRaWAN regulations. Thus, it is only possible to vary the number of transmitting devices and the re-synchronization probabil-

ity. To analyze p_{sync} in detail, we assume again a transmission of one message per duty cycle time frame d_t per device. For different rates, the clock drift values between two transmissions must be adapted accordingly, and the calculation remains the same. The possible number of devices a gateway can re-synchronize is then dependent on the ToA of the re-synchronization messages and the tolerated p_{sync}. This relation is shown in Figure 4.4 for a duty cycle time frame $d_t = 3600\,\text{s}$. A maximal duty cycle for the gateway of 1 % per hour is used according to most channels. The ToA of the re-synchronization messages is calculated with 6 B payload for all spreading factors. This is the largest possible payload leading to the minimal transmission airtime using spreading factor 12. The tolerated p_{sync} drops drastically, especially when a large spreading factor is used for the re-synchronization messages. It is still possible to re-synchronize the device clocks of 500 devices per hour if the re-synchronization message is sent with spreading factor 7 and for 96.79 % of them if spreading factor 8 is used. Only 7.76% of 500 device clocks can be re-synchronized once an hour with spreading factor 12 without exceeding the duty cycle limits.

Re-Synchronization Relationship

The re-synchronization probability for each device is affected by two factors: the clock drift of the transmitting device x modeled as constant $t_{\text{drift},x}$ and the drift limit t_l before a re-synchronization is required. This relationship can be denoted as follows:

$$\begin{aligned}
t_{\text{drift},x} > t_l &\implies p_{\text{sync},x} = 1 \\
\frac{t_l}{2} < t_{\text{drift},x} \leq t_l &\implies p_{\text{sync},x} = 0.5 \\
\vdots &\quad \vdots \\
\frac{t_l}{k_{\text{sync}}} < t_{\text{drift},x} \leq \frac{t_l}{k_{\text{sync}} - 1} &\implies p_{\text{sync},x} = \frac{1}{k_{\text{sync}}}
\end{aligned}$$

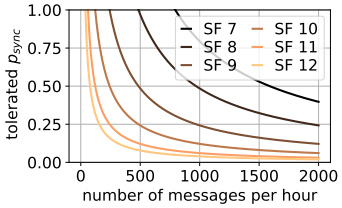

Figure 4.4: Tolerated p_{sync} for different numbers of messages.

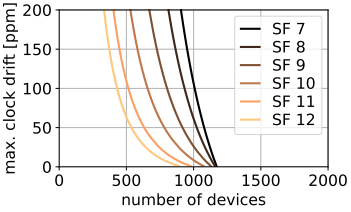

Figure 4.5: Allowed $t_{d,m}$ for different numbers of messages.

If the clock drift of the transmitting device $t_{\text{drift},x}$ is larger than the drift limit t_l, a re-synchronization message must be sent after each transmission. This leads to $p_{\text{sync},x} = 1$ for device x. The clock must be re-synchronized after every second message if $t_l/2 < t_{\text{drift},x} \leq t_l$, etc. Thus, to guarantee that each device in the network can be re-synchronized, we achieve

$$t_{d,m} \leq \frac{t_l}{k_{\text{sync}} - 1} \quad \Longrightarrow \quad p_{\text{sync}} \leq \frac{1}{k_{\text{sync}}} \tag{4.3}$$

where $t_{d,m}$ is the maximal clock drift of all devices in the network and p_{sync} is the re-synchronization probability of the devices. This solves to

$$t_{d,m} \leq \frac{t_l}{k_{\text{sync}} - 1} = \frac{t_l \cdot p_{\text{sync}}}{1 - p_{\text{sync}}} \tag{4.4}$$

for a general relationship between the maximal clock drift $t_{d,m}$, the drift limit t_l, and the re-synchronization probability p_{sync}. Note, if we have a continuous uniform distribution of the clock drifts T_{drift} between $[a, b]$ of all devices in the network with $E[T_{\text{drift}}] = \frac{a+b}{2}$, we can also replace $t_{d,m}$ with $E[T_{\text{drift}}]$ as expected clock drift in Equation 4.4. However, this would leave no drift space if clock drifts change over time. For that reason the maximal clock drift $t_{d,m}$ is used in the following.

Lastly, as the goal is to calculate the maximal possible clock drift $t_{d,m}$ to re-synchronize all sensors dependent on the drift limit t_l and the re-synchronization probability p_{sync}, we achieve $t_{d,m}$ by

$$t_{d,m} = \frac{t_l \cdot p_{\text{sync}}}{1 - p_{\text{sync}}}, \tag{4.5}$$

instead of the inequality in Equation 4.4.

General Time Scheduled Relationship

Based on the consideration of the clock drift and the re-synchronization probability, we aim at having a general relationship between the clock drifts and the ToA of the transmitting device, the ToA of the re-synchronization messages, the duty cycle, and the duty cycle time frame. The goal is to determine the possible number of end devices that are able to transmit per duty cycle time frame without exceeding the duty cycle limits, having no collisions in the channel. As worst case analysis, and to have equal sized slots in our time scheduled approach again, we use the maximal time drift $t_{d,m}$, the maximal message ToA $t_{t,m}$, and the maximal re-synchronization ToA $t_{s,m}$ for the following calculation again. This leaves free duty cycle space for potential additional gateway messages or re-transmissions if not all devices use these maximal values for their transmission. Having n end devices, we get

$$\frac{d_t - (t_{t,m} + t_{s,m} + t_{d,m}) \cdot n}{n} = t_l \tag{4.6}$$

as total available drift time t_l per sensor. The total duty cycle time frame usable by all sensors is d_t. If we subtract $t_{t,m} + t_{s,m} + t_{d,m}$ for all sensors, that is required for the transmission, potential re-synchronization messages, and the largest possible clock drift $t_{d,m}$ as guard time, we get the available drift time for all sensors. Dividing the result by n, we get t_l per sensor. Next, solving Equation 4.5 for t_l, we get $t_l = \frac{t_{d,m}}{p_{sync}} - t_{d,m}$. Adding this to Equation 4.6, we achieve

$$\frac{d_t - (t_{t,m} + t_{s,m} + t_{d,m}) \cdot n}{n} = \frac{t_{d,m}}{p_{sync}} - t_{d,m}. \tag{4.7}$$

Then, we can calculate the possible number of re-synchronization messages per hour by dividing the available time for re-synchronization $d \cdot d_t$ by the time for one re-synchronization message $t_{s,m}$. Thus, for all devices n, we achieve

$$p_{sync} = \frac{d \cdot d_t}{t_{s,m} \cdot n} \tag{4.8}$$

If we include this in Equation 4.7, we achieve

$$\frac{d_t - (t_{t,m} + t_{s,m} + t_{d,m}) \cdot n}{n} = \frac{t_{d,m} \cdot t_{s,m} \cdot n}{d \cdot d_t} - t_{d,m} \tag{4.9}$$

Solving this for n, we achieve

$$n = \left\lceil \frac{-d \cdot d_t(t_{t,m} + t_{s,m}) \pm \sqrt{k_1}}{2 \cdot t_{s,m} \cdot t_{d,m}} \right\rceil, \tag{4.10}$$

with

$$k_1 = (d{\cdot}d_t{\cdot}t_{t,m} + d{\cdot}d_t{\cdot}t_{s,m})^2 + 4d(d_t)^2 t_{d,m}{\cdot}t_{s,m}, \tag{4.11}$$

and $nddt \neq 0$, $t_{d,m}t_{s,m} \neq 0$. Thus, the limiting number of devices n transmitting once an hour can be determined dependent on the message and re-synchronization ToA limit, the duty cycle, and the clock drift limits of the devices. The result is presented in Figure 4.5 for a duty cycle limit of 1 % per hour with the maximal clock drift $t_{d,m}$ on the y-axis and the possible number of devices transmitting once an hour on the x-axis. The colors depict the used spreading factor for the re-synchronization message with darker colors for smaller spreading factors. For 100 ppm clock drift, 431 devices can be re-synchronized per hour if spreading factor 12 is used for the re-synchronization. Using spreading factor 7 for the re-synchronization, it is 1,008 devices. Thus, we can answer the first part of our first research question RQ4.1 with *yes, if the number of messages is chosen according to Equation 4.10, we can avoid collisions completely.*

4.2.3 Channel Access Simulation

To evaluate the performance of the time scheduled channel access approach in coexistence with other approaches, this section introduces the channel access simulation methodology and assumptions for the simulation.

General Simulation Idea

The simulation idea for LoRa messages is similar to the simulation from the previous chapter for a single gateway, introduced in Section 3.3.1. Again, one simulation iteration can be specified as a specific time frame during the complete simulation. This time frame is selected and simulated at once. Each sensor transmits once per time frame, while the transmission strategy is dependent on the channel access approach. In the following, we set the time frame to one hour. Note that different time frames work accordingly while duty cycle constraints must be recalculated then. In the following, details on the message, collision, and location simulation are described and simulation specific settings for the different channel access approaches are introduced.

Location and Transmission Distance: In a real network, devices that transmit messages are located in geographically distributed locations. Thus, not each device can hear all messages of a network because of the hidden node problem. To emulate this behavior, all devices are allocated at an x and y coordinate with the following strategy: first, the maximal transmission distance d_M of devices with spreading factor 12 is calculated with the Hata model for the same RSSI values introduced in the previous chapter in Equation 3.5. Both, the gateway and the device height is set to 3 m. Note that any other model, parameter setting, or maximal distance is reasonable too and will not change the conclusions. The device height and the gateway height are set to the same values to guarantee bidirectional communication and allow re-synchronization of all devices. This leads to different maximal transmission distances compared to the settings of Chapter 3, where the device and gateway height is different. For spreading factors 7

to 12, it is 714.64 m, 843.14 m, 994.75 m, 1,173.63 m, 1,240.12 m, and 1,463.11 m, respectively. Afterwards, a gateway is placed in the middle of the possible transmission area and all devices are randomly placed around the gateway within a maximal radius of d_M. Note that other device placement strategies are also applicable. Each device can transmit to the gateway in the following simulation but not every message is heard by any device. In a last step, the distance to the gateway is calculated for each device and the minimal possible spreading factor to access the gateway is assigned to the device. During the simulation, all messages are then transmitted with that assigned spreading factor.

Message Simulation and Collision: The transmission start time calculation for all messages in each iteration is different, dependent on the channel access approach. For random access and listen before talk, random transmission start times are calculated for each simulation iteration between 0 s and 3600 s, similar to the simulation in the previous chapter. For the time scheduled approach, the one-hour time frame of one simulation iteration is divided in n individual equal sized slots. Then, each device is assigned to one slot, where it transmits in. The start timestamp of the first transmission of each device is uniformly random calculated between the slot start timestamp and the first possible timestamp that would lead to a re-synchronization after the first transmission. This avoids an unrealistic start condition, where all devices start their transmission at the beginning of the slots without any time drift. Thus, no transient phase in the simulation is detected for the time scheduled approach and no simulated transmissions need to be discarded.

The ToA of all messages is calculated similar to the previous chapter, according to Equation 3.4. Each device achieves a random payload between 1 B and 51 B and the spreading factor according to the device's distance calculated above. The transmission end time is achieved by adding the ToA to the transmission start time. After each transmission with the time scheduled approach, the next transmission start timestamp is delayed by the pre-defined device clock drift. If the device drifts out of its slot, it is re-synchronized. Since larger spreading fac-

tors are more robust against interference [99], the re-synchronization messages are transmitted using spreading factor 12 and 6 B payload leading to a ToA of 0.926 s using spreading factor 12. Lastly, message collisions are again calculated by overlapping intervals as introduced in the previous chapter, if no listen before talk is used and a back-off is possible.

Listen before Talk: If listen before talk is used, each device listens to the channel before it transmits its messages to determine whether it is already occupied. If it is free, the transmission is started and otherwise, the message is delayed according to a predefined back-off strategy and the device starts a new transmission attempt. However, in a real world deployment, each device has a geographic location where it transmits from and a specific distance to the gateway. Thus, only other devices in close distance to the transmitting device can hear potential messages. If another device is outside this transmission radius but transmits to the same gateway, collisions can occur because of the hidden node problem. We see in [7] that a random back-off delay between 0.4 s and 1.75 s shows good results. For that reason, we use this delay in the following and refer to [7] for more details.

Message Recovery: Without additional message recovery mechanisms, colliding messages are always lost. This is especially crucial for re-synchronization messages if the time scheduled channel access approach is used. However, not all colliding messages are automatically lost according to literature [99, 18]. The simulation in this work is extended by such a recovery approach. If the recovery is used, it is assumed that messages transmitted with higher spreading factors always dominate messages transmitted with lower spreading factors and are thus, transmitted correctly [99]. Messages with the smaller spreading factor are lost. Especially the time scheduled approach benefits from this behavior, as the important re-synchronization messages are transmitted with spreading factor 12.

4.2.4 Channel Access Scenario Overview

The theoretical observation shows that the time scheduled approach has the potential to avoid all collisions in LoRaWAN. For that reason, the focus of the scenario study is on the coexistence analysis between the time scheduled approach and random access or listen before talk. This can emulate the behavior in a real network with cross-traffic. For each parameter combination, a simulation of 200 h is conducted that is long enough to cover all possible effects of the transmissions. Ten re-configurations are performed for each simulation, where slots and clock drifts are newly assigned to the devices and locations are newly calculated for the listen before talk devices. This avoids potential errors by a randomly chosen well suited setup. In all scenarios, the clock drifts are applied per device randomly between 0 ppm and a pre-defined clock drift limit $t_{d,m}$. This prevents all devices drifting with the same clock drift that is not realistic in reality. The drift limits are between 2 ppm and 150 ppm. The possible number n of devices per clock drift limit using the time scheduled approach is summarized in Table 4.3. In the following, details about all scenarios are given.

Scenario S 1: Time Scheduled and Random Access

Scenario S 1 simulates the time scheduled approach in coexistence with random access. Accordingly, the number of messages per hour transmitted with time scheduled are adapted according to the findings in Equation 4.10. For all messages, the clock drift is assigned randomly between 0 ppm and the limit for the specific simulation. Additionally, a 10 % randomness in positive and negative direction is added. Scenario S 1 has three sub-scenarios, S 1.1, S 1.2, and S 1.3.

Scenario S 1.1: The time scheduled approach is seen as main channel access approach, while random access is additional cross-traffic. The number of messages n transmitted with the time scheduled approach for specific maximal clock drifts $t_{d,m}$ is selected according to the values in Table 4.3. Additionally, 1 % – 50 % random access cross-traffic is added and thus, more messages must be handled

Table 4.3: Time scheduled device number n per clock drift limit $t_{d,m}$ in ppm.

$t_{d,m}$	150	125	100	75	50	25	20	15	10	5	2
n	373	398	431	475	538	644	676	714	761	824	872

by the channel. For example for a 50 ppm clock drift with 50 % random access traffic, 538 time scheduled and 269 random access messages are transmitted. The highest number of messages transmitted with random access for all scenarios is 436 with 50 % random access traffic in scenario S 1.1. The cross-traffic is chosen to not overload the network. If only random access traffic is simulated, the collision probability is always less than 15 % for all settings.

Scenario S 1.2: The maximum number of devices transmitting in the network is set to the numbers listed in Table 4.3 for specific maximal clock drifts $t_{d,m}$. If more random access traffic is transmitted, less traffic is sent with the time scheduled approach and vice versa. For example, for 50 % random access traffic and a clock drift of 50 ppm, 269 random access messages, and 269 time scheduled messages are transmitted summing up to 538 messages, as listed in Table 4.3.

Scenario S 1.3: The number of messages is analogous to S 1.2. In contrast, the slot length is adapted towards the number of messages sent with the time scheduled approach. If 50 % random access traffic is in the network, only 50 % of the original time scheduled traffic is in the network, and thus, the slot lengths are doubled. For each sub-scenario, in total, eleven different drift settings with eleven different random traffic percentages, ten re-configurations of clock drifts and slot allocations, and 200 h simulation per setting is performed. This sums up to a simulation of more than 230,000 messages per device for each sub-scenario.

Scenario S 2: Time Scheduled and Listen before Talk

In scenario S 2, time scheduled and listen before talk is used for channel access since listen before talk improves the collision probability against random access. Scenario S 2 is similar to S 1 with three sub-scenarios, S 2.1, S 2.2, and S 2.3 and the same settings as S 1. The goal is to determine if listen before talk cross-traffic coexists better with time scheduled. Again, more than 230,000 messages are simulated per device per sub-scenario.

4.3 Channel Access Simulation Results

This section presents evaluation results for the defined channel access scenarios. The main goal of the scenario study is to answer the question, whether the time scheduled approach can coexist with cross-traffic.

4.3.1 Scenario S 1: Time Scheduled and Random Access

The percentage of random access traffic and the clock drifts of the devices transmitting with the time scheduled approach are the main influencing factors in scenario S 1. The results of this simulation study are presented in the following.

Traffic Study

The collision probability for different random access traffic percentages is shown in Figure 4.6. The box plots show the percentage of random access traffic on the x-axis and the collision probability on the y-axis. One box includes the collision probability results for all possible clock drifts and re-configurations. The different colors show the different sub-scenarios as described in Section 4.2.4. The collision probability significantly increases with the percentage of random access traffic, in the first glance independently of the sub-scenario. For up to 15 % random access traffic, no clear difference between the sub-scenarios is visible. A median collision probability of less than 5 % is only achieved for less than 10 %

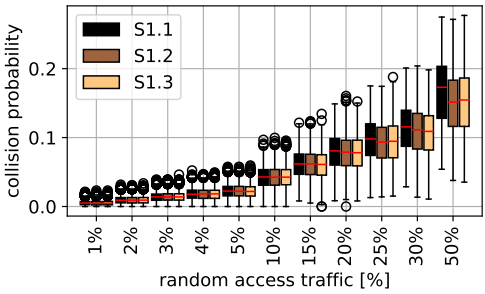

Figure 4.6: Time scheduled and random access traffic study.

random access traffic. Larger differences between the sub-scenarios are achieved for more random access traffic and the percentage of random access traffic is the largest influencing factor on the collision probability. In contrast, the total number of messages has a smaller influence. For 50 % random access traffic, in S 1.1 50 % more messages are transmitted than in S 1.2 or S 1.3; however, the median collision probability is increased by less than 2 %. Since we see in [7] that the total number of transmitted message per hour has a larger impact on the overall collision probability than the clock drift of the devices, we omit details for this study here and further investigate the collisions.

Individual Collision Study

To quantify the performance of a channel access approach, it is essential to know which messages collide. This is shown for S 1.1 in Figure 4.7. The collision probability for random access messages, shown by the black boxes, is much higher than the time scheduled traffic shown by the brown boxes. The median collision probability is larger than 20 % for all random access traffic values and increases slightly starting from 4 % random access traffic. The variance of the result is large, especially for few transmitted random access messages. In contrast, the

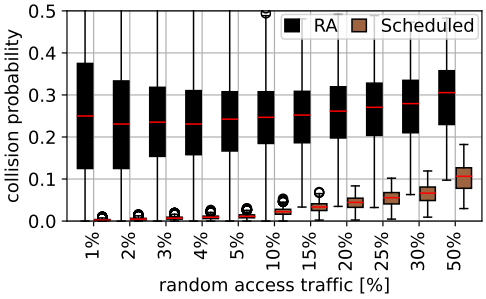

Figure 4.7: Collision by access type.

time scheduled approach performs much better for all percentages even if 50 % random access messages are sent. To quantify the quality of the time scheduled channel access in coexistence with random access, not only the collision probability for random access and time scheduled messages is taken into consideration. It is important to know whether time scheduled messages collided with random access messages or other time scheduled messages. If two time scheduled messages collide, a complete impairment of the general access methodology can happen. If the clock of one device drifted that far out of its channel without re-synchronization because of frequent collisions of the re-synchronization messages, two time scheduled messages collide. In the worst case, this collision re-occurs frequently each hour and makes re-synchronization impossible. This is called *systematic collision* in the following and is studied in addition to random access and time scheduled collisions, and collisions of re-synchronization (sync) messages. With regard to systematic collisions, for scenario S 1.1, a maximal collision probability of 0.54 % for a single iteration with 50 % random access traffic is achieved. For 1 % random access traffic, a systematic collision probability of 0.0003 % is achieved. A large increase is not detected with less than 30 % random access traffic. Thus, the re-synchronization is still stable, although random access messages are sent in the channel.

Table 4.4: Relative collision probability improvement for sub-scenarios and SF-based message recovery in comparison to baseline scenario S 1.1.

	all	random access	time scheduled	systematic	sync
S 1.1 SF	42.8 %	47.6 %	41.5 %	99.9 %	83.0 %
S 1.2	4.4 %	12.5 %	0.6 %	98.1 %	1.4 %
S 1.2 SF	42.9 %	47.8 %	41.2 %	97.2 %	83.4 %
S 1.3	4.4 %	12.6 %	0.4 %	47.2 %	−0.3 %
S 1.3 SF	43.3 %	48.3 %	41.2 %	96.0 %	83.2 %

Sub-Scenario Study

As a next step, it is determined whether the different sub-scenarios can significantly improve the collision probability. Furthermore, it is analyzed if the message recovery due to a larger used spreading factor can improve the results. Table 4.4 lists the decrease in collision probability for different scenarios compared to scenario S 1.1 as baseline. Each line in the table shows one scenario. The term *SF* suggests message recovery because of higher spreading factor. There, we assume message recovery potential for transmissions with the larger spreading factor if two messages collide. The columns present the collision probability decrease in percent, for *all*, *random access*, *time scheduled*, and *systematic* collisions. The last column shows the result for the synchronization message losses.

It is shown that all scenarios improve compared to the baseline scenario S 1.1. The largest improvement is visible for the systematic collisions. This is for two reasons: first, for S 1.2 and S 1.3, fewer messages per hour must be transmitted and thus, it is less likely that the clock of one device drifts out of the slot before re-synchronization is performed. Second, the spreading factor based recovery works especially good for the re-synchronization messages since they are transmitted with spreading factor 12. Furthermore, with regard to systematic collisions, it is visible that S 1.2 without recovery performs much better than

S 1.3. The larger slots in S 1.2 increase the number of re-synchronization messages that can collide before a systematic collision occurs. Since no other collision type shows a significant difference between S 1.2 and S 1.3, S 1.2 is always preferable. The spreading factor-based recovery further improves the result for all scenarios and all random access percentages. The highest improvement is achieved for systematic and sync collisions. The collision probability reduction between S 1.1 and S 1.1 SF is more than 40 % for all message types. Only 0.25 % of all time scheduled sync messages collided in S 1.1, while an improvement of 83 % is achieved for S 1.1 SF with only 0.14 % collisions. Furthermore, all approaches are stable and do not exceed duty cycle limitations. For each scenario, the clock drift is chosen randomly between 0 ppm and the set limit of 2 ppm up to 150 ppm. Because of the random drift, on average only a maximum of 50 % from the available gateway duty cycle is used for re-synchronization, as expected.

