Adaptive Multi-hop Networks for Industrial Applications with Dynamically Changing Traffic

Florian Meyer

Adaptive Multi-hop Networks for Industrial Applications with Dynamically Changing Traffic

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation

von

Florian Meyer

aus

Hamburg, Deutschland

2023

Date of Oral Examination	December 20th, 2022
Chair of Examination Board	Prof. Dr.-Ing. habil. Alexander Kölpin Institute of High-Frequency Technology Hamburg University of Technology
First Examiner	Prof. Dr. rer. nat. Volker Turau Institute of Telematics Hamburg University of Technology
Second Examiner	Prof. Dr.-Ing. Andreas Timm-Giel Institute of Communication Networks Hamburg University of Technology
Digital Object Identifier	https://doi.org/10.15480/882.5080

Acknowledgment

My thanks go to all those who have accompanied me on my way in the past years. I would like to single out Prof. Dr. Volker Turau, who not only contributed significantly to the outcome of this work through valuable advice and constructive feedback, but whose calm, appreciating and understanding manner also inspires me personally. I would also like to thank Prof. Dr. Timm-Giel for his expertise as a second supervisor.

Thanks to Ivonne Andrea Mantilla González for the joint research and many fruitful discussions. And, of course, I would also like to thank all my colleagues at the Institute of Telematics for the great time - I will never forget the sometimes absurd discussions during lunch. Especially many thanks to Christoph Weyer for the fun conversations about non-technical topics and the support in software and hardware questions. I also enjoyed talks about teaching and research with him and Dr. Marcus Venzke. I appreciate Florian Kauer for getting me in touch with openDSME.

Words cannot describe my deepest gratitude for the constant support and encouragement by my family, especially my parents, Claudia and Dirk Meyer. They have shown me a world of possibilities and always stood by my side with the most appreciated advice. Likewise, it goes to my friends who provided the necessary distraction in stressful times. Aiko Klostermann first sparked my interest in a PhD, and I am deeply grateful for it.

Last but not least, my thanks go to Franziska Gerfen, without whose love and support this work would not have been possible, and who also endured me working through the night from time to time.

Abstract

This dissertation explores techniques for adaptive multi-hop networks in the IIoT concerning time-varying, dynamically changing traffic. It demonstrates that efficient transmission of management traffic is crucial for high adaptability in IEEE 802.15.4 DSME, develops several techniques to increase adaptability and validates them through simulations, hardware experiments, and analytical models. In particular, QMA reduces collisions in management traffic by learning when it is feasible to transmit a packet. Sending multiple packets per GTS and group acknowledgments further relieve the management traffic load. Finally, dynamic CAP-reduction enables a fine-grained trade-off between management and data traffic.

Table of Contents

Chapter 1

Introduction

Over the last few years, wireless communication has reshaped the way we view and interact with our surroundings - not only in consumer products but also in industrial applications. Here, wired networks are gradually replaced by their wireless counterparts because they significantly reduce setup times and expenses. However, the transition is not trivial and numerous challenges lie ahead, some of them nearly solved, some of them subject to ongoing research. For example, the standardization of protocols and whole network stacks for the so-called Industrial Internet of Things (IIoT) enables efficient IPv6 communication between individual sensor nodes and cloud applications over existing Internet infrastructure. In short, it makes Wireless Sensor Networks (WSNs) more accessible and flexible, even for smaller setups, and paves the way towards Industry 4.0 and smart industrial plants [DNRC20, SOM14]. However, this flexibility comes at a cost: in comparison to its wired counterparts, wireless communication often features less throughput, higher delays, and significantly diminished determinism, i.e., unstable reliability and delays [SFDPH20]. This circumstance complicates application design and demands robust solutions throughout the entire network stack.

In general, communication in the IIoT can be classified as either primary or secondary traffic. Primary traffic comprises data messages from the application layer. To ensure high reliability, they are usually transmitted with exclusive channel access in dedicated time and/or frequency slots, albeit contention-based channel access is sometimes used, notably for application layer broadcasts. Consequently, secondary traffic consists of management messages, mainly to support primary traffic, e.g., for allocation of exclusive communications slots or broadcasts for route establishment. Secondary traffic is predominantly contention-based. Primary and secondary traffic

are closely linked and reciprocally interdependent. So congestion of secondary traffic inevitably leads to impaired performance of primary traffic because slots cannot be allocated and routes cannot be established.

The above statement especially holds in scenarios with time-varying or dynamically changing traffic conditions, where the changing conditions require constant adjustments of the communication partners. Thereby, fluctuating traffic patterns do not only emerge naturally, i.e., through external interference or fading effects but can also be introduced artificially, e.g., through the IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) protocol stack. Here, large IPv6 packets are segmented into several smaller Medium Access Control (MAC) layer packets for transmission on resource-restricted devices [SB11]. Consequently, bursts occur at the MAC layer for every IPv6 transmission. Irregular traffic may also arise because sensor nodes are usually battery-powered and thus rely on long sleep phases to sustain energy. Collected data is aggregated and then transmitted all at once, to keep the communication overhead at a minimum. In the worst case, bursts may cause queues to overflow, ultimately leading to packet loss.

These examples show that wireless communication requires mechanisms that ensure resilience against dynamically changing traffic conditions and external interference. Especially the MAC layer is responsible for guarding access to the shared wireless medium and, consequently, for coordinating the transmission and reception process of the whole network. As such, it has a substantial impact on the performance of a WSN. In the last years, various protocols have been introduced to address the demand for higher reliability, reduced energy consumption, and bounded delays. One notable example is IEEE 802.15.4 and in particular its sub-specifications Deterministic and Synchronous Multi-channel Extension (DSME) and Time-slotted Channel Hopping (TSCH). They are specifically designed for the novel challenges of the IIoT by employing frequency diversity, multi-hop communication, and fully configurable frame structures. Other examples include 5th Generation Cellular (5G), Bluetooth Low Energy (BLE), Long Range (LoRa), and IEEE 802.11ah [FLAA21]. They significantly increase reliability and delay in stable traffic conditions - usually by employing Time Division Multiple Access (TDMA)-based channel access. However, in the context of systems with dynamically changing traffic, this can be detrimental because TDMA slots first need to be allocated, introducing high delays and poor adaptability. In addition, slot allocations are typically conducted in the secondary traffic phase using contention-based channel access. Thus, they are highly susceptible to collisions in dense networks or when many management messages must be exchanged, e.g., when bursts in the primary traffic lead to many initiated slot allocations.

In IEEE 802.15.4 DSME, the primary traffic phase, or Contention Free Period (CFP), consists of so-called Guaranteed Time Slot (GTS), which are spread over time and frequency and allow exclusive access to the shared wireless medium. They are allocated during the secondary traffic phase, or Contention Access Period (CAP), in which slotted CSMA/CA is used. DSME permits the autonomous allocation of GTS between arbitrary pairs of nodes via a distributed 3-way handshake [IEE20]. Thus, DSME offers excellent prerequisites for a fast adaptation to dynamically changing traffic, yet also suffers from the problems discussed above. The agility of a DSME, and MAC protocols in general, is determined by a variety of factors - first and foremost by the message overhead for acquiring TDMA slots. Other influence factors are the number of transmittable packets per TDMA slot because fewer slots need to be allocated if more packets are sent per slot, and the contention-based protocol itself.

In this dissertation, we show that fluctuating and dynamically changing primary traffic results in a tremendous increase of management messages for slot allocation, and hence, in an oversaturation of the contention-based secondary traffic phase. To combat this effect and increase DSME's adaptability to such traffic patterns, we propose and assess several enhancements, such as sending multiple packets per GTS, Group Acknowledgments (GACKs), and delay-aware scheduling. In the process, novel and future techniques such as Machine Learning (ML) or Quantum Annealing (QA) are explicitly explored as solutions to the problems described. Most of the proposed techniques are applicable in other protocols with a TDMA-based channel access for primary traffic and a contention-based channel access phase for secondary traffic.

1.1 Contributions

In this dissertation, a variety of techniques is devised that facilitate the adaptation of MAC protocols to dynamically changing primary traffic by employing theoretical models, simulations and hardware experiments. They can be categorized as methods for secondary traffic, primary traffic, or a trade-off of both. In this context, the contributions of this work can be summarized as follows:

- A comparison of DSME and TSCH on hardware to identify a candidate for further experiments.

Secondary Traffic

- A thorough analysis of the impact of dynamically changing and fluctuating primary traffic on the slot negotiation process in DSME, i.e., on secondary traffic. The analysis identifies possible bottlenecks for further optimization.
- Proposal and evaluation of Q-learning-based Multiple Access (QMA), a Q-learning (QL) based multiple-access scheme for contention-based channel access during secondary traffic. QMA strives to increase reliability by dynamically learning when a transmission is likely to be successful and at which times it will result in a collision.

Primary Traffic

- Modification of the transmission process in DSME to enable sending multiple packet per GTS. An additional analysis shows whether this approach can effectively reduce the management overhead induced by fluctuating primary traffic.
- A study on reducing the acknowledgment overhead of primary traffic through different ways of transmitting GACKs in DSME.
- The formulation of diverse scheduling problems for TDMA-based systems in general and for protocols with alternating primary and secondary traffic phases in particular as Mixed-integer Linear Programs (MILPs). The resulting schedules can be used as baselines for dynamic schedulers.
- Reformulation of a selected MILP for delay-minimization as a Quadratic Unconstrained Binary Optimization (QUBO) problem for application in Simulated Annealing (SA) and future transfer to QA.

Primary and Secondary Traffic Trade-off

- Proposal of a novel CAP-Reduction (CAP-R) mechanism to allow for a dynamic trade-off between secondary and primary traffic.

1.2 Dissertation Structure

This dissertation is structured as follows: Chap. 2 provides an overview of the IIoT's state of the art. In particular, it attempts to highlight (dis)advantages of frequently utilized technologies with respect to dynamically changing primary traffic. Afterward, Chap. 3 covers the research approach, including a description of the utilized system model, relevant metrics, and tools. It also introduces ML techniques for resource-restricted devices in Sect. 3.6, which serve as a basis for QMA. The main part of the dissertation is devoted to four topics. First, Chap. 4 identifies DSME for further experiments, prevailing in a comparison with TSCH. Chap. 5 deals with *secondary traffic* in DSME by analyzing the slot allocation handshake in Sect. 5.1, which constitutes a majority of secondary traffic. It also presents QMA in Sect. 5.2, a QL based MAC for secondary traffic intending to increase reliability in the IIoT. Chap. 6 then describes improvements for *primary traffic*, where Sect. 6.1 describes the analysis of sending multiple packets per GTS and Sect. 6.2 different ways of transmitting GACKs. Afterward, scheduling methods for low delay through MILPs and QUBOs are described in Chap. 7. At last, Chap. 8 presents a method that enables dynamic use of the secondary traffic phase for primary traffic and discusses the trade-off in doing so. Chap. 9 concludes this dissertation.

Chapter 2

The Industrial Internet of Things

Efficient, flexible, affordable - these characteristics have fueled the Internet of Things (IoT)'s triumphal march across the globe, which has since changed the way we interact with our surroundings on a daily basis. Smartphones, televisions, cars, refrigerators, everything is connected and operates as part of a vast coherent system to ease our everyday lives. Hence, the transfer to industrial applications seems obvious and natural, as they can especially benefit from the underlying wireless communication [KRZ+20]. This way, the IIoT bridges the gap between business and industrial processes and offers endless possibilities through transparent Internet connectivity. It is thus an essential enabler for streamlined business processes and new technologies such as cloud, edge, and fog computing [KRZ+20, SSH+18] and hence, e.g., the live execution of ML algorithms on industrial data - something that seemed impossible only a few years ago.

On the technical side, there are still numerous challenges to solve to fully exploit the possibilities of the IIoT. First and foremost, it imposes several (and sometimes conflicting) requirements on the wireless transmission protocol due to its diversity. These could not yet be addressed by a single solution. In addition to safety and security, efficient transmission is a precursor to Industry 4.0. Therefore, this dissertation constitutes an essential contribution to the implementation of the IIoT in the near future.

The following sections first provide an overview of requirements, challenges and opportunities of the IIoT (Sect. 2.1) with a focus on time-varying and dynamically changing traffic (Sect. 2.3). Afterward, prominent existing technologies are presented, identifying their typical field of application (Sects. 2.4 and 2.6). The goal is to answer

the following questions:

- What prominent technologies exist in the IIoT?
- What is the cause of dynamically changing traffic?
- How is adaptation to dynamically changing traffic achievable and at what scale?
- Which technologies provide good adaptability to dynamically changing traffic?

2.1 Challenges and Opportunities

Due to the diversity of application scenarios, the IIoT imposes several conflicting requirements on MAC protocols while likewise offering promising opportunities [DNRC20, KRZ+20, MZC+21, SFDPH20, SSH+18]. These challenges and opportunities are further discussed below to ascertain possible application scenarios and (dis)advantages of the MAC protocols presented in Sect. 2.4.

Challenges

- **Energy:** Sensor devices are often battery-powered, so energy-efficient data transmission must be ensured at MAC level. At best, the transceiver should be switched on only for the actual transmission.
- **Interoperability / Coexistence:** The 2.4 GHz frequency band hosts many transmission protocols. A goal of the MAC layer is to minimize collisions between different protocols or even allow interoperability between them, e.g., with software defined radio techniques.
- **Latency:** In sensor-actuator networks, short response times are required. Therefore the MAC layer must guarantee low delays for transmission.
- **Reliability:** Reliability is not only important so that all data is transmitted, but also because packet loss leads to increased delay due to larger queues and energy consumption due to retransmissions.
- **Security:** Wireless networks are vulnerable to a variety of attacks due to the shared wireless medium. In addition, complex encryption algorithms usually cannot be implemented on resource-restricted devices.

- **Throughput:** Data-collection scenarios with thousands of sensor devices require high throughput close to the sink even if the collection interval of individual sensors is low.

Opportunities

- **Cost**: Depending on the application, the expenses for a wired IIoT deployment are determined to a non-neglectable part by the cost of wires, especially if remote sensors need to be connected. Wireless networks alleviate this problem as antennas are cheap and widely available.

- **Fault tolerance / Safety:** Wireless sensor networks are often constructed through dense mesh networks. In case of node failure, routes can be dynamically adapted so that data collection is still possible from other nodes.

- **Flexibility:** In wired networks, it is difficult to add or remove nodes at runtime as physical wires are required. Wireless mesh networks bypass this issue by allowing nodes to associate to the network at runtime over the shared wireless medium.

- **Mobility:** Similar to flexibility, mobility is usually supported in wireless sensor networks as nodes can dynamically disconnect from a network and join in another place. Several protocols even support a dynamic handoff so that nodes never have to leave the network.

2.2 The Industrial Internet of Things Stack

As discussed in Chap. 2, a variety of possible IIoT architectures and protocol stacks for different application scenarios exists. Several works identify and classify commonly used protocols [TS21, FLAA21, Zha19, DEA06], and Fig. 2.1 further elaborates their position in the IIoT protocol stack with their corresponding Open Systems Interconnection (OSI) layers. Sigfox, LoRaWAN, WirelessHART, and ZigBee represent full-stack solutions, with the latter two based on the physical layer of IEEE 802.15.4, thus achieving a high degree of specialization.

In recent years, there has been a trend towards IPv6 to make the large amount of IIoT devices addressable and connectable to the existing Internet infrastructure. Cellular protocols such as Long Term Evolution (LTE) or 5G, WiFi and Radio Frequency Identification Blink (RFID)/Near Field Communication (NFC) natively support IPv6,

while an abstraction layer is needed for most Wireless Personal Area Networks (WPANs). The Internet Engineering Task Force (IETF) designed 6LoWPAN specifically for this purpose [HCC09].

6LoWPAN, as specified in RFC4944 and RFC6282 [ZVKB07, Thu11], was originally designed for IEEE 802.15.4 and serves three main purposes: (De)fragmenting IPv6 packets with a minimum Maximum Transmission Unity (MTU) of 1280 B to / from MAC packets with a MTU of 127 B, mesh routing, and compressing IPv6 and User Datagram Protocol (UDP) headers to allow for more payload. Later RFCs added additional functionality like neighbor discovery (RFC6775 [CNB12]) and selective fragment recovery (RFC8931 [Thu20]). RFC7668 adds support for 6LoWPAN over BLE [NSI+15].

A number of application protocols exist for OSI-layer 5 through 7, including Advanced Message Queuing Protocol (AMQP), Constrained Application Protocol (COAP), Data Distribution Service (DDS), and Message Queuing Telemetry Transport (MQTT). AMQP is a message-oriented binary protocol with support for flow-control. It provides messaging guarantees (at-least-once, at-most-once, exactly-once) but requires a reliable transport protocol such as TCP. RFC7252 defines COAP [SHB14], a protocol specifically for low-power IIoT devices with a Representational State Transfer (REST)-like structure. This makes the protocol built on UDP compatible with traditional Internet protocols such as HTTP. COAP defines explicit rules for use with 6LoWPAN in RFC4944 [MKHC07]. At last, MQTT is a hierarchically organized protocol for systems with limited bandwidth based on publish-subscribe. So-called *message brokers*, which should run on non-resource-constrained devices, route data through the network and distribute it to subscribed clients without knowing the actual number of clients, due to the hierarchical structure.

In particular, the data link layer has a high impact on reliability and adaptability of an IIoT application. Therefore, Sects. 2.4 to 2.4.4.5 describe the MAC protocols displayed in Fig. 2.1 in detail.

2.3 Wireless Applications with Time-varying Traffic

There is a variety of reasons for dynamically changing traffic in WSNs. Some of them occur naturally, i.e., due to environmental effects or other reasons for packet loss, while others are artificially created in the application layer and network stack [PMSZ21, MPSZ18]. Here, dynamically changing traffic refers to any irregularities in the generation or reception process of primary traffic, especially bursts and time-

Figure 2.1: Protocol stack of the modern IIoT.

varying packets. In the following the main causes for dynamically changing traffic are further described.

- **Environmental Effects:** Dynamically changing traffic can occur due to environmental effects including external interference, fading, reflecting, and shadowing. The resulting packet loss leads to several retransmissions and thus to a time-varying reception at the communication partner.

- **Internal Interference:** Similarly to environmental effects, packets loss can occur due to internal interference, e.g., during contention-based channel access through CSMA/CA. Large topologies exhibiting many hidden-node problems amplify the effect.

- **Fragmentation:** An artificially introduced cause for irregular traffic is 6LoWPAN. Here, large IPv6 packets are fragmented into several smaller packets for transmission at the MAC layer. This entails a burst of packets at the MAC layer for every transmitted IPv6 packets.

- **Mobility:** Every application with mobility exhibits irregular traffic when mobile nodes are handed over from one access point to another.

- **Application:** A common application with sporadic, irregular traffic is alarm signaling. Here, packets are only generated in response to an event and then require timely transmission through the network to the gateway.

2.4 Medium Access Control Protocols

The MAC layer is a sublayer of the datalink layer responsible for managing multiple access to a shared transmission medium. For this purpose, MAC protocols usually use one of the following multiple access schemes:

- **Random access** is the simplest access scheme, but may result in collisions. For example with ALOHA, nodes send data as soon as it is available and, in case of a collision, again after a random backoff. CSMA/CA also senses the transmission medium before a transmission and transmits only if it is free. Thus, random access is useful for systems with sporadic traffic and scattered networks where the collision probability is low.

- **Time Division Multiple Access (TDMA)** provides exclusive channel access by dividing time into slots, which are assigned to pairs or groups of nodes for data exchange.

- **Frequency Division Multiple Access (FDMA)** works similar to TDMA, but instead of time it divides frequencies into multiple logical channels. TDMA and FDMA can be used in combination to expand the number of usable slots.

- **Code Division Multiple Access (CDMA)** allows simultaneous transmission of parallel data streams on a single frequency channel by augmenting the data signal with individual spread codes for each transmitter. If the codes are orthogonal, the receiver can then correlate the individual data signals using the original spread codes.

Other tasks of a MAC protocol include frame detection, addressing, transparent data transfer for higher layer, and transmission error detection.

2.4.1 Low Power Wide Area Networks

Low Power Wide Area Networks (LPWANs) constitute a class of wireless communication technology featuring long communication ranges up to 50 km, low power consumption, and low data rates. With these characteristics, LPWANs are best suited for monitoring and emergency signaling applications in secluded areas where they can sustain operation on battery or through energy harvesting. Two prominent LPWAN solutions are Long Range Wide Area Network (LoRaWAN) and Sigfox [MBCM19, KVN16]. Both protocols operate in the unlicensed ISM bands and are limited to star topologies.

LoRaWAN is a network protocol built upon the proprietary LoRa physical layer patented by Semtech. LoRa relies on a Chirp Spread Spectrum (CSS) modulation which spreads a narrowband frequency signal over a larger bandwidth and encodes bits in so-called *up* and *down* chirps through increasing and decreasing frequencies, respectively. Through a spreading factor, LoRa offers data rates between 50 bit s^{-1} and 300 bit s^{-1} with a payload length of up to 243 B in a tradeoff for transmission range. LoRaWAN enhances LoRa by making all base stations in an area receive transmitted messages and filtering duplicates on backend servers, also responsible for security, acknowledging, and localization.

Sigfox, by contrast, utilizes a Binary Phase Shift Keying (BPSK) modulation with 100 Hz Ultra Narrowbands (UNBs) limiting the data rate to 100 bit s^{-1} and payload

length to 12 B. This allows for low-power, low-cost, high-sensitivity antennas at the receiver. Base stations and backbone servers are proprietary, deployed by the Sigfox company, and do not support Acknowledgments (ACKs). Instead every end-device transmits packets over three (of 360) randomly selected orthogonal channels to ensure reliability. Downlink communication can only take place after uplink communication.

2.4.2 Cellular Communication Networks

Cellular networks have been providing ubiquitous connectivity in urban environments since the global introduction of the Global System for Mobile Communications (GSM) in the late 1990s. Since then, the primary mode of communication has changed from telephony to data exchange on the Internet, driven by ever higher data rates on licensed unimpeded frequencies.