4.3.2 Scenario S 2: Time Scheduled and Listen before Talk

The coexistence simulation of time scheduled and listen before talk shows similar results as the random access and time scheduled study. However, since listen before talk avoids some collisions, the overall collision probability is smaller. Since differences are small, the results are only compared to the findings for scenario S 1. Figure 4.8 shows the collision probability for the coexistence study between time scheduled and random access traffic (S 1.1, black) and time scheduled and listen before talk traffic (S 2.1, brown). The y-axis presents the collision probability and the x-axis the cross-traffic other than time scheduled traffic.

The figure shows that S 2.1, where the cross-traffic is simulated with the listen before talk approach, always performs better than S 1.1, where the cross-traffic is from the random access approach. The difference is increasing with the percentage of not time scheduled traffic. This is expected since listen before talk only avoids certain collisions and does not influence other access methodologies with additional messages or overhead. Thus, for 10 % cross-traffic, the median collision probability for S 1.1 is 3.44 %, for S 2.1 it is 2.96 %, and the mean colli-

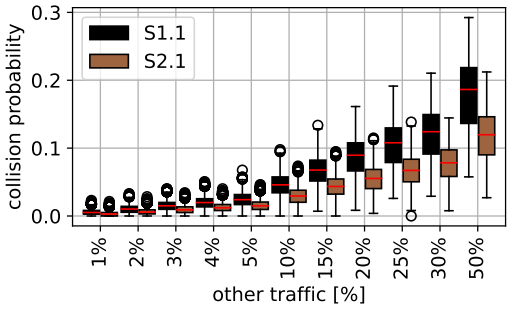

Figure 4.8: Comparison of time scheduled with random access (S 1.1) and listen before talk (S 2.1).

sion probability is 3.46 % and 2.94 %. The mean collision probability difference is more than 5 % for 50 % cross-traffic.

For that reason, we can answer the second part of our first research question RQ4.1 as follows: *In general, the results show that the time scheduled channel access approach can only coexist with random access or listen before talk as cross-traffic in a meaningful way if the predominant form is the time scheduled approach. Otherwise, there is an increasing risk of message collisions or, in the worst case, systematic collisions of time scheduled messages. This can lead to a loss of all messages sent in specific slots. However, we see that the time scheduled approach always outperforms random access and listen before talk by means of collision probability if used in coexistence.*

4.4 LoRaWAN Channel Access Energy Model

Different channel access approaches do not only influence the collision probability but also the energy consumption of sensors in a LoRaWAN. For that reason, this section presents the methodology to quantify the energy consumption for LoRa messages transmitted with different channel access approaches by a simple process diagram first. Afterwards, the idea is extended to be parameterized for arbitrary hardware and a metric to quantify energy efficiency of LoRa transmissions is introduced. Based on this generic energy model, the energy requirements of the channel access approaches discussed above are modeled.

4.4.1 Energy Consumption for LoRaWAN Channel Access

LoRaWAN channel access can be displayed as a process diagram, dependent on the access mechanism. Figure 4.9 shows this diagram of a transmission cycle for listen before talk (LBT), random access (RA), time scheduled (Scheduled), and slotted ALOHA (Slotted). The relevant part of a LoRaWAN transmission cycle with respect to energy consumption starts when the transceiver is powered and ends before the sensor returns to sleep. Table 4.5 summarizes the states during a message transmission with an example range of theoretical (t) and experimental (e) (if available) current consumption from literature [184] for comparison. Note, we model the energy consumption of one transmission for different channel access approaches and introduce an abstraction that allows the aggregation of different individual states with a different behavior for energy modeling.

Process Diagram

A general LoRaWAN channel access process diagram can be established with the following states, as shown in Figure 4.9. After the transceiver is powered on in state (S1), the channel access approach is selected. Note, devices do not perform this distinction in general, but we aggregate multiple channel access mechanisms in the same diagram to save space.

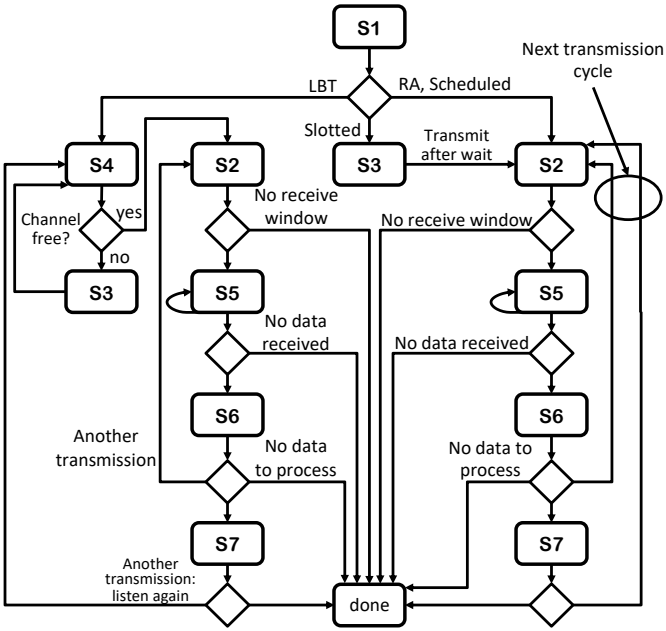

Figure 4.9: General LoRaWAN channel access process diagram.

Random Access and Scheduled: After the transceiver wakes up (S1), a data transmission (S2) is immediately started if random access or time scheduled is used. Afterwards, optional receive windows may be opened (S5), data is received (S6), and further processing is performed (S7). When data is received or processed, another transmission cycle is possible, shown by the arrows back to (S2) after (S6) and (S7). The difference between random access and time scheduled is the state possibilities. Random access can be used without receive windows or data reception but for time scheduled, both (S5) and (S6) are required after the transmissions to receive potential re-synchronizations. This can also lead to a higher probability for state (S7) where received data is processed.

Table 4.5: Transmission states and current consumption ((t): theoretical, (e): experimental).

State	Name	Description	Exemplary current consumption [184]
S1	wake up	activation of transceiver module	(e): 2.268 mA
S2	transmit	transmission of payload data	(t): 18 mA – 28 mA (e): 21.86 mA – 39.43 mA
S3	waiting	waiting before next state transition	(e): 2.033 mA
S4	listen	channel sensing for ongoing transmission	(t): 10.5 mA (e): 10.76 mA – 11.12 mA
S5	open receive window	preparation for listening to incoming messages	(e): 1.996 mA
S6	receive	active receiving of a message	(t): 10.5 mA (e): 10.76 mA – 11.12 mA
S7	processing	processing of data before next transmission	task dependent

Slotted ALOHA: In contrast, the sensor waits (S3) for the next sending slot before it starts its transmission if slotted ALOHA is used. In the wait state, we assume the sensor being in stand-by mode. Afterwards, the state transition is according to random access or time scheduled. However, if another transmission cycle is started after (S6) or (S7), potentially another waiting state (S3) is required (this is not added to the diagram since it is only optional for slotted ALOHA). This depends on the length of the slots and the ToA of the already transmitted messages in this slot.

Listen before Talk: When listen before talk is used, first, it is determined whether the channel is occupied (S4). If this is the case, the sensor waits (S3) until the next channel sensing is performed. If the channel is free, the sensor starts a transmission, similar to the random access procedure. Furthermore, receiving data in state (S6) can be seen as channel sensing and the next transmission can immediately start. Thus, a transition between (S6) and (S2) is possible. Note that this transition is only possible if data reception is similar to channel sensing. If this is not possible, a transition to state (S4) is required. A transition from (S7) to (S4) is possible if additional data is processed. Here, we assume that additional channel sensing is required after a specific time of data processing.

Energy State Reduction: While all states in the process diagram are required for a comprehensive description of a LoRa transmission, several states can be aggregated from an energy consumption perspective. First, each *open receive window* is practically spoken a *wait* and a *listen* operation if no data is received, and a *receive* operation if data is received. Furthermore, each *listen* operation is similar to *receive*. In addition, the *wake up transceiver* state (S1) is required for all channel access mechanisms, and thus independent of them. For that reason, it is not further investigated.

4.4.2 Energy Model Description

This section introduces a general energy model for LoRaWAN channel access approaches. First, the energy consumption for a LoRa message transmission is modeled. Afterwards, different channel access specific characteristics are explained and modeled in detail. The energy models from literature lack in two main situations: (1) a more complex behavior during channel access, i.e., multiple 'listening on the channel' operations, wait times, or optional states that do not occur for all transmissions are not properly covered and (2) a comparison of different channel access approaches without specific hardware specifications and thus, the energy requirement between them is not possible. To overcome these limitations, we express the transmission behavior for different channel

access approaches in more detail in the following. This includes the specific behavior for different channel access approaches when messages collide, during listening on the channel, waiting for a transmission, and if a time synchronization is required. According to literature [7], listen before talk and time scheduled are the most promising alternatives to current random access. For that reason, these approaches are studied in more detail in this work. Since slotted ALOHA is only suggested in very specific scenarios, in which the possible message ToA has a small range [7, 20], it is omitted in the following consideration.

LoRa Transmission Processes

Different channel access approaches can be compared by analyzing the sensor's behavior during *transmit*, *wait*, *receive*, and *process* [19]. The probability to start one of these procedures or processes, the duration of each process, and the energy consumption of a sensor in each process are of major interest. However, the energy requirement to process any received data is not directly related to the transmission behavior itself and is highly dependent on the used hardware. To cover this, a random variable E_0 is added with $E_0 = p_0 \cdot T_0$ to model every additional energy consumption with a constant power p_0 and a random time T_0, being the time between two data transmission starts of the sensor. It covers, among others, the sensor's general energy requirement for data measurement, processing of any data, and the energy consumption during the sleep mode. In this work, no parameter study of the named procedures is done as the energy consumption of them is independent of the channel access approach. Please note that we omit the analysis of minor changes in the duration of the sleep mode as a result of different channel access approaches, since the energy consumption for idle or sleep modes is several orders of magnitude smaller compared to all active processes [162, 184, 207]. The general equation for the total energy consumption E_{total} based on the energy consumption for transmit (E_1), wait (E_2), and receive (E_3) and independent of the access approach is given by

$$E_{\text{total}} = E_1 + E_2 + E_3 + E_0. \tag{4.12}$$

General Notation Details

In the following, this very general equation is further analyzed. All important variables for the following energy model with an explanation are summarized in Table 4.6. The convention with the indices is as follows. Every variable with a single index (e.g. E_i) describes the behavior in a specific process with $i = 1$ for transmit, $i = 2$ for wait, $i = 3$ for receive, and $i = 0$ for channel access independent procedures. The second index is for channel access specific variables (e.g. $E_{i,j}$) with $j = \mathrm{ra}$ for random access, $j = \mathrm{lbt}$ for listen before talk, and $j = \mathrm{sc}$ for the time scheduled approach.

LoRaWAN Energy Model

The total energy consumption E_{total} described in Equation 4.12 is still dependent on the used channel access approach. We can describe the required energy for a LoRa transmission by the power p_i over time T_i the end device is in the different processes i. Note, since we do not consider any power peaks, e.g. at the beginning of a transmission, we use a constant average power consumption p_i in process i. Then, we normalize the time with a standard time t_{min} during the well known transmission process. Therefore, we use the minimal ToA $t_{\mathrm{toa,SF=7,sy=1}} = t_{\mathrm{min}}$, a sensor requires transmitting data in LoRaWAN for time normalization. This is the ToA to transmit one symbol sy with spreading factor SF 7 using random access without wait, receive, or any collisions. We obtain

$$E_{\mathrm{total}} = \sum_{i=1}^{3} p_i \cdot T_i + E_0 = \sum_{i=1}^{3} p_i \cdot t_{\mathrm{min}} \cdot \frac{T_i}{t_{\mathrm{min}}} + E_0 \tag{4.13}$$

$$= \sum_{i=1}^{3} E_i^* \cdot T_i^* + E_0 \tag{4.14}$$

with E_i^* as required energy for the minimal duration t_{min} in process i and T_i^* as normalized time in process i. With this normalization by the minimal transmission ToA t_{min}, we can directly express the difference between the investigated

Table 4.6: Summary of important energy model notations. Random variables in capital letters, constants in lowercase letters.

Variable	Explanation
E_i	energy consumption in process i
E_0	channel access independent energy consumption between two transmission starts
E_i^*	energy in process i in time $t_{\min}$
T_i	time in process i
$t_{\min}$	minimal transmission ToA for normalization
T_i^*	normalized time in process i
p_i	required constant average power in process i
T_{toa}	ToA for transmission
c_i	constant scaling parameter
E_{total}	energy consumption of the complete transmission cycle
E_{lora}	energy consumption of all transmission cycles in the complete LoRaWAN
E_{eff}	energy efficiency of the LoRaWAN
	Channel access specifics
N_j	req. number of receive windows if access approach j is used
$T_{i,j}$	time in process i if access approach j is used
$t_{i,j,o}$	constant time in process i with access approach j if receive window is opened
$T_{\text{drift,total}}$	clock drift since last re-synchronization
t_{slot}	slot length in seconds
T_{rcf}	duration between drift out of a slot and actual successful re-synchronization
T_{drift}	time drift in seconds between two transmissions
$p_{\text{sync,sc,perfect}}$	probability to receive sync message in perfect time slotted
$p_{\text{sync,sc}}$	probability to receive sync in collision affected time slotted
$p_{\text{coll,sync}}$	collision probability of sync message
X	number of listening windows
$p_{\text{coll,lbt}}$	probability detecting a listen before talk channel as occupied when listened on the channel

transmission and the best case with $T_1 = t_{\min}$ and no overhead from other processes. Dependent on the underlying hardware, the required energy in the different processes can vary, while the normalized time T_i^* is usually dependent on the channel access approach and the amount of transmitted data. For example, if a transmission takes twice as long as the above-mentioned optimal one, e.g. due to double the number of transmitted symbols, we achieve $T_1^* = 2$ for the transmission process.

Next, we express differences in the energy consumption between transmit, wait, and receive for the same duration with constant scaling parameters c_2 for wait and c_3 for receive. These scaling parameters are the energy cost differences for wait and receive respectively compared to transmit. They can reflect the influence of different sensors or the difference in the energy consumption between processes and need to be adjusted based on the underlying hardware. In this way, we can describe the energy consumption of all processes in relation to the consumption E_1^* for a data transmission with

$$E_{\text{total}} = E_1^*(T_1^* + c_2 \cdot T_2^* + c_3 \cdot T_3^*) + E_0. \tag{4.15}$$

If a message transmission of one second requires ten times more energy than waiting for one second, c_2 is equal to 0.1.

Process Duration Modeling

While the total energy consumption of a LoRa transmission cycle is described in Equation 4.15, the different behavior during each process dependent on the used channel access approach is not covered yet. The main influencing factor of the different processes on the overall energy consumption is its individual duration. As introduced above, the transmit process is always required during a LoRa message transmission cycle. Its duration is the transmission ToA T_{toa} of the transmitted message, and it is independent of the channel access approach.

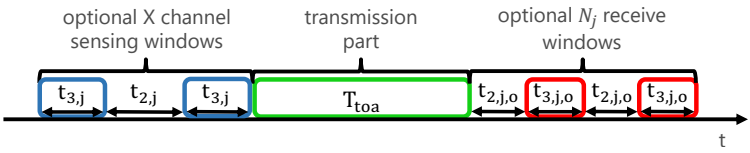

Figure 4.10: Visualization of transmission cycle.

Thus, we can describe the ToA for all approaches as

$$T_{\text{toa}} = T_{1,\text{ra}} = T_{1,\text{lbt}} = T_{1,\text{sc}} = T_1. \tag{4.16}$$

However, the total duration for wait and receive during a complete transmission cycle is dependent on the channel access approach. An example transmission cycle is visualized in Figure 4.10. Ahead of the actual data transmission, shown in green, it is possible to open optional listening windows in blue, as it is done with listen before talk. After a transmission, optional receive windows are possible, shown in red. In the following, the optional listening and receive windows are investigated for each approach individually. In particular each duration that is relevant for the energy consumption calculation is studied.

Random Access: Random access has the least overhead if data is transmitted without any possibility to receive messages and the sensor immediately returns to sleep mode after a transmission. Besides this simple procedure, a transmission with random access can contain an optional data reception possibility, where one or more receive windows are opened. This adds one or more wait and reception process to the actual data transmission, since according to LoRaWAN standardization, the sensor waits for a pre-defined duration ahead of each reception window [164]. This is symbolized by the optional receive windows after the transmission part in Figure 4.10.

The wait time $T_{2,\mathrm{ra}}$ during a transmission cycle can then be calculated by

$$T_{2,\mathrm{ra}} = N_{\mathrm{ra}} \cdot t_{2,\mathrm{ra},o} \tag{4.17}$$

with N_{ra} as random variable for the number of receive windows and $t_{2,\mathrm{ra},o}$ as pre-defined constant single wait time. Considering all transmissions in the network where random access is used, the expected total wait time $E[T_{2,\mathrm{ra}}]$ is

$$E[T_{2,\mathrm{ra}}] = E[N_{\mathrm{ra}}] \cdot t_{2,\mathrm{ra},o}. \tag{4.18}$$

The random variable to describe the reception time $T_{3,\mathrm{ra}}$ during a transmission process with random access can be defined by

$$T_{3,\mathrm{ra}} = N_{\mathrm{ra}} \cdot t_{3,\mathrm{ra},o} \tag{4.19}$$

with $t_{3,\mathrm{ra},o}$ as duration of the opened reception window. Again, considering all transmissions in the network, the expected reception time $E[T_{3,\mathrm{ra}}]$ during a transmission procedure if random access is used can be defined by

$$E[T_{3,\mathrm{ra}}] = E[N_{\mathrm{ra}}] \cdot t_{3,\mathrm{ra},o}. \tag{4.20}$$

Time Scheduled: Next, the wait and reception time is analyzed if the time scheduled (sc) approach is used. The wait time $T_{2,\mathrm{sc}}$ and the reception time $T_{3,\mathrm{sc}}$ calculation if a reception window is opened is similar to random access. However, the constant wait duration $t_{3,\mathrm{sc},o}$ and the number of receive windows N_{sc} can be different. Data that need to be received if time scheduled is used can include, similar to random access, message acknowledgments and, different to random access, re-synchronizations. While the decision, whether receive windows are opened to receive acknowledgments is dependent on message acknowledgment settings, similar to random access, it is different for the probability $p_{\mathrm{sync,sc}}$ to re-synchronize the sensor's clock. This re-synchronization

probability is dependent on the clock drift of the end device, the slot length, and the collision probability for the re-synchronization message, as we elaborate in Section 4.2.2. In the following, we assume no additional required receive windows for message acknowledgments besides the ones used to receive re-synchronization messages, to not require additional resources at the end device. In a working time scheduled approach without cross-traffic, no messages collide due to distinct message slots. Otherwise, for example if messages collide with random cross-traffic, the sender is either not informed about the data loss or the required energy for additional receive windows needs to be added to the energy consumption calculation afterwards. Considering the re-synchronizations only, the probability $p_{\mathrm{sync,sc,perfect}}$ to open a reception window and receive a re-synchronization message in perfect conditions without any collision is

$$p_{\mathrm{sync,sc,perfect}} = P(T_{\mathrm{toa}} + T_{\mathrm{drift,total}} > t_{\mathrm{slot}}) \tag{4.21}$$

with t_{slot} as total slot length and $T_{\mathrm{drift,total}}$ as total time drift of the sensor's clock since the last re-synchronization. However, real LoRaWANs suffer from message collisions or losses. For that reason, we extend the equation to

$$p_{\mathrm{sync,sc}} = P(T_{\mathrm{toa}} + T_{\mathrm{drift,total}} > t_{\mathrm{slot}} + T_{\mathrm{rcv}}) \tag{4.22}$$

with T_{rcv} as total duration between the message transmission exceeded the slot due to the clock drift and the re-synchronization message arrived successfully.

Considering the expected value of the random variable T_{drift} for the clock drift, a fixed slot length t_{slot}, and a collision probability for the re-synchronization messages $p_{\mathrm{coll,sync}}$, we can achieve a single equation for the probability $p_{\mathrm{sync,sc}}$ to receive a re-synchronization message in the time scheduled approach after a message is sent in a collision affected channel with

$$p_{\mathrm{sync,sc}} = \frac{E[T_{\mathrm{drift}}]}{(t_{\mathrm{slot}} - E[T_{\mathrm{toa}}]) + \frac{E[T_{\mathrm{drift}}]}{1-p_{\mathrm{coll,sync}}} - E[T_{\mathrm{drift}}]}. \tag{4.23}$$

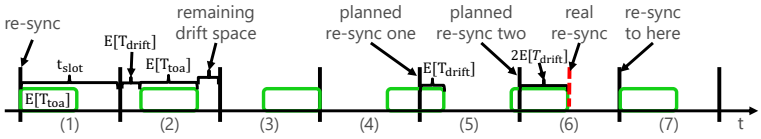

Figure 4.11: Example visualization of slots and clock drift.

We assume a loss of all colliding messages and since we assume transmissions of individual sensors in fixed slots and thus, after constant time intervals as introduced in Section 4.2.2, we can explain the calculation with Figure 4.11.