In particular, later cellular standards such as Long Term Evolution Advanced (LTE+) enable easy-to-deploy, mobile, and geographically widespread IoT applications through pre-existing infrastructure and seamless handover between cells. However, a number of enhancements are required to make mobile networks fully attractive for these applications, as detailed in [ASHAM18]. The following sections provide a short insight into such extensions and an overview of the 5th generation cellular standard.

2.4.2.1 LTE-M

LTE-M or LTE-MTC is an Low Power Wide Area (LPWA) extension of the LTE standard with enhanced Machine Type Communication (eMTC) by the 3GPP. It allows for lower latency, extended coverage and higher energy efficiency at moderate data rates of 1 Mbit s^{-1} (LTE-Cat-M1) and 5 Mbit s^{-1} (LTE-Cat-M2) with bandwidths of 1.4 MHz and 5 MHz, respectively, and a reduced transmission energy of 20 dBm. EMTC is designed to be to be further used in 5G, while being backward compatible with existing cellular networks.

2.4.2.2 NB-IoT

Narrowband-Internet of Things (NB-IoT) is a LPWA technology targeting even lower power devices and is broadly perceived as an enabler for massive IoT deployments. It can be integrated into existing infrastructure by using LTEs guard band, multiplexed in-band with LTS signals without affecting its performance or operated as a standalone. Thus a data rate of up to 250 kbit s^{-1} can be achieved.

2.4.2.3 5th Generation Cellular

5G is the latest cellular technology that is currently being rolled out in many locations [ASHAM18]. It offers a number of improvements over previous versions and targets three main classes of use cases: enhanced Mobile Broadband (eMBB), massive Machine Type Communication (mMTC), and Ultra-reliable Low Latency Communication (URLLC). Thereby, eMBB provides high data rates of up to 20 Gbit s^{-1}, mMTC supports massive static deployments, e.g., in the IIoT, and URLLC satisfies strict delay and reliability requirements. 6G networks are not expected before 2030.

2.4.3 IEEE 802.11

Besides the basis for consumer Wi-Fi (IEEE 802.11ac/ax/be), the IEEE 802.11 standard defines several low-power extensions suitable for the IIoT [IEE21b]. Generally, IEEE 802.11 features a variety of modulation schemes (BPSK, Quadrature Phase Shift Keying (QPSK), 16-Quadrature Amplitude Modulation (QAM) to 1024-QAM) with different coding rates and channel bandwidths, allowing for high data rates (up to 1.201 Gbit s^{-1} in IEEE 802.11ax) or more robust communication depending on the application's requirements. Traditionally, a Distributed Coordination Function (DCF) utilizing CSMA/CA with Ready To Send (RTS) / Clear To Send (CTS) messages is used, but newer standards also provide a scheduled channel access (Point Coordination Function (PCF)), Multi-User Multiple Input Multiple Output (MU-MIMO) and beam forming. Multi-hop capability was introduced with IEEE 802.11s in 2007 [IEE11].

IEEE 802.11ah [IEE17], better known as Wi-Fi HaLow, is specifically designed for the IIoT, operating in the sub-1 GHz band (e.g., 863 – 868 MHz in Europe) allowing for ranges up to 1 km. It supports a wide range of physical layer configurations with channel bandwidths between 1 MHz and 16 MHz resulting in data rates between 150 kbit s^{-1} and 78 Mbit s^{-1} [TSS+21]. Intending to achieve a similar power consumption as Bluetooth, it significantly reduces the MAC header overhead, allows nodes to stay sleeping and ignore beacons (Target Wake Time (TWT)), groups nodes' channel access according to a TDMA scheme to avoid collisions (Restricted Access Window (RAW)), and indicates potential traffic in beacons (Traffic Indication Map (TIM)) [IEE17].

The IEEE 802.11ba amendment, issued in March 2021, further reduces energy consumption through the use of Wake-up Radios (WURs) [IEE21b]. The WUR achieves a power consumption of less than 1 mW and is merely switched on periodically to listen for WUR beacons for synchronization and wake-up frames signaling downlink

traffic. After receiving a wake-up frame, the device turns on a high-power transceiver for further communication [DLL+20].

2.4.4 Wireless Personal Area Networks

WPANs are wireless networks with ranges of 10 – 100 m and low power consumption, allowing long runtime using batteries. To increase coverage, WPAN participants can be interconnected to form mesh networks. As the name suggests, WPANs are tailored for geographically limited applications,, e.g., for data collection in industrial plants.The following sections describe prevalent WPAN protocols.

2.4.4.1 IEEE 802.15.4

Defining a communication protocol for Low-Rate Wireless Personal Area Networks (LR-WPANs), the IEEE 802.15.4 standard emphasizes low-cost, low-power applications [IEE20]. It classifies devices into two groups, namely Reduced-Function Devices (RFDs) and Full-Function Devices (FFDs), where RFDs can only communicate with FFDs but FFDs can communicate with all other devices. Consequently, RFDs are mostly small sensor devices while FFDs can also fulfill the role of a Personal Area Network (PAN) coordinator and thus perform essential tasks in the network, e.g., association and time synchronization. Additionally, it manages the network topology as a star, peer-to-peer, or cluster-tree topology.

Two modes of operation are available: beacon-enabled and non beacon-enabled. In the non beacon-enabled mode, nodes are not synchronized but communicate freely in a contention-based manner using unslotted CSMA/CA. The superframe (SF) structure of the beacon-enabled mode is illustrated in Fig. 2.2, where the PAN coordinator periodically transmits beacons for synchronization. This beacon interval, is divided into an inactive period and an active period with 16 equally sized time slots. Up to seven preassigned GTS can be used for exclusive channel access during the so-called CFP, while the rest is used for contention-based communication using slotted CSMA/CA in the so-called CAP.

The IEEE 802.15.4 standard specifies different physical layers, some of which operate on region-restricted frequencies. It provides 16 non-overlapping channels with 5 MHz spacing in the 2.4 GHz band using a Offset Quadrature Phase Shift Keying (OQPSK) modulation.

As the size of wireless IIoT applications grows, IEEE 802.15.4 quickly reaches the limits of its scalability - especially regarding reliability and bounded delays. Because of this, the IEEE 802.15.4e amendment from 2012 extends the existing standard

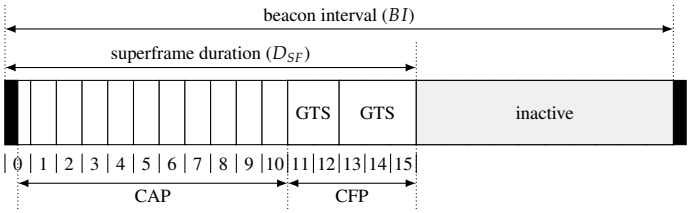


Figure 2.2: Superframe structure according to the IEEE 802.15.4 standard.

with additional MAC behavior modes to bridge this gap [ABM20, IEE12], each for a specific application:

- **Asynchronous Multi-channel Adaptation (AMCA)** for large deployments found in smart utility and infrastructure monitoring applications. AMCA does not support beacon-enabled PANs.
- **Deterministic and Synchronous Multi-channel Extension (DSME)** for commercial, industrial, and healthcare applications with stringent delay and reliability requirements such as sensor-actuator networks.
- **Low Latency Deterministic Network (LLDN)** for applications like factory automation aiming for minimal delays below 10 ms. LLDN is limited to star topologies.
- **Radio Frequency Identification Blink (RFID)** for providing an id and optional payload to other devices without association and acknowledgments, e.g., in access control or tracking applications.
- **Time-slotted Channel Hopping (TSCH)** for process automation applications with support for channel hopping to minimized the influence of multipath fading and external interference.

Of these MAC modes, AMCA and LLDN are omitted in recent versions of the standard [RN20, IEE20]. DSME and TSCH hence seem to be most relevant and are described in more detail in Sects. 2.4.4.4 and 2.4.4.5 respectively.

2.4.4.2 ZigBee

Since 2002, ZigBee is developed as a high-level communication protocol based on IEEE 802.15.4's physical and MAC layer specification. Providing a full-stack

solution, it is intended for low-power, low-data-rate applications like home automation or small sensor networks. As an enhancement to pure IEEE 802.15.4, it offers self-organized mesh networking built upon the Ad-hoc On-demand Distance Vector (AODV) algorithm for routing, i.e., every device is responsible for route discovery itself, and a coordinator is only required for gateway tasks. In contrast the IEEE 802.15.4e standards DSME and TSCH, ZigBee does not feature frequency diversity and is hence prone to external interference. ZigBee operates in a non-beaconed mode or a beacon-enabled mode, where devices are synchronized and the underlying IEEE 802.15.4 SF is divided into 16 time slots. Here, up to seven time slots can be assigned to specific devices increasing reliability and timeliness, since access to the slots is usually contention-based through CSMA/CA [LSH08].

Due to ZigBee's low robustness, it is rarely used in industrial applications. Nevertheless, the ZigBee IP standard enables IPv6 networking over 6LoWPAN and Routing Protocol for Low Power and Lossy Networks (RPL).

2.4.4.3 WirelessHART

The HART Communication Foundation (HCF) develops *Wireless HART* as a counterpart to the Highway Addressable Remote Transducer (HART) protocol, commonly used for industrial automation. Similarly to ZigBee, it is based on IEEE 802.15.4's physical layer but redefines the data-link, network, transport, and application layers to accomplish essential goals of the protocol. This includes simplicity of deployment, flexibility, scalability, reliability, security, and possibility for self-organization and self-healing. For this, it utilizes mesh routing organized by a single *Network Manager*, which also takes care of resource scheduling, path configuration, and time synchronization.

To diminish interference with other protocols in the 2.4 GHz frequency band, Wireless HART employs frequency diversity and a Frequency-hopping Spread Spectrum (FHSS) mechanism on 16 channels. Additionally, it uses a TDMA-based medium access with pre-scheduled, fixed-length slots for communication. All in all, it is considered one of the most popular wireless protocols for industrial process automation applications [PC09].

2.4.4.4 Deterministic and Synchronous Multi-channel Extension

IEEE 802.15.4 DSME divides time into multi-superframes (MSFs) and further into superframes (SFs) [IEE20], as shown in Fig. 2.3. SFs consist of 16 time slots, the first of which is used to transmit beacons containing network and time information

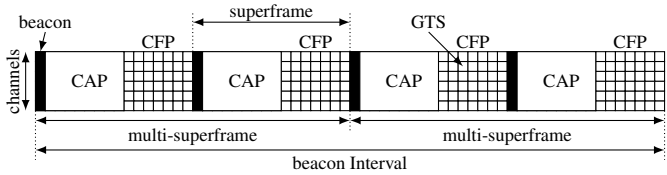


Figure 2.3: Structure of a DSME multi-superframe.

once every beacon interval. The following 15 slots are separated into two phases: a Contention Access Period (CAP) with 8 time slots and a Contention Free Period (CFP) with 7 time slots. The CFP contains so-called Guaranteed Time Slot (GTS) which are spread over time and frequency and allow exclusive access to the shared medium after initial allocation. They must be allocated during the CAP through a 3-way-handshake. Additionally, the CAP can be used for exchanging secondary traffic via CSMA/CA. To increase throughput in a trade-off with reactivity, DSME provides CAP-Reduction (CAP-R). This converts CAPs to CFPs, preserving only the first CAP of an MSF for secondary traffic. Other features of DSME include frequency hopping and automatic detection of beacon collisions.

The structure of a MSF is defined by the superframe order (SO), multi-superframe order (MO), and beacon order BO. Here, SO determines the duration D_S of a slot and thus the duration D_{SF} of a SF. The duration D_{MSF} of a MSF is specified by MO, and the length of a beacon interval BI is determined by BO. All durations can be calculated as multiples of 15.36 ms: $D_{SF} = 15.36\,\text{ms} \cdot 2^{SO}$, $D_{MSF} = D_{SF} \cdot 2^{MO-SO}$, and $BI = D_{MSF} \cdot 2^{BO-MO}$. Therefore, the number of SFs per MSF is equal to 2^{MO-SO}. A schedule of allocated GTS is repeated after every MSF. In order to transmit a packet with the maximum payload length of of 127 B, $SO \geq 3$ is required. With $SO = 2$ and $SO = 1$, 66 B and 18 B of payload can be transmitted respectively. Hence, MO, SO, and BO are natural choices for parameter studies regarding DSME's adaptability, as they determine the frequency and the length of the CAP and the CFP.

Distributed Slot Negotiation Fig. 2.4 shows the 3-way GTS allocation handshake of IEEE 802.15.4 DSME. It is initialized by the unicast transmission of a *GTS request* from a node v_0 to a node v_1. v_1 responds with a broadcast of the *GTS response* to inform all nodes in its neighborhood about the GTS that is going to be allocated. When v_0 receives the *GTS response*, it finalizes the handshake with a broadcasted *GTS notify* to also inform all nodes in its neighborhood about the GTS that is allocated.

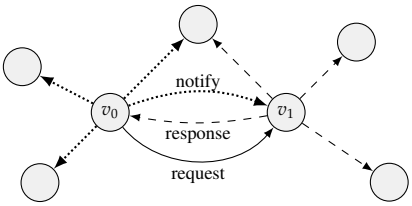


Figure 2.4: 3-way slot allocation handshake in IEEE 802.15.4 DSME.

If any of v_0's or v_1's neighbors have already allocated the GTS, i.e., a duplicate allocation occurs, or if any of the 3 messages is lost, the GTS allocation is rolled back using the same 3-way handshake. Afterwards, v_0 and v_1 can try to allocate another GTS.

2.4.4.5 Time-slotted Channel Hopping

Similarly to DSME, TSCH features a TDMA/FDMA-based channel access enabling bounded delays and a guaranteed bandwidth per node [IEE20]. Hence, it also increases the network capacity by employing a multi-channel approach, and mitigates the influence of external interference through a channel hopping algorithm, where a new channel is selected before every transmission. Consequently, it is well-suited for process automation and monitoring applications. In TSCH, beacons are only utilized for initial time synchronization, but all nodes synchronize on so-called *slotframes* afterward [IEE20], as shown in Fig. 2.5. Slotframes are divided into multiple time slots with a fixed length sufficient for the transmission of a maximum length data packet and the respective ACK. Several slotframes can simultaneously be active on different channels. Communication is realized as the assignment of time slots with an arbitrary channel offset to a pair of nodes. Thereby, *shared links* are dedicated to secondary traffic with CSMA/CA for channel access, while *dedicated links* provide exclusive channel access, e.g., for primary traffic. At last, it should be said that TSCH offers mesh networking and is thus not topology dependent. Algorithms for link scheduling, however, are not in the scope of the standard, and calculating efficient schedules is not a trivial task that even gains in complexity in networks with dynamically changing traffic. As discussed in Sect. 2.2, standardized protocols like 6TiSCH Operation Sublayer (6top) and IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH) help with this task but introduce a higher complexity to the network stack

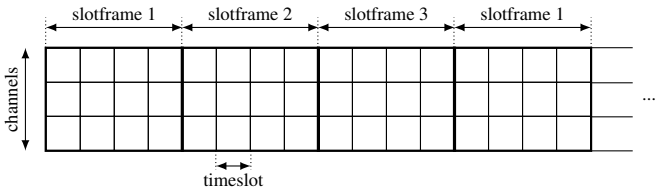


Figure 2.5: Structure of a TSCH slotframe over the time and frequency domain.

[CVV+21, VW18, SB11]. Additionally, TSCH has been proven to work well with static link schedules [DANLW15].

2.4.5 Discussion

This section summarizes some of the most prominent MAC protocols for the IIoT. A particular focus lies on adaptability to dynamically changing primary traffic. Two technologies seem particularly interesting for data collection in dense multi-hop networks: DSME and TSCH.

Although TSCH is characterized by a standardized network stack in the form of 6TiSCH and a high degree of flexibility, these aspects can be detrimental from an agility point of view. For slot allocations, the next higher layer, i.e., 6top, must be contacted in TSCH. This indirection results in additional overhead. An advantage of TSCH for the given application scenario is the already built in channel hopping.

For DSME, on the other hand, not the whole stack is standardized but it has been proven to work with 6LoWPAN without any further adaptation layers. That is because DSME comes with its own distributed slot allocation mechanism through a 3-way handshake, which provides excellent prerequisites for high agility. In addition, DSME offers a strict division into primary and secondary traffic, whereby the phase for secondary traffic is slightly longer. This increases adaptivity when a network is setup and many management messages need to be exchanged.

All in all, both DSME and TSCH are well suited for application in data collection scenarios with dynamically changing traffic. Therefore, a conclusive answer to the question whether DSME or TSCH is more suitable for quick adaptability is postponed and taken up again in Chap. 4 under consideration of realistic hardware experiments.

2.5 Routing Strategies

A variety of routing protocols exist for the IIoT, developed for a wide range of application scenarios and constraints. They can be classified by five properties: *Proactive* vs *reactive*, *hop-by-hop* vs *source*, *static* vs *adaptive*, *centralized* vs *distributed*, and *single-sink* vs *multi-sink* routing.

Greedy Perimeter Stateless Routing (GPSR) is a proactive, hop-by-hop, adaptive, distributed routing algorithm based on the geographic positions of nodes. These can be either preconfigured or determined via the Global Positioning System (GPS). Packets are not sent to specific addresses but are forwarded to given coordinates, so the geographic position of the sink node must be known in advance. Routes are formed by nodes periodically broadcasting their position in the network. These are then recorded in routing tables by nodes in the vicinity, so that a packet can always be forwarded to a node that is closer to the destination coordinates than the current node. If no closer node is found, the packet is forwarded greedily to a node in the rough direction of the destination coordinates.

Popular representatives for other routing protocols are, e.g., AODV, Optimized Link State Routing (OLSR), Open Shortest Path First (OSPF) or RPL. Especially AODV, OLSR and OSPF are not suitable for adaptive multi-hop networks, because AODV calculates routes on demand and OLSR and OSPF as link-state protocols require topology information of the entire network and therefore exhibit an excessive overhead. RPL, which is the de-facto routing protocol for 6LoWPAN, overcomes these problems but requires considerable initialization time since Destination Oriented Directed Acyclic Graphs (DODAGs) are built for routing. These are built through broadcasts and a target function to minimize, e.g., hop towards the sink. GPSR is used as the routing protocol in this dissertation, except for experiments in Chap. 4 using RPL.

2.6 Scheduling Strategies

Scheduling, i.e., the allocation of exclusive time and frequency slots to specific nodes, has a major impact on the performance of IIoT applications. There are three main types of scheduling: *distributed* scheduling, where any two nodes directly negotiate a slot, *centralized* scheduling, where a coordinator possesses information about the entire network and assigns slots to all other nodes, and autonomous scheduling where nodes schedule slots independently based on local information. Distributed scheduling exhibits swifter adaptability but cannot guarantee perfect scheduling in contrast to

centralized scheduling due to information asymmetry. The following sections given an insight into selected, existing autonomous (Sect. 2.6.2) and distributed (Sects. 2.6.1 and 2.6.3) scheduling methods.

2.6.1 Traffic-aware and Prediction-based Slot Scheduling

Traffic-aware and Prediction-based Slot Scheduling (TPS) is a distributed scheduling algorithm that aims to maximize reliability for nodes with traffic rates fluctuating around a fixed average. For this, each node determines the number of incoming packets per MSF P_t at time t and predicts the required slot count λ_t using an Exponentially Weighted Moving Average (EWMA) as

$$\lambda_t = \beta \cdot P_t + (1-\beta) \cdot \lambda_{t-1} \qquad \text{with } \lambda_0 = 0. \tag{2.1}$$

The particular slots are then chosen randomly. TPS uses a hysteresis to prevent frequent (de)allocations and includes over-provisioning of one slot or less to reduce potential queues. Therefore, the final slot target is given by

$$slot\ target = \begin{cases} \lceil \lambda_t \rceil & \text{for } \lambda_t - Scd_{t-1} > 0 \\ \lceil \lambda_t \rceil + 1 & \text{for } \lambda_t - Scd_{t-1} < -2\,, \\ Scd_{t-1} & \text{otherwise} \end{cases} \tag{2.2}$$

where Scd is the number of already scheduled slots.

2.6.2 Orchestra

Orchestra is an autonomous scheduler for TSCH networks in combination with RPL routing [DANLW15]. Autonomous means that scheduling decisions are neither centralized nor distributed, but solely based on local information - RPL neighbors and parents to be precise. Thus, Orchestra exhibits no signaling overhead. Orchestra schedules over-provisioned communication slots for MAC, routing, and application at distinct, mutually prime intervals. Four slot types are defined for communication:

- **Common Shared Orchestra Slots (CSs)** are shared between all network nodes for Reception (RX) and Transmission (TX) and are scheduled at a predefined time and frequency. This enables RPL to transmit broadcasts and discover neighbors.

- **Receiver-based Shared Orchestra Slots (RBSs)** are shared for TX between all neighbors of a node. The node itself schedules an RX slot at the same time and frequency so all neighbors can send packets to it.

- **Sender-based Shared Orchestra Slots (SBSs)** are shared for RX between all neighbors of a node. The node itself schedules an TX slot at the same time and frequency so it can send packets packets to any of its neighbors. This also enables broadcasting.

- **Sender-based Dedicated Orchestra Slots (SBDs)** are dedicated slots for transmission of packets from one node to another. This is the most reliable and energy-efficient solution for steady traffic.

2.6.3 6top Scheduling Functions

The IETF 6TiSCH working group standardized the 6top and 6top Protocol (6P) that enable distributed slot negotiation for neighboring TSCH nodes. 6P Scheduling Functions (6P-SFs) define when slots are added or deleted and execute 6P transactions. Requirements for 6P-SFs are listed in RFC8480 [VW18].