The figure shows seven example slots over time for a message with an expected length $E[T_{\text{toa}}]$, symbolized by the green boxes. For simplicity reasons, we did not visualize any guard times in this example. In a real setup, they are required between slots to avoid collisions with messages in the following slots. However, adding guard times does not influence the calculation of $p_{\text{sync,sc}}$. From slot to slot, the expected time drift of the messages is $E[T_{\text{drift}}]$. The message has been re-synchronized to start its transmission at the beginning of its first slot. That is why the sensor has $t_{\text{slot}} - E[T_{\text{toa}}]$ 'space' to drift. Dividing this by the expected drift time $E[T_{\text{drift}}]$, the synchronization probability is achieved in a perfect network without any collision. But since we also consider message collisions in our LoRaWAN, this synchronization probability must be adjusted, as explained in detail in the following. A sensor is re-synchronized when its message transmission drifts out of the assigned slot, as shown between slot four and five. However, the re-synchronization message is assumed to be lost in this example. In contrast to other operations, where a lost message directly influences the device behavior and increases or reduces the energy consumption, in LoRaWAN both the sensor and the gateway do not know the loss until the next message is sent from the sensor to the gateway. Then, the next message exceeds its slot limits again, as shown between slot five and six, and the gateway sends a new re-synchronization. Until this point, no additional energy is consumed by the sensor and the probability to receive a re-synchronization message $p_{\text{sync,sc}}$

is not adjusted. This is only required, when the sensor actually receives the re-synchronization message.

Since a message loss is equal to a less frequent reception of re-synchronization messages at the end device, the calculation of the re-synchronization probability $p_{\mathrm{sync,sc}}$ can be adjusted by extending the slot length by one expected time drift $E[T_{\mathrm{drift}}]$ for each unsuccessfully transmitted re-synchronization. Thus, we add $\frac{E[T_{\mathrm{drift}}]}{1-p_{\mathrm{coll,sync}}}$ for each required re-synchronization message to the denominator and subtract $E[T_{\mathrm{drift}}]$ for the successfully received re-synchronization. Contrary to common sense, many colliding messages reduce the sensor's energy consumption since fewer messages are received and the re-synchronization rate is reduced. However, a too high collision rate of re-synchronization messages may lead to sensors drifting out of their slots and colliding with messages transmitted from other sensors in adjacent slots. This can lead to a complete break in the time scheduled channel access approach as described in Section 4.2.2. Thus, it is essential to set up the slot length in a suitable way, reducing both energy consumption and avoiding collisions. Finally, we achieve a wait time $T_{2,\mathrm{sc}}$ during a transmission cycle if receive windows are only opened when a re-synchronization message arrives correctly of

$$T_{2,\mathrm{sc}} = p_{\mathrm{sync,sc}} \cdot N_{\mathrm{sc}} \cdot t_{2,\mathrm{sc},o} \tag{4.24}$$

and an expected total wait time $E[T_{2,\mathrm{sc}}]$ of

$$E[T_{2,\mathrm{sc}}] = p_{\mathrm{sync,sc}} \cdot E[N_{\mathrm{sc}}] \cdot t_{2,\mathrm{sc},o}. \tag{4.25}$$

The random variable $T_{3,\mathrm{sc}}$, for the reception time when the time scheduled approach is used is

$$T_{3,\mathrm{sc}} = p_{\mathrm{sync,sc}} \cdot N_{\mathrm{sc}} \cdot t_{3,\mathrm{sc},o} \tag{4.26}$$

and the expected reception time $E[T_{3,\mathrm{sc}}]$ considering all transmissions in the network is

$$E[T_{3,\mathrm{sc}}] = p_{\mathrm{sync,sc}} \cdot E[N_{\mathrm{sc}}] \cdot t_{3,\mathrm{sc},o}. \tag{4.27}$$

Note that in reality it is very unlikely that the sensor opens receive windows only when a re-synchronization message arrives correctly. Thus, this consideration is the best case investigation and the actual probability to open receive windows is most likely higher.

Listen before Talk: For listen before talk, the time to transmit data and the handling of receive windows is similar to random access. However, the listening on the channel process, shown ahead of the transmission in Figure 4.10 must be taken into consideration to avoid collisions.

Channel Listening Procedure: The process of listening on the channel determines whether a channel is free for a device to send its data. Besides avoiding potential collisions, it adds additional receive and wait times and increases the energy consumption of the device. Listening on a channel is equal to a data reception process and occurs before each transmission. In addition, the wait process occurs if the channel is detected as occupied during receive. Wait and receive before each transmission can be described by the geometric distribution, while wait is not necessarily always needed, in particular not, if the first receive process detects no other message. Let X be a random variable describing the number of listen windows until encountering an empty one. In this case, X follows the geometric distribution that describes the number of trials k until first success ($k = 1, 2, ...$) with $p_{\mathrm{coll,lbt}}$ being the mean probability to detect a channel as occupied when listened on a channel. We can hence compute the mean number of trials before the channel is free as the first moment of the geometric distribution with

$$E[X] = \frac{1}{1 - p_{\mathrm{coll,lbt}}}. \tag{4.28}$$

Since the wait process is not required when the channel is free, the random variable $T_{2,\mathrm{lbt}}$ to describe the wait time in a complete transmission cycle can be achieved with

$$T_{2,\mathrm{lbt}} = t_{2,\mathrm{lbt}} \cdot (X - 1) + N_{\mathrm{lbt}} \cdot t_{2,\mathrm{lbt},o}. \tag{4.29}$$

The pre-defined wait time $t_{2,\mathrm{lbt}}$ when another message occupies the channel is multiplied with $X - 1$ to achieve the expected wait time ahead of the actual transmission. A single wait time $t_{2,\mathrm{lbt}}$ can be defined as constant value or as random variable. For example a general simulation study in [7] shows good results with a random value between 0.4 s and 1.75 s. Note, we define it as constant in the following for simplicity reasons. Furthermore, similar to random access, the expected wait time after a transmission is added, achieved by $N_{\mathrm{lbt}} \cdot t_{2,\mathrm{lbt},o}$. Again, this leads to an expected total wait time $E[T_{2,\mathrm{lbt}}]$ considering all listen before talk transmissions in the LoRaWAN of

$$E[T_{2,\mathrm{lbt}}] = t_{2,\mathrm{lbt}} \cdot (E[X] - 1) + E[N_{\mathrm{lbt}}] \cdot t_{2,\mathrm{lbt},o}. \tag{4.30}$$

The reception time $T_{3,\mathrm{lbt}}$ is achieved by

$$T_{3,\mathrm{lbt}} = t_{3,\mathrm{lbt}} \cdot X + N_{\mathrm{lbt}} \cdot t_{3,\mathrm{lbt},o}, \tag{4.31}$$

where the channel listening time $t_{3,\mathrm{lbt}}$ to avoid collisions is set to a pre-defined constant value and is multiplied with X for the number of channel listening operations. Again, with $N_{\mathrm{lbt}} \cdot t_{3,\mathrm{lbt},o}$, the reception time after the actual message transmission is added similar to random access. Thus,

$$E[T_{3,\mathrm{lbt}}] = t_{3,\mathrm{lbt}} \cdot E[X] + E[N_{\mathrm{lbt}}] \cdot t_{3,\mathrm{lbt},o} \tag{4.32}$$

is the expected total receive time considering all listen before talk transmissions in a LoRaWAN. While all wait and receive times can be defined in the network setup phase, the probability $p_{\mathrm{coll,lbt}}$ to detect another message at the channel

when a sensor listens to it during listen before talk is dependent on the load in the network and the network setup. However, to fully describe the behavior of listen before talk, an in-depth understanding of this probability is required.

Collision Avoidance Probability: In a well planned LoRaWAN, each sensor transmits to its closest gateway, interferes with a minimal number of other sensors for a minimal duration to avoid collisions [17, 6], but also avoids the hidden node problem. The hidden node problem in LoRaWAN can occur, if obstacles are in the line of sight transmission of sensors, but also if sensors are placed in opposite directions from a gateway with possible transmission distances to the gateway but not to the other sensor. Since the location of obstacles is highly dependent on, among others, the network setup, urbanization, and topology, it is omitted here, and the focus is on sensor locations only. The influence of the sensor's location on the collision probability by the hidden node problem is visualized in Figure 4.12. The figure displays a center gateway (GW) and two sensors s_1 and s_2. Both sensors transmit to the central gateway GW indicated by the black arrows. Dependent on the distance to the gateway, a sensor receives a spreading factor. Larger spreading factors allow transmissions across longer distances at the cost of longer transmission airtime [17]. Thus, unnecessarily large spreading factors should be avoided to minimize the ToA of the transmissions [6]. In the example figure, areas where sensors are able to transmit to the gateway are marked with different spreading factors. These areas are placed circularly around the gateway starting with spreading factor 7 in the middle and using spreading factor 9 in the outside circle. Higher spreading factors are not added for better visibility.

In this example, sensor s_1 uses spreading factor 9 for its transmissions in the spreading factor 9 area to make sure its data can be transmitted to the gateway. In contrast, sensor s_2 uses spreading factor 8. The usage of larger spreading factors and thus, longer transmission distances lead to larger interference areas for both sensors. Sensor s_1 interferes all sensors in the red shaded area in this example. In this area, the gateway is included showing that a transmission to

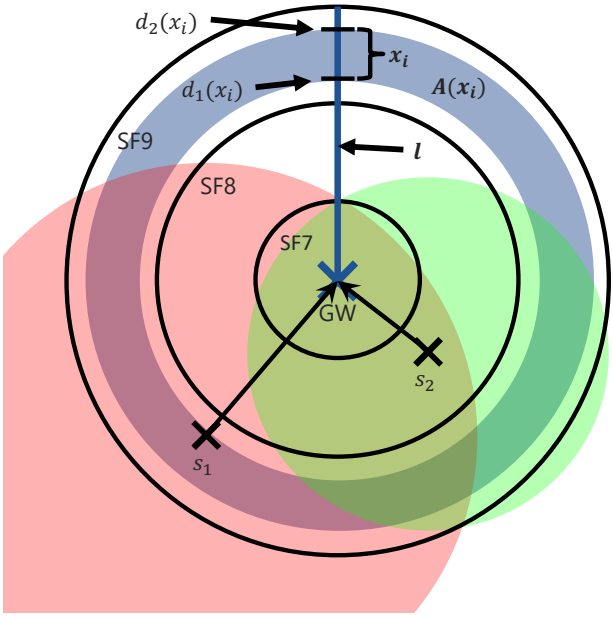

Figure 4.12: Schematic representation of the proposed collision avoidance model for listen before talk

the gateway is possible. In addition, sensor s_2 is also in this area and if sensor s_1 transmits data, sensor s_2 can also hear the transmission. For that reason, it is not influenced by the hidden node problem when sensor s_1 is transmitting. In such a case, sensor s_2 could avoid a collision by delaying its own transmission if listen before talk is used.

However, the situation is different if sensor s_2 starts a transmission. Its green interference area does not include sensor s_1 and if sensor s_2 transmits data while sensor s_1 plans to start a transmission, it can not hear sensor s_2's transmission. Thus, sensor s_1 would also transmit and the messages collide.

To model this behavior for all sensors s_i in a network with a set of sensors S, we determine the probability that a sensor s_i can hear any other sensor s_j with $s_i, s_j \in S$. Then, we can calculate the probability for a wait procedure when listen before talk is used. Therefore, we map the area of a complete LoRaWAN to a single line l. In Figure 4.12, this line l connects the gateway with the outer circle of spreading factor 9, while for the calculation it also includes the other spreading factors up to spreading factor 12 in more outside regions. In the next step, the line is discretized in n sections, highlighted by the example section x_i for section i. It is delimited by the distance $d_1(x_i)$ and $d_2(x_i)$ to the gateway GW. By calculating the area $A(x_i)$ with $i \in 1, 2, ..., n$ of the complete circle, shown in blue, with a minimal distance to the gateway of $d_1(x_i)$ and a maximal distance to the gateway of $d_2(x_i)$, the circular representation of the network can be mapped to a one line representation with $l(i) = A(x_i)$. For the area calculation, the following holds true

$$\sum_{i=0}^{n} A(x_i) = \sum_{i=7}^{12} A(\mathrm{SF}_i) = A(\mathrm{GW}), \tag{4.33}$$

with $A(\mathrm{SF}_i)$ as area where sensors transmit with spreading factor i and $A(\mathrm{GW})$ as complete area the gateway GW 'covers'. After this step, each area $A(x_i)$, or $l(i)$ contains the following information: area size, minimal, maximal, and mean distance to the gateway, number of sensors in the area it represents, and the spreading factor all these sensors transmit with to be able to reach the gateway. In this work, we assume a uniform spatial distribution of sensors in the complete network to achieve the average hidden node probability per sensor transmitting with spreading factor i. However, the calculation works similarly for each area $A(x_i)$ if a uniform spatial distribution of sensors in $A(x_i)$ but differences between $A(x_i)$ and $A(x_j)$ are visible. Otherwise, the calculation is done for each sensor individually.

In a last step, the interference percentage is calculated for each $l(i)$ with each other $l(j)$ with $j \in 1, , 2...n$. Thus, we receive for each area $l(i)$ the number of sensors located in the area and the number of sensors that can be heard in

area $l(i)$ from any other area $l(j)$ in the network. If we map this to sensors and spreading factors, we can, for example, calculate the number of sensors with their respective spreading factors that are heard by a sensor s transmitting with its spreading factor sf located in $l(i)$ or in general in a specific spreading factor area. If this is done for all sensors in the network, or averaged for all spreading factors in the network, we can determine the hidden node probability for all sensors in the network.

Listen Before Talk Back-off Model: The wait probability for listen before talk is equal to the probability a sensor s_i detects another transmission from any other sensor s_j when it wants to start its own transmission or during a predefined sensing interval before the transmission attempt. To determine the wait probability and duration, the probability another message occupies the channel and is detected must be calculated. This wait duration, a so called back-off, is usually a delay for the next transmission attempt for a specific duration. Thus, besides the arrival rate of new transmission attempts, also these retrials influence the general load in the network and must be taken into consideration.

Assuming exponentially distributed inter-arrival times, we can model our network as a simple $M/GI/1$ queuing model if the number of transmissions, and thus, arrivals is large enough and sensor transmission start times are random [149, 211]. The transmission channel is a single processing unit with general independent processing time distribution equal to the ToA. Messages arriving when the channel is occupied are immediately rejected (perfect listen before talk without retrial queue). If we extend the system to listen before talk with retrials after a channel was occupied, an additional finite-source retrial queue can be modeled, presented for example in [212] and surveyed in [213]. It extends the system to a two-dimensional queuing model with the number of messages in the back-off phase in the second dimension. A numerical solution for the system for an exponential processing and back-off time is provided by [214]. However, an exponential back-off time is not reasonable in LoRaWAN [7]. Especially very short and long back-offs lead to unnecessary more back-offs or require

long waiting times. Both would increase the energy consumption unnecessarily. Thus, the final listen before talk model for LoRaWAN can be described as $GI/GI/1$ model with a finite-source retrial queue. A similar model is evaluated by Wüchner et al. in [215]. The authors rely on the MOSEL-2 evaluation tool [216] to obtain numerical results. Similarly, we also use a simulation in this work to validate our modeling approach and obtain a performance evaluation for the presented channel access approaches in LoRaWAN.

Energy Consumption and Energy Efficiency

Last, the average energy consumption E_{lora} of all transmission cycles in the LoRaWAN can be calculated with

$$E_{\mathrm{lora}} = \frac{\sum_{i=1}^{|S|} (E_{\mathrm{total},s_i})}{|S|} \tag{4.34}$$

for all sensors $s \in S$ with S as the set of all sensors, dependent on the used channel access approach. However, this does not consider potentially lost messages due to message collisions. As we outline above, the collision and back-off probability for listen before talk can be obtained by a simulation. Furthermore, as we elaborate in Section 4.2.2, perfect time scheduled can avoid collisions completely if slots are chosen large enough, sensors are re-synchronized correctly and timely, and the amount of cross-traffic is low. If the current state-of-the-art channel access approach random access is used, the arrival process of new messages can be estimated as Markov arrival process according to Metzger et al. [149], if the number of devices in the network is large. We assume a sufficiently large number of messages since for a few sensors, this analysis is pointless due to the low number of collisions. Thus, the expected collision probability $p_{\mathrm{coll},s}$ for a sensor s can be calculated in the random channel access case for a given time frame T with

$$p_{\text{coll},s} = 1 - \left(\prod_{a \in S \setminus s} (1 - P_{s,a}) \right) \tag{4.35}$$

with

$$P_{s,a} = \frac{T_{\text{toa},s} + T_{\text{toa},a}}{T} \tag{4.36}$$

and $P_{s,a}$ as probability for a collision of a message from sensor s with transmission ToA $T_{\text{toa},s}$, and sensor a with transmission ToA $T_{\text{toa},a}$ in the time interval T. By the product of the complementary probabilities $(1 - P_{s,a})$ of the collision probabilities between s and all $a \in S \setminus s$, we achieve the probability for no collision of the message transmitted from s in the time interval T. As a consequence, $p_{\text{coll},s}$ is the collision probability of a message transmitted by s in the time interval T. Thus, the collision probability is dependent on the load in the network and the sensor's sending rate. Dependent on the collision probability and additional overhead from wait and receive, the energy efficiency E_{eff} can then be determined for the complete LoRaWAN. It is achieved by normalizing the expected required energy to transmit data $E[E_1]$ with the average energy consumption E_{lora} for a complete transmission cycle. Furthermore, the collision probability p_{coll} needs to be considered. This leads to

$$E_{\text{eff}} = \frac{E[E_1] \cdot (1 - p_{\text{coll}})}{E_{\text{lora}}} = \frac{E^*}{E_{\text{lora}}}. \tag{4.37}$$

We can extend the energy efficiency consideration by adjusting the required energy to transmit data. Instead of using $E[E_1]$ as expected required energy to transmit a message, E^* can be defined as minimal possible required energy to successfully transmit a message. With this adjustment, additional parameters like message overhead as a result of the coding rate or the message header can be included. Furthermore, the influence of the spreading factor, an unnecessarily large payload, and message collisions can be considered. Thus, with the adjustment, further studies on transmission quality in a LoRaWAN can be con-

ducted by comparing the minimal required payload, spreading factor, and thus, the message transmission ToA with the minimal possible energy consumption. However, such studies directly influence the collision or loss probability, e.g. if the used spreading factor is adjusted. For that reason we focus on energy efficiency studies without additional message parameter or spreading factor adjustments to compare the channel access approaches only.

To this end, we can answer our second research question RQ4.2 as follows: *It is possible to model the energy consumption of a LoRa transmission in a general way for arbitrary hardware, if the energy requirement is normalized by the minimal possible ToA. The energy requirement of processes like listening on the channel or additional wait times can be modeled. To further extend the energy study for LoRaWAN channel access, the energy efficiency is a valuable metric to benchmark different approaches. It includes the actual required energy at the sensor and the collision probability.*

Scenario Description

To validate the presented model and outline influencing parameters leading to variances in the results or deviations from the model, a LoRaWAN channel access simulation is done, as described in Section 4.2.3. For each simulation, we simulate between 50 and 800 sensors in steps of 50 to achieve results with small and larger load and expected collision probability in the network. To obtain statistically significant results, 200 simulation runs are conducted for each individual parameter setting. The duration of a single simulation run is 3600 s and each sensor is transmitting once in this time frame. The locations of all sensors and the payload is re-assigned 20 times to investigate the influence of different payloads and spreading factors.

S 1: Model Validation and Variation Determination: In the first scenario S 1, the presented energy model and the listen before talk modeling approach is studied in detail by a simulation. Key influence factors on the results are discussed and differences between model and simulation are quantified. These key

factors are the spreading factor, the payload, the collision probability, the sensors in range that can be heard to avoid collisions by listen before talk and their respective spreading factors. Variations of these factors have a direct influence on the resulting energy consumption and efficiency.

S 2: Channel Access Approach Comparison: The focus of the second scenario S 2 is on the different channel access approaches. For each approach, variable parameters are studied with the goal of obtaining the most energy efficient parameter setting. For random access, the number of sensors, the probability and duration before a reception window is opened, and the actual receive window length can be varied. When listen before talk is used, in addition, different sensing and back-off durations are investigated to achieve the most energy efficient implementation. Lastly, the main focus for listen before talk is on the probability to receive a re-synchronization message. This can be influenced by the slot length and the average clock drift of sensors.

S 3: Energy Value Adjustment and Approach Comparison: In the last and third scenario S 3, the goal is to determine the best channel access approach for different LoRaWAN deployments and discuss benefits and drawbacks of the approaches. Therefore, to consolidate our model with realistic energy values for transmission, message reception, and wait operations, we consulted the work of other authors [184]. From these, we extracted a theoretical current consumption in the range of 18 mA to 28 mA for the power consumption during transmission and 10.5 mA for receive. During waiting or idle periods the transceiver needs 2.033 mA if kept in standby. Based on this, the power consumption to receive data is varied between 30 % and 60 % of transmit, and wait requires between 7 % and 12 % of transmit. With these values as input, we compare the different channel access approaches, random access, listen before talk, and time scheduled.

4.5 Energy Study Evaluation

This section evaluates the defined scenarios and answers the remaining research question. Therefore, the energy model is validated by a simulation and key influence factors on a LoRa transmission and the energy consumption are presented. Then, an energy efficiency study for the channel access approaches is presented, followed by a numerical evaluation with real energy values from literature.

4.5.1 Scenario S 1: Model Validation

First, key transmission factors are discussed to determine the possible difference in the transmission behavior influencing the presented energy model. If the general structure of a LoRa message is not adjusted and the focus is only on the transmitted data and the distance to the gateway, the message payload and the used spreading factor for the transmission are of key interest. Both have an influence on the ToA and thus, on the energy consumption of a transmission.

Spreading Factor and Payload: For that reason, the expected energy consumption E_1 for the actual LoRa transmission is calculated first. Besides a random payload assignment between 1 B and 51 B, the spreading factor the sensors transmit with is achieved for each sensor by placing them circularly around the gateway, as described in the simulation approach in Section 4.2.3. This leads to 23 % of all sensors transmitting with spreading factor 7, 9 % with spreading factor 8, 12 % with spreading factor 9, 18 % with spreading factor 10, 7 % with spreading factor 11, and 28 % with spreading factor 12. By assigning a random payload to all sensors, an average ToA of 0.789 s is achieved.

This average theoretical value is compared to the result from the simulation study for random access in Figure 4.13 for different numbers of sensors in steps of 100 for better readability. The y-axis shows the deviation between the model and the simulation, the x-axis shows the number of sensors. The errorbars are the 90 % confidence intervals. A positive deviation indicates a higher value by the simulation compared to the model, and vice versa. For the *known SF* result,

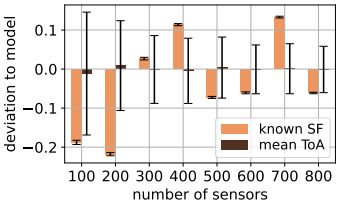

Figure 4.13: Comparison random access (normalized by $t_{\min}$): Impact of payload and spreading factor.

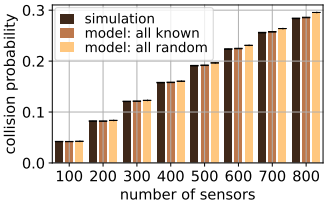

Figure 4.14: Comparison random access: Collision probability with and without knowledge.

presented by the orange bar, the spreading factor is achieved from the sensor and only the payload is unknown. Thus, the ToA is calculated for the model with a random payload and the known spreading factor. For the *mean ToA* result, presented by the brown bar, both the payload and the spreading factor are not known in the model and the simulation result is compared to the average theoretical value of $T_1 = 0.789$.

First, the results show a small difference between model and simulation. The maximum deviation is about $-0.2 \cdot t_{\min}$, and thus, only $0.2 \cdot 0.029$ s. Second, we see that knowing the spreading factor reduces the deviation a lot but increases the average deviation between simulation and model. If an average ToA is used, the model matches with the simulation result very accurately on average and the confidence intervals always overlap with zero deviation. With a deviation of less than $0.2 \cdot t_{\min}$, the result is also acceptable. We can conclude that the model can describe the transmission process accurately, although the ToA of LoRa messages has a large range from several milliseconds up to a few seconds.

Collision Probability: Next, the collision probability is of key interest, as it determines the percentage of messages arriving correctly at the receiver side. This value also influences the energy efficiency of the channel access approaches. Figure 4.14 presents a comparison between simulation and modeling

results to answer the question, whether the model can describe the average collision probability in a LoRaWAN, although the transmission ToA of individual message transmissions is very large. The y-axis of the figure represents the collision probability and the x-axis, again, the number of sensors in steps of 100. The black bar presents the simulation results, the brown bar the modeling results, where the spreading factor and the payload is known. Both are compared to the yellow line where, again, the average ToA of 0.789 s is applied. This is similar to no prior knowledge about the transmission ToA of the sensors. The 90 % confidence intervals are very small and indicate only a small variance in the results. In general, the simulation matches with the model if the ToA is known with less than 0.2 % deviation in the worst case. Furthermore, the confidence intervals for the simulation and the model, where all information is known overlap for all number of sensors, except 800. Thus, we see no statistically significant difference. If the ToA is not known, highlighted by the yellow bar, the model matches with the simulation or overestimates it slightly. Differences are between less than 0.1 % and 0.9 %. This minor difference is a result of the large range of possible ToAs of all transmissions. Thus, we can conclude that the model can deal with the large range of possible ToAs as a result of different spreading factors also when we model the collision probability. This is relevant for all channel access approaches but with this insight, the model is validated for random access.

Hidden Node Problem: To model listen before talk, the validation must be extended to cover the hidden node problem. Therefore, the question is answered, whether the model is accurate if the collision avoidance probability is determined. The result is presented in Figure 4.15. The figure shows the difference in percent between the model and the simulation for the probability of a sensor transmitting with a specific spreading factor SFx (y-axis) that it is heard by another sensor transmitting with SFy (x-axis). This is a measure for collision avoidance potential, as all sensors that hear each other can avoid collisions using listen before talk. For example, the -0.053 % value in the top column of Figure 4.15 is the probability difference between the simulation and the model

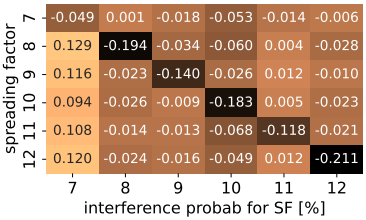

Figure 4.15: Comparison listen before talk: Hidden node study.

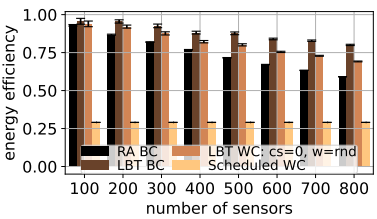

Figure 4.16: S2 - theoretical comparison: energy efficiency of different approaches.

that a sensor transmitting with spreading factor 7 can be heard by any other sensor transmitting with spreading factor 10. The largest difference is detected, when a sensor transmits with spreading factor 12 and whether it is heard by another sensor in the spreading factor 12 area. With a maximal deviation of about 0.2 %, the model is validated as working correctly with only minor mismatch for each single spreading factor. The average probability that a random sensor hears any other sensor is calculated as 35.13 % by the model. Using the simulation, the results are between 34.76 % and 35.37 % for 50 to 800 sensors in steps of 50, and as a result, very small for all numbers of sensors. For that reason, if a sensor does not re-attempt its transmission, both the collision probability and the probability for an unsuccessful transmission attempt can be modeled.

4.5.2 Scenario S 2: Channel Access Approach Comparison

Besides model validation, the evaluation aims to compare channel access approaches regarding energy efficiency. Therefore, random channel access, listen before talk, and time scheduled are compared in the following with different parameter settings. Since sensor locations and thus, the spreading factor assignment and payload determination is the same for each approach, only the performance of the channel access influences the energy efficiency.

Energy Efficiency Comparison: For that reason, different parameter settings for the three channel access approaches, random access, listen before talk, and time scheduled are compared initially, summarized in Figure 4.16. The y-axis represents the energy efficiency defined in Equation 4.37, with an energy efficiency $E_{\text{eff}} = 1$ defined as optimal case and a collision free data transmission without any wait, back-off, listen, or reception time. The error bars indicate the 90 % confidence intervals again. The x-axis plots the number of sensors and the different colors represent different parameter settings for the channel access approaches. The black bar shows the best case random access (RA BC) scenario from an energy efficiency point of view. There, no receive window is opened and thus, only data transmission is considered. The deviation from the most energy efficient transmission is only a result of message collisions, increasing with more sensors because of a load increase with more messages in the network. Similarly, the best case listen before talk (LBT BC) depicts listen before talk with listening on the channel, a random back-off between 0.4 s and 1.75 s, as suggested in [7] if the channel is occupied, and message transmission if no other message is detected. In this best case scenario, it is assumed that both listening on the channel and any back-off consumes no energy. Thus, the difference between the base case random access and the best case listen before talk presents the maximal possible energy efficiency improvement potential if listen before talk is used. Since in reality, additional listening on the channel and back-offs consume energy, this is included in the next listen before talk scenario, presented by the orange bar (LBT WC; $cs = 0$, $w = \text{rnd}$). The cs parameter describes the channel listening duration, and the w parameter the wait duration for the back-off if another message has been detected. Thus, if $cs = 0$, it is only checked whether the channel is free at the point in time, a sensor attempts to transmit data. In this example, we assume no energy consumption for this procedure. However, if another message is detected, a back-off is started and additional energy is consumed. Here, we assume the same energy consumption for the same duration of wait and transmit as worst case assumption. The last scenario discussed in Figure 4.16 is the worst case time scheduled (Scheduled WC). An analysis of the best case time sched-

uled approach is pointless, since its energy efficiency is equal to 1. No collisions occur and no re-synchronization is required because of no device clock drifts in that case. In the theoretical worst case, each sensor is re-synchronized after each transmission which leads to an open receive window wait time and a data reception after each transmission. In this worst case assumption, we further assume that the energy required for wait and receive is equal to transmit. The wait time is set to 1 s before a reception window is opened, stated as default value to open the first receive window in literature [105]. The reception duration is set to 0.926 s as maximal duration to transmit data with spreading factor 12. Thus, we guarantee that a sensor can receive data, independent of the used spreading factor. The results show a drop in the energy efficiency of time scheduled down to 29 % in the worst case. However, there is different improvement potential by limiting the reception or wait duration, consuming less energy during wait or receive, or trigger the re-synchronization process less frequently by less clock drift of the sensors or larger slots. In general, we can draw several conclusions from these studies. First, when the load in the network is small, random access performs similar to listen before talk from an energy efficiency point of view. In this case, only few collisions occur anyway. Second, time scheduled performs, although avoiding all collisions, much worse than listen before talk and random access in the worst case. However, there is improvement potential to perform a lot better than the worst case setup. Third, if the load is increased, listen before talk improves in general against random access if the energy requirement to listen on the channel is not taken into consideration. Lastly, the performance of listen before talk is highly dependent on the energy consumption ratio for the back-off and for data transmission. If the additional energy consumption during the back-off is small, the resulting energy efficiency converges towards the brown bar. In contrast, if it is high, it converges towards the orange bar and an energy consumption maximum during back-off similar to the value during transmit can be assumed. Note, to determine the final energy efficiency of listen before talk we also need to include the consumption during listening on the channel discussed in the following.

Listen Before Talk Energy Efficiency Study: To study the energy efficiency of listen before talk, the impact of different listening and back-off duration on the collision probability and the total back-off duration is investigated. Longer channel listening duration and additional back-offs are only meaningful if in return, the collision probability or the total back-off duration is reduced. However, our simulation results show that a longer back-off does not impact the collision probability since it only changes the next transmission attempt time and does not modify the load in the network. In contrast, longer channel listening durations even increase the average collision probability. This behavior is expected since longer channel listening requires a free channel for a longer time to start another transmission. Thus, short free channel slots are not used and the total load increases. Taking the total back-off duration and the total number of back-offs into consideration, a good trade-off is already achieved with a random back-off between 0.4 s and 1.75 s as described in [7]. A longer listening time detects more messages but again, also avoids short free channel slots. Thus, the total number of back-offs and the total back-off duration is increased. For that reason, a minimal duration to listen on the channel followed by a back-off between 0.4 s and 1.75 s is suggested for listen before talk. To this end, the general energy efficiency for listen before talk is delimited by the energy efficiency of the best case listen before talk (LBT BC) and the worst case listen before talk with up to no channel listening time and the random back-off (LBT WC; $cs = 0, w = \mathrm{rnd}$), as plotted in Figure 4.16. Note, in addition to the energy efficiency for LBT WC; $cs = 0, w = \mathrm{rnd}$, the required additional energy to listen on the channel must be added. This impairs the overall worst case energy efficiency for an additional percentage, dependent on the energy requirement for channel listening and is considered in the following.

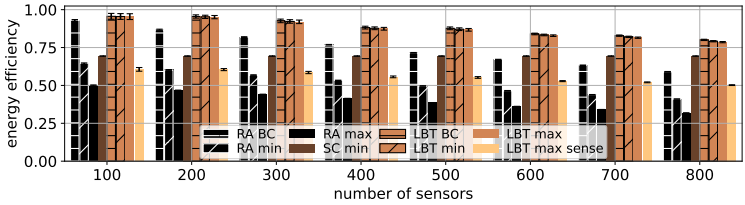

Figure 4.17: S3 - realistic energy values from literature: Comparison of energy efficiency of different approaches.

4.5.3 Scenario S 3: Realistic Energy Value Study

In the last scenario, the energy efficiency for the different channel access approaches is compared, using realistic energy values from literature [184, 207]. The result is plotted in Figure 4.17 for different parameter settings. The best case random access (RA BC), with no receive windows, and the best case listen before talk (LBT BC), with no required energy for the listening and the back-off process are taken from the previous evaluation as reference results. Both bars are plotted with horizontal line markers for better visibility. The results are compared to minimal (min, diagonal line marker) and maximal (max, no line marker) consumption results for random access and listen before talk. In the minimal (min) case, the power consumption to receive data is 30 % of transmit as defined as minimal value in the scenario definition (Section 4.2.4). For wait, 7 % is used. In the maximal (max) case, 60 % and 12 % is used for receive and wait, respectively. In addition, the time scheduled approach with minimal energy consumption for wait and receive is plotted by the brown bar. Furthermore, the *LBT max sense* scenario, presented by the yellow bar in the figure, also considers the energy consumption of a one-second listening interval. An additional energy consumption of 0.45 % of transmit is added to listen on the channel, as average receive energy consumption. The results show that, again, listen before talk performs similar in little load situations and better than random access for high

load, if no energy is required to listen on the channel. In the channel listening case (LBT max sense), listen before talk performs worse than best case random access (RA BC) but similar or better than random access with receive windows (RA min and RA max). Furthermore, if random access with receive windows is considered, the open receive window and receive procedure is similar to the time scheduled approach. Though, in contrast to random access, no collisions occur when time scheduled is used. This leads to a better energy efficiency than random access with receive windows if the load is small and even an improvement against the best case random access for a large load and many sensors. Furthermore, if a data reception is only assumed when a message has been successfully transmitted to the gateway without any collision, the energy efficiency for random access is improved by up to 3 % for 800 sensors (RA min scenario). For RA max, it is an improvement of up to 4 %. In the time scheduled case, the energy efficiency can be improved by 10 % if every second transmission is re-synchronized only. A further improvement by 7 % and close to 10 % is achieved if only 20 % or 10 % of all messages must be re-synchronized, respectively. With 10 % message re-synchronization rate, we achieve an energy efficiency of 89 %.

Finally, we can answer our last research question RQ3 as follows. *From an energy efficiency point of view, random access is only suggested if the load in the network is small and no receive windows are required. Listen before talk can improve the energy efficiency against random access if the required energy to listen on the channel can be minimized. Nevertheless, the best channel access approach is time scheduled, as it can avoid all collisions and works in a very energy efficient way if the re-synchronization rate as a result of sensor's clock drifts is small.*

4.6 Lessons Learned

The recent opening of Amazon Sidewalk to the public, using LoRaWAN as long range communication technology will further foster adoption of the LoRa technology for IoT use cases and our everyday life. Network access is simplified since Amazon Smart Home devices can operate as gateways and provide direct cloud connectivity for a multitude of new applications and sensors. However, to work in a reliable and energy efficient way, alternatives to the error-prone random channel access are required when networks grow and the amount of traffic increases. In this context, listen before talk and a time scheduled channel access have been proven as suitable alternatives.

There are several characteristics a future channel access approach must satisfy to comply with the unique selling points of LoRaWAN. Channel access needs to be simple without unnecessary overhead keeping energy requirements low. For that reason, required message synchronization, channel sensing, or additional processing at the devices should be minimized. In addition, the goal is to improve current unreliable random access by reducing the message collision probability leading to severe data loss. And lastly, to provide enough network resources for all participants in a LoRaWAN, future channel access must comply with transmission regulations and is supposed to work with cross-traffic from other devices of other, not controllable networks. With these challenges and target characteristics, the following three research questions have been identified for future LoRaWAN channel access, investigated in this chapter.

RQ4.1) Is it possible to schedule all messages in a LoRaWAN channel to avoid collisions completely, despite the device and LoRaWAN specific challenges such as device clock drifts, different duration to transmit messages, the gateway duty cycle, and cross-traffic?

RQ4.2) Is it possible to model a LoRaWAN with different channel access approaches and assess the resulting energy consumption and energy efficiency, usable for various underlying hardware?

RQ4.3) What is the best channel access approach for LoRaWAN from an energy efficiency point of view, dependent on the load in the network?

To address the questions, we provide a theoretical consideration for a time scheduled channel access approach and can avoid collisions in a LoRaWAN completely. By a theoretical investigation, a general relationship between the number of sensors, the transmission behavior, and the device capability by means of random clock drifts and duty cycle regulations is presented. In a large scale simulation study, the approach has proven to work efficiently, reducing the collision probability significantly if the amount of cross-traffic is manageable and device clock drifts are reasonable (RQ4.1).

We propose a general energy consumption and energy efficiency model for LoRaWAN, describing data transmission, wait times, and data reception. With the model, we do not rely on any specific hardware but provide a parameterizable generic model to describe LoRa message transmissions. Furthermore, we model the behavior of random access, listen before talk, and time scheduled, and compare the approaches. Our results show that the ToA and the collision probability, as the most important factors to determine the energy efficiency, can be modeled very accurately. The hidden node model for listen before talk achieves only a deviation of 0.2 % between the model and the simulation in the worst case. In general, we can identify the energy efficiency as a valuable metric to be considered besides the general energy requirements or other influencing factors like the expected collision probability. Furthermore, the energy efficiency can also be extended to cover the influence of, among others, different spreading factors or different device behavior (RQ4.2).

Finally, our general energy efficiency study suggests to use random access only, if no additional receive windows are opened, and network load is small. Listen before talk can improve the energy efficiency in a LoRaWAN against random access by up to 20 % in high load situations, if minimal energy is required to listen on the channel. Thus, we suggest to focus on energy consumption reduction for channel sensing in the listen before talk case. Lastly, the time scheduled approach is the best choice, if the random clock drift of sensors is small. Never-

theless, this is the most complex approach and further studies are required on the additional energy consumption at the sensor to re-synchronize the clock. In addition, not all sensors are capable of clock re-synchronizations or some clock drifts can be too large for a practical usage (RQ4.3).

For a comprehensive network planning of a real, large scale LoRaWAN, an in-depth investigation of the expected load for different geographic areas, sensor capability, and required SLAs is essential. If the expected load is small, sensors are cheap and large clock drifts are expected, or a specific percentage of lost messages can be tolerated, random access is a viable solution for channel access. If many messages must be acknowledged or re-transmitted, listen before talk can already improve the general performance of a LoRaWAN, especially if the influence of the hidden node problem is small and many devices can hear each other. Then, the focus is on deploying devices with a small energy consumption to listen on the channel. The highest service quality can be achieved with a complete time scheduled approach, if the end devices show little or moderate device clock drifts and are able to re-synchronize. However, if random access cross-traffic is very high, systematic collisions between several messages transmitted with the time scheduled approach can break the complete system. For that reason, this needs to be avoided at all cost. From an energy efficiency point of view, the usage of a time scheduled channel access can even improve against random access, if many collisions can be avoided, providing end users and their applications with a better quality LoRaWAN.

5 Conclusion

Our way of life, social interactions, work routines, and leisure activities are undergoing rapid transformations, driven by an unprecedented rate of change. A major driver behind this evolution is the ever-present connectivity to the Internet. This constant connection empowers us to stream videos, engage with various social media, hold video calls with friends, automate tasks, and remotely monitor home conditions from any corner of the world, at any time. Furthermore, implementing thorough monitoring through extensive networks of autonomous sensors can contribute to addressing social challenges. These challenges include, among others, the enhancement of urban living standards, simplifying daily life, refining agricultural practices, and the provisioning of vital data to tackle the climate change. To meet these demands, we have carefully designed networks with the general goal of providing each user with satisfactory resources to access their desired applications seamlessly, regardless of their location. We have conducted extensive quality assessments across various application domains, particularly in the field of video content streaming, an entertainment medium enjoyed by a significant portion of the global populace on a daily basis. Yet, as the network sustains a constantly growing load, attributed to the dissemination of high-quality content and the increasing number of interconnected devices facilitating diverse applications, the implications of these rising demands on network performance and quality often remain unknown. This emphasizes the necessity for comprehensive investigations that ensure appropriate resource allocation and stable network quality for all participants. While we have gained substantial insight into conventional Internet Protocol (IP) networks, the impact of changes, especially those in multimedia applications and

video streaming, and the key factors contributing to QoE degradation remain subjects of active research. Notably, novel network access technologies like various LPWANs, which enable network connectivity for a wide array of emerging IoT applications, are still in their arising research stages.

Hence, it becomes mandatory to address both the assessment of application quality within existing networks and an exhaustive exploration of emerging network possibilities. These considerations are essential to tackle the challenge of ensuring robust network quality for all end users in the upcoming decade. On one front, the increasing traffic load in IP-based networks, largely driven by the growth in high-quality video streams, necessitates a profound network monitoring approach and predictive streaming quality analysis. This endeavor currently results in an extensive full-packet trace monitoring and the prediction of QoE degradation factors, often accomplished through complicated, ML-driven methodologies reliant on numerous extracted features from streaming traces. Nonetheless, these resource-intensive techniques exhibit limited scalability, requiring specialized hardware that impedes large-scale, distributed deployment. As a result, there is an urgent demand for quick, efficient, and lightweight alternatives that maintain comparable quality prediction accuracy. On the other front, the increasing number of end devices entering existing networks catalyzes the evolution of novel technologies tailored to diverse use cases. This requires a comprehensive exploration of these technologies with regard to performance, data transmission quality, reliability, resource and energy consumption, and meaningful network design, among other facets.

This thesis comprehensively addresses both efforts, tackling the monitoring and quality assessment of streaming application, a leading contributor to total Internet traffic, as well as conducting an exhaustive analysis of network performance within a LoRaWAN, one of the rapidly emerging LPWAN solutions. The domain of streaming research, and particularly the prediction of streaming quality, has witnessed several years of examination. Nevertheless, an observable trend in literature leans towards more complex and resource-intensive solutions. This context enabled us to identify a wide spectrum of research investigations

that remain unexplored, concerning specifically to the prediction of streaming quality using remarkably lightweight methodologies that have not yet been addressed in existing literature. Additionally, while present LoRaWAN research largely centers on the objective of providing network coverage to all end devices, a thorough exploration of network quality throughout the entire network design process has been largely uncovered until now. This thesis bridges that gap by conducting a comprehensive examination of network quality during the entirety of the network's design phase. The research questions outlined in the introductory sections of this thesis have been effectively addressed and answered. For an in-depth analysis of these questions and a comprehensive overview of major contributions, we refer to the respective chapters.

In Chapter 2, our initial objective was to predict key QoE degradation factors, namely initial delay, quality changes, playback quality, and video re-buffering events, solely using uplink data, in a lightweight and resource-efficient manner. To achieve this, we compiled an extensive dataset, which we have made available to the research community. By analyzing this dataset along with a generated broad streaming dataset encompassing diverse video content, we explored the expected monitoring or processing effort needed to estimate QoE degradation factors using uplink, downlink, or full packet trace data. Our findings, based on a basic queuing model, indicate an approximately 86 % reduction in required data for uplink-based monitoring. The reduction potentially allows us to monitor 100 to 1,000 times more streaming flows in parallel using the same hardware. Consequently, our results demonstrate that relying solely on uplink data can decrease the necessary monitoring resources or enable less complex hardware to perform monitoring tasks. This understanding can significantly assist network providers in implementing distributed streaming quality prediction approaches, leading to resource, energy, and cost savings.