The Minimal Scheduling Function (6P-MSF) (RFC9033 [VVDD21]) is a simplified version of the On-the-fly Bandwith Reservation (OTF) protocol proposed in [PWW+16] and the default 6P-SF of 6P. It incorporates and combines advances of prior, discontinued 6P-SFs Scheduling Function 0 (SF0) [DGPA17], Experimental Scheduling Function (SFX) [DGPA18], and Autonomous Scheduling Function (ASF) [DVW18]. 6P-MSF schedules slots randomly based on the traffic demand of neighboring nodes and topology changes, and relocates them in case of collisions. It

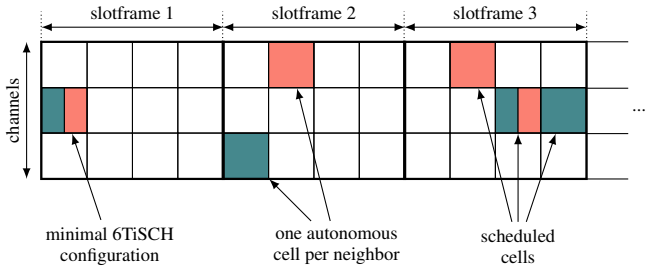


Figure 2.6: Structure of 6P-MSF with three slotframes and channels for two nodes.

also schedules autonomous slots with an offset based on the MAC address of nodes. Hence, the 6P-MSF does not optimize reliability, nor latency [TMV+21]. At its core, 6P-MSF utilizes three TSCH slotframes as shown in Fig. 2.6. The first hosts a minimal 6TiSCH configuration with a single slot to enable communication, the second is dedicated to said autonomous cells, and the third is used for scheduling shared and dedicated cells.

Several works describe 6P-SFs for the purpose of delay minimization to solve this issue, including Low Latency Scheduling Function (LLSF) [CWWV16], Recurrent Scheduling Function (RESF) [DSLF18], and YSF [TMV+21].

Chapter 3

Research Approach

This work relies on different methodologies to address the covered topics: theoretical models, simulations, and hardware experiments. For simulations, openDSME is executed within OMNeT++, whereas it is embedded into CometOS or Contiki-NG for hardware experiments. Therefore, the following sections first formally introduce the systems model (Sect. 3.1) and then cover commonly used metrics (Sect. 3.2) for easier replicability. Afterward, an overview of openDSME (Sect. 3.3), and insights into the simulation environment (Sect. 3.4) and hardware setup (Sect. 3.5) are provided. At last, techniques utilized in this dissertation are presented, i.e., machine learning for resource-restricted devices (Sect. 3.6).

3.1 System Model

This section formally introduces the system model underlying the following sections. In general, we consider data collection scenarios with a tree structure at the routing layer. Such a routing tree can be easily established in any network by using an appropriate routing protocol like GPSR or RPL. The network is assumed to be homogeneous in the sense that all nodes gather data at a common rate and send it towards the root, which either acts as a gateway or directly processes the sensor data. An exception are Sects. 5.1 and 6.1 that consider full-mesh routing with multiple source-sink pairs to resemble realistic sensor-actuator networks. Node failures are out of the scope of this work.

The network can be modeled as an undirected graph $\mathbb{G} = (\mathbb{V}, \mathbb{E})$. Every node $v_i \in \mathbb{V}$ represents a network device and an edge $(v_m, v_n) \in \mathbb{E}$ symbolizes a wireless link between devices v_m and v_n. For every v_i, denote $\mathbb{I}_i$ the set of nodes within its

interference range. Let the directed rooted tree $T = (\mathbb{V}, \mathbb{E}')$ with root v_0 be a routing tree of $\mathbb{G}$ so that $\mathbb{E}' \subseteq \mathbb{E}$. Denote by $\mathbb{N}_i \subseteq \mathbb{I}_i$ the neighbors of v_i in T, i.e., all communication partners of v_i. Similarly, $N_i^- \subseteq \mathbb{N}_i$ is the set of children and N_i^+ the parent of v_i in T. Let γ_i be the number of nodes in the subtree T_i rooted in v_i and $|\mathbb{V}| = \gamma_0$ be the total number of nodes in the network.

All nodes $v_i \in \mathbb{V} \setminus \{v_0\}$ generate an average of $\delta > 0$ packets with payload S per second. Specifically, packet generation occurs on average in intervals of length γ with bursts of ϵ packets per generation, so that $\delta = \epsilon \cdot \gamma^{-1}$. Two patterns are considered for the exact distribution of γ: A normal distribution with mean $\mu = \gamma$, and an exponential distribution with mean $\lambda^{-1} = \gamma$. Consequently, the latter case describes the number of bursts per second by a Poisson distribution with mean $\lambda = \gamma^{-1}$ and a one second observation interval. All packets P_i generated by node v_i must be forwarded one by one over the path $\rho_i = \{(v_i, v_j), (v_j, v_k), \ldots, (v_m, v_0)\}$ to v_0, i.e., via the edges of T. Every node has a finite buffer to store packets for transmission, where generated and received packets are immediately added to the buffer. Hence, the communication model can be described by *store and forward*.

3.2 Metrics and Statistics

To ensure traceability and reproducibility of the results obtained in the following chapters, this section first formally describes the utilized metrics. A fantastic *"survey on the metrics that matter"* is provided in [YKH17], covering metrics of different network stack layers. This section roughly follows the definitions of this paper but introduces changes for practical implementation and only presents a selection of the metrics most used in this dissertation.

Local metrics

- **Throughput (TP)** defines the amount of bytes a node v_i is transmitting per unit time, e.g., per second. Similarly, **Goodput (GP)** is the amount of (payload) data in bytes that v_i transmits in said time, i.e., without header and ACK overhead. TP and GP can also be reported in packet(s) s^{-1} if S is constant.

- **Dwell time (DwT)** is the duration that a packet remains at a particular node, i.e., the time between receiving or generating a packet and transmitting it.

- The **Energy consumption** of sensor nodes can be assessed in two ways: Either by measuring the total power consumption at the power supply or by gathering

information about time spent in different transceiver states. As the required power in each state is constant, the time in each state directly corresponds to the total power consumption.

- **Queue level** is the average number of packets queued for transmission at each node. It is closely related to delay and the Packet Delivery Ratio (PDR) because more packets in the queue lead to longer waiting time. Similarly, the PDR diminishes if queue drops occur.

End-to-end metrics

- The **Packet Delivery Ratio (PDR)** indicates the reliability of a network by expressing the generated packets P^i_{TX} by a node v_i to the number of packets P^{0i}_{RX} received by sink v_0 from v_i, i.e., $\forall_{v_i \in \mathbb{V} \setminus \{v_0\}} : PDR_i = \frac{P^i_{TX}}{P^{0i}_{RX}}$. Consequently, the PDR is a *multi-hop metric* for which $P^i_{TX} \geq 1000\,\text{packet(s)}$ must apply in order to obtain percentages with at least one decimal place.

- **Delay (D)** is the time between a packet generation at a node v_i and its reception time at the sink v_0. Obviously, there is a strong correlation between the path length $|\rho_i|$ and delay. Hence we normalize delay by the number of hops to ensure comparability between all nodes in the network

In general, the mentioned metrics are not recorded directly after the start of each network participant but with a certain delay. Specifically, the Start time (ST) ensures that all nodes are associated with the network before any packets are generated. Afterward, dummy packets, which are not recorded in the statistics, are generated during the Warmup Duration (WD). This enables evaluating a network in a settled state where an initial schedule is already created. Otherwise, the PDR would indicate severe packet loss for the first few transmissions where transmission slots are not yet available.

For simplified presentation, a network-wide average of the listed metrics is often calculated. It is the average over all nodes in the network, e.g., for the PDR it is given by $|\mathbb{V}|^{-1} \sum_{v_i \in \mathbb{V} \setminus \{v_0\}} PDR_i$. Confidence intervals are calculated over several averages, which are obtained by repeating the experiment. If not stated otherwise, experiments are conducted with at least 15 repetitions per parameter and all results are shown with a 95 % confidence level.

3.3 openDSME

openDSME is an open-source implementation of IEEE 802.15.4 DSME, developed by the Institute of Telematics at Hamburg University of Technology [Kau19, KKLT16b, BK16]. It is designed for portability to a variety of platforms, including network simulators like OMNeT++ (see Sect. 3.4.1) or Cooja, but also hardware platforms. In particular, the ATmega256RFR2 and the M3 Open Node are supported through CometOS (see Sect. 3.5.2) and the latter can also be used in conjunction with Contiki-NG (see Sect. 3.5.3) for experiments in the FIT IoT-Lab. Recently, Alamos et al. ported openDSME to RIOT, opening up a variety of other platforms as evidenced by their initial experiments with openDSME over LoRa [ÁKSW21a].

Fig. 3.1 shows openDSME's internal structure, which can be roughly divided into four central entities. The `DSMEPlatform` represents an abstraction layer between openDSME and a hardware or software platform and provides, among other things, transceiver switching, timer operation, and an interface for transmitting messages from a higher layer and receiving messages from the physical layer. The `DSMELayer` implements the core functionality according to the IEEE 802.15.4 standard and can be controlled via the standardized MAC Common Part Sublayer Service Access Point (MCPS-SAP) and MAC Sublayer Management Entity Service Access Point (MLME-SAP) interfaces. Alternatively, the `DSMEAdaptionLayer` can be used which provides additional functionality outside the scope of the standard such as scheduling and automatic association on top of the aforementioned interfaces to allow seamless integration of openDSME into a network stack.

openDSME is entirely written in C++ without any dependencies on the standard library to reduce code size. Instead, all required data structures are shipped with it. To the author's knowledge, there is currently no other implementation of IEEE 802.15.4 DSME available, apart from analytical models. For a comprehensive tutorial on openDSME refer to [KKT19b].

3.4 Simulation Environment

The following two sections introduce the simulation environment, i.e., the simulation library and framework OMNeT++, and INET, a standard model library for OMNeT++ which is utilized in this work.

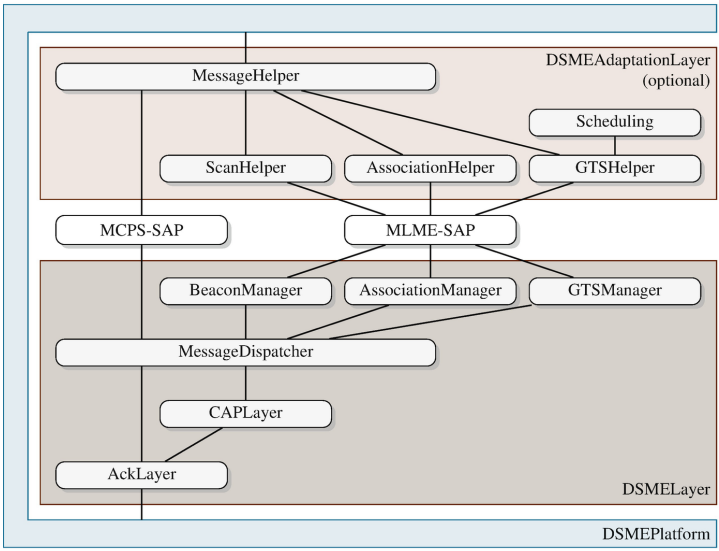


Figure 3.1: Internal structure of openDSME according to F. Kauer [Kau19].

3.4.1 OMNeT++

OMNeT++ is a discrete, event-based simulation library and framework intended for building network simulators [VH08, Var19]. It is written in C++ and allows users to implement models as instantiable components, which can be abstracted and bundled into compound components with help of OMNeT++'s NEtwork Description (NED) language. Networks can be defined similarly, where the term network applies to everything from wireless networks to wired networks to simple queuing networks. Simulation time is decoupled from real time, which allows the execution and assessment of large simulation models, multitudes faster than executing them on hardware.

Other features of OMNeT++ include an IDE based on Eclipse with extensive debugging support, which simplifies model development. OMNeT++ does not include a model library for network protocols but instead relies on INET as described in Sect. 3.4.2 [Var19, MVK19].

3.4.2 INET

INET is the de facto standard model library for OMNeT++, maintained by the developers of OMNeT++ with contributions from the community [MVK19]. It contains implementations for several application protocols, Internet protocols (e.g. Transmission Control Protocol (TCP), UDP, Internet Protocol Version 4 (IPv4), and IPv6), link and physical layer protocols (e.g. Ethernet, IEEE 802.11, and IEEE 802.15.4), and physical models. Following the example of OMNeT++, it employs modules that can be freely combined via gates.

Several simulation frameworks extend INET for specific applications like vehicular communication (Veins [SGD11]), LTE communication (SimuLTE [VSN15]), and Mobile Adhoc Networks (MANETs) (INETMANET [AI19]). For openDSME, a slightly modified fork of INET is utilized.

3.5 Hardware Environment

The following sections provide an overview of the setup and environment for the conducted hardware experiments .

3.5.1 FIT IoT-LAB

All hardware experiments are conducted on physical nodes in the Future Internet Testing (FIT) IoT-LAB [ABF+15], a group of seven testbeds scattered all over France. The testbeds feature different types of hardware and diverse environmental conditions and deployments. For example, the Grenoble testbed exhibits realistic conditions with nodes deployed in offices and hallways under the floor and in the ceiling, while the Strasbourg testbed features a laboratory room with little external interference, as shown in Fig. 3.2. All testbeds can be access via a unified web interface or REST API.

The target hardware are M3 Open Nodes, featuring 32 bit Cortex M3 CPUs with a maximum clock speed of 72 MHz, 64 kB Random-access Memory (RAM) and 256 kB Read-only Memory (ROM). It is shown in Fig. 3.3. The M3 Open Node is supported by the real-time operating systems CometOS and Contiki-NG, which are further explained in Sects. 3.5.2 and 3.5.3.

(a) Grenoble testbed. **(b)** Strasbourg testbed.

Figure 3.2: Testbeds of the FIT IoT-LAB utilized for experiments in this dissertation [IL22b]. In Grenoble, nodes are deployed in the floor and ceiling of an office building, in Strasbourg they are located in a shielded room [IL22a].

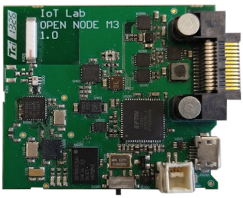


Figure 3.3: The M3-Open Node used for all hardware experiments.

3.5.2 CometOS

CometOS is a small, extensible real-time operating system for WSNs written in C++ and developed by the Institute of Telematics at Hamburg University of Technology [Unt14]. It is designed similarly to OMNeT++ in the sense that it implements functionality in modules that exchange messages via gates. These modules also can be executed in OMNeT++ through adapter classes. For hardware platforms, CometOS offers a non-preemptive scheduler, modules, gates, and message passing [Wei17]. It only requires a slim, hardware-dependent Platform Abstraction Layer (PAL) and supports dynamic memory allocation. A key feature of CometOS is its ability to deploy hybrid testbeds, i.e., one part of an experiment is simulated in OMNeT++ and another part on physical hardware, with both parts communicating as usual [UWT12]. This allows for an easier code transition from simulation to hardware platforms [Unt14]. In CometOS, openDSME is not evaluated with an IPv6 stack, but with flooding-based routing and a simple application layer that sends data periodically.

3.5.3 Contiki-NG

Contiki-NG (short for next generation) was created in 2017 as a fork of the Contiki real-time operating system [Kur18]. It is open-source, cross-platform, and specifically designed for the next generation of IoT devices, e.g., 32 bit Microcontroller Units (MCUs) like the ARM Cortex M3. In contrast to Contiki, it focuses on improved documentation, code structure, and RFC-compliant IPv6 communication. For this, it does not only contain an implementation of a standard IPv6 protocol stack according to the IETF consisting of IPv6, 6LoWPAN, 6TiSCH, RPL, and COAP, but also protocols like BLE, IEEE 802.15.4, and MQTT. In Contiki-NG, openDSME is executed as a sublayer of 6LoWPAN with IPv6 addressing and RPL for routing.

3.6 Machine Learning for Resource-restricted Devices

Throughout this dissertation, ML algorithms are utilized for optimizing the operation of the MAC layer, specifically the contention-based channel access (see Sect. 5.2). Thus, the following sections provide a brief overview of existing ML classes and explicate specific algorithms. A particular emphasis is put on applicability on resource-restricted sensor devices with less than 100 kB RAM, 200 Mbit ROM, a 32-bit processors with less than 100 MHz clock speed, and no floating-point unit. A more detailed overview of the target hardware is provided in Sect. 3.5.

The following sections introduce the three main classes of ML: Supervised Learning (SL) in Sect. 3.6.1, Unsupervised Learning (UL) in Sect. 3.6.2, and Reinforcement Learning (RL) in Sect. 3.6.3. Other learning problems, such as semi-Supervised Learning, self-Supervised Learning, or Transfer Learning (TL), usually borrow concepts from these three main classes, and are out of the scope of this work. For a good reference refer to [Zha20]. Sects. 3.6.3.2 to 3.6.3.4 in particular deal with Q-learning (QL) and a special QL variant for cooperative multi-agent systems, which forms the basis for the algorithm presented in this thesis.

3.6.1 Supervised Learning

Supervised Learning (SL) is a ML technique, which iteratively updates an approximation function m_{SL}, also called model, using tuples of training data (x_i, y_i) from

given sets of structured input data $\mathbb{X}_{SL} = \{x_0, \ldots, x_{|\mathbb{X}_{SL}|-1}\}$ and labeled output data $\mathbb{Y}_{SL} = \{y_0, \ldots, y_{|\mathbb{Y}_{SL}|-1}\}$ [Liu11] so that ultimately

$$m_{SL}(x_i) \approx y_i \qquad \text{for } 0 \leq i < |\mathbb{X}_{SL}| = |\mathbb{Y}_{SL}|. \tag{3.1}$$

Afterwards, m_{SL} can be used to predict outputs of unseen input data [CCD08]. In this context, the expressiveness of the input data, also called *features*, plays a major role for the required training time and the quality of the approximation function m_{SL}. Hence, careful feature selection is essential in SL [ZNLW19]. Popular SL models and algorithms include, but are not limited to, Naive Bayes (NB), Support Vector Machines (SVMs), Decision Trees (DTs), Random Forests (RFs) and Artificial Neural Networks (ANNs) [CNM06, SHG19], with the main task being the prediction of quantitative and qualitative outputs, i.e., regression and classification tasks [HTF08].

A disadvantage of SL is the necessity to acquire labeled, well-balanced datasets from realistic environments. This is especially difficult in the IIoT, where environmental conditions are constantly changing. Rare fault cases are often unknown and can hardly be covered sufficiently or reproducibly [WSM+21, AAG+21]. Hence, adaptation is often only possible with large amounts of data and complex models, the collection and design of which increase costs and are usually not suitable for resource-restricted devices [MYH+19].

The most popular machine learning models today are presumably ANNs, which can be efficiently trained through backpropagation [RHW86] and show great potential for abstracting and solving complex tasks [AJO+18]. For this very reason, there has been a lot of effort recently to bring complex ANNs to resource-restricted devices, as further elaborated in Sect. 3.6.1.1.

3.6.1.1 Lightweight Supervised Learning

A common workflow for SL is to train ANNs offline and then utilize the trained model on another device. On resource-restricted devices without floating-point unit and limited memory, however, even their execution, or *inference*, is a challenging task.

One solution, called *deep compression*, is characterized by a three-step pipeline [HMD15]. The idea is to compress an ANN to reduce its size and speed up inference. The first step is *network pruning*, where weights below a certain threshold are discarded because they do not influence the calculation. The next step is *quantization and weight sharing*, during which weights are first sorted into bins so that they can be accessed via common indices. At last, *Huffman encoding* is applied. This approach

achieves remarkable results by not only reducing the ANN's size but also enhancing execution time and prediction accuracy. It is also integrated into state-of-the-art frameworks such as *TensorFlow Lite* [AAB+15], increasing the accessibility of SL on resource-restricted devices.

3.6.2 Unsupervised Learning

In contrast to SL, Unsupervised Learning (UL) does not require labeled output data but discovers intrinsic structures and patterns in the input data $\mathbb{X}_{SL} = \{x_0, \ldots, x_{|\mathbb{X}_{SL}|-1}\}$ itself. Common applications are clustering, where data is grouped into logical sets based on similarity of the input features [Liu11], and dimensionality reduction [HSLZ21]. Popular UL algorithms include K-means, Principal Component Analysis (PCA), Singular Value Decomposition (SVD), and ANN autoencoders [Liu11].

In general, UL performs worse than SL in real-world applications as the number of input features tends to be large [UQR+19], and clusters may not be clearly separable [Zha20]. However, semi-supervised learning methods offer a promising, yet more unstable, solution where a model is first trained with supervised methods, and afterward, unlabeled data is introduced to it [vEH19].

3.6.3 Reinforcement Learning

The main idea of Reinforcement Learning (RL) is that an agent interacts with an environment by observing its state S_t at time t and executing an action $a_t \in \mathbb{A}_t$ in response to it, changing the state of the environment. Here, $\mathbb{A}_t$ is the set of available actions in state S_t. The agent then observes the next state S_{t+1} and receives a reward R_t in response to the executed action. The goal is to maximize the total reward, i.e., the sum of rewards attained in S_t and all future states S_{t+i}. This way, the agent may learn an optimal policy of actions for any finite Markov Decision Process (MDP). The interaction between the agent and the environment is also depicted in Fig. 3.4.

3.6.3.1 Exploration and Exploitation

Some works have already described the importance of exploration for the quality of RL, particularly when applied to MAC protocols [KCM+16, LJS20]. In essence, exploration ensures that a QL agent performs different (usually random) actions from time to time to evaluate their effectiveness. On the other hand, executing too many random actions leads to a sub-optimal behavior. Thus, there is a fine balance between exploration and exploitation of already learned behavior [SP17].

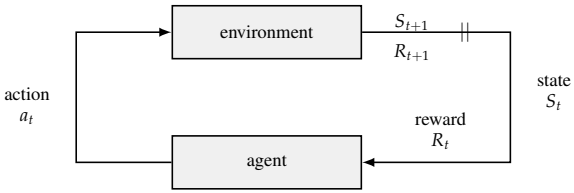


Figure 3.4: Interaction of an agent with its environment in RL.