With this insight, our objective was to address the research question of whether it is feasible to predict key QoE degradation factors accurately using uplink data exclusively. To achieve this goal, we developed two straightforward and lightweight prediction methodologies. The first method leverages the inher-

ent characteristics of video data transmission to create a simplified predictive model for identifying impairments during video streaming. This real-time applicable approach can be implemented with or without expert knowledge, yielding comparable or even better prediction outcomes when compared to complex state-of-the-art ML techniques available in existing literature. The second approach employs a lightweight random forest model, utilizing a maximum of only ten features to effectively estimate QoE degradation factors with a high degree of accuracy in real-time. As a result, we have demonstrated that both techniques can serve as valuable tools to predict severe QoE degradation factors, eliminating the need for complicated and resource-intensive solutions. These simplified methods can be executed on middleboxes with moderate processing resources, making them ideal for comprehensive monitoring and QoE degradation factor prediction at last mile towards the customer. This contributes to an enhanced overall streaming QoE for end users and resource saving in provider's networks.

In the context of LoRaWAN planning and performance analysis, Chapter 3 introduces a novel and innovative gateway placement strategy. Addressing the challenge of designing an efficient and universally applicable solution to improve network quality with a gateway placement in a LoRaWAN, we propose an approach grounded in graph metrics. Our investigation highlights the relevance of the distance between sensors and gateways as a critical factor influencing effective gateway placement solutions, affecting network quality through collision probability. Through an extensive simulation study, we examined diverse network configurations and compared the performance of our approach to existing state-of-the-art techniques. Our results showcase that our placement strategy reduces the required number of gateways while enhancing collision probability in comparison to current solutions. The approach remains versatile across various networks deployments and end-device transmission patterns. Additionally, we assessed our approach's effectiveness across emulated deployments for different global cities, demonstrating its value regardless of deployment specifics. Our graph-based gateway placement ensures resilience against expected load increase in future networks and is adaptable to computational

constraints by a simple network segmentation and a divide and conquer solution. As a result, our placement methodology serves as a valuable tool in the network planning phase for both small and large LoRaWANs. Furthermore, it offers potential for network improvement by minimizing collision probability through strategic addition of gateways to established LoRaWAN setups.

Finally, Chapter 4 addresses the optimization of channel access within a LoRaWAN, focusing on collision probability and energy efficiency. We present a fully time-scheduled channel access approach designed to eliminate message collisions within the network. Through theoretical analysis, we establish the maximum feasible number of devices that can transmit using our method, while complying with network regulations. Therefore, we could formulate a closed formula describing this maximal number dependent on various device and network related parameters. A comprehensive simulation study validates our approach's effectiveness in collision reduction and its performance in the presence of concurrent interfering traffic. This research contributes insights for academia and providers, offering a solution to enhance the often unreliable LoRaWAN channel access, thereby mitigating data loss in such networks. Moreover, we explore distinct channel access strategies in terms of energy consumption and efficiency. Our study introduces a versatile model to characterize the energy consumption of LoRaWAN sensors transmitting data via diverse channel access approaches. Extending our investigation, we devise a novel and very general energy efficiency metric for comparative assessment of channel access methods, enabling determination of the most energy-efficient data transmission approach in a LoRaWAN. This universally applicable energy efficiency metric holds potential value for network operators, academia, and industry, extending its utility beyond the realm of LoRaWAN network design to areas such as data centers and large-scale application deployments.

The methods, methodologies, concepts, and insights explored and discussed in this thesis extend beyond the boundaries of individual aspects of streaming or IoT quality monitoring, encompassing a broader spectrum. In an era where the information and communication technology sector's resource and energy

demands continue to rise, while energy consumption reduction is essential to mitigate the carbon footprint and address climate change, the significance of lightweight, comprehensive monitoring solutions and the understanding and design of innovative LPWANs cannot be overvalued. These factors will play a crucial role in addressing the challenges presented by climate change over the next decade. The application of our monitoring methodologies using partial data to other domains presents the foundation for substantial resource conservation. Thus, a comprehensive understanding of every data-intensive application is fundamental to develop resource-efficient monitoring solutions. This in turn leads to further potential research studies focused on optimizing network monitoring efficiency and resource demand for a diverse range of emerging applications.

This thesis marks an initial stride towards enhancing data transmission quality through the utilization of novel IoT access network technologies that are essential for the development and roll-out of future 6G networks. Looking ahead, future networks should be tailored to deploy the most fitting technology based on specific use cases, user demands, and applications. This involves a comprehensive study of various network access technologies both in isolation and in coexistence with other, interfering deployments. The comprehensive goal is to create versatile networks providing a high quality to diverse application domains. Integrating different technologies into a unified network and addressing associated requirements and challenges is a central objective required in 6G development, necessitating a thorough examination of resource and energy demands, as well as the potentials presented by distinct network technologies. Moving forward, the vision is a dynamic network architecture where technology selection is dynamically adapted based on current demands, network load, cost considerations, and application or end device preferences and capabilities. The manner in which data is transmitted may not be of direct concern to the end user. The crucial factors lie in the resources needed, and the resulting quality achieved, independent of the network technology. In this context, LoRaWAN assumes a significant role, particularly in scenarios demanding long-distance, resilient, and energy-efficient data transmission.

Likewise, a thorough investigation into various network components and entire networks is essential to assess energy consumption and efficiency. While this work presents an initial insight in this research direction by analyzing the complete transmission cycle rather than just data transmission by a LoRa sensor, further expansion of this study is required. Beyond measuring and transmitting data at the sensor level, the behavior of elements like access points, backend networks, and data centers requires careful investigation, both individually and within the context of complete applications or service function chains. This results in the requirement of comprehensive energy consumption and efficiency studies from a holistic perspective of the entire network deployment. In the future, enhancing energy efficiency will not be limited to optimizing individual components within a data transmission service chain. It will encompass the entire service from data measurement and processing to responding back to end devices. This approach allows for a comparison of network quality improvements achieved through methods such as comprehensive data processing and resource-intensive ML solutions, compared to lightweight solutions like the one presented for video streaming in this thesis. Moreover, it enables a comprehensive evaluation of the impact of resource and energy requirements, leading to profound decisions on the most suitable approach to enhance overall service quality for end users, encompassing application quality, resource efficiency, and energy demands. To this end, the developed mechanisms, methodologies, and concepts substantially advanced the state-of-the-art and open up novel possibilities in research and technology deployment beyond communication systems.

Appendices

A Streaming Measurement Procedure and Dataset

The procedure to measure the dataset used in this chapter is based on the design and deployment of a testbed and the definition of a measurement procedure and scenarios. In addition, post-processing steps to enrich the dataset with additional streaming information is presented. This information is later important for an accurate streaming phase and QoE degradation factor estimation.

A.1 Testbed and Measurement Description

To ensure that real-world scenarios are replicated as precisely as possible, a client-server-based setup has been developed to record both network and application data. The full setup is presented and explained in the following. Afterwards, details about measurements and measured scenarios are given.

A.1.1 Streaming Testbed

The streaming testbed contains three main components, shown in Figure A.1. The management server, the measurement control unit, and the measurement device. The management server is an entity that validates and organizes measurements and does not take an active part in the measurement process. Instead, its responsibilities include the validation, post-processing, and storage of data. Examples for measurement validation steps are scans for empty or erroneous files, the correct enforcement of bandwidth limitations, and extensive logging

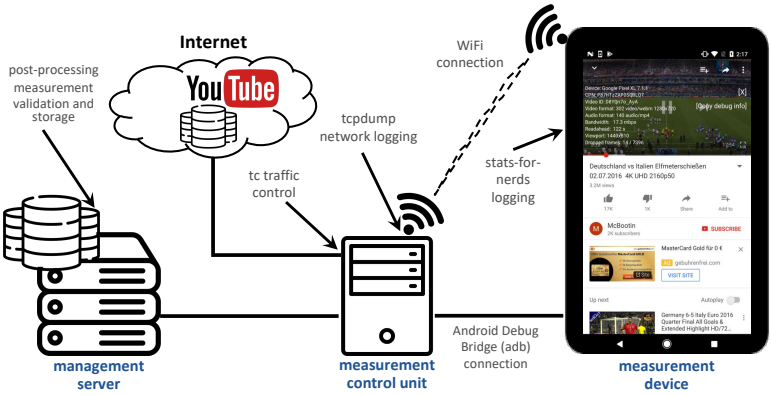

Figure A.1: Streaming testbed overview.

of events occurring during the measurement process. The measurement control unit is responsible for the actual measurement process and is connected to the management server. It is equipped with a sufficiently powerful i7 - 4770 processor with 8 x 3.40 GHz and 16 GB RAM to avoid bottlenecks during the actual measurements. It is connected to the Internet via the German national research and education network (Deutsches Forschungsnetz) via fixed connection to guarantee that measurements are not impacted by physical bandwidth limitations. To control uplink and downlink bandwidths, the state-of-the-art Linux command line tool *tc* [217] is used for traffic shaping and control in the Linux kernel. To perform actions, such as starting measurements or logging application data, the measurement control unit is connected to the measurement device, a smartphone, via the Android Debug Bridge (adb) [218]. This way can be used to directly connect to Android devices via Universal Serial Bus (USB) without interfering with measurements. Finally, the measurement control unit provides a wireless connection with variable and controlled network parameters like bandwidth or packet delays for the measurement device itself. The interface of the

wireless access point is the monitoring point for the complete network traffic, containing network and transport layer data. All upstream and downstream traffic between the wireless network interface and the connected device is captured using *tcpdump* [219] within the Linux kernel ring buffer. Data is stored locally on the measurement control unit before it is offloaded to the management server for further processing. The measurement device is connected to the Internet via the provided 2.4 GHz WiFi access point. To exclude unintended bottlenecks at the smartphone, for the measurement setup, a brand new Google Pixel XL with Android SDK version 28 released in 2020 is used. The display resolution of 2560 x 1440 pixel (equal to the 1440p YouTube video resolution) does not limit the video playback, and the 2.15 GHz and 1.6 GHz quad-core Qualcomm Snapdragon 821 processor and 4 GB RAM are sufficient for the playback of videos. Note that playback related decisions are triggered by the YouTube app and not by the phone. Thus, it is not expected to achieve other results with other Android-based smartphones if the available resources are sufficient. Furthermore, no additional applications are running at the smartphone and the battery is kept at sufficient health during all measurements. During one measurement run, a video is played, and relevant application layer information is logged directly by the device. To achieve this, a specially developed wrapper app has been used to monitor the native Android YouTube app exactly as it is distributed through Google Play Store. The source code of this tool is freely available on GitHub [220] and a detailed description is available in [21].

A.1.2 Measurement Description

The steps of a single measurement run are defined as follows. Every measurement is started by the measurement unit. In a first step, it checks for available connections to the management server, the Internet, and the measurement device via the adb. Upon success, a WiFi access point is opened to provide Internet connectivity to the measurement device. Subsequently, the network scenario is defined. Either no bandwidth limitation can be set for the complete

measurement, which results in approximately 400 Mbit/s downlink bandwidth (and thus, no impairment when streaming videos between 144p and 1080p) or a predefined bandwidth setting schedule can be used. In the latter case, bandwidth limitations are planned for the complete measurement by dynamically applying different limits based on either synthetic traffic limitations or real-world bandwidth measurements. Real world bandwidth measurements contain traces from 3G mobile networks, emulating realistic poor network conditions, and nowadays widely used 4G mobile networks. Afterwards, prior to the actual playback of the desired video, a *setup video* (ID FiO0iLzTyVg) is played for 10 s. This is done to ensure that all network, transport, and application data of the desired video can be logged, and the bandwidth setting is applied correctly. During this *setup video*, the player can adapt the requested playback quality towards the initial bandwidth setting. This avoids unwanted playback behavior that is not a result of the defined scenario but of the switch to the initial bandwidth limitation. Afterwards, a YouTube video is selected for the measurement based on a predefined list of video URLs.

Beginning with the measurement start, after connectivity to all components has been established, all network and transport layer data transmitted and received at the WiFi access point is logged. This includes especially the uplink and downlink video data from the measurement device. Furthermore, as already mentioned during the testbed description, the application data is logged directly at the measurement device by parsing and storing the *stats-for-nerds* data provided by the native YouTube app once a second. In these *stats-for-nerds* information, the complete application behavior like buffer filling status, played video, or the number of already played frames is available. This data is transmitted immediately to the measurement control unit and written to a file via the USB-connection to not interfere with the WiFi connection used for the measurement.

After video playback is finished, all network, transport, and application data points are sent from the measurement control unit to the management server for validation and further post-processing steps. The network and transport layer data include the timestamp, the source and destination IP address, source and

destination port as well as packet lengths for all TCP and QUIC packets observed during the measurement period. The application data include a timestamp, the currently played out video and audio quality, the frames per second, buffer status information, the number of dropped and already played out frames as well as the video ID. Additional information include the current App version, Operating System (OS) version, number of connections, and the battery status during the measurement. A large subset of the data is published for researchers in [5].

A.1.3 Measurement Scenarios

With the described testbed, more than 75 different bandwidth scenarios are applied to study the streaming behavior of the native YouTube app. The bandwidth settings are chosen to study three main situations during YouTube streaming: (1) understand the streaming procedure in general, (2) gain knowledge about scenarios with playback issues and limited bandwidth, and (3) study the streaming process under conditions as similar as possible to reality. For that reason, the bandwidth settings have been chosen as follows.

For the first goal, constant bandwidth settings to monitor streaming in very regular, and for high bandwidth limitations, good conditions have been selected. With these settings, one can understand the streaming process in general and get many baseline details for streaming that helps, for example, in streaming issue prediction. Furthermore, the steps of bandwidth limitations are increased for larger limits since more than 10 Mbit/s is usually sufficient for a good streaming experience. For lower bandwidth limitations, smaller changes affected the playback behavior more severe, and are thus measured in smaller steps. There, the bandwidth ranges from 200 kbit/s to 2 Mbit/s in 100 kbit/s steps and from 2 Mbit/s to 10 Mbit/s in 500 kbit/s steps. Additionally, constant bandwidth settings of 20 Mbit/s, 25 Mbit/s, and 100 Mbit/s are studied to get insights into best case streaming situations. This sums up to 38 constant bandwidth scenarios to analyze the general streaming and requesting behavior with the app. The second goal is to generate playback issues for the app. Therefore, video re-buffering

events with abruptly changing bandwidth have been triggered to achieve a better understanding of this condition. With more slowly changing bandwidth settings, quality changes, buffer level changes, or in general changing conditions in the app are detected. Last, the goal is to test the behavior in realistic conditions with emulated 3G and 4G scenarios. This helps to understand whether stalling or varying playback quality is really an issue in real networks. This understanding can help to react on decreasing buffer situations early and improve the buffering behavior in general to avoid stalling and increase the user perceived quality. Therefore, traces from mobile video streaming scenarios according to [221, 222] are used. The measurements show that the 4G traces of [221] are most likely sufficient to stream the video in consistently high quality. Furthermore, no stalling events could be detected applying these scenarios. In contrast, the traces in [222] have smaller bandwidths and trigger quality changes most likely. Furthermore, the quality is automatically chosen by the YouTube App to not influence the common behavior of the data requesting process. In total, 243 different YouTube videos are measured for evaluation to cover a broad range of typical content. All data is aggregated to a large dataset that serves as the quality determination process. The data post-processing strategy after the measurements are conducted and an overview about the final dataset is presented in detail in the following paragraphs.

A.2 General Data Post-processing

The general data post-processing is done by three major steps according to the overview in Figure A.2. After the measurement is completed, erroneous, invalid, or partly missing data is discarded in the first post-processing step to ensure complete and high quality data. Second, to simplify a streaming behavior study, flows related to YouTube streaming are identified based on their IP-addresses. Application data is cleaned by removing advertising data points related to the playback of commercials. In the last step, streaming phases are defined describing the current *player health* which means whether the video player receives

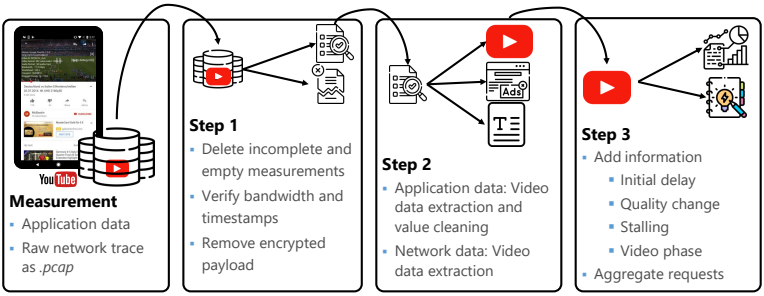

Figure A.2: Step by step post-processing after measurements.

enough video data to fill or keep the current playback buffer health or if the buffer fill level is declining. In addition, all data is aggregated to uplink requests to be used in the session model and the ML-based QoE degradation factor prediction approaches in this thesis. To receive a more general idea about the post-processing, these three steps are explained in detail in the following.

Post-processing Step 1: Data Integrity Check and Data Cleaning

In the data integrity and cleaning step, all invalid measurements are discarded. Therefore, the following tests are performed.

Empty or Incomplete Measurements: First, all measurements where either the raw network file or the application *stats-for-nerds* file does not contain data are deleted. This may occur if the YouTube app is not opened correctly during measurement pre-start. Furthermore, inconsistent measurement runs for which network traffic is measured longer than application data or vice versa are deleted. This occurs if the *tcpdump* capture crashes or if the application data logging is faulty.

Correct Bandwidth Setting: Next, it is determined whether the bandwidth is set correctly. Therefore, bandwidth limit changes logged during the measurement process by the testbed are compared to the predefined bandwidth setting. Furthermore, all network data is analyzed to determine if the network throughput exceeded the possible bandwidth limitation. All error-prone measurements are discarded.

Timestamp Verification: The timestamps of all measurements are checked for plausibility. Since both sources, network and application data, contain timestamps which are supposed to give a complete view about the measurement duration, the overlay of the timestamps is validated. There is no exact congruence because the network layer first establishes a connection and starts downloading video data before the application logs are filled. Similarly, no more data is downloaded at the end of the video when the remaining content is already in the buffer, but the application logs still show video playback.

Encrypted Payload Removal: Since the raw network data from the packet capture file contains the encrypted payload from which no further information can be extracted, the payload is dropped. By means of tshark [223], relevant network and transport layer traffic information is extracted from the packet headers. This includes the UNIX timestamp of each packet, the IPv4 addresses of source and destination and the source and destination port as these are needed to identify YouTube traffic, the protocol type and the packet payload size. In addition to the described network and transport layer information, all raw application data from *stats-for-nerds* is received with a 1 s to 3 s granularity, dependent on the current player status or on player issues after the first post-processing step. Player issues can occur if not enough data is downloaded because of, for example, very little bandwidth during the measurements. Then, the player crashes or error-prone values are displayed and logged.

Post-processing Step 2: YouTube Streaming Data Extraction

In the second post-processing step, relevant video streaming data is separated from cross-traffic and all error-prone values found in the application *stats-for-nerds* file are thoroughly cleaned.

Application Data Video Extraction: The video ID is available for each measurement point and thus, only the application information for the measured video ID is kept. Other video IDs, especially the *setup video* described earlier, as well as advertisement played during the measurement process is excluded. Furthermore, all measurement runs where the correct video is never played because of too long advertisement are discarded.

Value Cleaning: Buffer Health: In this step, all values in the *stats-for-nerds* logs are checked for valid ranges. During this process, negative buffer health values are found occasionally in the proximity of a video quality change. Via manual validation in the application, this inconsistency is confirmed to be a logging issue of the YouTube application itself.

If the quality of a video changes, the buffer health may change suddenly since the adaptive streaming algorithm switches the input for its buffer health prediction from the buffered content of the previous quality to the buffered content of the upcoming quality. This can lead to very low or even negative buffer health values in the logs. However, these negative values occur regardless of changing to a higher or lower video quality.

For example, the playback may be uninterrupted, and quality changes to a higher level. The negative buffer health values occur when the client decides to request a new quality, but neither has the client changed to the new quality nor downloaded enough content in this quality to allow for a quality change without playback interruption. However, cleaning these error-prone buffer health values has no influence on the methodology or the evaluation results. For the

prediction, no buffer health values are used and during labeling, quality changes are labeled by the quality information in the *stats-for-nerds* data. Furthermore, it is determined whether the complete playback before the quality change could be played out to not mislabel stalling.

Network Data Video Extraction: There are two possible ways to separate video network and transport layer data from cross-traffic. For many measurement runs, it is possible to extract the video flows by following IP-port-tuples based on the Domain Name System (DNS) resolution for *googlevideo.com* [4]. These flows are identified as video flows and separated from other traffic. However, to filter not only cross-traffic, but also the traffic of the *setup video* at the beginning of each measurement run and specific advertisements, another approach is used in addition.

Firstly, the start and end time of the correct video is determined from the application data information for each measurement run. Next, from the network data, all flows which are active within that time window are considered candidates for the video stream. Candidate flows are marked and listed with the complete traffic in descending order by traffic volume. Afterwards, the candidate flows are added as streaming flows to the dataset until the traffic volume accumulates to at least 90 % of the total traffic during the video stream. This is valid since video flows are identified as dominant flows in YouTube streaming measurements [29].

With this method, cross-traffic like loading of video comments or video recommendations, transmission of DNS requests, and other background processes are excluded. Furthermore, including only the largest flow would not be sufficient because YouTube may change the connection to a different server during a video stream in case of, for example, data transmission issues, quality changes, or video re-buffering.

Post-processing Step 3: Data Preparation

The input of this step is all valid network and application data without cross-traffic. Out of the network data, all video uplink requests are extracted first and downlink traffic belonging to these requests is aggregated. The resulting traffic is afterwards combined with the application data for streaming phase and QoE degradation factor labeling and prediction. The detailed process of request extraction, data aggregation, network and application data combination, and the final data labeling is described in the following.

Request Extraction and Aggregation: Consecutive download of video data is required in video streaming to guarantee playback without interruptions. Therefore, HTTP requests are detected in the uplink, asking for video data from the YouTube server. All packets containing at least 300 B payload to the YouTube server are extracted from the received packets, similar to [55], and are named as uplink chunk requests. Here, it is assumed that the video is downloaded in a consecutive manner according to Figure A.3. This means video request x is always requested before video request $x+1$ and downloaded completely. Thus, all downlink traffic following a request in the same flow is marked as associated to that specific request, before the next one is sent to the server. In this way, information such as the amount of uplink and downlink traffic in bytes, number of uplink and downlink packets, and protocol associated with one request can be extracted. Furthermore, the last downlink packet of request x, the end of *downlink duration* x in Figure A.3, need not necessarily match with the next uplink request timestamp. Note that video and audio data can be requested in parallel via different flows but also multiplexed to a single flow. Furthermore, short before a quality change, different qualities can be requested in parallel but only one of them is played out later. Last, all empty uplink requests, i.e., those followed by less than 1 KB of download data from the server are discarded since no relevant buffer change is expected by them. After this step, the following information is available for each uplink request: UNIX timestamp of the request start, inter-request time, downlink duration, request size of the uplink request

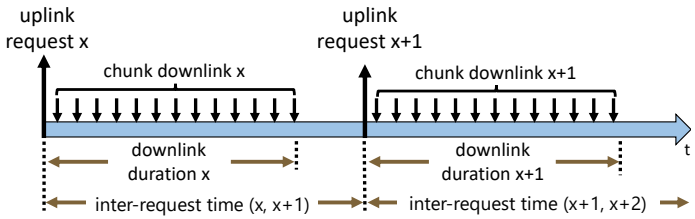

Figure A.3: Timeline of YouTube chunk requests and data download.

packet, summed up chunk downlink and uplink in byte for the request, number of uplink and downlink packets between requests, server port of the video flow, and the protocol. These are the network information used for QoE degradation factor estimation.

Request and Application Data Combination: In this step, the application data from the *stats-for-nerds* is added to the aggregated request information. Relevant information are the playback quality and the buffer health. If the playback quality keeps the same during the complete request, this value is added, if it is changing, the first and the last quality is added, and the request is labeled as quality change. Please note that the goal is only to predict whether quality changes occur during one request. Thus, if multiple quality changes are logged in the *stats-for-nerds* between two requests, they are not labeled explicitly. This happens, if the player requests a different quality short before a bandwidth change. Then, the player can switch back to the old quality and the new quality is never played out but is visible in the logs. Furthermore, very short inter-request times lead occasionally to no application data points for these intervals. Then, the buffer health and quality information from the last application data point is used. On top of buffer health information, the network data is labeled with the playback quality and quality changes as first QoE degradation factors after this step.

Streaming Phase and Stalling Labeling: For each video, the player is always in one of four streaming phases during an ongoing streaming session as introduced in Section 2.1.1: (1) *Filling* if the available playback time in the buffer is filling, or (2) *steady* if the buffer level remains constant as a result of a regular download and playout of data. In this phase, the buffer is filled enough and no stalling or quality change to a worse quality is expected. The player is in (3) *depletion*, if the buffer level decreases. If the duration of this phase is too long and the buffer drops below the quality change threshold, the player requests a lower quality if possible. Consequently, the buffer depletion phase is of high interest for quality change estimation and is also the first indicator for a potential imminent stalling event. If the player is already requesting the lowest quality, no change to a lower quality is possible. Then, if the buffer runs empty, the playback interrupts and (4) the *stalling* phase is entered and stalling occurs. To label the dataset with the four phases, the following procedure is applied. All values in the first and last 5 s of a video measurement are assigned as filling and depletion respectively, since in the beginning and at the end of playback the video is always in these phases. Afterwards, all other logs with a buffer level below 1.2 s are set to stalling. This shows good results in practice since the player can only play out completely downloaded video segments. For that reason, in most cases some playback time is left, and the buffer does not drop to zero. The threshold of 1.2 s is chosen by looking at the maximum buffer level during a stalling event in the data. The remaining logs are listed as steady if the buffer level does not change for more than 0.3 s between two logged values and the overall buffer is larger than 5 s. The slope boundary of 0.3 s is chosen by looking at the occurring slopes in unlimited bandwidth scenarios where a steady phase can be determined manually. This ensures that small changes in the buffer health do not prevent the algorithm from detecting a steady phase. Furthermore, too small buffer levels are not set to steady since a buffer health level of less than 5 s is not enough to guarantee a smooth video playback experience if the bandwidth fluctuates. All other values are set to filling if the buffer level is increasing or depletion if it is decreasing. Note, if the quality of a video

changes, the buffer health level changes suddenly based on the already downloaded data for the upcoming video quality. To correct false assignments of a stalling phase in this case, it is checked whether it is possible to have played out all data since the previously observed log value. If this is not the case, the value is set to the previously selected phase value.

After this correction, the phase detection values are smoothed to prevent frequent jumps out of and back into the steady phase. The minimum duration of a steady phase is 15 s. If within 10 s after a steady phase another data point is labeled as steady the entire period is set to steady. The same procedure is applied for short jumps out of the stalling phase similar to [55]. To validate the stalling labeling, it is assumed that a complete video playback interruption occurs if the playback of a stream is interrupted because of a buffer under run. Since playback is never paused during the measurements, this information is achieved by comparing consecutive *playedFrames* values, available in the *stats-for-nerds* information. If no frames are played out between two application log entries, or less than time passed between two requests with regard to the played out Frames per Second (FPS), the stream is assumed to stall. With this validation and addition, also very short stalling occurrences can be labeled. Considering the requesting behavior in the different phases after labeling, the following is observed: Video segments are downloaded in the buffer filling phase in a best effort manner using the complete available bandwidth. Thus, the average inter-request time of video chunk requests is smaller than the duration of each chunk, until the buffer is filled. As a consequence, the smallest inter-request time is expected in the filling phase. If the buffer is filled to the target buffer level, regular chunk requests are required to keep the buffer at a constant level. This behavior takes place in the steady phase. Thus, a regular pattern of uplink requests can be detected. If the mean inter-request time of a chunk request download is longer than the video playback time the chunk involves, the player is in the buffer depletion phase. Thus, longer inter-request times are expected here compared to the filling and the steady phase. In the stalling phase, no significant amount of data is downloaded and also a large inter-request time is expected.

Table A.1: Information after post-processing for each request.

Network data	
request start timestamp	inter-request time
duration from request to last downlink packet of request	size of uplink request packet
number of bytes in downlink and uplink	number of packets in downlink and uplink
server port of video flow	used transport layer protocol
Application and labeled data	
information whether video is stalling	information whether quality is changed
if quality is changing: initial and target quality	information about streaming phase
buffer filling level at beginning and end of request	statistics about buffer filling level
played video ID	initial playback delay

Initial Delay Determination: By using the first non-zero played out frames value, information is received when the first playback is logged in the *stats for nerds* data. Since the logging granularity there is about 1 s, the exact play start can not be extracted from the data. Using the number of played out frames and the video FPS, showing how many frames are played out within one second, the initial playback start can be calculated. Furthermore, the initial video chunk request timestamp is logged in the network data. It is subtracted from the playback start timestamp to receive the ground truth initial delay. For later initial delay prediction, the first request start timestamp is subtracted from all request start timestamps. Consequently, the UNIX timestamp of the measurement has

no influence on the prediction result and each video run starts with 0.00 s as initial request start timestamp. More details about the initial delay and a small dataset is published in [14]. After the post-processing is completed, all information available for each request are summarized in Table A.1.

A.3 Dataset Overview: Quality of Experience Degradation Factors

This section details on the main QoE degradation factors available in the dataset. First, details about the initial delay are given, followed by the playback quality with specific network layer data that indicate different played out qualities at the application. At the end of this section, information about quality change and stalling information available in the dataset is given.

A.3.1 Initial Delay

The initial delay analysis shows that the mean ground truth initial delay is 2.64 s. It is achieved by subtracting the playback start timestamp from the initial uplink request timestamp. The playback start is either directly available in the YouTube *stats-for-nerds* file or can be calculated with the frames per second and the already played out video time. Playback can start when only a single request is sent to the YouTube server but in more than 60 % of all measurements, five requests are sent to start the video and in total up to ten requests can be required until playback starts. This happened if the bandwidth at the beginning of a video is very small and thus, the player could not download a complete chunk to start playback. Furthermore, more chunk downloads can be started before playback starts than chunk downloads are ended. Thus, we see the beginning of video playback during the download of data chunks, for example in the middle of an open chunk request download. This happened in 69.72% of all runs. However, no additional information is achieved by further analyzing this behavior.

Table A.2: Overview requests per quality.

Type	# Requests	%	Type	# Requests	%
144p	153,262	13.00	480p	190,575	16.16
240p	87,002	7.38	720p	499,237	42.34
360p	111,545	9.46	1080p	108,638	9.21
quality change	28,773	2.44			

A.3.2 Playback Quality

The playback quality is the played out video resolution. Since this chapter deals with the analysis of the playback quality from network data without studying the effect of packet loss, no frame errors like blur or fragments in single frames are observed. Only the played out resolution by YouTube is taken into consideration as quality in this work. For each quality, more than 85,000 requests are measured as summarized in Table A.2. The highest representative is 720p, which is the target resolution for many videos at the smartphone without triggering higher quality manually. Since only 1,344 requests are measured for 1440p and YouTube did not automatically trigger this quality, it is omitted from further evaluation. In addition, all 28,773 requests labeled as quality change are not considered for the quality estimation but only for the quality change estimation.

Inter-Request Time: Furthermore, since the inter-request time is the main network layer information to describe the streaming process in this chapter, an overview of it for all available video resolutions is given in Figure A.4. The x-axis of this CDF shows the inter-request time in seconds, whereas the line colors represent the different resolutions, starting at 144p and ending with 1080p, with lighter colors for larger resolutions. The figure shows that the inter-request time for 10 % up to 20 % of all requests, regardless of the resolution, is very small. The measurements show that this behavior is visible most likely at the beginning of the video, where many requests are sent if sufficient bandwidth is available and

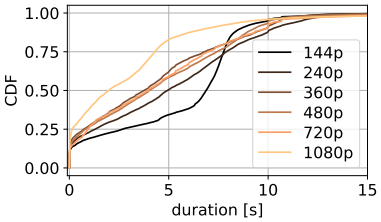

Figure A.4: Inter-request time by resolution.

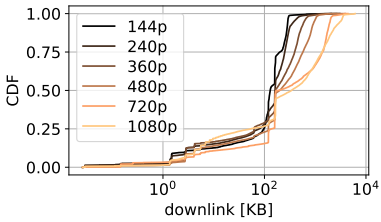

Figure A.5: Downlink bytes by resolution.

the buffer is filled. Furthermore, more than 70 % of all inter-request times are between 0.1 s and 12 s. A rather linear distribution for resolutions between 240p and 720p in that time interval is visible, with 240p having the lowest gradient.

The 144p and 1080p resolutions show a different behavior. While 144p shows a larger inter-request time more often, with only 38 % having an inter-request time less than 6 s, more short inter-request times are visible for 1080p, with 88 % being below 6 s. This observation shows the tendency of having larger inter-request times for poorer qualities. One reason is the different amount of data that must be downloaded for specific resolutions, with fewer data for smaller ones. To minimize the overhead, more seconds of video playback can be requested with a single request for smaller resolutions. Another reason is that lower resolutions are used during phases with bandwidth issues. There, the next request can be delayed because the download of the previous one is not finished.

Downlink Duration: One argument for this assumption is also revealed by the downlink duration. For 144p, more requests have again a higher duration. More than 50 % of all requests show a downlink duration of more than 6 s. For 240p, it is only 32 %, and less than 20 % for the other resolutions. Thus, a request download for smaller resolutions often takes longer than downloading larger ones. The possible reasons are bandwidth issues or a higher number of requested playback seconds. Furthermore, the result for 720p and 1080p shows

that only 12 % of the downlink duration is longer than 6 s. Since these resolutions are often the highest available ones, and thus, also used with much higher bandwidth rates than required for the download, the data is downloaded fast and the downlink duration is shorter. These results show that the inter-request time and the downlink duration can be a valuable input to determine the playback quality. For that reason, it is essential to use them for the estimation task. However, a detailed differentiation, especially between small resolution changes or for the duration distribution as a whole, is not seen. Thus, neither the inter-request time nor the downlink duration is enough for adequate quality estimation, without the use of additional parameters or pattern detection.

Chunk Downlink: The amount of downloaded data per chunk request or the chunk downlink, as visualized in Figure A.3, is another valuable information, and is in particular important for the quality estimation. An overview of the chunk downlink for all requests of each resolution as CDF is given in Figure A.5. The x-axis shows the downlink in kilobytes in a logarithmic scale for better visualization. It is obvious that larger resolutions have larger chunk downlinks, which is visible for the largest 50 % of all requests. Here, except when comparing 720p and 1080p, a clear distinction between all resolutions is evident. This is different for all requests smaller than the median for all resolutions. Only minor differences are visible in those cases, with only 720p and 1080p having a slightly different distribution. One possible reason are different sizes of audio and video chunks downloaded in parallel. Nevertheless, the number of downlink bytes is seen to be a valuable input source for quality estimation. Other parameters like the chunk uplink size, the number of packets in uplink and downlink direction, and the packet size of the requesting packet are also valuable information for the prediction in this thesis. However, since no additional information can be extracted from these parameters at this point, no further details are given here.

A.3.3 Quality Change

The other video quality based QoE degradation factor are quality changes. Especially, the difference in playback quality before and after a quality change is important; that is, whether the next higher or lower resolution is chosen after a quality change or if available resolutions are skipped [224]. In total, 21,971 quality changes are detected during the measurements. More than 30 % of all quality changes are triggered away from 720p. Only about 15 % are triggered away from 240p, 360p, 480p, or 1080p, respectively. The reason is that 720p is used for a broad range of initial bandwidth settings from about 2 Mbit/s up to unlimited bandwidth, if no higher resolution is available. Then, when the bandwidth drops, the resolution changes to a smaller one. Only in 8.16 % of all quality changes 144p is the initial quality before the quality changes. When playing a video with 144p from the start, too low bandwidth results occasionally in a crash of the YouTube app, whereas higher bandwidth results in a playback start with 240p. Thus, 144p is used less often at the beginning of a video stream in the dataset. Compared to that, the target quality shows a rather even distribution regarding 144p up to 720p. Only 1080p shows a lower percentage with 2.54 %, since not all videos are available in this resolution.

To determine whether the resolution is changed to the next higher or lower level or if intermediate resolutions are skipped, the exact resolution transition is studied. This evaluation is given in Figure A.6. The figure shows the overall percentage of all quality changes per resolution at the y-axis and the resolutions before a quality change at the x-axis. The different colors of the stacked bar plot show the target resolution, with the same coloring as used in previous graphs. This means all target resolutions are summed up to 100 % to describe all quality changes that may occur. It is evident that for lower resolutions, several levels of quality are skipped less frequently. In more than 70 %, an adjacent resolution is chosen when 144p is streamed before and for 240p it is in more than 80 % of all cases. This is different for higher resolutions. For 720p, 35 % of the changes are to the next lower resolution and with a percentage of less than 1 % to the next higher resolution. On the other hand, more than 20 % of all changes are

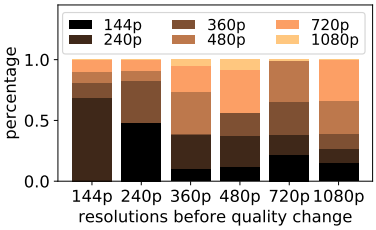

Figure A.6: Resolution transitions.

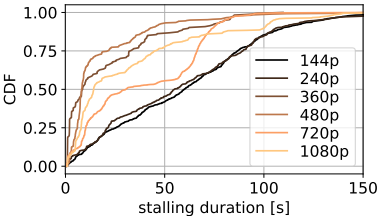

Figure A.7: Stalling duration.

to 144p as lowest resolution. This must be taken into consideration, especially for higher resolutions, by analyzing potential erroneous predictions regarding a quality change, since a small quality change to an adjacent resolution is less severe for the overall streaming experience of an end user than a change to a much lower or higher resolution [224, 225].

A.3.4 Stalling

It might be not sufficient to only monitor single large inter-request times or delayed requests for stalling detection, since one up to 30 request downloads were started during stalling events throughout all measurements. In addition, the influence of the video resolution that is played out before stalling starts is investigated in Figure A.7. The figure overviews the stalling duration on the x-axis based on the resolution as CDF. The colors are kept as earlier. A difference in the stalling duration for different resolutions is detected. When 360p, 480p, or 1080p is played, long stalling is less likely. In total, 42 % of all stallings when 1080p quality is played and even less if 360p and 480p is played are longer than 20 s. In contrast, it is about 80 % for 144p and 240p and even about 50 % of all stalls are longer than 60 s for these resolutions. Afterwards, the stalling duration is normalized for the duration when a specific quality is played out, to avoid comparing resolutions that are played out very long with the very short ones.

Then, in relation to the playtime, the resolution differences are smaller. There is still a tendency for about 70 % of all videos played with 144p or 240p to have a higher percentage of stalling. On the other hand, the 1080p resolution has a higher percentage of videos stalling for 20 % of the total playtime or more.

A.4 Data Preparation for Machine Learning

This section covers further information on data preparation steps for the ML approach to determine QoE degradation factors, described in Section 2.5.

Bootstrapping: The dataset is balanced by class balancing to contain equal data points for each target output value to improve the results of ML models. For example, the stalling feature is 0 for most data points since videos are in a playback state most of the time and stalling is rare in comparison. Therefore, data points are randomly sampled from the target values using bootstrapping, with progressively fewer occurrences, until eventually the dataset is balanced.

Shuffling: Shuffling helps to prevent problems arising from dependency on consecutive data points in the original dataset. Some ML tasks update their models before one iteration over the training data is completed. Then, initial updates suffer if data is not shuffled and similar, statistically dependent data are fed into the model. Moreover, algorithms that in theory process the whole training set at once may be implemented to split and process mini-batches on a machine, due to resource limitations for large datasets. Thus, shuffling ensures that each data point leads to an independent change on the model.

Normalization: Normalization of features and targets is done as a last step since the numeric stability that is achieved through normalization benefits certain evaluation and analysis metrics. It is performed by Scikit-learn's *MinMaxScaler* which determines the minimum and maximum values for each feature and scales them accordingly between 0 and 1.

Table A.3: Hyperparamters for the prediction.

QoE deg. factor	Bootstrap	Criterion	max_features	min_sample_split	n_estimators
initial delay	True	gini	sqrt	5	2000
quality	False	entropy	auto	5	500
streaming phase	True	gini	auto	2	30
quality change	True	gini	auto	2	30
stalling	True	gini	auto	2	10

Test and Trainings Set: To reproduce results for the scientific community, it is a good practice to use fixed random seeds for a test- and training-set split. However, the split becomes deterministic with a fixed random state. For that reason, different seeds are tested during evaluation to ensure that the results are valid for different splits. Furthermore, two dataset splits are performed: First, a random 80:20 training- and test-dataset split. Second, the test set only contains videos that are not included in the training set, to avoid fitting the model to video specific features. This means that the algorithm trains the behavior on different videos than those it is tested on. These two datasets are not sufficient if hyperparameter tuning is used to find the best parameter configuration for the models. With the random forest prediction model presented in this chapter, three-fold cross-validation is used, and the best configuration is evaluated.

Hyperparameter Optimization: To find the best parameter configuration in a given parameter space, a grid search [226] is performed with the F1 score as the target metric for hyperparameter optimization. The result of the optimization and the hyperparameters used for the prediction are summarized in Table A.3. All other hyperparameters are set to the default values.

Acronyms

3GPP 3rd Generation Partnership Project.

ABR Adaptive Bitrate.

adb Android Debug Bridge.

CDF Cumulative Distribution Function.

CDN Content Delivery Network.

CNN Convolutional Neural Network.

CRC Cyclic Redundancy Check.

CSMA Carrier Sense Multiple Access.

CSS Chirp Spread Spectrum.

DE Low Datarate Optimize.

DNS Domain Name System.

DPI Deep Packet Inspection.

FIFO first in first out.

FPS Frames per Second.

HAS HTTP Adaptive Streaming.

HTTP Hypertext Transfer Protocol.

ILP Integer Linear Program.

IoT Internet of Things.

IP Internet Protocol.

ISP Internet Service Provider.

KQI Key Quality Indicator.

LoRa Long Range.

LoRaWAN Long Range Wide Area Network.

LPWAN Low Power Wide Area Network.

LSTM Long Short Term Memory.

LTE-M Long-Term Evolution Machine Type Communication.

M2M Machine-To-Machine.

MAC Medium Access Control.

ML Machine Learning.

MOS Mean Opinion Score.

NB Narrow-Band.

NN Neural Network.

OS Operating System.

PDR Packet Delivery Ratio.

QoE Quality of Experience.

QoS Quality of Service.

RNN Recurrent Neural Network.

RSSI Received Signal Strength Indication.

SLA Service Level Agreements.

TCP Transmission Control Protocol.

TLS Transport Layer Security.

ToA Time on Air.

USB Universal Serial Bus.

Bibliography and References

Bibliography of the Author

Journal Papers

[1] F. Metzger *et al.*, "An Introduction to Online Video Game QoS and QoE Influencing Factors," *Communications Surveys & Tutorials*, 2022.

[2] M. Seufert, V. Burger, K. Lorey, A. Seith, F. Loh, and P. Tran-Gia, "Assessment of Subjective Influence and Trust with an Online Social Network Game," *Computers in Human Behavior*, 2016.

[3] T. Hoßfeld, A. Seufert, F. Loh, S. Wunderer, and J. Davies, "Industrial User Experience Index vs. Quality of Experience Models," *IEEE Communications Magazine*, 2022.

[4] F. Loh, F. Poignée, F. Wamser, F. Leidinger, and T. Hoßfeld, "Uplink vs. Downlink: Machine Learning-Based Quality Prediction for HTTP Adaptive Video Streaming," *Sensors*, 2021.

[5] F. Loh, F. Wamser, F. Poignée, S. Geißler, and T. Hoßfeld, "YouTube Dataset on Mobile Streaming for Internet Traffic Modeling and Streaming Analysis," *Scientific Data*, 2022.

[6] F. Loh, N. Mehling, S. Geißler, and T. Hoßfeld, "Efficient Graph-Based Gateway Placement for Large-Scale LoRaWAN Deployments," *Computer Communications*, 2023.

[7] F. Loh, N. Mehling, and T. Hoßfeld, "Towards LoRaWAN without Data Loss: Studying the Performance of Different Channel Access Approaches," *Sensors*, 2022.

[8] F. Loh, S. Raffeck, S. Geißler, and T. Hoßfeld, "Plan the Access? Generic Hardware Independent Energy Consumption and Efficiency Model for Different LoRaWAN Channel Access Approaches," *IEEE Internet of Things Journal*, 2024.

Conference Papers

[9] F. Loh, S. Geißler, F. Schaible, and T. Hoßfeld, "Talk to Me: Investigating the Traffic Characteristics of Amazon Echo Dot and Google Home," in *International Conference on Communications and Electronics*, IEEE, 2021.