For application on resource-restricted devices, a basic exploration strategy is a constant exploration rate ρ, however, it tends to be slow and increasing ρ would results in too many random actions. Another popular exploration strategy is ϵ-greedy. It selects random actions with probability $\rho > 0$, where ρ exponentially decreases over time. The goal is to reach an optimal stable state as fast as possible, however, it is a one-shot learning task. That means, ρ is never increased again and changes in the environment cannot be explored sufficiently fast. Additionally, ϵ-greedy initially results in many collisions because many random actions are selected [ALNT14]. A variety of other exploration strategies exists that can be categorized to different exploration classes - for an overview refer to [AGS+21].

3.6.3.2 Q-learning

Q-learning (QL) is a model-free, off-policy RL algorithm which optimizes a policy of taken actions using rewards. Model-free means that no underlying model is required for learning, but the algorithm finds an optimal policy by exploring $\mathbb{A}_t$. Exploration can be realized by taking random actions so that the desired policy does not have to be followed during training. i.e., off-policy learning [WD92].

QL uses so-called Q-values, associated with S_t and a_t, to express the quality of executing a_t in S_t. To incorporate future rewards while updating Q-value, QL relies on a weighted form of the Bellman equation which is expressed as

$$Q(S_t, a_t) \leftarrow (1 - \alpha_{RL}) Q(S_t, a_t) + \alpha_{RL} \left(R_t + \gamma \max_a Q(S_{t+1}, a) \right). \tag{3.2}$$

Here, α_{RL} is the learning rate, determining the influence of a newly calculated Q-value, and γ is a discount factor, determining the influence of future rewards [WD92]. In other words, γ expresses how far the agent looks into the future. For $\gamma = 0$, it only considers R_t, while it considers all future rewards for $\gamma = 1$. For systems without

final state, $\gamma < 1$ is chosen to ensure that $Q(S_t, a_t)$ does not become infinite by summation of all future rewards [Mel01].

3.6.3.3 Q-value Representation

One of the main challenges of QL is to represent the Q-values for every state-action pair. In theory, any linear function approximator can be exploited without influencing the stability of the algorithm. Especially for problems with large or continuous action spaces, this reduces the number of states, and thus memory usage, and the number of required training steps [DAMH21]. The idea is to approximate Q-values $Q(S_t, a_t)$ by carefully selecting features and updating the approximation using gradients. However, execution can still be quite resource-intensive, e.g., if linear combinations of many features are used.

As an alternative, a simple table with $|S_t \times \mathbb{A}_t|$ values can be employed [LJS20]. Training is as simple as writing a value $Q(S_t, a_t)$ to the designated cell so that a maximum of $|\mathbb{A}_t| + 1$ lookups is required for each training step, according to Eq. (3.2). Hence, this approach is applicable on resource-restricted devices.

At last, it should be said that there has been a lot of effort to enable QL with non-linear function approximators, e.g., deep ANNs [AAT+19]. Training ANNs through backpropagation is quite resource-intensive, rendering them unsuitable for online-training on embedded devices. Additionally, storing weights consumes plenty of memory and hence only pretrained ANNs can be utilized as elaborated in Sect. 3.6.1.1.

3.6.3.4 Cooperative Multi-agent Q-learning

In the IIoT, agents are usually represented by network nodes that execute the QL algorithm locally but are expected to learn a globally optimized behavior. This is not feasible with regular QL, so adapted QL algorithms for multi-agent systems are required.

Lauer and Riedmiller propose an enhanced QL algorithm for cooperative multi-agent systems [LR00] under the assumption that complex interactions between multiple agents cannot be modeled by a local MDP. Hence, they utilize a global Q-table which models rewards for all combinations of actions in the network. An example is given by Tbl. 3.1 for two agents and two actions a' and a".

Global Q-table		agent 0	
		a'	*a*"
agent 1	*a*'	1	-1
	a"	-1	10

agent 0 (local)	
a'	*a*"
1	10

agent 1 (local)	
a'	*a*"
1	10

Table 3.1: Example of a global Q-table for two agents with two actions *a*' and *a*", and the respective local Q-tables.

Global Q-table		agent 0	
		a'	*a*"
agent 1	*a*'	10	-1
	a"	-1	10

agent 0 (local)	
a'	*a*"
10	10

agent 1 (local)	
a'	*a*"
10	10

Table 3.2: Example of a global Q-table of a wrong final policy for two agents with two actions *a*' and *a*", and the respective local Q-tables.

Under the assumption that all agents behave cooperatively, i.e., they try to choose actions which maximize the reward from the global Q-table, every agent updates its local Q-table per state S_t and action $a_t \in \mathbb{A}_t$ according to

$$Q(S_t, a_t) \leftarrow \max\{Q(S_t, a_t), R_t + \gamma \max_a Q(S_{t+1}, a)\} \tag{3.3}$$

Here, $Q(S_t, a_t)$ is only updated if a newly calculated Q-value is larger than $Q(S_t, a_t)$. This ensures an optimal policy after all possible actions have been explored. For example, agent 1 could choose action *a*". Depending on agent 0, agent 1 experiences two different rewards, -1 and 10. It only stores a reward of 10 for action *a*", as given by Eq. (3.3), under the assumption that agent 0 also chooses action *a*" to maximize the global reward. Thus, the agents achieve cooperation.

A problem arises if multiple policies lead to an optimal reward as shown in Tbl. 3.2. The combination of actions (*a*', *a*') and (*a*", *a*") both yield a reward of 10 and in compliance with Eq. (3.3), both agents update their local Q-values to $Q(S_t, a') = 10$ and $Q(S_t, a'') = 10$. Therefore, agent 0 could choose action *a*' while agent 1 selects action *a*" as both actions yield the same reward. The combination (*a*', *a*"), however, only yields a reward of -1. Thus, the agents fail to agree on an optimal policy.

A solution is to maintain the optimal policy $\pi(S_t)$ per state as a separate, additional table. $\pi(S_t)$ is learned in parallel to local Q-values as

$$\pi(S_t) \leftarrow \begin{cases} \underset{a}{\operatorname{argmax}}\, Q(S_t, a) & \text{if } \max_a Q(S_t, a) > \\ & R_t + \gamma \max_a Q(S_{t+1}, a) \\ \underset{a}{\operatorname{argmax}}\, R_t + \gamma Q(S_{t+1}, a) & \text{otherwise} \end{cases} . \tag{3.4}$$

That means, an agent only selects a new action for S_t if the associated Q-value is strictly greater than the Q-value of the current policy $\pi(S_t)$. This way, a switch of actions is avoided for duplicate global rewards, and all agents agree on policy $\pi(S_t)$ which yielded the duplicate reward first.

In their work, Lauer and Riedmiller mention that the described algorithm is not applicable to stochastic environments, and they later propose an alternative algorithm [LR04]. It, however, requires more memory and is hence less feasible for resource-restricted devices. Sect. 5.2.2 proposes an alternative solution.

Chapter 4

A Comparison of DSME and TSCH on Hardware

As clarified in the discussion in Sect. 2.4, both TSCH and DSME exhibit promising properties for high adaptivity to dynamically changing traffic. For TSCH, this includes the standardized IIoT stack, dynamic frequency hopping, and a flexible frame structure that allows cells to be used for several purposes. In DSME, the distributed slot negotiation handshake and the CAP stand out, which enable fast (de)allocation of slots.

Despite several works in the literature concerning the theoretical evaluation and comparison of DSME and TSCH, there are - to the authors knowledge - little to no works performing a hardware-based comparison. Consequently, the following sections address this shortcoming by comparing both protocols on hardware with a focus on their adaptability to dynamically changing traffic. In particular, Sect. 4.2 briefly describes the integration of openDSME into the Real-time Operating System (RTOS) Contiki-NG, and Sects. 4.3.2, 4.3.4 and 4.3.5 are dedicated to the comparison of DSME's and TSCH's association time, slot allocation time, reliability, and throughput in data collection scenarios, respectively.

The results show that both protocols are suitable for application in the IIoT but DSME provides better adaptability, where the major differences can be summarized

---

Excerpt of an extended work to be published in collaboration with I. Mantilla-González. The author was responsible for the integration of openDSME into Contiki-NG and for gathering and describing the following results. Additional insights and results have been collected by I. Mantilla-González.

as follows:

- TSCH features faster association by waiving dedicated messages
- DSME exhibits a significantly lower slot allocation time
- DSME achieves higher reliability, but TSCH a lower dwell time and higher throughput

4.1 Related Work

Few works on the comparison of DSME and TSCH exist [PCN21, APMB15, CMML20, KSKT18, JAG+16], most of them built upon theoretical models. Alderisi et al. evaluate DSME's and TSCH's delay in realistic process automation environments by means of simulation in OMNeT++ [APMB15]. They show that DSME outperforms TSCH in terms of end-to-end delay for networks with more than 30 nodes. Similarly, Kurunathan et al. evaluate all IEEE 802.15.4e protocols using a theoretical model in terms of throughput and delay [KSKT18]. They confirm the results from [APMB15] and also attribute higher throughput to DSME.

Another theoretical model in the form of a Markov chain is provided in [CMML20]. They extend the consideration of throughput and delay to energy consumption. Additionally, they verify their results by simulating DSME and TSCH in two different simulators due to the lack of a common implementation. Contradicting the previously presented works, they find that TSCH provides higher throughput and lower latency than DSME. This reveals a serious problem of both protocols, namely that there is no recommendation for setting the relevant configuration parameters, and thus comparability across multiple works is rarely given.

Finally, it should be said that openDSME has recently been ported to RIOT [ÁKSW21b], which provides an implementation of TSCH and the 6LoWPAN stack similar to Contiki-NG [BGH+18]. Thus, a comparison of the two protocols in RIOT is an obvious next step. At this time, however, to the author's best knowledge there exist no comparisons of DSME and TSCH on hardware.

4.2 Integration with Contiki-NG

In the context of this dissertation, openDSME has been ported to the RTOS Contiki-NG and interfaced with the existing 6LoWPAN stack and MQTT application layer, where Fig. 4.1 details its position in the Contiki-NG stack. As highlighted, mainly the

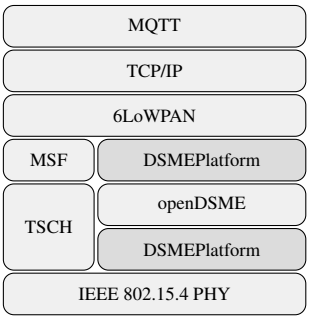


Figure 4.1: Integration of openDSME in the Contiki-NG software stack.

implementation of the `DSMEPlatform` class is required, which was reimplemented based on a deprecated openDSME port for Contiki [Kau19]. The class handles essential communication with the upper protocol layers and the IEEE 802.15.4 physical layer, memory and timer management, configuration of the MAC layer, and logging. Contrary to the TSCH implementation, an additional hardware timer for DSME has been set up, since the software timers provided by Contiki-NG did not attain the required precision for the execution of DSME. This insight was achieved through an elaborated series of tests.

The TSCH implementation has been extended by an existing 6P-MSF implementation, which is currently under review for incorporation into Contiki-NG [ÁKSW21b]. Corresponding MQTT clients implementations have been written for both protocols.

4.3 Hardware Evaluation

The following sections describe the realization and results of the hardware-based comparison of TSCH and DSME in the FIT IoT-LAB. It is performed using Contiki-NG on M3 Open Nodes as described in Sect. 3.5.1. The evaluation is divided into three parts, preceded by the scenario description in Sect. 4.3.1. The first part in Sect. 4.3.2 examines DSME's and TSCH's association process, Sect. 4.3.4 the slot allocation process, and Sect. 4.3.5 their reliability, dwell time, and energy consumption in realistic data-collection scenarios.

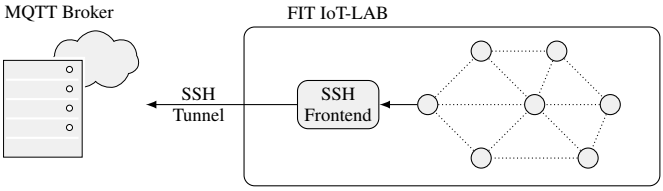


Figure 4.2: Scenario description for the hardware-based comparison of DSME and TSCH.

4.3.1 Scenario Description

The following experiments are performed in FIT IoT-LAB's Strasbourg testbed, where nodes employ MQTT and periodically collect and send data to an MQTT broker in an external network to mimic realistic conditions. To avoid interference from external parties, a private MQTT broker is set up, which is accessible to the gateway via an Secure Shell (SSH) tunnel. The scenario is detailed in Fig. 4.2. For the experiments, nodes are deployed in a grid-like physical topology as shown in Fig. 4.5c, for which RPL then creates a logical routing tree with node 28 acting as the gateway. It should be noted that this logical topology is subject to constant variation and MQTT is only used as an application layer - all measurements are done directly at the MAC layer.

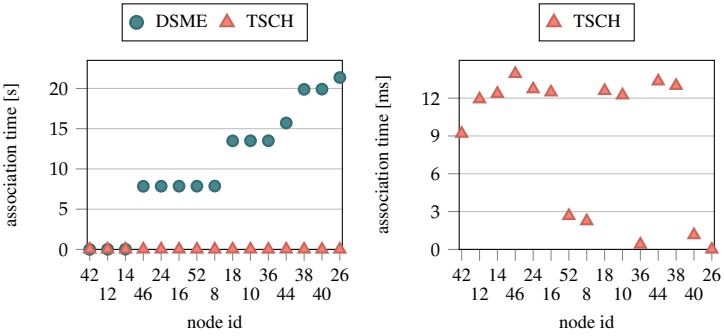
Tbl. 4.1 summarizes the most important parameters for the experiments with TSCH and DSME. Since both protocols are quite different in detail and have a large number of configuration parameters, the configuration was selected as recommended by standard documents or related work. This is especially true for the scheduling algorithms 6P-MSF and TPS. Obviously, both protocols can continuously be further optimized for specific scenarios, but the following experiments are intended to compare both protocols in a realistic configuration and show fundamental differences on protocol level.

4.3.2 Association Time

The association process, i.e., how nodes join the network, is an elementary difference between DSME and TSCH. DSME utilizes a handshake while TSCH synchronizes on any message transmitted by nodes that have already joined the PAN. One should note that RPL, and hence the routing from Fig. 4.5, does not have an influence on the association process. Fig. 4.3 shows the association time of all network participants using DSME and TSCH for a single run - yet several runs have been performed

Protocol	Scheduler	Slots	Topology	Nodes	Collection Interval
Association Experiment					
DSME	TPS	28	grid	10 - 30	1000 ms
TSCH	6P-MSF	101	grid	10 - 30	1000 ms
Allocation Experiment					
DSME	TPS	28	pair, star, grid	1 - 15	100 – 1000 ms
TSCH	6P-MSF	101	pair, star, grid	1 - 15	100 – 1000 ms
Data Collection Experiment					
DSME	TPS	28	grid	15	1000 ms
TSCH	6P-MSF	101	grid	15	1000 ms

Table 4.1: Parameters for the hardware-based comparison of DSME and TSCH.



(a) Network association time of DSME and TSCH.

(b) Network association time of TSCH.

Figure 4.3: Network association time of DSME and TSCH for all network participants.

leading to identical results. It can be seen that TSCH outperforms DSME in terms of association time due to the shortened allocation process, resulting in about 20 ms using TSCH and about 21 s using DSME for all 15 nodes to associate. An interesting observation is that nodes associate in specific intervals using DSME corresponding to a beacon interval and a CAP, as illustrated in Fig. 4.3a. On the other hand, the association timings are more diverse with TSCH because a node is considered associated as soon as it receives any message from an associated node.

Fig. 4.4 shows the total network association time, i.e., the time between the first association of a node with the coordinator and the association of the last node for

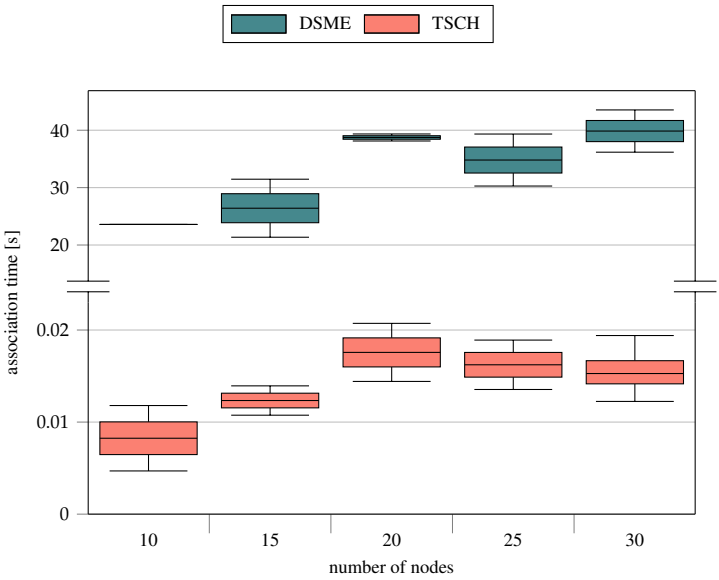
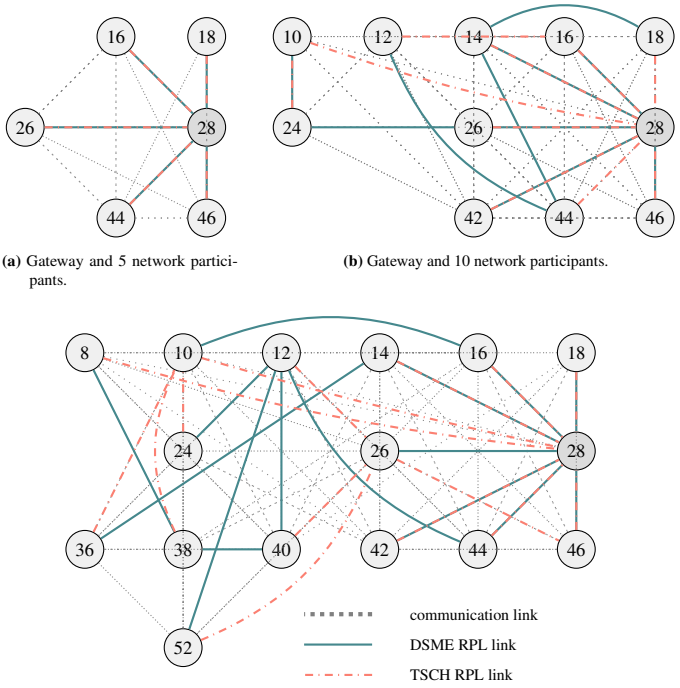


Figure 4.4: Total network formation time for an increasing number of nodes.

increasing network sizes. Notice the break in the y-axis. As discussed, TSCH achieves significantly lower association time. For both protocols, the association time increases with increasing number of network participants, although the total network association time is rather defined by the maximum number of hops to the sink than the total number of participants. Congestion of the respective secondary traffic phases could not be detected.

4.3.3 RPL Network Configuration

The following sections describe experiments with RPL routing in the same base configuration, applied to TSCH and DSME networks. To gain a better understanding of the established networks' structures, Fig. 4.5 shows the neighbor graph using RPL for fixed physical topologies with different numbers of nodes. From this graph, RPL generates routing graphs for TSCH and DSME, respectively. One should note that the established routes are subject to ongoing changes and Fig. 4.5 only represents a configuration at one point in time.



(a) Gateway and 5 network participants.

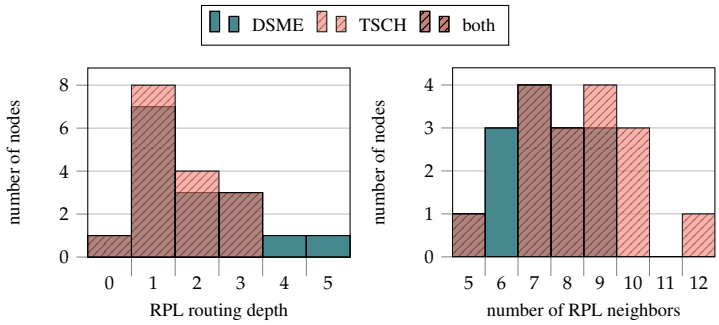
(b) Gateway and 10 network participants.

(c) Gateway and 15 network participants.

Figure 4.5: Neighbor graph and RPL routing graph for DSME and TSCH with gateway and RPL root 28.

As illustrated in Fig. 4.5c, DSME and TSCH use similar connections close to the sink (node 28) but different routes everywhere else in the network. That is because RPL greedily selects a parent with minimum distance to the sink and only changes it if another node advertises a significantly shorter routes. Insignificant changes are ignored so that nodes often remain with their initial selection. Fig. 4.5 also details the RPL network for five and ten network participants, where a star topology is created for both DSME and TSCH with five participants.

For the following experiments, it is yet particularly important that all networks have the same characteristics. Fig. 4.6 shows histograms of the relevant statistics: depth and density. For the exemplary routing graphs from Fig. 4.5c, DSME and TSCH



(a) Nodes per RPL depth.

(b) Nodes per RPL neighbors.

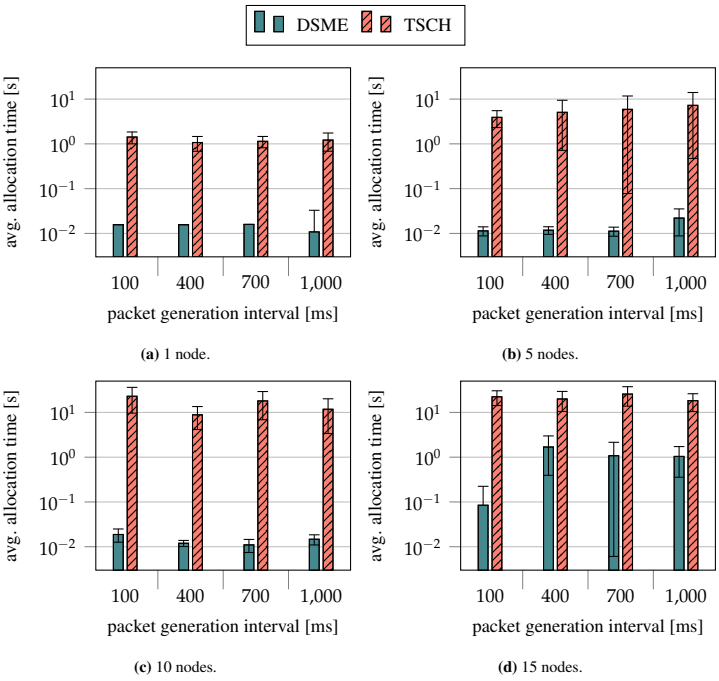
Figure 4.6: Number of nodes with a given depth and with a given number of neighbours in a RPL network using DSME and TSCH.

exhibit about the same number of nodes per depth but DSME also has nodes with depth four and five, as shown in Fig. 4.6a. On the other hand, the TSCH is slightly denser as illustrated by Fig. 4.6b. However, with both protocols nodes have at least five neighbors so that the network is quite dense.

4.3.4 Slot Allocation Time

The slot allocation time is crucial for the adaptivity of MAC protocols, since it determines how fast they can react to changes in traffic. Therefore, Fig. 4.7 shows the average slot allocation time of DSME and TSCH for increasing packet generation rates and increasingly larger and denser networks. For this purpose, we define the slot allocation time as the duration between sending an allocation request and receiving the allocation response with TSCH, or until the *GTS notify* is sent with DSME.

It can be seen that the slot allocation time increases with increasing network density, i.e., with an increasing number of nodes. That is because both protocols utilize contention-based channel access for exchanging secondary traffic and are hence susceptible to collisions that result in retransmissions and additional delay. Across the different packet generation intervals, however, there is little difference in allocation time because the Contiki-NG MQTT implementation limits throughput to increase reliability. This problem is explained in detail in Sect. 4.3.5. It should be noted that the median slot allocation time is generally low for both protocols, but high outliers occur caused by collisions during the allocation process as shown in Fig. 4.8. DSME



(a) 1 node.

(b) 5 nodes.

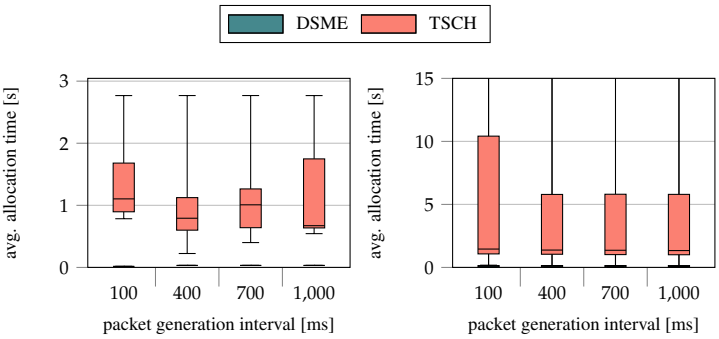
(c) 10 nodes.

(d) 15 nodes.

Figure 4.7: Average slot allocation time for DSME and TSCH for increasing packet generation intervals and network sizes.

exhibits a significantly lower slot allocation time than TSCH, and thus, higher agility. This is mainly attributable to the CAP and the built-in allocation mechanism.

DSME's and TSCH's allocation timing is broken down over multiple runs in Fig. 4.9, thus also showing the time variance in slot negotiation. For DSME, the scheduler is executed at the beginning of the CFP so that slot requests are initialized outside of the CAP and then dispatched in the next CAP. The other messages are delayed primarily due to the randomness of the CSMA/CA algorithm but transmitted within a single CAP, so that the total allocation time is low. For TSCH, as shown in Fig. 4.9b, the request is always sent shortly after its initialization. The other messages, however, are significantly delayed and exhibit a large spread due to the indirection via 6top and the high number of slots in the frame structure.



(a) 1 node. **(b)** 15 nodes.

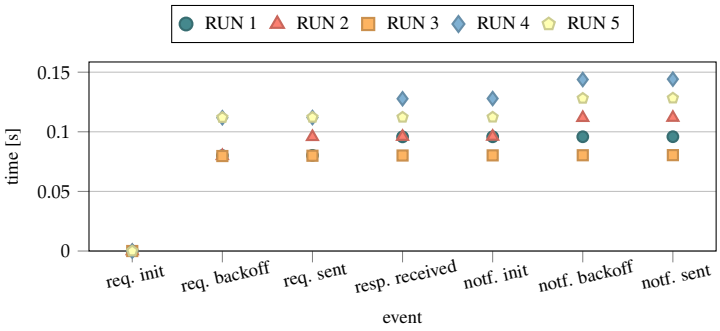
Figure 4.8: Median slot allocation time and upper and lower quartile for DSME and TSCH for increasing packet generation intervals and network sizes. The whiskers indicate minimum and maximum values.

Finally, Fig. 4.10 shows the number of successful and failed slot allocation attempts. It can be seen that DSME performs slightly more successful allocations than TSCH but also exhibits a significantly higher number of failed transmissions. The main reason for failed transmissions are lost packets and resulting timeouts, i.e., when no ACK is received. The shown results and further testing reveal that the packet generation interval has no consistent influence on the success rate and the variation is caused by external influences, e.g., interference.

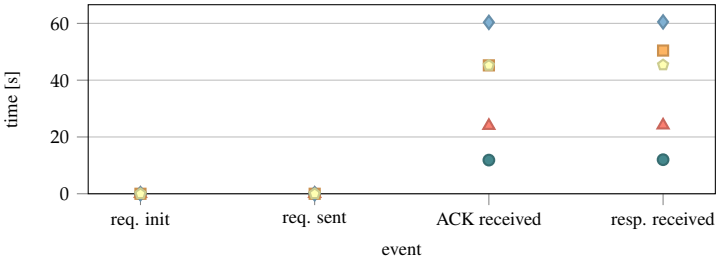
4.3.5 Data Collection

The following subsections examine TSCH and DSME in a realistic data-collection scenario, as described in Sect. 4.3.1. One should notice that metrics are not obtained at the application layer but at the MAC layer to exclude unwanted side effects.

Through a series of tests, it became apparent that the MQTT backend implementation in Contiki-NG drops packets received from the MQTT frontend. The ratio of dropped packets is shown in Fig. 4.11 and is high for both protocols and all packet generation intervals. This behavior results in the MAC protocols receiving about the same number of packets from the higher layers for transmission for all packet generation intervals. The behavior suggests that Quality of Service (QoS) level 1 is used internally, where an ACK must be received before a new packet can be sent. However, the MQTT frontend defaults to QoS level 0, so the reason could not be



(a) Slot negotiation process of DSME.



(b) Slot negotiation process of TSCH.

Figure 4.9: Timing of the slot negotiation process in DSME and TSCH.

finally clarified and should be reconsidered in another work. The following data collection evaluation therefore only considers a packet generation interval of 1 s.

4.3.5.1 Reliability and Delay

Fig. 4.12 shows DSME's and TSCH's link PDR, i.e., the ratio of successfully delivered packets to transmission attempts to an immediate neighbor. At all nodes, DSME reaches a PDR of 100 % while TSCH loses packets at all nodes. Especially at nodes with high contention like node 14, 26, 38, and 40, many packets are lost. The results indicate that, e.g., node 26 and 14 are subject to reciprocal interference in shared or dedicated slots.

Despite the lower PDR, TSCH attains a significantly higher throughput as shown in Fig. 4.13a where node 16 and node 26 represent nodes in one-hop distance to the

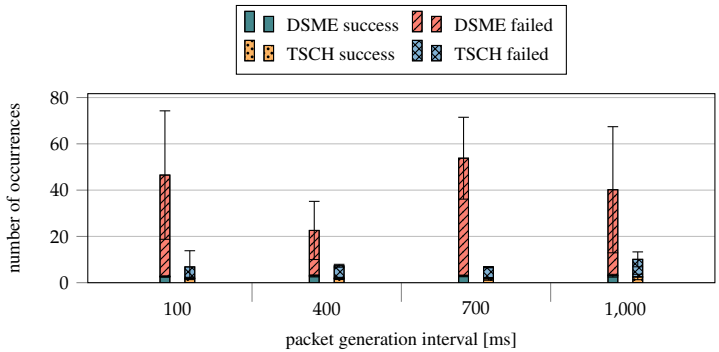


Figure 4.10: Successful and failed slot allocations using DSME and TSCH for increasing packet generation intervals.

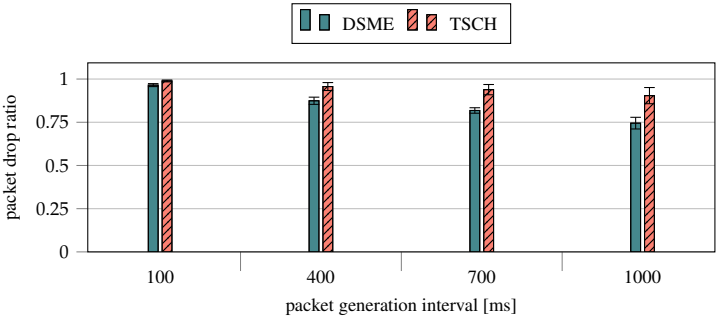


Figure 4.11: Ratio of dropped packets in the Contiki-NG MQTT backend implementation.

sink, and node 36 and 52 represent leaves with little interference. In general, the throughput of both protocols is low due to the reasons discussed in Sect. 4.3.5. For both protocols, the throughput is higher close to the sink because node 16 and 26 forward packets from other nodes. At this point, it should be said that DSME has the potential to increase throughput beyond the limits of TSCH's throughput through CAP-Reduction (CAP-R). However, tests have shown that utilizing CAP-R leads to a prolonged network setup time, and slots cannot be allocated fast enough. That is because only a single CAP can be used, which also has to serve RPL broadcasts. After 30 min of execution, almost no data messages have been transmitted through DSME

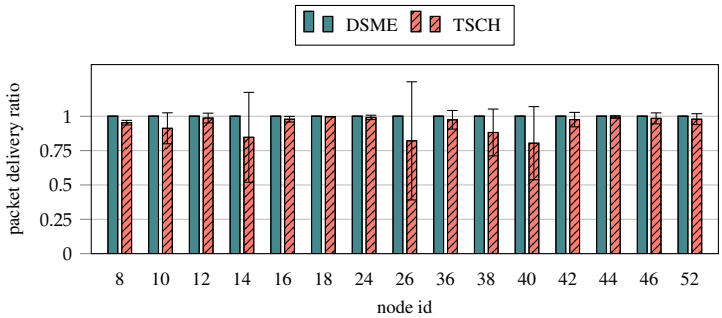
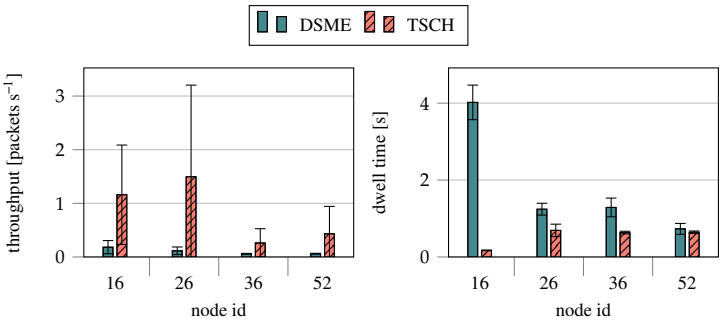


Figure 4.12: Packet Delivery Ratio of DSME and TSCH for all network participants.



(a) Throughput.

(b) Dwell time.

Figure 4.13: Single hop throughput and dwell time of DSME and TSCH on MAC level for increasing packet generation intervals.

with CAP-R. Therefore, a dynamic CAP-R algorithm that reduces CAPs gradually, as proposed in Chap. 8, would be favorable to improve performance.

The increased throughput also results in a lower dwell time using TSCH as indicated in Fig. 4.13b. In other words: packets are queued for less time using TSCH indicating the creation of a more efficient schedule by 6P-MSF in comparison to TPS. Additionally, 6P-MSF has the advantage of using a minimal 6TiSCH configuration in the second slotframe which allows nodes to directly transmit packets without allocating slots and hence reduces initial queuing time. In TPS, queued packets are not considered by the scheduler, so it can take a long time to empty the queue.

4.3.5.2 Energy Efficiency

At last, we consider the energy consumption of DSME and TSCH expressed by the active times of the transceiver for data reception and data transmission. Here, the time in the RX state also includes time for the reception idle time. The active times where collected in software through the `Energest` module of Contiki-NG because hardware-based measurements were overshadowed by the consumption of the processor. Furthermore, the active times of the processor and time spent in the sleep state were recorded.

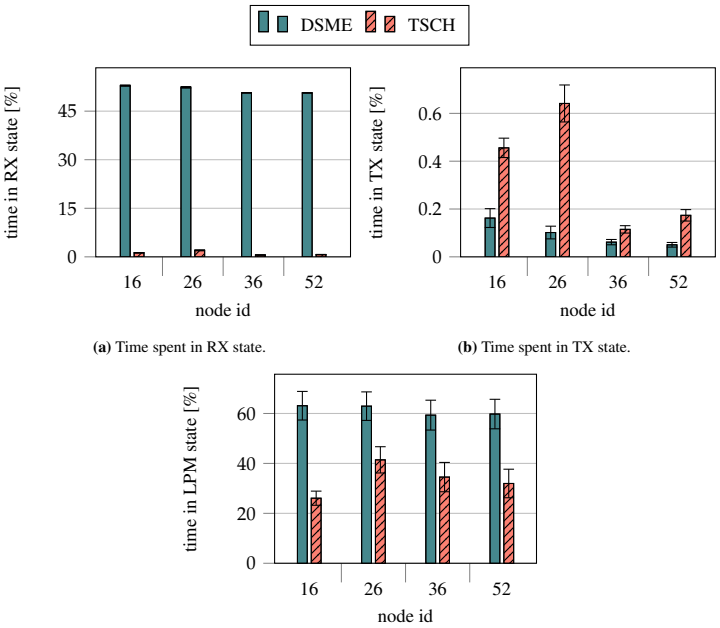
Fig. 4.14a shows the time share of both protocols in the active RX state. As expected, it is just over 50 % for DSME, since nodes must be active during the CAP which accounts for exactly 50 % of the frame structure. In addition, there is active time for receiving beacons and data messages. For TSCH with 6P-MSF on the other hand, only one slot per neighbor of 101 available slots in slotframe 1 is initially used for dedicated management messages and synchronization takes place based on data messages. Another influence factor is slotframe 0, where some slots are allocated for 6TiSCH bootstrapping. Therefore, the energy consumption in the RX state is significantly lower than in DSME.

For the fraction of time in TX state, as shown in Fig. 4.14b, TSCH exhibits a higher value than DSME. Again, it can be seen that node 16 and 26 require more energy for both protocols which can be attributed to their increased throughput. At node 52 and 36, however, DSME's throughput is lower than TSCH's but more time is spent in TX state. This likely stems from retransmissions and an extended slot allocation procedure.

Finally, Fig. 4.14c shows the fraction of the processor's time in low-power mode. It is visible that it is larger for DSME than for TSCH, which is likely due to its significantly lower throughput. Both protocols are expected to require processing time mainly for handling data and management messages, since the scheduling algorithms are kept quite simple.

4.4 Discussion and Conclusion

The experiments conducted in the course of this chapter compare DSME and TSCH in realistic scenarios on hardware and show that TSCH is particularly suitable when a fast network setup and high flexibility are required. However, in contrast to DSME, the shortened slot negotiation handshake leads to frequent inconsistencies during slot (de)allocation, which require protracted resolution. In addition, the frame syn-



(a) Time spent in RX state.

(b) Time spent in TX state.

(c) Processor's time spent in low-power mode (LPM).

Figure 4.14: Energy consumption of DSME and TSCH given by transceiver active time spent in the RX state, TX state, low-power mode.

chronization through data messages can lead to more frequent changes of the parent node and consequently to instabilities of the network. Combined, these reasons result in a reduced reliability compared to DSME, e.g., with 82 – 99 % versus 100 % successfully transmitted packets in a data-collection scenario with 15 nodes.

It should be noted that the association time is hardly relevant in the context of this work, since some set-up and settle time must always be factored in for the practical application in wireless sensor networks. Thus, it is more important that no data loss occurs during regular operation, especially when considering changing traffic conditions. The experiments show that DSME excels at fast slot (de)allocation and hence offers a significantly higher adaptability to time-varying traffic. Another factor is DSME's CAP, which allows sending secondary traffic without allocating dedicated

slots first.

All points considered, DSME seems better suited for the scenario described in Chap. 2 and will mainly be considered for the techniques proposed and experiments conducted in the following chapters. However, many of the techniques are generally applicable to TDMA-based protocols with alternating phases for primary and secondary traffic and can consequently also be transferred to TSCH.

Chapter 5

Secondary Traffic in DSME

In the context of this work, we refer to the term secondary traffic as management messages exchanged mainly but not only to support primary traffic. Thus, this category notably comprises negotiation messages for exclusive communication slots during primary traffic and broadcasts conveying routing information. Thereby, secondary traffic is a vital part of modern MAC protocols. Secondary traffic is mostly realized through contention-based channel access, i.e., with protocols like CSMA/CA and Aloha. With these, many collisions can occur at high data rates and in dense networks due to increasing contention so that the overall reliability of secondary traffic diminishes significantly. This performance degradation often implies reduced reliability of primary traffic because slot allocations are delayed, packets cannot be transmitted sufficiently fast, and queue drops occur. Consequently, a highly reliable protocol for secondary traffic is required.

In IEEE 802.15.4 DSME, slotted CSMA/CA is utilized for secondary traffic during the CAP where management messages for GTS (de)allocations, beacon allocations, and network association are exchanged. Therefore, this chapter first analyzes the influence of the slot allocation handshake on primary traffic and vice versa in Sect. 5.1, and shows how the setup time changes for various parameters and GTS allocation rates. One result worth highlighting is that DSME's adaptability to time-varying primary traffic can be further improved by mechanisms that relieve secondary traffic. Afterward, Sect. 5.2 exploits the findings of this analysis to design QMA, a source-efficient multiple-access scheme based on QL. Although QMA is designed in the context of DSME, it is generally applicable for transmitting secondary traffic in any other protocol, and can even be utilized for primary traffic. Results indicate that QMA

is capable of solving the hidden-node problem and provides excellent scalability to larger scenarios.

Analyzing DSME's Slot Allocation Handshake
Section 5.1

A distinguishing feature of DSME is its flexibility and adaptability to time-varying network traffic and to changes in the network topology by purely local actions with low overhead. In particular, DSME allows to dynamically establish dedicated links between any two neighboring nodes. These GTS (de)allocations are performed in a distributed manner as describe in Sect. 2.4.4.4, i.e., each pair of nodes can autonomously (de)allocate GTS according to their needs.

As discussed in Sect. 2.4.4.4, the (de)allocation of a GTS is based on a three-way handshake that takes place in the CAP via CSMA/CA, so it is subject to collisions (hidden node problem) and race-conditions. When collisions occur while allocating a new GTS, the internal data structures containing information about the slots assigned to the node and the channel/timeslot pairs allocated to neighboring nodes can become inconsistent. This can lead to the abandonment of the (de)allocation process and a rollback of the handshake to regain consistency. The procedure works similarly for duplicate allocation notifications. All this makes the allocation process a complex undertaking and estimating the required times is challenging.

Therefore, the following sections analyze the performance of the slot (de)allocation process in DSME and evaluated DSME's ability to adapt to fluctuating and dynamically changing traffic. In particular, Sects. 5.1.3.2 and 5.1.3.3 show that the setup time is tightly correlated with the chosen frame structure and accordingly with the values of their configuration parameters. In this context, Sect. 5.1.3.4 also makes recommendations for the configuration parameter of the CSMA/CA algorithm. At last, the limits of slot allocation rates as dictated by the primary traffic rate are analyzed in Sect. 5.1.3.5 for different topologies. The analysis is intended to elucidate the reciprocal effects of primary and secondary traffic in detail.

5.1.1 Related Work

Few works exist which analyze the slot allocation process in DSME. In [HSER14, XZY+18] the CSMA/CA-based medium access is analyzed, while the collision probability in networks with unsaturated traffic is evaluated in [PMS+09]. In the same

Sect. 5.1.2 is an extension of, and Sect. 5.1.3 was previously published in "Meyer, F., Mantilla-González, I., Kauer, F., Turau, V.: Performance Analysis of the Slot Allocation Handshake in IEEE 802.15.4 DSME. In: Ad-Hoc, Mobile, and Wireless Networks. pp. 102–117. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-31831-4_8." [MMGKT19]. In addition to the simulative evaluation, the author is responsible for extending the theoretical model with a Markov chain.

direction, Koubaa et al. show the influence of the parameter $minBE$ on slotted CSMA/CA for broadcast transmissions in small and large scale networks [KAT06]. To properly dimension IEEE 802.15.4 networks, adaptive protocols for estimating CSMA/CA parameters have been proposed. There are some centralized approaches [ATHD13], in which the coordinator adjusts BE for nodes depending on the traffic patterns, and some distributed approaches [FAC+11], in which nodes locally compute the CSMA/CA parameters to meet a predefined PDR.

In TSCH networks, nodes communicate using shared and dedicated links. [YTB18] formulates the collision probability to access a shared link for non-periodic traffic patterns to determine the number of nodes to allocate per link. In [DGAND+17], measurements of delivery probability, packet latency, and energy consumption are obtained from an analytical model of the TSCH's CSMA/CA algorithm, to determine the performance per node in transmissions over shared links. Different works have been focused on analysis and improvements in DSME networks [LLS+13, HN14]. [SPW17] shows that modifying the Clear Channel Assessment (CCA), the channel busy condition due to acknowledge can be avoided. Additionally, a restrictive contention access scheme is proposed to reduce the probability of collisions.