[10] F. Loh and T. Hoßfeld, "Quantification of Energy Efficiency for 6G Ready Internet of Things," in *International Conference on Networked Systems*, 2023.

[11] F. Loh *et al.*, "Is the Uplink Enough? Estimating Video Stalls from Encrypted Network Traffic," in *Network Operations and Management Symposium*, IEEE, 2020.

[12] F. Loh, A. Pimpinella, S. Geißler, and T. Hoßfeld, "Uplink-Based Live Session Model for Stalling Prediction in Video Streaming," in *Network Operations and Management Symposium*, IEEE, 2023.

[13] T. Karagkioules *et al.*, "A Public Dataset for YouTube's Mobile Streaming Client," in *Network Traffic Measurement and Analysis Conference*, IEEE, 2018.

[14] F. Loh *et al.*, "From Click to Playback: A Dataset to Study the Response Time of Mobile YouTube," in *Multimedia Systems Conference*, ACM, 2019.

[15] F. Loh, K. Hildebrand, F. Wamser, S. Geißler, and T. Hoßfeld, "Machine Learning Based Study of QoE Metrics in Twitch.tv Live Streaming," in *Network Operations and Management Symposium*, IEEE, 2023.

[16] F. Loh, D. Bau, J. Zink, A. Wolff, and T. Hoßfeld, "Robust Gateway Placement for Scalable LoRaWAN," in *Wireless and Mobile Networking Conference*, IEEE, 2021.

[17] F. Loh, N. Mehling, S. Geißler, and T. Hoßfeld, "Graph-Based Gateway Placement for Better Performance in LoRaWAN Deployments," in *Mediterranean Communication and Computer Networking Conference*, IEEE, 2022.

[18] F. Loh, S. Raffeck, F. Metzger, and T. Hoßfeld, "Improving LoRaWAN's Successful Information Transmission Rate with Redundancy," in *International Conference on Wireless and Mobile Computing, Networking and Communications*, IEEE, 2021.

[19] F. Loh, S. Raffeck, S. Geißler, and T. Hoßfeld, "Generic Model to Quantify Energy Consumption for Different LoRaWAN Channel Access Methods," in *International Conference on Wireless and Mobile Computing, Networking and Communications*, IEEE, 2022.

[20] F. Loh, N. Mehling, S. Geißler, and T. Hoßfeld, "Simulative Performance Study of Slotted ALOHA for LoRaWAN Channel Access," in *Network Operations and Management Symposium*, IEEE, 2022.

[21] M. Seufert *et al.*, "A Wrapper for Automatic Measurements with YouTube's Native Android App," in *Network Traffic Measurement and Analysis Conference*, IEEE, 2018.

Workshop Papers

[22] C. Moldovan, F. Loh, M. Seufert, and T. Hoßfeld, "Optimizing HAS for 360-Degree Videos," in *Network Operations and Management Symposium*, IEEE, 2020.

[23] V. Vomhoff, S. Geißler, F. Loh, W. Bauer, and T. Hoßfeld, "Characterizing Mobile Signaling Anomalies in the Internet-of-Things," in *Network Operations and Management Symposium*, IEEE, 2022.

[24] F. Loh, S. Geißler, and T. Hoßfeld, "LoRaWAN Network Planning in Smart Environments: Towards Reliability, Scalability, and Cost Reduction," in *KuVS Fachgespräch - Würzburg Workshop on Next-Generation Communication Networks*, 2022.

[25] F. Loh, V. Burger, F. Wamser, P. Tran-Gia, G. Schembra, and C. Rametta, "Performance Evaluation of Video Streaming Service Chains in Softwarized 5G Networks with Task Graph Reduction," in *International Teletraffic Congress*, IEEE, 2017.

[26] F. Loh, S. Raffeck, S. Geißler, and T. Hoßfeld, "Paving the Way for an Energy Efficient and Sustainable Future Internet of Things," in *KuVS Fachgespräch - Würzburg Workshop on Next-Generation Communication Networks*, 2023.

[27] F. Loh, V. Vomhoff, F. Wamser, F. Metzger, and T. Hoßfeld, "Traffic Measurement Study on Video Streaming with the Amazon Echo Show," in *Internet-QoE Workshop on QoE-Based Analysis and Management of Data Communication Networks*, ACM, 2019.

[28] K. Nguyen, F. Loh, T. Nguyen, D. Doan, H. T. Nguyen, and T. Hoßfeld, "Serverless Computing Lifecycle Model for Edge Cloud Deployments," in *International Conference on Communications Workshops*, IEEE, 2023.

[29] A. Pimpinella, A. E. Redondi, F. Loh, and M. Seufert, "Machine-Learning Based Prediction of Next HTTP Request Arrival Time in Adaptive Video Streaming," in *International Conference on Network and Service Management*, IEEE, 2021.

[30] F. Loh, C. Baur, S. Geißler, H. ElBakoury, and T. Hoßfeld, "Collision and Energy Efficiency Assessment of LoRaWANs with Cluster-Based Gateway Placement," in *International Conference on Communications Workshops*, IEEE, 2023.

[31] T. Hoßfeld, S. Raffeck, F. Loh, and S. Geißler, "Analytical Model for the Energy Efficiency in Low Power IoT Deployments," in *International Conference on Network Softwarization*, IEEE, 2022.

Software Demonstrations

[32] F. Wamser, F. Loh, M. Seufert, P. Tran-Gia, R. Bruschi, and P. Lago, "Dynamic Cloud Service Placement for Live Video Streaming with a Remote-Controlled Drone," in *Symposium on Integrated Network and Service Management*, IEEE, 2017.

[33] F. Loh, F. Wamser, T. Hoßfeld, and P. Tran-Gia, "Quality of Service Assessment of Live Video Streaming with a Remote-Controlled Drone," in *Conference on Network Softwarization and Workshops*, IEEE, 2018.

[34] F. Loh *et al.*, "A Wrapper for Automated Measurements with YouTube's Native App," in *Network Operations and Management Symposium*, IEEE, 2018.

[35] F. Loh, N. Mehling, F. Metzger, T. Hoßfeld, and D. Hock, "LoRaPlan: A Software to Evaluate Gateway Placement in LoRaWAN," in *International Conference on Network and Service Management*, IEEE, 2021.

Dataset

[36] Loh, Frank, *Uplink vs. Downlink: Machine Learning-Based Quality Prediction for HTTP Adaptive Video Streaming*, Accessed: 2023-01-12, 2021. [Online]. Available: https : / / zenodo . org / record / 4890859 # .Y7 _ 3DOzML0p.

General References

[37] Statista, *Number of Internet Users Worldwide from 2005 to 2022*, Accessed: 2023-08-08, 2023. [Online]. Available: https://www.statista.com/statistics/273018/number-of-internet-users-worldwide/.

[38] Statista, *Number of Internet and Social Media Users Worldwide as of April 2023*, Accessed: 2023-08-08, 2023. [Online]. Available: https://www.statista.com/statistics/617136/digital-population-worldwide/.

[39] Gitnux Marketdata, *Internet Traffic Statistics and Trends in 2023*, Accessed: 2023-08-08, 2023. [Online]. Available: https://blog.gitnux.com/internet-traffic-statistics/.

[40] Sandvine, *2023 Global Internet Phenomena Report*, Accessed: 2023-08-08, 2022. [Online]. Available: https://www.sandvine.com/global-internet-phenomena-report-2023.

[41] Computer Weekly, *Global IoT Roaming Data Traffic to Reach 650PB in 2026*, Accessed: 2023-08-08, 2021. [Online]. Available: https://www.computerweekly.com/news/252510007/Global-IoT-roaming-data-traffic-to-reach-650PB-in-2026.

[42] Cisco, *Cisco Annual Internet Report*, Accessed: 2023-08-08, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

[43] IoT Analytics, *State of IoT 2023*, Accessed: 2023-08-08, 2023. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/.

[44] S. Wassermann, M. Seufert, P. Casas, L. Gang, and K. Li, "Vicrypt to the Rescue: Real-time, Machine-Learning-Driven Video-QoE Monitoring for Encrypted Streaming Traffic," *Transactions on Network and Service Management*, 2020.

[45] M. Shen, J. Zhang, K. Xu, L. Zhu, J. Liu, and X. Du, "DeepQoE: Real-Time Measurement of Video QoE from Encrypted Traffic with Deep Learning," in *International Symposium on Quality of Service*, IEEE, 2020.

[46] S. C. Madanapalli, A. Mathai, H. H. Gharakheili, and V. Sivaraman, "ReCLive: Real-Time Classification and QoE Inference of Live Video Streaming Services," in *International Symposium on Quality of Service*, IEEE, 2021.

[47] Sandvine, *2022 Global Internet Phenomena Report*, Accessed: 2023-01-04, 2022. [Online]. Available: https://www.sandvine.com/global-internet-phenomena-report-2022.

[48] Gsma, *The State of Mobile Internet Connectivity*, Accessed: 2023-01-05, 2021. [Online]. Available: https://www.gsma.com/r/wp-content/uploads/2021/09/The-State-of-Mobile-Internet-Connectivity-Report-2021.pdf.

[49] D. Do, P. Huynh, P. Vo, and T. Vu, "Customer Churn Prediction in an Internet Service Provider," in *International Conference on Big Data*, IEEE, 2017.

[50] K. Brunnström *et al.*, "Qualinet White Paper on Definitions of Quality of Experience," 2013.

[51] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *Communications Surveys & Tutorials*, 2014.

[52] R. K. Mok, E. W. Chan, and R. K. Chang, "Measuring the Quality of Experience of HTTP Video Streaming," in *International Symposium on Integrated Network Management and Workshops*, IEEE, 2011.

[53] Google, *HTTPS Encryption on the Web*, Accessed: 2023-01-05, 2022. [Online]. Available: https://transparencyreport.google.com/https/overview?hl=en.

[54] Statista, *Global Mobile Video Traffic from 2017 to 2022*, Accessed: 2023-01-05, 2022. [Online]. Available: https://www.statista.com/statistics/252853/global-mobile-video-traffic-forecast/.

[55] C. Gutterman *et al.*, "Requet: Real-time QoE Metric Detection for Encrypted YouTube Traffic," *Transactions on Multimedia Computing, Communications, and Applications*, 2020.

[56] C. Sieber, A. Blenk, M. Hinteregger, and W. Kellerer, "The Cost of Aggressive HTTP Adaptive Streaming: Quantifying YouTube's Redundant Traffic," in *International Symposium on Integrated Network Management*, IEEE, 2015.

[57] A. Finamore, M. Mellia, M. M. Munafo, R. Torres, and S. G. Rao, "YouTube Everywhere: Impact of Device and Infrastructure Synergies on User Experience," in *Internet Measurement Conference*, ACM, 2011.

[58] T. Hoßfeld, C. Moldovan, and C. Schwartz, "To each According to his Needs: Dimensioning Video Buffer for Specific User Profiles and Behavior," in *International Symposium on Integrated Network Management*, IEEE, 2015.

[59] Y. Sani, A. Mauthe, and C. Edwards, "Adaptive Bitrate Selection: A Survey," *Communications Surveys & Tutorials*, 2017.

[60] F. Wamser, D. Staehle, J. Prokopec, A. Maeder, and P. Tran-Gia, "Utilizing Buffered YouTube Playtime for QoE-Oriented Scheduling in OFDMA Networks," in *International Teletraffic Congress*, IEEE, 2012.

[61] D. Tsilimantos, T. Karagkioules, A. Nogales-Gómez, and S. Valentin, "Traffic Profiling for Mobile Video Streaming," in *International Conference on Communications*, IEEE, 2017.

[62] A. Seufert, F. Wamser, D. Yarish, H. Macdonald, and T. Hoßfeld, "QoE Models in the Wild: Comparing Video QoE Models Using a Crowdsourced Data Set," in *International Conference on Quality of Multimedia Experience*, IEEE, 2021.

[63] Y. Qi and M. Dai, "The Effect of Frame Freezing and Frame Skipping on Video Quality," in *international Conference on Intelligent Information Hiding and Multimedia*, IEEE, 2006.

[64] T. Hoßfeld, R. Schatz, and S. Egger, "SOS: The MOS is not Enough!" In *Workshop on Quality of Multimedia Experience*, IEEE, 2011.

[65] F. Laiche, A. Ben Letaifa, and T. Aguili, "QoE-Aware Traffic Monitoring Based on User Behavior in Video Streaming Services," *Concurrency and Computation: Practice and Experience*, 2023.

[66] V. K. Adhikari *et al.*, "Measurement Study of Netflix, Hulu, and a Tale of Three CDNs," *Transactions on Networking*, 2014.

[67] S. C. Madanapalli, A. Mathai, H. H. Gharakheili, and V. Sivaraman, "Modeling Live Video Streaming: Real-Time Classification, QoE Inference, and Field Evaluation," *arXiv preprint arXiv:2112.02637*, 2021.

[68] A. Reed and M. Kranch, "Identifying HTTPS-Protected Netflix Videos in Real-Time," in *Conference on Data and Application Security and Privacy*, ACM, 2017.

[69] D. Tsilimantos, T. Karagkioules, and S. Valentin, "Classifying Flows and Buffer State for YouTube's HTTP Adaptive Streaming Service in Mobile Networks," in *Multimedia Systems Conference*, ACM, 2018.

[70] X. Xiao, F. Zeng, H. Jiang, Z. Xiao, and P. Zhu, "A Novel Dynamic Channel Assembling Strategy in Cognitive Radio Networks with Fine-grained Flow Classification," *Internet of Things Journal*, 2022.

[71] T. T. Nguyen and G. Armitage, "A Survey of Techniques for Internet Traffic Classification Using Machine Learning," *Communications Surveys & Tutorials*, 2008.

[72] M. Jiang *et al.*, "FA-Net: More Accurate Encrypted Network Traffic Classification Based on Burst with Self-Attention," in *International Joint Conference on Neural Networks*, IEEE, 2023.

[73] A. Shahraki, M. Abbasi, A. Taherkordi, and A. D. Jurcut, "A Comparative Study on Online Machine Learning Techniques for Network Traffic Streams Analysis," *Computer Networks*, 2022.

[74] Z. Jin, Z. Liang, M. He, Y. Peng, H. Xue, and Y. Wang, "A Federated Semi-Supervised Learning Approach for Network Traffic Classification," *International Journal of Network Management*, 2023.

[75] M. H. Mazhar and Z. Shafiq, "Real-Time Video Quality of Experience Monitoring for HTTP and QUIC," in *Conference on Computer Communications*, IEEE, 2018.

[76] I. Orsolic and L. Skorin-Kapov, "A Framework for in-Network QoE Monitoring of Encrypted Video Streaming," *IEEE Access*, 2020.

[77] F. Bronzino, P. Schmitt, S. Ayoubi, G. Martins, R. Teixeira, and N. Feamster, "Inferring Streaming Video Quality from Encrypted Traffic: Practical Models and Deployment Experience," *Measurement and Analysis of Computing Systems*, 2019.

[78] G. Dimopoulos, I. Leontiadis, P. Barlet-Ros, and K. Papagiannaki, "Measuring Video QoE from Encrypted Traffic," in *Internet Measurement Conference*, ACM, 2016.

[79] M. Lopez-Martin, B. Carro, J. Lloret, S. Egea, and A. Sanchez-Esguevillas, "Deep Learning Model for Multimedia Quality of Experience Prediction Based on Network Flow Packets," *Communications Magazine*, 2018.

[80] T. Mangla, E. Halepovic, M. Ammar, and E. Zegura, "eMIMIC: Estimating HTTP-Based Video QoE Metrics from Encrypted Network Traffic," in *Network Traffic Measurement and Analysis Conference*, IEEE, 2018.

[81] R. Schatz, T. Hoßfeld, and P. Casas, "Passive YouTube QoE Monitoring for ISPs," in *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, IEEE, 2012.

[82] G. Dimopoulos, P. Barlet-Ros, and J. Sanjuas-Cuxart, "Analysis of YouTube User Experience from Passive Measurements," in *International Conference on Network and Service Management*, IEEE, 2013.

[83] M. Ghosh, D. C. Singhal, and R. Wayal, "DeSVQ: Deep Learning Based Streaming Video QoE Estimation," in *International Conference on Distributed Computing and Networking*, 2022.

[84] P. Casas, M. Seufert, S. Wassermann, B. Gardlo, N. Wehner, and R. Schatz, "DeepCrypt-Deep Learning for QoE Monitoring and Fingerprinting of User Actions in Adaptive Video Streaming," in *International Conference on Network Softwarization*, IEEE, 2022.

[85] L. R. Jiménez, M. Solera, and M. Toril, "A Network-Layer QoE Model for YouTube Live in Wireless Networks," *IEEE Access*, 2019.

[86] U. K. Sarkar, S. Ramakrishnan, and D. Sarkar, "Modeling Full-Length Video Using Markov-Modulated Gamma-Based Framework," *Transactions on Networking*, 2003.

[87] M. M. Krunz and A. M. Makowski, "Modeling Video Traffic Using M/G/∞ Input Processes: A Compromise Between Markovian and LRD Models," *Journal on Selected Areas in Communications*, 1998.

[88] S. Tanwir and H. Perros, "A Survey of VBR Video Traffic Models," *IEEE Communications Surveys & Tutorials*, 2013.

[89] D. P. Heyman and M. J. Sobel, *Stochastic Models in Operations Research: Stochastic Processes and Operating Characteristics*. McGraw-Hill, 1982.

[90] J. D. Little, "A Proof for the Queuing Formula: L=λ W," *Operations Research*, 1961.

[91] Google, *Choose Live Encoder Settings, Bitrates, and Resolutions*, Accessed: 2023-01-12, 2022. [Online]. Available: https://support.google.com/youtube/answer/2853702?hl=e.

[92] Sidneys, *YouTube Video Stream Format Codes*, Accessed: 2023-01-11, 2018. [Online]. Available: https://gist.github.com/sidneys/7095afe4da4ae58694d128b1034e01e2.

[93] I. Orsolic, D. Pevec, M. Suznjevic, and L. Skorin-Kapov, "A Machine Learning Approach to Classifying YouTube QoE Based on Encrypted Network Traffic," *Multimedia Tools and Applications*, 2017.

[94] Scikit-learn Developers, *Sklearn Feature Selection: SelectKBest*, Accessed: 2023-01-12, 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html.

[95] H. Laurent and R. L. Rivest, "Constructing Optimal Binary Decision Trees is NP-Complete," *Information Processing Letters*, 1976.

[96] Scikit-learn Developers, *Sklearn Preprocessing Standard Scaler*, Accessed: 2023-01-12, 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[97] Global Data, *Global IoT Market will Surpass the $1 Trillion Mark by 2024, Says Global Data*, Accessed: 2023-03-20, 2021. [Online]. Available: https://www.globaldata.com/media/thematic-research/global-iot-market-will-surpass-1-trillion-mark-2024-says-globaldata/.

[98] Statista, *Internet of Things (IoT) and non-IoT Active Device Connections Worldwide from 2010 to 2025*, Accessed: 2023-07-31, 2023. [Online]. Available: https://www.statista.com/statistics/1101442/iot-number-of-connected-devices-worldwide/.

[99] J. Haxhibeqiri, F. Van den Abeele, I. Moerman, and J. Hoebeke, "LoRa Scalability: A Simulation Model Based on Interference Measurements," *Sensors*, 2017.

[100] Ubidots, *LoRaWAN vs NB-IoT: A Comparison Between IoT Trend-Setters*, Accessed: 2023-03-20, 2020. [Online]. Available: https://ubidots.com/blog/lorawan-vs-nb-iot/.

[101] N. Matni, J. Moraes, H. Oliveira, D. Rosário, and E. Cerqueira, "LoRaWAN Gateway Placement Model for Dynamic Internet of Things Scenarios," *Sensors*, 2020.

[102] Design and Reuse, *Semtech Acquires Wireless Long Range IP Provider Cycleo*, Accessed: 2023-01-20, 2012. [Online]. Available: https://www.design-reuse.com/news/28706/semtech-cycleo-acquisition.html.

[103] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the Limits of LoRaWAN," *Communications Magazine*, 2017.

[104] The Things Network, *Frequency Plans*, Accessed: 2023-01-20. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/frequency-plans/.

[105] D. Bankov, E. Khorov, and A. Lyakhov, "On the Limits of LoRaWAN Channel Access," in *International Conference on Engineering and Telecommunication*, IEEE, 2016.

[106] M. A. A. Khan, H. Ma, S. M. Aamir, and Y. Jin, "Optimizing the Performance of Pure ALOHA for LoRa-Based ESL," *Sensors*, 2021.

[107] N. Abramson, "The Throughput of Packet Broadcasting Channels," *Transactions on Communications*, 1977.

[108] M. Hata, "Empirical Formula for Propagation Loss in Land Mobile Radio Services," *Transactions on Vehicular Technology*, 1980.

[109] The Things Network, *Message Types*, Accessed: 2023-03-13. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/message-types/.

[110] The Things Network, *LoRa Physical Layer Packet Format*, Accessed: 2023-03-13. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/lora-phy-format/.

[111] The Things Network, *Forward Error Correction and Code Rate*, Accessed: 2023-03-13. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/fec-and-code-rate/.

[112] Semtech, *Product Details SX1276*, Accessed: 2023-01-20. [Online]. Available: https://www.semtech.com/products/wireless-rf/lora-connect/sx1276#download-resources.

[113] T. Polonelli, D. Brunelli, A. Marzocchi, and L. Benini, "Slotted ALOHA on LoRaWAN–Design, Analysis, and Deployment," *Sensors*, 2019.

[114] N. Chinchilla-Romero, J. Navarro-Ortiz, P. Muñoz, and P. Ameigeiras, "Collision Avoidance Resource Allocation for LoRaWAN," *Sensors*, 2021.

[115] G. Ferré, "Collision and Packet Loss Analysis in a LoRaWAN Network," in *European Signal Processing Conference*, IEEE, 2017.

[116] M. R. Islam, M. Bokhtiar-Al-Zami, B. Paul, R. Palit, J.-C. Grégoire, and S. Islam, "Interference Issues in LoRaWAN: A Comparative Study Using Simulator and Analytical Model," in *Region 10 Symposium*, IEEE, 2022.

[117] J. El Rayess, K. Khawam, S. Lahoud, M. El Helou, and S. Martin, "Study of LoRaWAN Networks Reliability," in *Conference on Cloud and Internet of Things*, IEEE, 2023.