An analysis of the GTS allocation process [VBPA17] leads to an analytical method to select CSMA/CA parameters that minimize the number of contention access periods demanded to complete this procedure. They consider different multi-hop topologies, and simulate the networks using Cooja. Finally, [KKLT16a] presents a formal analysis of the distributed slot allocation procedure in DSME, identifying weaknesses related to packet loss, which eventually lead to inconsistent slot allocations. Despite the fact that DSME's allocation allows reliable data transmissions, vulnerabilities of CSMA/CA affect the slot management.

5.1.2 Theoretical Evaluation

To illustrate the importance of high reliability for the GTS (de)allocation process, one can calculate the expected number of transmissions and expected setup time to (de)allocate a single GTS. The setup time corresponds to the elapsed time between the transmission of the first *GTS request* and the reception of the *GTS notify*. Some approaches disregard the *GTS notify* assuming that it does not affect the (de)allocation process once the *GTS response* is delivered [VBPA17]. However, the *GTS notify* is required for updating neighbor's Slot Allocation Bitmap (SAB) to avoid collisions, and is hence considered in the following calculations.

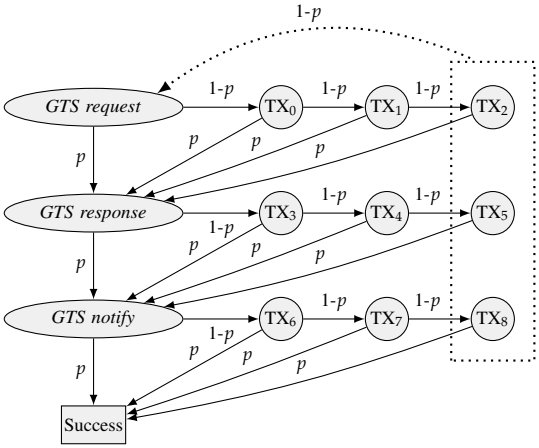


Figure 5.1: Absorbing Markov chain for GTS negotiation through the 3-way handshake where p is the probability of a successful packet transmission, i.e., the PDR.

5.1.2.1 Expected Transmission Attempts

Especially in scenarios with highly fluctuating traffic, it is crucial to keep the number of transmission attempts per 3-way handshake message (*GTS request*, *GTS response*, *GTS notify*) minimal. This way, more GTS allocations can be conducted within a single CAP.

The (de)allocation handshake can be modeled as an absorbing Markov chain as shown in Fig. 5.1. The states TX_i symbolize CSMA/CA retransmission after an unsuccessful transmission attempt. A total of $macMaxFrameRetries = 3$ retransmissions is conducted (per default) before a packet is dropped. `Success` is the absorbing state, and the transition probabilities are given on the transitions, where p is the probability of a successful transmission, i.e., the PDR. Note that the ACK for the *GTS request* is not considered because the handshake can continue if the *GTS response* was successfully received [IEE20].

The Markov chain with $t = 12$ transient states and $r = 1$ absorbing state can be represented in the canonical form of its transition matrix P as

$$P = \begin{pmatrix} Q & R \\ \mathbf{0} & I_r \end{pmatrix}, \tag{5.1}$$

	Probability of successful transmission p				
metric	1.0	0.9	0.8	0.7	0.6
Exp. TX attempts	3	3.33	3.76	4.32	5.13
Exp. setup time [ms]	8.16	9.07	10.22	11.75	13.96
metric	0.5	0.4	0.3	0.2	0.1
Exp. TX attempts	6.41	8.67	13.49	27.81	123.63
Exp. setup time [ms]	17.43	23.59	36.70	75.65	336.28

Table 5.1: Expected number of transmission attempts and expected setup time for GTS negotiation through the 3-way handshake.

where Q is a $t \times t$ matrix, describing the probability to transition from a transient state to another transient state, R is a $t \times r$ matrix, describing the probability to transition from a transient state to an absorbing state, $\mathbf{0}$ is an $r \times t$ zero matrix and I_r is an $r \times r$ identity matrix. It allows calculation of the fundamental matrix M as

$$M = (I_t - Q)^{-1}, \tag{5.2}$$

where I_t is a $t \times t$ identity matrix. The expected number of steps S until absorption, i.e., the number of transmission attempts until the handshake is successful, can be expressed as

$$S = M\mathbf{1}, \tag{5.3}$$

where $\mathbf{1}$ is a vector of ones of size t.

Tbl. 5.1 shows the expected number of transmission attempts for different probabilities, p, of successful transmission. The number of required transmission attempts increases exponentially for decreasing p. This illustrates the necessity for a high PDR during the CAP which can, e.g., be attained by reducing the CAP's load and hence the number of collisions during contention-based access.

5.1.2.2 Expected Setup Time

Fig. 5.2 illustrates the composition of individual timings during the GTS handshake. Consequently, the expected setup time t_{setup} can be expressed as

$$t_{setup} = t_{RQ} + t_{ACK} + t_{RSP} + t_{NTF}, \tag{5.4}$$

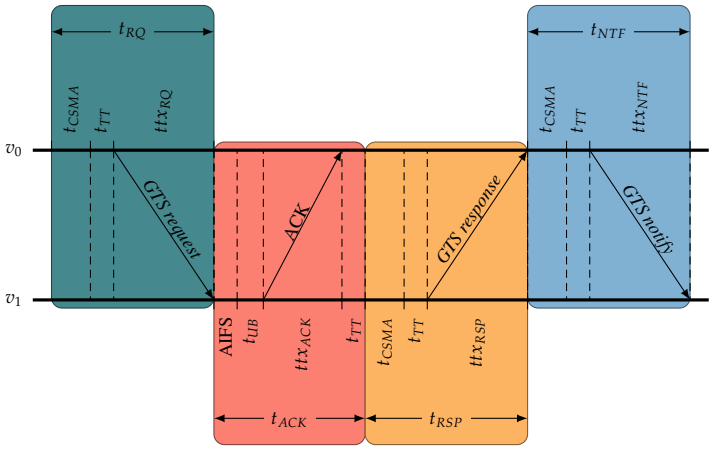


Figure 5.2: Message sequence and timing of the 3-way handshake.

where t_{RQ}, t_{RSP}, and t_{NTF} are the expected times for sending a *GTS request*, *GTS response*, and *GTS notify*, respectively. Furthermore, t_{ACK} is the expected time for transmitting an ACK. In the following, we assume that the channel is always idle during a CCA and, thus, CSMA/CA only performs one backoff.

According to Fig. 5.2, the expected times for sending a *GTS request*, ACK, *GTS response*, and *GTS notify* are given by

$$t_{RQ} = t_{CSMA} + t_{TT} + ttx_{RQ} = 2.72\,\text{ms}, \tag{5.5}$$

$$t_{ACK} = AIFS + \frac{t_{UB}}{2} + ttx_{ACK} + t_{TT} = 0.896\,\text{ms}, \tag{5.6}$$

$$t_{RSP} = t_{CSMA} + t_{TT} + ttx_{RSP} = 2.72\,\text{ms}, \tag{5.7}$$

$$t_{NTF} = t_{CSMA} + t_{TT} + ttx_{NTF} = 2.72\,\text{ms}, \tag{5.8}$$

with $t_{TT} = aTurnaroundTime = 12\,\text{symbol(s)}$ being the turnaround time of the transceiver and $t_{UB} = aUnitBackoffPeriod = 20\,\text{symbol(s)}$ as given in Tbl. 5.2. An ACK is sent within the time window $[AIFS, AIFS + t_{UB}]$ so that the expected time of transmission lies $AIFS + \frac{t_{UB}}{2}$ after receiving the *GTS request*. Additionally, $ttx_{ACK} = 0.352\,\text{ms}$ and $ttx_{RQ} = ttx_{RSP} = ttx_{NTF} = 1.28\,\text{ms}$ are the physical transmission times for the respective handshake messages and the ACK. The expected

	Duration of handshake components						
	$AIFS$	t_{TT}	t_{UB}	ttx_{RQ}	ttx_{ACK}	ttx_{RSP}	ttx_{NTF}
duration [symbol(s)]	12	12	20	80	22	80	80
duration [ms]	0.192	0.192	0.32	1.28	0.352	1.28	1.28

Table 5.2: Duration of 3-way handshake components in DSME.

time to access the shared wireless medium t_{CSMA} using CSMA/CA is calculated as

$$t_{CSMA} = t_B + t_{CCA} = 1.248\,\text{ms}. \tag{5.9}$$

Here, $t_{CCA} = 0.128\,\text{ms}$ the constant time for conducting a CCA and

$$t_B = \frac{(2^{BE} - 1) \cdot aUnitBackoffPeriod}{2} = 1.12\,\text{ms} \tag{5.10}$$

is the expected time for the channel access in unslotted CSMA/CA, where a random backoff exponent from the interval $[0, 2^{BE} - 1]$ is used for every transmission. By default $BE = 3$.

In the fundamental matrix M from Sect. 5.1.2.1, every entry $(0, j)$ indicates the expected number of times one of the states depicted in Fig. 5.1 is visited. This allows to calculate the total expected setup time for different PDRs. Tbl. 5.1 shows the resulting expected setup times, which show a similar behavior to the expected number of transmission attempts. It should be noted that t_{setup} does not depend on MO and SO in an ideal scenario. The dominant factor is the minimum backoff exponent $minBE$.

5.1.3 Simulative Evaluation

Sect. 5.1.3.1 describes the setup of the simulative evaluation. In order to evaluate the influence of frame structure parameters such as SO and MO on the allocation process, we define the *setup time* as the total time required by nodes to successfully allocate a GTS. Hence, the setup time is measured from the time a node issues a *GTS request* until the last *GTS notify* has successfully been received. In particular, Sect. 5.1.3.2 evaluates a contention-free scenario and Sects. 5.1.3.3 to 5.1.3.5 consider contention-based scenarios with a line and grid topology. The contention-free scenario allows to obtain an upper bound for the number of allocations per GTS. Through varying burst sizes, networks are evaluated under non-saturated conditions. Sect. 5.1.3.3 also scrutinizes the reasons for failure during setup.

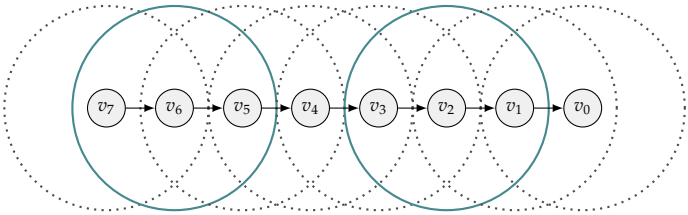


Figure 5.3: Line topology with eight nodes and the respective communication ranges. Only two nodes can communicate at the same time.

5.1.3.1 Scenario Description

For evaluation, we consider three scenarios, the first of which is a network with two nodes - sink and source - where the source periodically transmits packets to the sink. In this contention-free scenario, further discussed in Sect. 5.1.3.2, packets are generated at high rates to evaluate the behavior of a saturated network.

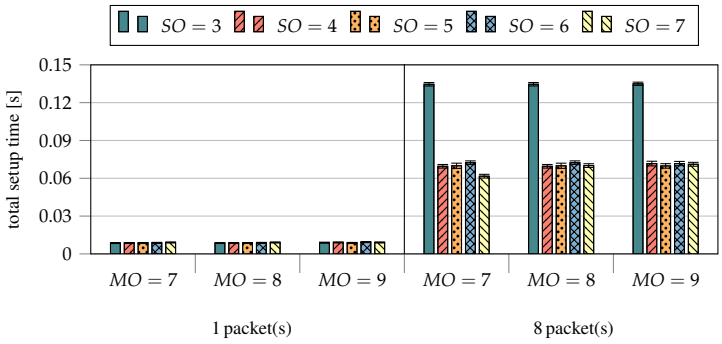
Sects. 5.1.3.3 to 5.1.3.5 consider a scenario with increasing contention. For this, the line topology depicted in Fig. 5.3 is utilized, where every node $v_{i\neq 0}$ sends its self-generated packets towards its parent v_{i-1}. Packets received from its child v_{i+1} are not forwarded. The communication range of v_i is chosen in a way that it can only reach its parent and its child.

At last, Sect. 5.1.3.5 utilizes a grid topology with $m \times m$ nodes, where $2 \leq m \leq 7$. The nodes' communication range is configured so they can communicate with their nine neighbors. Thus, contention is higher than in the line topology. Every node communicates with an exclusive, randomly chosen communication partner that changes every run.

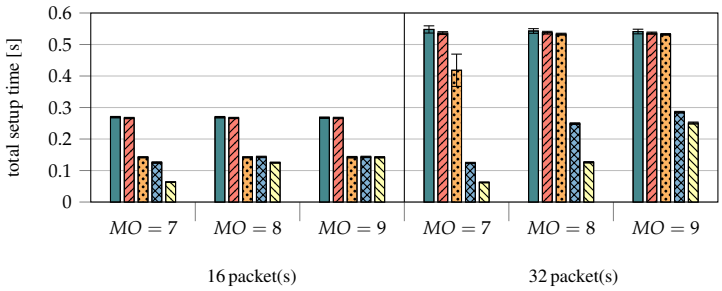
For all experiments, a burst of packets ϵ is generated at fixed intervals, equivalent to the duration of a MSF of order $MO = 9$, i.e., every 7.864 s. Evaluation starts after the association phase is complete. Tbl. A.2 summarizes the relevant configuration parameter for all scenarios. According to [IEE20], $minBE = 3$ and $maxBE = 5$ are used for the CSMA/CA algorithm during the CAP.

5.1.3.2 Performance in Contention-free Scenarios

In this section, the setup time of a network without contention is analyzed, i.e., under optimal conditions as described in Sect. 5.1.3.1. Fig. 5.4 shows the setup time of two nodes for various values of SO and MO and an increasing number of generated packets. As expected and shown in Fig. 5.4a, the setup time for 1 packet(s) does



(a) Setup time for $\epsilon = 1$ packet(s) and $\epsilon = 8$ packet(s).



(b) Setup time for $\epsilon = 16$ packet(s) and $\epsilon = 32$ packet(s).

Figure 5.4: Setup time for two nodes with varying number of packets per burst ϵ and different values of *SO* and *MO*.

not differ significantly for different *SO*s and *MO*s, since the packet is transmitted at the start of the CAP. For 8 packet(s), however, the setup time for $SO = 3$ increases drastically in relation to the other *SO*s. The reason for this is that in a CAP with $SO = 3$, a maximum of seven handshakes can be realized. The eighth handshake does not fit into the same CAP and the node has to wait a full CFP until the next handshake can be performed. For larger *SO*s, the CAP is longer and all GTS can be allocated in one CAP.

For 8 packet(s) and $SO = MO = 7$ (see Fig. 5.4a), the setup time is slightly lower than for the other configurations because the scenario only allows for a single SF. In other words: the network is saturated when seven GTS are allocated and

DSME does not try to allocate more slots afterwards because all allocated slots are marked in the Allocation Counter Table (ACT), resulting in a lower setup time but an increased transmission time for data packets. This also holds true for the scenarios with 16 packet(s) and 32 packet(s), where $7 * 2^{MO-SO} < 16$ GTS and $7 * 2^{MO-SO} < 32$ GTS are available, respectively.

Looking at Fig. 5.4b, it may seem counter-intuitive at first that the setup time for $SO = 3, 4, 5$, $MO = 9$, and 32 packet(s) is the same because less packets can be sent with a smaller SO as shown in the 8 packet(s) scenario. However, by reducing SO by one, the length of a CAP halves but the number of CAPs per MSF doubles. Therefore, the accumulated duration of all CAPs of a MSF stays the same, resulting in the same setup time for a scenario without contention. The influence of $minBE$ and $maxBE$ is not explicitly shown here but in Sect. 5.1.3.4, since a larger $minBE$ simply results in a larger setup time.

At last, the theoretical estimation from Sect. 5.1.2.2 can be compared with the results depicted in Fig. 5.4. Without contention and external interference, the theoretical model yields an expected negotiation time of 8.16 ms but does not include the acknowledgment duration of $t_{ACK} = 0.896$ ms. Thus, the total setup time lies well within the 95 % confidence interval of, e.g., $SO = 3$ and $MO = 9$ with $8.93\,\text{ms} \pm 0.38\,\text{ms}$.

5.1.3.3 Performance in Contention-based Scenarios

Fig. 5.5 shows the setup times for different line topologies. It can be seen that the setup time increases for all SOs for up to six nodes. Afterward, it stagnates. This is due to the fact that in a network with up to five nodes, only one node can transmit at same time. With six to eight nodes, two nodes can transmit at the same time through spatial reuse, speeding up the setup process when the number of nodes increases. The setup time increases for an increasing SO because slot allocations are triggered at the start of every CAP and when the last handshake finishes successfully (see Sect. 5.1.3.1). Because of this, nodes have to wait longer until the next allocation is triggered for a higher SO.

There can be several reasons for a failing 3-way handshake. For the line topology, these are presented in Fig. 5.6. It can be seen that increasing SO does not increase the number of initiated handshakes. In fact, the number of failed handshakes is slightly lower. The main reason for a failing handshake is that the *GTS response* or *GTS notify* message does not arrive in time. The latter case cannot be detected by DSME, as given by the standard [IEE20]. By slightly adapting the handshake, openDSME

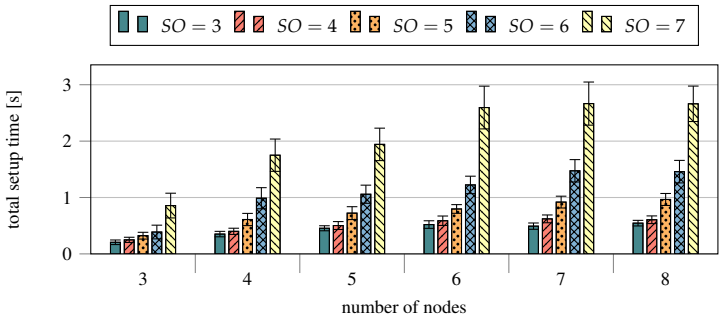


Figure 5.5: Setup time for a line topology with an increasing number of nodes, $\epsilon = 8$ packet(s), $SO = 3$, and $MO = 9$.

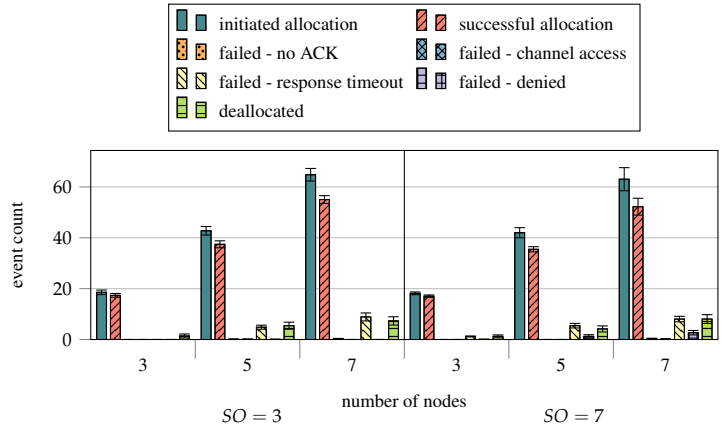


Figure 5.6: Number of initiated, successful, and failed GTS allocations in a line topology with $SO = 3$ and $SO = 7$, $MO = 9$, and $\epsilon = 8$ packet(s).

guarantees the detection of this case and the timely deallocation of the involved slots [Kau19]. This is denoted by *deallocated* in Fig. 5.6.

5.1.3.4 Influence of CSMA/CA Parameters

Apart from the slot length (SO), the minimum backoff exponent ($minBE$) and maximum backoff exponent ($maxBE$) are the most relevant parameter influencing the

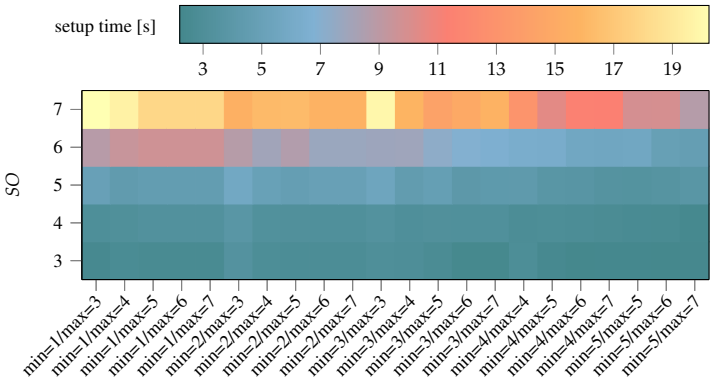


Figure 5.7: Setup time for combinations of SO, $minBE$, $maxBE$, and $MO = 9$.

network setup time. They determine the backoff durations of the CSMA/CA algorithm. Fig. 5.7 shows the impact of different combinations of $minBE$ and $maxBE$ on the setup time of a line topology with six nodes, 5 packet(s), $MO = 9$, and different SOs. The default parameters for DSME, as specified in [IEE20], are $minBE = 3$ and $maxBE = 5$. The color gradient gives the setup time in seconds from 20 s to about 1 s.

It can be seen that the network setup time strongly depends on the combination of $minBE$ and $maxBE$. For the given scenario, higher values for both parameters lower the setup time, e.g., the setup time for $SO = 6$ decreases from 8.73 s to 4.65 s for the smallest ($minBE = 1, maxBE = 3$) and the largest ($minBE = 5, maxBE = 7$) combination of exponents, respectively. The determining factor here is $minBE$, since the CSMA/CA algorithm calculates a random backoff exponent smaller than $\min\{minBE + NB, maxBE\}$, where NB is the number of executed backoffs for a specific transmission. Therefore, large values for $minBE$ and $maxBE$ increase the chance that two nodes choose a different backoff exponent. Thus, less collisions occur and packets are transmitted faster. As explained in Sect. 5.1.3.3, a smaller SO results in a shorter setup time.

5.1.3.5 Influence of the GTS Allocation Frequency

At last, the influence of the GTS-handshake triggering interval is evaluated in a grid topology, as described in Sect. 5.1.3.1. Fig. 5.8 shows the setup time for triggering

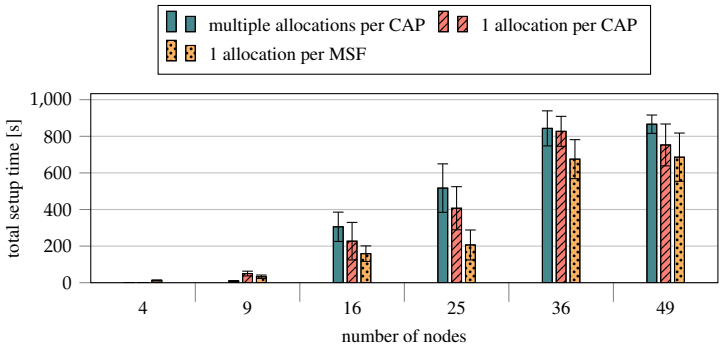


Figure 5.8: Setup time for different GTS allocation triggering intervals with $SO = 3$, $MO = 9$, $\epsilon = 1\,\text{packet(s)}$, and an increasing number of nodes.

allocations multiple times per CAP, and once per CAP, once per MSF. The SF within the MSF is chosen randomly for the last method. It can be seen that triggering handshakes multiple times per CAP performs better than the other two methods for small networks with four and nine nodes because contention is low, and packets do not collide frequently. Hence, it is possible for a node to transmit multiple packets in a single CAP, speeding up the allocation process. From 16 nodes on, however, the other two methods perform better. Triggering allocations once per MSF performs strictly better than triggering once per CAP because less traffic and therefore less congestion is created. This is also illustrated by Fig. 5.9. Here, it can be seen that triggering allocations once per MSF initiates significantly less allocations than the other methods, increasing the energy-efficiency of the network. The main reasons for a failed allocation are unacknowledged requests, most probable due to hidden nodes problems.

5.1.4 Discussion and Conclusion

The results obtained from Sects. 5.1.3.2 to 5.1.3.5 demonstrate that the CAP is the critical component for DSME's adaptability to bursty and dynamically changing traffic. Even if only few GTS have to be (de)allocated, many collisions might occur due to the utilized CSMA/CA algorithm. Especially, Sect. 5.1.3.5 evidences that the setup time highly depends on the handshake triggering interval. Triggering multiple allocations per CAP in small networks performs better in terms of the setup time. However, as the size of the network increases this performance is degraded given

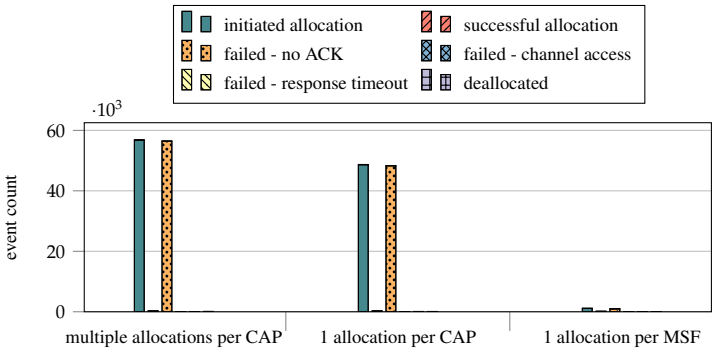


Figure 5.9: Initialized, successful and failed GTS allocations for different triggering intervals with $SO = 3$, $MO = 9$, $\epsilon = 1\,\text{packet(s)}$, and 49 nodes.

the increasing congestion. Therefore, disseminating slot allocations (i.e., a single allocation per CAP or a single allocation per MSF) is desirable for medium and large scale networks.

Hence, to make DSME more adaptable to applications with bursty and dynamically changing traffic, we conceive three solutions:

- Delay GTS handshakes on a higher layer to reduce collisions. This could be done in a time-based scheme, where nodes allocate GTS one after another.

- CSMA/CA can be replaced with another, yet compatible, contention-based multiple access scheme offering higher reliability and implying less collisions. This idea is investigated in Sect. 5.2, where QMA is proposed, a QL-based multiple access scheme for resource-restricted devices.

- The overhead of management messages per GTS allocation should be reduced - either by allocating multiple GTS per handshake, or by sending multiple packets per GTS to reduce the number of required GTS. Sect. 6.1 explores the latter option.

Apart from that, this chapter explores the performance impact of the pertinent parameters. For small networks with less than six nodes, the setup time increases almost exponentially as SO increases. Although the slot allocation process speeds up as the number of nodes increases, due to spatial reuse, the setup time keeps increasing

with SO. Additionally, it is verified that a larger $minBE$ and $maxBE$ leads to a smaller probability of collisions due to nodes selecting the same backoff period.

QMA: Q-learning Based Multiple Access
Section 5.2

The theoretical and simulative evaluations from Sect. 5.1 suggest that a more reliable contention-based channel access is essential for proper operation of DSME. This requirement is also inherent in other protocols that implement secondary traffic through contention-based channel access.

Managing contention-based channel access is difficult because no node posses information about other nodes in the network. However, often hidden patterns introduced by primary traffic govern the secondary traffic. One example is DSME's 3-way handshake, as illustrated in Sect. 5.1, where each slot (de)allocation during fluctuating primary traffic leads to four management messages with distinct timing in secondary traffic. Another example are routing protocols transmitting periodic broadcasts for route discovery such as GPSR or RPL. Of course, such patterns also emerge in primary traffic, but the scenarios are more restricted. For example, it is impossible to identify hidden patterns if nodes are subject to frequently and spontaneously changing traffic conditions, i.e., when the transmission process is random. On the other hand, the induced secondary traffic might still exhibit hidden patterns as discussed above. Present contention-based channel access schemes like CSMA/CA or ALOHA do not utilize information about these patterns as it is hard to extract.

Machine Learning (ML) offers a promising solution to find such hidden patterns efficiently and to deduce the state of the network, e.g., traffic patterns of other nodes, from local observations [LR00]. The following sections aim to develop a contention-based multiple access scheme that continuously adapts to changes in the network and external interference. This goal excludes offline ML techniques where a model is trained on large-scale datasets and then deployed in nodes as discussed in Sect. 3.6. Hence, we rely on Reinforcement Learning (RL) as a method that enables the design of adaptive controllers that learn online, in real-time, and improve solutions to control problems.

The following sections introduce Q-learning-based Multiple Access (QMA), a QL-based multiple access scheme designed to increase throughput and reliability of the contention-based medium access. QMA learns hidden patterns in the contention-based traffic, allowing it to assess the probability of successfully sending a packet at

---

Minor revision of the previously published "Meyer, F., Turau, V.: QMA: A Resource-efficient, Q-learning-based Multiple Access Scheme for the IIoT. In: IEEE 41st International Conference on Distributed Computing Systems (ICDCS). IEEE (2021). https://doi.org/10.1109/icdcs51616.2021.00087." [MT21].

a given time. For this, Sect. 5.2.2 extends the cooperative QL algorithm presented in Sect. 3.6.3.4 to stochastic environments, which then serves as a basis for QMA as described in Sect. 5.2.3. To enhance QMA's adaptability in frequently changing environments, Sects. 5.2.4 and 5.2.5 introduce a parameter-based exploration algorithm and a delayed startup mechanism, respectively. At last, Sect. 5.2.7 demonstrates QMA's capability to solve the hidden node problem without overhead for additional RTS/CTS messages and Sect. 5.2.8 verifies QMA on hardware in a realistic environment.

5.2.1 Related Work

Communication in wireless mesh networks provides an optimal playground for RL due to its compulsion to adapt to constant environmental changes. Over the last few years, much work has been done optimizing selected aspects of the MAC layer [WBCJ17, MHKH19, KMH+19] up to replacing the whole MAC layer by learning agents [LE06, LJS20, BYSMK18].

Liu and Elhanany propose Reinforcement Learning based MAC (RL-MAC) [LE06], a RL-based MAC protocol adapting a node's duty cycle, i.e., active and sleeping times, to reduce energy consumption. The core idea is to allocate time slots for the active time, allowing a node to exchange messages with its neighbors. Afterwards, nodes transition into sleep mode. Through QL, nodes learn to maximize the ratio of the effective transmission and reception time to total active time and throughput. Even though their approach allows for asymmetric transmission rates, communication is still based on CSMA/CA with RTS/CTS messages. Thus, it is prone to collisions during high-load traffic scenarios.

Another approach is given in [LJS20] in the form of Collaborative Reinforcement Learning-based MAC (CoRL), a collaborative RL-based MAC-protocol. CoRL is based on ALOHA-Q [CKM+15]. It divides time into frames which are further subdivided into time slots. Learning is done through stateless QL, i.e., the state is invariant over time. Thus, Q-values are directly calculated for every action, which correspond to choosing a transmission slot in the current frame. Additionally, they incorporate a collaborative Q-value. This design, however, severely limits CoRL's flexibility because a node must select exactly one slot per frame. Compared to QMA, asymmetric traffic and adaptability to hidden traffic patterns is not possible. On the other hand, CoRL requires less memory due to stateless Q-Learning.

Chu et al. propose an extension of ALOHA and Q-Learning based MAC (ALOHA-Q) [CMG12], ALOHA and Q-Learning based MAC with Informed Receiving

Global Q-table		agent 0	
		a'	*a*"
agent 1	*a*'	-1	1
	a"	1	0

agent 0 (local)	
a'	*a*"
1	1

agent 1 (local)	
a'	*a*"
1	1

Table 5.3: Example for the global rewards during the resource acquisition of two agents and the according local Q-tables.

(ALOHA-QIR), which applies QL to slotted ALOHA but employs Information Receiving (IR) to increase energy efficiency. IR appends a number m to every transmitted packet which indicates the number of future frames for which the transmitter is going to use a selected slot. This allows the receiver to efficiently manage its transceiver and learn good reception slots over m Q-value updates. Additionally, they employ *ping*-packets to allow low-traffic nodes to signal that a slot is still active when no data packet is available for transmission. Such mechanisms are currently not considered in QMA. ALOHA-QIR allows for asymmetric traffic rates through ping-packets but compared to QMA, additional messages are required. Furthermore, every node must select at least one transmission slot.

Apart from the mentioned RL-algorithms, many works deal with general resource allocation and coordination using RL [AAT+19, WBCJ17, MHKH19]. These algorithms can potentially be employed for channel access but often lack domain-specific mechanisms to cope with the rapidly changing nature of the IIoT. Additionally, it should be noted that most of the presented works do not utilize dedicated algorithms for multi-agent MDP as proposed in [LR00] or [AM08]. In [AM08] a learning automaton is employed to solve the problem, while [LR00] alters the original QL algorithm to deduce local rewards from a global reward table.

5.2.2 Distributed Q-Learning for Stochastic Environments

The following sections build on the cooperative Q-learning algorithm by Lauer and Riedmiller explained in Sect. 3.6.3.4. In this algorithm, a problem arises in stochastic environments, i.e., when actions do not have a deterministic outcome [LR00]. Consider two agents competing for access to a shared resource, where action *a*' acquires the resource for the current state and action *a*" backs off until the next state. The resource cannot be used by both agents at the same time. Tbl. 5.3 summarizes the global rewards.

As one can see, a punishment is given if both agents try to acquire the shared resource in the same state and a reward if only one of the agents acquires the resource.

However, there is a small probability that an agent chooses *a*' but does not acquire the resource because it is not needed by the agent in the current state. If the other agent also executes *a*' and acquires the resource, it experiences the full reward, thinking that the first agent chose *a*". This might happen vice-versa and will eventually result in a collision during the acquisition. As both agents only store the maximum experienced reward per action, according to Eq. (3.3), they are stuck with action *a*' which will result in many failed acquisitions over time.

We offer a solution by introducing a small penalty ξ for Q-value updates, so that

$$Q(S_t, a_t) \leftarrow \max\{Q(S_t, a_t) - \xi, R_t + \gamma \max_a Q(S_{t+1}, a)\} \tag{5.11}$$

This way, $Q(S_t, a_t)$ is gradually reduced if a maximum reward was only experience once. On the other hand, $Q(S_t, a_t)$ is reset to the maximum Q-value every other update if the maximum reward is continuously experienced. In other words: ξ only affects fluctuating Q-values because stable and optimal Q-values are reupdated to their original value every other update. At last, the learning rate α_{RL} can be incorporated as

$$\begin{aligned} Q(S_t, a_t) \leftarrow \max\{&Q(S_t, a_t) - \xi, (1 - \alpha_{RL}) Q(S_t, a_t) \\ &+ \alpha_{RL}(R_t + \gamma \max_a Q(S_{t+1}, a))\} \end{aligned} \tag{5.12}$$

5.2.3 Description of QMA

QMA's core idea is to utilize QL to determine at which times transmission is beneficial and at which times it is likely to result in a collision. For this, QMA employs a mechanism similar to slotted CSMA/CA, where time is divided into M subslots $m_0, \ldots, m_{M-1}$. This discretization of time helps to keep the state space small. For application in DSME, e.g., eight CAP slots are further subdivided into 54 subslots. During this time, the transceiver is turned on to guarantee compatibility with CSMA/CA.

5.2.3.1 Q-value Representation

For QMA, we employ a simple table with $|S_t| \cdot |\mathbb{A}_t|$ values, as described in Sect. 3.6.3.3. Thus, at maximum two multiplications, three additions and $|\mathbb{A}_t| + 1$ array lookups are required for each training step, using Eq. (5.12). It is possible, e.g., by selecting $\alpha_{RL} = 0.5$ and integer rewards, to replace multiplication with the learning rate by an efficient right shift by one. This allows for the execution on resource-restricted and embedded devices without a floating-point unit.

5.2.3.2 Action Space

The action space of QMA is given by the set $\mathbb{A}_t = \{QBackoff, QCCA, QSend\}$, where the actions are defined as follows:

1. *QBackoff* lets a node back off to the next subslot.
2. *QCCA* performs a CCA and transmits a packet on success or backs off to the next subslot on failure.
3. *QSend* transmits a packet immediately without assessing the channel.

Separating *QCCA* and *QSend* allows for priority-based transmissions, i.e., nodes can select *QSend* if they have to transmit a packet urgently. Other nodes can check, by selecting *QCCA*, if there is already a high-priority transmission going on and back off to a later subslot.

5.2.3.3 State Space

QMA's state space only consists of the current subslot m_t, allowing nodes to synchronize their transmission slots solely based on time information. At first, it might seem beneficial to include other features like the queue level, number of overheard messages, or number of successive idle slots. However, several experiments have shown that these additional inputs withhold the agent from building a collision-free schedule. For example, nodes learn to transmit packets when their queue is full but do not care if the according subslot is already used by another node. Instead, queue levels are incorporated into the exploration function, as discussed in Sect. 5.2.4.

5.2.3.4 Internal States

Fig. 5.10 shows QMA's internal structure based on five states. In `QDecision`, a node checks if a packet needs to be transmitted and selects an action $a_t \in \mathbb{A}_t$ depending on m_t. Then a_t is executed, and QMA transitions to state `QEvaluation`. If *QBackoff* is executed, `QEvaluation` is guaranteed to be reached by the start of the next subslot. A transmission triggered by *QCCA* or *QSend*, however, can span multiple subslots depending on the packet length. Consequently, rewards are not directly observable after an action is performed but, e.g., after an ACK is received. Therefore, we save state S_t and action a_t until the outcome of a_t is known. Afterward, a_t is evaluated according to Eqs. (5.13) and (5.15), $Q(S_t, a_t)$ is updated, and the next packet can be transmitted [DAMH21].

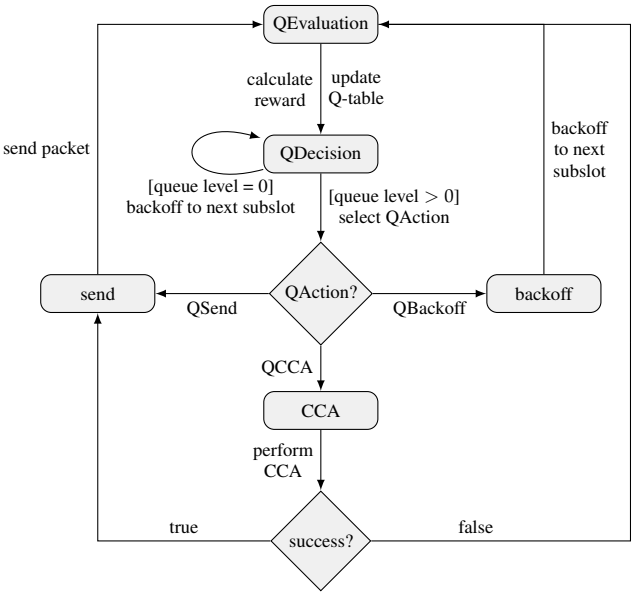


Figure 5.10: Flow chart for the QMA state machine.

5.2.3.5 Algorithmic Description

Alg. 1 outlines the complete QMA algorithm, based on the action space described in Sect. 5.2.3.2, the state space described in Sect. 5.2.3.3, and internal structure described in Sect. 5.2.3.4. Given a non-empty queue at the start of a subslot (line 4), QMA selects $a_t \in \mathbb{A}_t$ randomly with probability ρ (line 6), or according to policy $\pi(m_t)$ with probability $1-\rho$ (line 8). Here, ρ is calculated by parameter-based exploration as discussed in Sect. 5.2.4. Afterward, a_t is executed (line 10). As discussed in Sect. 5.2.3.4, the execution of a_t may span multiple subslots due to processing time and, in case of actions *QSend* and *QCCA*, waiting time for an ACK. During the waiting time, a node is inactive and does not select further actions. At last, the Q-table is updated (see line 15) and policy $\pi(m_t)$ is adjusted accordingly if a better action is found (line 13). Notice, that policy $\pi(m_t)$ is initialized with action *QBackoff* for every subslot m_t and only changed if a strictly greater Q-value is found, as described in Eq. (5.12).

At last, it should be said that a packet is dropped after *macMaxFrameRetries*

Algorithm 1: Learning multiple access with QMA.

Input: learning rate α_{RL}, discount factor γ, penalty ξ, exploration rate ρ

initialize $\forall_{m_t, a_t \in \mathbb{A}_t}$: $Q(m_t, a_t) \leftarrow -\infty$
initialize $\forall_{m_t}$: $\pi(m_t) \leftarrow$ *QBackoff*
on *subslot* $m_t \in \{m_0, \ldots, m_{M-1}\}$ **do**
  if *queue level* > *0* **then**
    if *random()* $\leq \rho_t$ **then**
      select $a_t \leftarrow$ random action `// exploration phase`
    else
      $a_t \leftarrow \pi(m_t)$
    end
    execute a_t, **wait** until a_t is finished, $i \leftarrow$ passed subslots
    $R_t \leftarrow$ received reward
    if $Q(m_t, a_t) < (1-\alpha_{RL})Q(m_t, a_t) + \alpha_{RL}(R_t + \gamma \max_a Q(m_{t+i}, a))$ **then** `// see Eq. (3.4)`
      $\pi(m_t) \leftarrow \operatorname{argmax}_a Q(m_t, a)$
    end
    $Q(m_t, a_t) \leftarrow \max\{Q(m_t, a_t) - \xi, (1-\alpha_{RL})Q(m_t, a_t) + \alpha_{RL}(R_t + \gamma \max_a Q(m_{t+i}, a))\}$ `// see Eq. (5.12)`
  end
end

retransmission as in CSMA/CA [IEE20]. Yet, a packet is not discarded after $macMaxCsmaBackoffs$ backoffs since QMA's main idea is to synchronize transmission times which might require several backoffs until an appropriate subslot for transmission is reached.

5.2.3.6 Reward Function Design

The goal of multi-agent QL is to find a globally optimized policy using local information. Thus, it is necessary to define rewards for the interaction of nodes, i.e., rewards for the set of actions $\{a_t^{(0)}, \ldots, a_t^{(|\mathbb{V}|-1)}\}$ of nodes $\{v_0, \ldots, v_{|\mathbb{V}|-1}\}$ at time t. The interactions are illustrated by the hidden Markov decision process in Fig. 5.11. Since actions *QBackoff* and *QCCA*, after a failed CCA, force an agent to backoff until the next subslot, there is no interaction with other nodes in the network. After subslot m_{M-1}, the agent returns to the first subslot m_0. On the other hand, actions *QSend* and *QCCA*, after a successful CCA, interact with the network by transmitting a packet and possibly interfering with other transmissions. Clearly, in an optimal policy, a

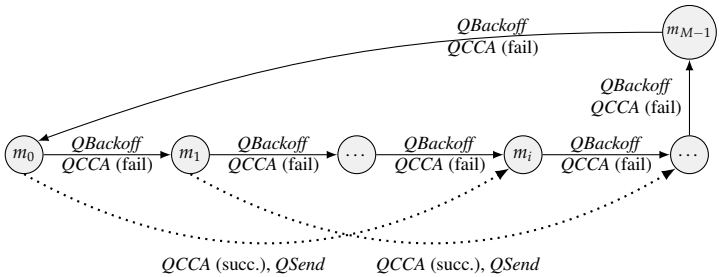


Figure 5.11: Exemplary multi-agent hidden Markov decision process. Hidden interactions are the result of concurrent actions of nodes with an overlapping transmission range, they are symbolized by dotted lines.

single node performs *QSend* while all other nodes choose action *QBackoff* so that no collision occurs.

Without loss of generality and for sake of simplicity, we consider the interaction of three Q-agents with actions *QBackoff*, *QCCA* and *QSend*. This is sufficient because there is no difference in a collision of two or more packets and all observing nodes behave the same. Local rewards are necessary because a single node cannot observe the global state of the network, i.e., the actions executed at other nodes. Tbl. 5.4 shows the local rewards for all possible node interactions and the resulting global rewards.