[118] X. Xia, Y. Zheng, and T. Gu, "FTrack: Parallel Decoding for LoRa Transmissions," in *Conference on Embedded Networked Sensor Systems*, IEEE, 2019.

[119] M. O. Shahid, M. Philipose, K. Chintalapudi, S. Banerjee, and B. Krishnaswamy, "Concurrent Interference Cancellation: Decoding Multi-Packet Collisions in LoRa," in *SIGCOMM*, ACM, 2021.

[120] A. Petroni and M. Biagi, "Interference Mitigation and Decoding Through Gateway Diversity in LoRaWAN," *IEEE Transactions on Wireless Communications*, 2022.

[121] A. Waret, M. Kaneko, A. Guitton, and N. El Rachkidy, "LoRa Throughput Analysis with Imperfect Spreading Factor Orthogonality," *Wireless Communications Letters*, 2018.

[122] J. Markkula, K. Mikhaylov, and J. Haapola, "Simulating LoRaWAN: On Importance of Inter Spreading Factor Interference and Collision Effect," in *International Conference on Communications*, IEEE, 2019.

[123] P. Maurya, A. Singh, and A. A. Kherani, "A Review: Spreading Factor Allocation Schemes for LoRaWAN," *Telecommunication Systems*, 2022.

[124] J. M. Finochietto, O. Dieng, D. Mosse, and R. M. Santos, "Adding Empirical Real-Time Guarantees to LoRaWAN," in *International Conference on Real-Time Networks and Systems*, 2023.

[125] M. Micheletto, P. Zabala, S. F. Ochoa, R. Meseguer, and R. Santos, "Determining Real-Time Communication Feasibility in IoT Systems Supported by LoRaWAN," *Sensors*, 2023.

[126] A. Dino, D. Garlisi, F. Giuliano, D. Croce, and I. Tinnirello, "Dynamic Adaptation of LoRaWan Traffic for Real-time Emergency Operations," in *International Conference on Wireless and Mobile Computing, Networking and Communications*, IEEE, 2022.

[127] G. G. M. de Jesus, R. D. Souza, C. Montez, and A. Hoeller, "LoRaWAN Adaptive Data Rate with Flexible Link Margin," *Internet of Things Journal*, 2020.

[128] 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*, 2020.

[129] C. Goursaud and J.-M. Gorce, "Dedicated Networks for IoT: PHY/MAC State of the Art and Challenges," *EAI Endorsed Transactions on Internet of Things*, 2015.

[130] K. Mikhaylov *et al.*, "On the Performance of Multi-Gateway LoRaWAN Deployments: An Experimental Study," in *Wireless Communications and Networking Conference*, IEEE, 2020.

[131] H. A. Cruz *et al.*, "Methodology for LoRa Gateway Placement Based on Bio-Inspired Algorithms for a Smart Campus in Wooded Area," *Sensors*, 2022.

[132] B. Ousat and M. Ghaderi, "LoRa Network Planning: Gateway Placement and Device Configuration," in *International Congress on Internet of Things*, IEEE, 2019.

[133] B. Citoni, S. Ansari, Q. H. Abbasi, M. A. Imran, and S. Hussain, "Impact of Inter-Gateway Distance on LoRaWAN Performance," *Electronics*, 2021.

[134] Y. Jin, L. Ding, F. Yang, L. Qian, and C. Zhi, "LoRa Network Planning Based on Improved ISODATA Algorithm," in *International Conference on Wireless Communications and Signal Processing*, IEEE, 2020.

[135] S. Mnguni, P. Mudali, A. M. Abu-Mahfouz, and M. Adigun, "Impact of the Packet Delivery Ratio (PDR) and Network Throughput in Gateway Placement LoRaWAN Networks," *Southern Africa Telecommunication Networks and Applications Conference*, 2021.

[136] C. N. da Silva, P. F. F. de Abreu, J. D. da Silveira, *et al.*, "Estimating the Number of Gateways Through Placement Strategies in a LoRaWAN Network," in *Brazilian Symposium on Computing Systems Engineering*, IEEE, 2022.

[137] K. S. Abakar, I. Bennis, A. Abouaissa, and P. Lorenz, "A Multi-Gateway Behaviour Study for Traffic-Oriented LoRaWAN Deployment," *Future Internet*, 2022.

[138] K. Tutschku, "Demand-Based Radio Network Planning of Cellular Mobile Communication Systems," in *Conference on Computer Communications*, IEEE, 1998.

[139] W. Wu, J. Luo, and M. Yang, "Gateway Placement Optimization for Load Balancing in Wireless Mesh Networks," in *International Conference on Computer Supported Cooperative Work in Design*, IEEE, 2009.

[140] S. Srivastava and A. K. Jaiswal, "Clustering Based Load Balanced Gateway Placement Approach," *International Journal of Computer Applications*, 2013.

[141] S. Mnguni, A. M. Abu-Mahfouz, P. Mudali, and M. O. Adigun, "A Review of Gateway Placement Algorithms on Internet of Things," in *International Conference on Advances in Big Data, Computing and Data Communication Systems*, IEEE, 2019.

[142] M. Heusse, T. Attia, C. Caillouet, F. Rousseau, and A. Duda, "Capacity of a LoRaWAN Cell," in *Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ACM, 2020.

[143] F. P. Correia, S. R. d. Silva, F. B. S. d. Carvalho, M. S. d. Alencar, K. D. R. Assis, and R. M. Bacurau, "LoRaWAN Gateway Placement in Smart Agriculture: An Analysis of Clustering Algorithms and Performance Metrics," *Energies*, 2023.

[144] E. Harinda, S. Hosseinzadeh, H. Larijani, and R. M. Gibson, "Comparative Performance Analysis of Empirical Propagation Models for LoRaWAN 868MHz in an Urban Scenario," in *World Forum on Internet of Things*, IEEE, 2019.

[145] M. Paredes *et al.*, "Propagation Measurements for a LoRa Network in an Urban Environment," *Journal of Electromagnetic Waves and Applications*, 2019.

[146] Semtech, *Product Details SX1257*, Accessed: 2023-01-20. [Online]. Available: https://www.semtech.com/products/wireless-rf/lora-core/sx1257.

[147] A. Jain and B. Reddy, "Node Centrality in Wireless Sensor Networks: Importance, Applications and Advances," in *International Advance Computing Conference*, IEEE, 2013.

[148] S. Wandelt, X. Shi, and X. Sun, "Approximation of Interactive Betweenness Centrality in Large Complex Networks," *Complexity*, 2020.

[149] F. Metzger, T. Hoßfeld, A. Bauer, S. Kounev, and P. E. Heegaard, "Modeling of Aggregated IoT Traffic and its Application to an IoT Cloud," *Proceedings of the IEEE*, 2019.

[150] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive Configuration of LoRa Networks for Dense IoT Deployments," in *Network Operations and Management Symposium*, IEEE, 2018.

[151] B. Reynders, Q. Wang, and S. Pollin, "A LoRaWAN Module for ns-3: Implementation and Evaluation," in *Workshop on ns-3*, ACM, 2018.

[152] J. C. da Silva, D. d. L. Flor, V. A. de Sousa Junior, N. S. Bezerra, and A. A. de Medeiros, "A Survey of LoRaWAN Simulation Tools in ns-3," *Journal of Communication and Information Systems*, 2021.

[153] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, "Do LoRa Low-Power Wide-Area Networks Scale?" In *International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, ACM, 2016.

[154] C. Caillouet, M. Heusse, and F. Rousseau, "Optimal SF Allocation in LoRaWAN Considering Physical Capture and Imperfect Orthogonality," in *Global Communications Conference*, IEEE, 2019.

[155] *Amazon Sidewalk Paves the Way for more Connected Communities*, Accessed: 2023-04-06. [Online]. Available: https://developer.amazon.com/en-US/blogs/alexa/device-makers/2020/09/amazon-sidewalk-paves-the-way-for-more-connected-communities.

[156] *Enabling Amazon Sidewalk With LoRa*, Accessed: 2023-04-05. [Online]. Available: https://info.semtech.com/sidewalk.

[157] *Coverage Sidewalk Amazon*, Accessed: 2023-04-05. [Online]. Available: https://coverage.sidewalk.amazon/.

[158] *LoRaWAN Market Report*, Accessed: 2023-04-05. [Online]. Available: https://www.gminsights.com/industry-analysis/lorawan-market.

[159] L. Beltramelli, A. Mahmood, P. Österberg, and M. Gidlund, "LoRa Beyond ALOHA: An Investigation of Alternative Random Access Protocols," *Transactions on Industrial Informatics*, 2020.

[160] J. Ortín, M. Cesana, and A. Redondi, "Augmenting LoRaWAN Performance with Listen before Talk," *Transactions on Wireless Communications*, 2019.

[161] L. Casals, B. Mir, R. Vidal, and C. Gomez, "Modeling the Energy Performance of LoRaWAN," *Sensors*, 2017.

[162] T. Bouguera, J.-F. Diouris, J.-J. Chaillout, R. Jaouadi, and G. Andrieux, "Energy Consumption Model for Sensor Nodes Based on LoRa and LoRaWAN," *Sensors*, 2018.

[163] The Things Network, *Duty Cycle*, Accessed: 2023-07-24. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/duty-cycle/.

[164] The Things Network, *Device Classes*, Accessed: 2023-03-30. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/classes/.

[165] K. Spathi, A. Valkanis, G. BeletsIoTi, G. Papadimitriou, and P. Nicopolitidis, "Performance Evaluation of Slotted ALOHA Based IoT Networks under Symmetric Traffic," in *International Conference on Communications, Computing, Cybersecurity, and Informatics*, IEEE, 2020.

[166] F. Tirado-Andrés and A. Araujo, "Performance of Clock Sources and their Influence on Time Synchronization in Wireless Sensor Networks," *International Journal of Distributed Sensor Networks*, 2019.

[167] L. Tessaro, C. Raffaldi, M. Rossi, and D. Brunelli, "Lightweight Synchronization Algorithm with Self-Calibration for Industrial LoRa Sensor Networks," in *Workshop on Metrology for Industry 4.0 and IoT*, IEEE, 2018.

[168] J. Ortín, M. Cesana, and A. Redondi, "How do ALOHA and Listen before Talk Coexist in LoRaWAN?" In *International Symposium on Personal, Indoor and Mobile Radio Communications*, IEEE, 2018.

[169] L. Leonardi, L. Lo Bello, F. Battaglia, and G. Patti, "Comparative Assessment of the LoRaWAN Medium Access Control Protocols for IoT: Does Listen before Talk Perform Better than ALOHA?" *Electronics*, 2020.

[170] A Xanthopoulos, A. Valkanis, G Beletsioti, G. I. Papadimitriou, and P. Nicopolitidis, "On the Use of Backoff Algorithms in Slotted ALOHA LoRaWAN Networks," in *International Conference on Computer, Information and Telecommunication Systems*, IEEE, 2020.

[171] M. A. A. Khan, H. Ma, S. M. Aamir, A. B. Cekderi, M. Ahamed, and A. A. A. Alsumeri, "Performance of Slotted ALOHA for LoRa-ESL Based on Adaptive Backoff and Intra Slicing," in *International Conference on Communication and Information Systems*, IEEE, 2022.

[172] G. Yapar, T. Tugcu, and O. Ermis, "Time-Slotted ALOHA-Based LoRaWAN Scheduling with Aggregated Acknowledgement Approach," in *Conference of Open Innovations Association*, IEEE, 2019.

[173] C. Garrido-Hidalgo *et al.*, "LoRaWAN Scheduling: From Concept to Implementation," *Internet of Things Journal*, 2021.

[174] M. O'kennedy, T. Niesler, R. Wolhuter, and N. Mitton, "Practical Evaluation of Carrier Sensing for a LoRa Wildlife Monitoring Network," in *Networking Conference*, IEEE, 2020.

[175] K. Mikhaylov, J. Petaejaejaervi, and T. Haenninen, "Analysis of Capacity and Scalability of the LoRa Low Power Wide Area Network Technology," in *European Wireless Conference*, VDE, 2016.

[176] S. Herrería-Alonso, A. Suárez-González, M. Rodríguez-Pérez, and C. López-García, "Enhancing LoRaWAN Scalability with Longest First Slotted CSMA," *Computer Networks*, 2022.

[177] N. Kouvelas, V. Rao, and R. Prasad, "Employing p-CSMA on a LoRa Network Simulator," *arXiv preprint arXiv:1805.12263*, 2018.

[178] C. Pham, "Investigating and Experimenting CSMA Channel Access Mechanisms for LoRa IoT Networks," in *Wireless Communications and Networking Conference*, IEEE, 2018.

[179] C. Pham, "Robust CSMA for Long-Range LoRa Transmissions with Image Sensing Devices," in *Wireless Days*, IEEE, 2018.

[180] L. Leonardi, L. Lo Bello, G. Patti, A. Pirri, and M. Pirri, "Simulative Assessment of the Listen before Talk Adaptive Frequency Agility Medium Access Control Protocol for LoRaWAN Networks in IoT Scenarios," *Applied System Innovation*, 2023.

[181] L. Leonardi, L. Lo Bello, G. Patti, A. Pirri, and M. Pirri, "Combined Use of LoRaWAN Medium Access Control Protocols for IoT Applications," *Applied Sciences*, 2023.

[182] S. Raffeck, S. Geißler, and T. Hoßfeld, "DBM: Decentralized Burst Mitigation for Periodic LoRa Devices Using Self-Organizing Radio Access," in *International Conference on Communications*, IEEE, 2023.

[183] C. Pham and M. Ehsan, "Dense Deployment of LoRa Networks: Expectations and Limits of Channel Activity Detection and Capture Effect for Radio Channel Access," *Sensors*, 2021.

[184] S. Maudet, G. Andrieux, R. Chevillon, and J.-F. Diouris, "Refined Node Energy Consumption Modeling in a LoRaWAN Network," *Sensors*, 2021.

[185] P. J. Basford, F. M. Bulot, M. Apetroaie-Cristea, S. J. Cox, and S. J. Ossont, "LoRaWAN for Smart City IoT Deployments: A Long Term Evaluation," *Sensors*, 2020.

[186] N. Vatcharatiansakul, P. Tuwanut, and C. Pornavalai, "Experimental Performance Evaluation of LoRaWAN: A Case Study in Bangkok," in *International Joint Conference on Computer Science and Software Engineering*, IEEE, 2017.

[187] S.-Y. Wang, J.-J. Zou, Y.-R. Chen, C.-C. Hsu, Y.-H. Cheng, and C.-H. Chang, "Long-Term Performance Studies of a LoRaWAN-Based PM2.5 Application on Campus," in *Vehicular Technology Conference*, IEEE, 2018.

[188] T. Fedullo *et al.*, "An IoT Measurement System Based on LoRaWAN for Additive Manufacturing," *Sensors*, 2022.

[189] A. Lombardo, S. Parrino, G. Peruzzi, and A. Pozzebon, "LoRaWAN vs NB-IoT: Transmission Performance Analysis within Critical Environments," *Internet of Things Journal*, 2021.

[190] D. Tamang, A. Pozzebon, L. Parri, A. Fort, and A. Abrardo, "Designing a Reliable and Low-Latency LoRaWAN Solution for Environmental Monitoring in Factories at Major Accident Risk," *Sensors*, 2022.

[191] A. Pozzebon, I. Cappelli, F. Campagnaro, R. Francescon, and M. Zorzi, "LoRaWAN Transmissions in Salt Water for Superficial Marine Sensor Networking: Laboratory and Field Tests," *Sensors*, 2023.

[192] N. Souza Bezerra, C. Åhlund, S. Saguna, and V. A. de Sousa, "Temperature Impact in LoRaWAN–A case Study in Northern Sweden," *Sensors*, 2019.

[193] I. G. B. Astawa, "Implementation of Temperature and Humidity Monitoring System Using LoRaWAN for Pharmaceutical Industry," in *First Mandalika International Multi-Conference on Science and Engineering*, Springer Nature, 2023.

[194] M. M. Erbati, G. Schiele, and G. Batke, "Analysis of LoRaWAN Technology in an Outdoor and an Indoor Scenario in Duisburg-Germany," in *International Conference on Computer and Communication Systems*, IEEE, 2018.

[195] E. Harinda, A. J. Wixted, A.-U.-H. Qureshi, H. Larijani, and R. M. Gibson, "Performance of a Live Multi-Gateway LoRaWAN and Interference Measurement Across Indoor and Outdoor Localities," *Computers*, 2022.

[196] A. Farhad, D.-H. Kim, and J.-Y. Pyun, "Scalability of LoRaWAN in an Urban Environment: A Simulation Study," in *International Conference on Ubiquitous and Future Networks*, IEEE, 2019.

[197] P Ferrari *et al.*, "Simulating Scalability of a Transparent LoRaWAN Enhancement for Emergency Communication," in *International Workshop on Metrology for Industry 4.0 & IoT*, IEEE, 2022.

[198] D. Todoli-Ferrandis, J. Silvestre-Blanes, V. Sempere-Payá, and S. Santonja-Climent, "Adaptive Beacon Period Configurator for Scalable LoRaWAN Downlink Applications," *IEEE Access*, 2023.

[199] R. Marini, K. Mikhaylov, G. Pasolini, and C. Buratti, "LoRaWanSim: A Flexible Simulator for LoRaWAN Networks," *Sensors*, 2021.

[200] J. M. Marais, A. M. Abu-Mahfouz, and G. P. Hancke, "A Review of LoRaWAN Simulators: Design Requirements and Limitations," in *International Multidisciplinary Information Technology and Engineering Conference*, IEEE, 2019.

[201] M. Chen, L. Mokdad, and J. B. Othman, "MELoNS-A Modular and Extendable Simulator for LoRaWAN Network," in *International Wireless Communications and Mobile Computing*, IEEE, 2023.

[202] M. A. Almuhaya, W. A. Jabbar, N. Sulaiman, and S. Abdulmalek, "A Survey on LoRaWAN Technology: Recent Trends, Opportunities, Simulation Tools and Future Directions," *Electronics*, 2022.

[203] S. Idris, T. Karunathilake, and A. Förster, "Survey and Comparative Study of LoRa-Enabled Simulators for Internet of Things and Wireless Sensor Networks," *Sensors*, 2022.

[204] M. Słabicki, *FLoRa - A Framework for LoRa Simulations with OMNeT++*, https://flora.aalto.fi/, Accessed: 2023-08-22, 2022.

[205] S. Trendov, M. Gering, and E. Siemens, "Impact of LoRaWAN Transceiver on End Device Battery Lifetime," in *International Conference on Systems, Signals and Image Processing*, IEEE, 2023.

[206] H. H. R. Sherazi, L. A. Grieco, M. A. Imran, and G. Boggia, "Energy-Efficient LoRaWAN for Industry 4.0 Applications," *Transactions on Industrial Informatics*, 2020.

[207] R. K. Singh, P. P. Puluckul, R. Berkvens, and M. Weyn, "Energy Consumption Analysis of LPWAN Technologies and Lifetime Estimation for IoT Application," *Sensors*, 2020.

[208] J. Finnegan, S. Brown, and R. Farrell, "Modeling the Energy Consumption of LoRaWAN in ns-3 Based on Real World Measurements," in *Global Information Infrastructure and Networking Symposium*, IEEE, 2018.

[209] K. Banti, I. Karampelia, T. Dimakis, A.-A. A. Boulogeorgos, T. Kyriakidis, and M. Louta, "LoRaWAN Communication Protocols: A Comprehensive Survey under an Energy Efficiency Perspective," in *Telecom*, MDPI, 2022.

[210] T. Schmid, Z. Charbiwala, J. Friedman, Y. H. Cho, and M. B. Srivastava, "Exploiting Manufacturing Variations for Compensating Environment-Induced Clock Drift in Time Synchronization," *SIGMETRICS Performance Evaluation Review*, 2008.

[211] C. Palm, "Intensitatsschwankungen im Fernsprechverker," *Ericsson Technics*, 1943.

[212] T. Phung-Duc, "Retrial Queueing Models: A Survey on Theory and Applications," *arXiv preprint arXiv:1906.09560*, 2019.

[213] A. Nazarov, J. Sztrik, and A. Kvach, "A Survey of Recent Results in Rinite-Source Retrial Queues with Collisions," in *Workshop on Retrial Queues and Related Topics*, Springer, 2018.

[214] P. Tran-Gia and T. Hoßfeld, *Performance Modeling and Analysis of Communication Networks: A Lecture Note*. 2021.

[215] P. Wüchner, J. Sztrik, and H. de Meer, "Finite-Source Retrial Queues with Applications," in *International Conference on Applied Informatics*, 2010.

[216] P. Wüchner, H. de Meer, J. Barner, and G. Bolch, "A Brief Introduction to MOSEL-2," in *GI/ITG Conference-Measuring, Modelling and Evaluation of Computer and Communication Systems*, 2006.

[217] Linux manual page, *Tc*, Accessed: 2023-07-31, 2001. [Online]. Available: https://man7.org/linux/man-pages/man8/tc.8.html.

[218] Android Studio, *Android Debug Bridge*, Accessed: 2023-01-11, 2023. [Online]. Available: https://developer.android.com/studio/command-line/adb.

[219] Tcpdump & Libpcap, *Tcpdump Man Page*, Accessed: 2023-01-11, 2022. [Online]. Available: https://www.tcpdump.org/manpages/tcpdump.1.html.

[220] Wamser, Florian, *YoMo Wrapper App*, Accessed: 2023-01-11, 2018. [Online]. Available: https://github.com/lsinfo3/yomo-wrapperapp.

[221] J. Hooft *et al.*, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks," *Communications Letters*, 2016.

[222] H Riiser, P Vigmostad, C Griwodz, and P Halvorsen, *DATASET: HSDPA-Bandwidth Logs for Mobile HTTP Streaming Scenarios*, 2012.

[223] Wireshark, *Tshark Manual Page*, Accessed: 2023-01-11, 2018. [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html.

[224] P. Ni, R. Eg, A. Eichhorn, C. Griwodz, and P. Halvorsen, "Flicker Effects in Adaptive Video Streaming to Handheld Devices," in *International Conference on Multimedia*, ACM, 2011.

[225] N. Barman and M. G. Martini, "QoE Modeling for HTTP Adaptive Video Streaming–A Survey and Open Challenges," *IEEE Access*, 2019.

[226] Scikit-learn Developers, *Sklearn Model Selection: Grid Search CV*, Accessed: 2023-01-12, 2022. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

ISSN 1432-8801