Here, it can be seen that a reward is given individually for every action in response to the observed state of the channel. In particular, the local rewards R_B, R_C, R_S for the respective actions *QBackoff*, *QCCA* and *QSend* are given by:

$$R_B = \begin{cases} 2 & \text{if DATA or ACK packet overheard} \\ 0 & \text{otherwise} \end{cases} \tag{5.13}$$

$$R_C = \begin{cases} 3 & \text{if CCA success and TX success} \\ -2 & \text{if CCA success and TX failed} \\ 1 & \text{if CCA failed} \end{cases} \tag{5.14}$$

$$R_S = \begin{cases} 4 & \text{if TX success} \\ -3 & \text{if TX failed} \end{cases} \tag{5.15}$$

actions			local rewards			global reward
$a_t^{(0)}$	$a_t^{(1)}$	$a_t^{(2)}$	r_0	r_1	r_2	R
Successful transmission						
QBackoff	*QSend*	*QBackoff*	2	4	2	8
QBackoff	*QCCA*	*QBackoff*	2	3	2	7
QCCA	*QSend*	*QCCA*	1	4	1	6
No transmission						
QBackoff	*QBackoff*	*QBackoff*	0	0	0	0
Failed transmission						
QCCA	*QBackoff*	*QCCA*	-2	0	-2	-4
QSend	*QBackoff*	*QCCA*	-3	0	-3	-6
QCCA	*QCCA*	*QCCA*	-2	-2	-2	-6
QSend	*QCCA*	*QSend*	-3	1	-3	-5
QSend	*QSend*	*QSend*	-3	-3	-3	-9

Table 5.4: Local and global rewards for all interactions of Q-agents with the action set $\mathbb{A}_t = \{QBackoff, QCCA, QSend\}$.

QSend is a high-risk, high-reward action that results in the highest reward for successful communication but the largest punishment in the case of a failed transmission. Consequently, the reward for successful transmissions with *QCCA* is slightly lower because it implies the CCA's additional time and energy overhead. The same applies for *QCCA* and *QBackoff* if no packet is sent - a higher reward is given for *QBackoff* because it does not result in a potential collision. Different rewards have been experimentally evaluated, including more significant differences between the rewards. However, larger differences often favor one action over another. For example, increasing the reward for a successful transmission using *QSend* to eight results in a policy where every node executes *QSend* in every subslot. This is not desired. Therefore, the presented reward function is a careful balance between all actions supporting the desired objectives of increased throughput and reliability. It is the result of many experiments. The design of the reward function is an art in itself, there are no common blueprints known.

Initially, Q-values are initialized to $-\infty$ because Eq. (5.12) only updates the policy if a calculated Q-value is higher than the one in the Q-table. In practice, it is sufficient to choose a number smaller than the largest punishment. Thus, we initialize the Q-table to -10. It should be mentioned that, by observing the state of the channel and transmissions, QMA cannot distinguish between internal interference, external interference, and computational overload of a communication partner. Thus, QMA

also learns the patterns of external interference and adapts to them dynamically.

5.2.4 Parameter-based Exploration

As already insinuated in Sect. 3.6.3.1, a constant exploration rate ρ tends to be too slow for applications with dynamically changing conditions, and increasing it would result in too many random actions in a stable state. On the other hand, converging algorithms such as ϵ-greedy never reset ρ so that changes in the environment are only explored slowly.

For this reason, we propose a parameter-based exploration mechanism for efficient exploration in dynamically changing environments. The main idea is that actions are randomly selected with probability ρ, where ρ increases exponentially based on given parameters, e.g., queue size. A stable state can usually be recognized by utilizing local information: queues are empty, as all packets are transmitted successfully. Changing a network's state, e.g., when a node joins the network, results in many collisions because the new node first executes several random actions. In this case, queue levels increase, and more random actions are executed to find a new stable state. Actions are selected greedily when a stable state is reached. However, when the network is oversaturated, it is not sufficient to only look at the local queue level because all queues in the network are full. Therefore, the average queue level of all neighboring nodes is subtracted from the local queue level to assess if the network is oversaturated or if all other nodes transmit their packets successfully. Fig. 5.12 illustrates how ρ is chosen based on the difference between the local queue level and the average queue level of all neighbors in QMA. It is not desirable to execute actions completely randomly as it would destroy an already established action schedule. Therefore, $\rho = 0.3$ is used for the maximum queue level of $Q_{CAP} = 8\,\text{packet(s)}$. Since Q_{CAP} is usually small, ρ is stored in a table and can be used efficiently by resource-restricted devices without any computational overhead. If the average queue level of all neighboring nodes is larger than the local queue level, $\rho = 0$.

The application of parameter-based exploration is also conceivable in other contexts. For example, nodes could start exploring based on available energy and stop exploration when energy becomes scarce.

5.2.5 Cautious Startup

Nodes joining a stable network are likely to destroy an already created schedule, as argued in Sect. 5.2.4. We combat this effect with a cautious startup mechanism, where

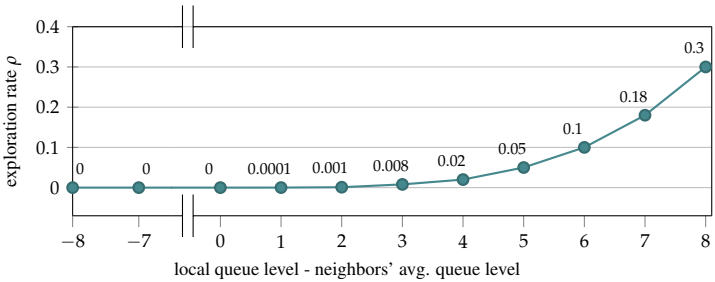


Figure 5.12: Dependency of ρ for parameter-based exploration on the queue level in QMA.

nodes only execute action *QBackoff* and observe the wireless medium for the first Δ subslot iterations after first selecting an action. During this time, overheard packets are registered, and a reward is given according to Eq. (5.13). Additionally, action *QCCA* and *QSend* are punished with negative rewards of -2 and -3, respectively. This way, subslots utilized by other nodes are likely indicated in the Q-table after cautious startup.

Although cautious startup cannot prevent nodes from randomly selecting a used slot, there is an initial bias that prevents them from fully committing to it. That means, e.g., if a node transmits a packet and there was no collision because the other node did, by chance, not have a packet to send, it received the positive reward. However, with $\alpha_{RL} = 0.5$, it takes multiple successful transmissions to change the policy for the slots because *QBackoff* also received a positive reward initially.

5.2.6 QMA: An Example

To illustrate how QMA's components work together, an example with three nodes v_1 , v_2 , v_3 is given in Fig. 5.13, showing the evolution of Q-tables with actions *QBackoff* (B), *QCCA* (C), and *QSend* (S) over the course of three frames. For simplicity, consider four subslots where the executed action is highlighted in gray, and the respective reward is given in brackets. Stars (*) indicate random actions that are triggered by parameter-based exploration. Notice that the Q-table's state is depicted *after* every frame, and Q-values are initialized to -10 as described in Sect. 5.2.3.6. Every executed action finishes within a single subslot, and $\pi(t)$ is initialized to *QBackoff*, as described in Alg. 1. Additionally, all updates are calculated

with Eq. (5.12) and Eq. (3.4) using $\alpha_{RL} = 1$ and $\gamma = 1$. At last, nodes always have at least one queued packet for transmission.

As depicted in Fig. 5.13, v_3 abides in cautious startup during the first frame. Thus, as described in Sect. 5.2.5, it selects action *QBackoff* and observes the channel. On the other hand, v_1 and v_2 started exploration resulting in v_1 executing *QSend* and v_2 executing *QCCA* randomly in the first subslots. For successfully transmitting the packet, v_1 receives a reward $R_S = 4$ (see Eq. (5.15)) while v_2 receives $R_C = 1$ for a failed CCA (see Eq. (5.14)). The Q-values are updated according to Eq. (5.12), where $\max_a Q(S_{t+1}, a) = -10$ applies during the first frame, as the next state has not been visited yet. In subslot four of the first frame, the Q-value of the next state (subslot one) has already been updated. Thus, the Q-value in subslot four of v_1 is updated to -4 because $R_B = 2$ and $\max_a Q(S_{t+1}, a) = -6$ (see Eq. (5.12)). v_3 overhears the successful transmission in subslot one and four and receives a reward of two (see Eq. (5.13)).

An interesting case occurs in subslot three of frame one. Here, v_1 and v_2 choose *QSend* and receive a reward $R_S = -3$ for a collision (see Eq. (5.15)). However, the Q-values are not updated to -13 but -12 because the newly calculated Q-value is not higher than the Q-value in the Q-table. Instead, $\xi = 2$ is subtracted from the Q-value in the Q-table. Here, v_3 does not overhear a packet due to the collision. One should note that the current policy is only changed if a larger Q-value is found. Thus, v_1 and v_2 execute *QBackoff* in the next frame.

During the next two frames, the Q-tables start to settle, and less randomness is involved in the action selection. *QCCA* is randomly selected by v_3 in subslot two of frame two, resulting in a successful transmission. Therefore, every node now has one subslot for transmission. At this point, it has to be said that more exploration and consequently more collisions occur in the actual execution of QMA. However, mechanisms like *cautious startup and* parameter-based exploration help reaching a stable state faster.

If $\gamma < 1$, the Q-table converges to a state where most executed actions are associated with a positive Q-value. In particular, subslots with successful transmission are associated with large positive Q-values while failed transmissions are indicated with large negative Q-values for action *QSend*. A more detailed analysis of QMA's convergence is given in Sects. 5.2.7.3 and 5.2.7.4. Additionally, a more extensive example of QMA, illustrating its convergence behavior over a longer period, can be found in Appendix A.2.

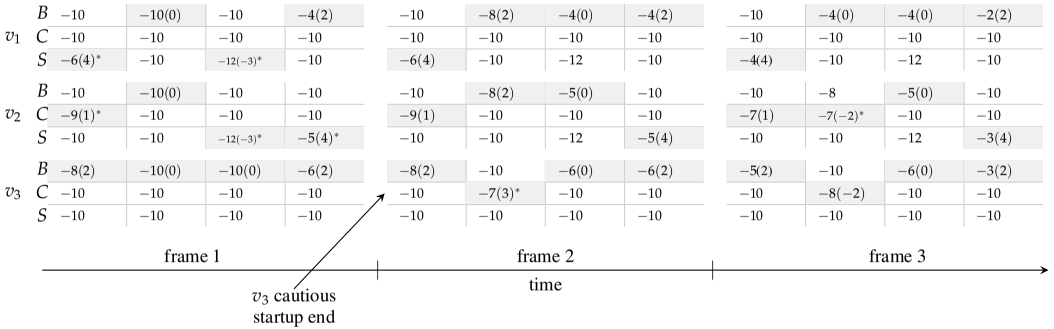


Figure 5.13: Example Q-tables during QMA's execution with $\xi = 2$, three nodes, four subslots, and actions *QBackoff* (B), *QCCA* (C), and *QSend* (S). Actions taken by an agent are highlighted in gray. The received reward for any action is given in brackets. Random actions are indicated by a star (*).

5.2.7 Simulative Evaluation

The following subsections describe the evaluation of QMA in different scenarios. For this, it is compared to slotted[1] and unslotted CSMA/CA, both of which are specified as multiple access schemes in IEEE 802.15.4 [IEE20]. The simulative evaluation is divided into two parts. After the simulation setup in Sect. 5.2.7.1, Sects. 5.2.7.2 to 5.2.7.4 describe QMA's capability to solve the hidden node problem without the overhead of RTS / CTS messages. Afterward, Sect. 5.2.7.5 demonstrates QMA's scalability in large networks with up to 91 nodes.

5.2.7.1 Scenario Description

In the first scenario, a hidden node problem with three nodes (v_0,v_1,v_2) is presented, where v_1 is the center node and v_0 and v_2 are not within interference range of each other, as depicted in Fig. 5.14a. Here, a CCA of v_0 or v_2 only fails if v_1 is currently sending an ACK. Data packets sent by v_0 or v_2 cannot be recognized by the opposite node using CCAs. Therefore, the CCA is successful most of the time even though the channel is not idle. We consider primary traffic with varying δ. The utilized MAC protocol features alternating active and sleeping phases with a length of 61.44 ms respectively, where QMA is utilized for transmission during the active phase. Generation of data packets starts after $ST = 100\,\mathrm{s}$ to allow for association and exchange of management information. This scenario enables easy estimation of the PDR for different channel utilizations.

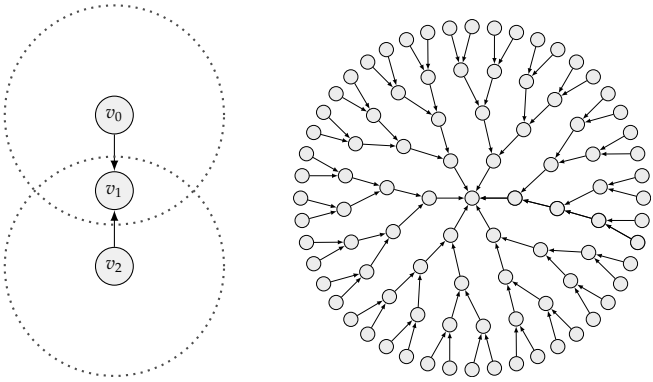
In the second scenario, QMA's scalability is investigated in a concentric topology with an increasing number of nodes, shown in Fig. 5.14b. A total of 7, 19, 43, and 91 nodes is evaluated, corresponding to one to four rings around the sink node. In contrast to the above configuration, QMA is evaluated for secondary traffic during the CAP of DSME, i.e., for (de)allocation of GTS and broadcast transmission. The latter are generated by the routing protocol GPSR for route discovery. Nodes alternately generate data packets with $\delta = 1\,\mathrm{packet(s)\,s^{-1}}$ and $\delta = 10\,\mathrm{packet(s)\,s^{-1}}$ for 5 s respectively. All data packets are transmitted during the CFP but imply secondary traffic for GTS (de)allocation. Similar to the first scenario, $WD = 200\,\mathrm{s}$ is utilized. Note that such data-collection scenarios often suffer from multiple hidden node problems [Kau19].

The relevant configuration parameters for the following simulations are summarized in Tbl. 5.5, and Tbl. A.3 describes the whole simulation setup.

[1] The implementation of slotted CSMA/CA has been advanced by T. Grewe in the context of his bachelor thesis.

topology	$\lvert\mathbb{V}\rvert$	α_{RL}	γ	WD	traffic	δ
hidden-node	3	0.5	0.9	100 s	Poisson	1 – 100 packet(s) s^{-1}
concentric	7 - 91	0.5	0.9	200 s	Poisson	1 – 10 packet(s) s^{-1}

Table 5.5: Parameters for the evaluation of QMA in a hidden-node and concentric topology.



(a) Hidden node problem with three nodes and the communication range of nodes v_0 and v_2.

(b) Concentric topology with 91 nodes and according routing tree.

Figure 5.14: Topologies for the evaluation of QMA.

5.2.7.2 Solving the Hidden Node Problem

Fig. 5.15 shows the PDR of slotted and unslotted CSMA/CA and QMA for nodes v_1 and v_3 of the hidden node problem. As one can see, QMA outperforms both CSMA/CA mechanisms for all packet generation rates. At higher packet generation rates, e.g., $\delta = 25\,\text{packet(s)}\,\text{s}^{-1}$, QMA maintains a PDR of 97 % while both CSMA/CA mechanisms show a PDR below 3.5 %. However, the performance difference becomes smaller for lower rates due to two reasons. Firstly, the probability of a collision is smaller for fewer transmitted packets utilizing CSMA/CA. Secondly, QMA learns faster if more packets are transmitted because different subslots can be tested and only the best are used for transmission. If only a few packets have to be sent, QMA still works but the certainty about good and bad transmission subslots is not that high.

QMA's main idea is to hold back packets until an appropriate transmission subslot is reached. This can lead to increased queue levels, as illustrated in Fig. 5.16. For

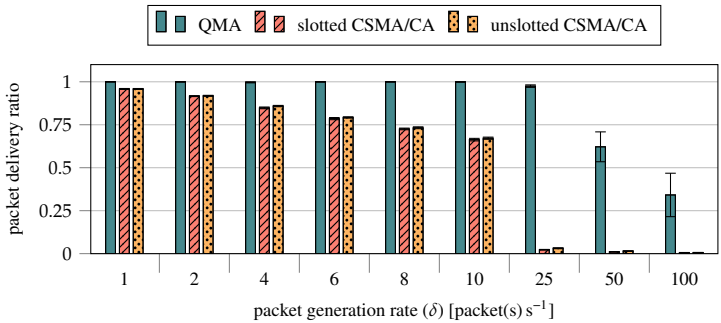


Figure 5.15: PDR of nodes v_1 and v_3 for varying packet generation rates δ.

$\delta \leq 10\,\text{packet(s)}\,\text{s}^{-1}$, QMA's queue level is slightly higher than the queue level of the CSMA/CA mechanisms, which consequently leads to a slightly increased delay as packets remain in the queue longer. The main cause of packet loss of the CSMA/CA mechanisms for $\delta \leq 10\,\text{packet(s)}\,\text{s}^{-1}$ are dropped packets due to too many retransmissions or, in rare cases, too many backoffs. For $\delta \geq 25\,\text{packet(s)}\,\text{s}^{-1}$, on the other hand, the probability for a collision is high, and the average queue level converges towards the maximum queue size of $Q_{CAP} = 8$. Here, most packets are lost due to queue drops as packets cannot be transmitted fast enough. QMA manages to reduce the queue level significantly for $\delta \geq 25\,\text{packet(s)}\,\text{s}^{-1}$ because it finds hidden patterns in the traffic and holds back packets to avoid collisions with the other nodes. This also reduces delay, as packets stay in the queue for a shorter time.

One should notice that the transmission behavior of CSMA/CA is rather random. Sometimes many consecutive packets are transmitted successfully, sometimes many transmissions fail due to the randomness in the exponential backoff algorithm. The same applies to QMA while it learns hidden traffic patterns, however, as soon as a feasible policy is found and if traffic conditions are stable, it offers deterministic reliability and delay.

5.2.7.3 Convergence and Adaptability

One of the main challenges in data-collection scenarios is adaptability to environmental changes, e.g., fluctuating traffic conditions and node failure. QMA offers good adaptability in these changing environments, as shown in Fig. 5.17. It depicts the cumulative Q-values per frame over time, i.e., the sum of Q-values for all subslots

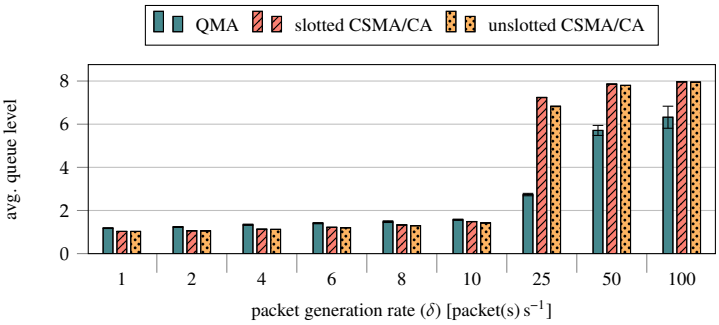


Figure 5.16: Average queue level of nodes v_1 and v_3 for varying packet generation rates δ.

following the best policy at that time. This metric cannot express the quality of the learned solution but indicates its stability. In other words: A higher cumulative Q-value does not mean that an agent performs better than an agent with a lower cumulative Q-value. However, a stable cumulative Q-value indicates that the Q-table does not change and the policy is stable [MKS+15].

Fig. 5.17 shows QMA's stability for different values of δ. During the first 100 s, the MAC protocol exchanges management information via QMA. Afterward, additional data packets are transmitted. It can be seen that QMA immediately reacts to the first transmitted management packets and settles on a stable policy about 9 s after the first transmission, as the rate of the management traffic is low. For $\delta = 100\,\text{packet(s)}\,\text{s}^{-1}$, the policy changes once more at 100 s when data packets start to be transmitted and after 265 s. The cumulative Q-value becomes constant about 30 s later. One should notice that the network is oversaturated for $\delta = 100\,\text{packet(s)}\,\text{s}^{-1}$ and nodes v_1 and v_3 compete for access to the channel. Thus agreeing on a common policy takes more time than in a non-saturated network. For $\delta = 1\,\text{packet(s)}\,\text{s}^{-1}$ and $\delta = 10\,\text{packet(s)}\,\text{s}^{-1}$, the policy does not change significantly when data packets start to be transmitted, as the management traffic exhibits about the same data rate.

Fig. 5.18 shows the according exploration rate ρ using parameter-based exploration. As one can see, exploration is triggered earlier for higher δ because queues fill quicker. As the network is oversaturated, ρ increases multiple times for $\delta = 100\,\text{packet(s)}\,\text{s}^{-1}$, and nodes v_1 and v_3 compete for access to the shared medium. A maximum of $\rho = 0.1$ is experienced for $\delta = 100\,\text{packet(s)}\,\text{s}^{-1}$, corresponding to a queue level of 6 packet(s).

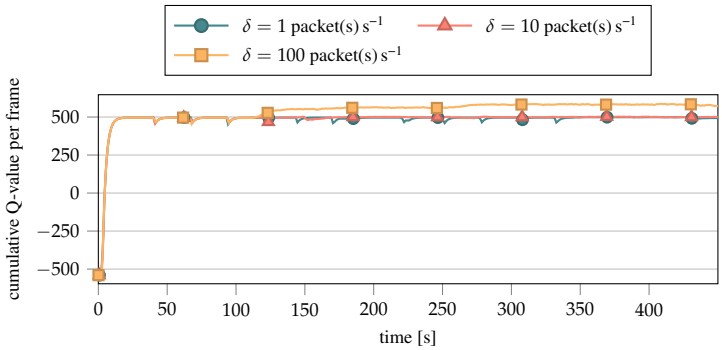


Figure 5.17: Cumulative Q-value per frame for varying packet generation rates δ.

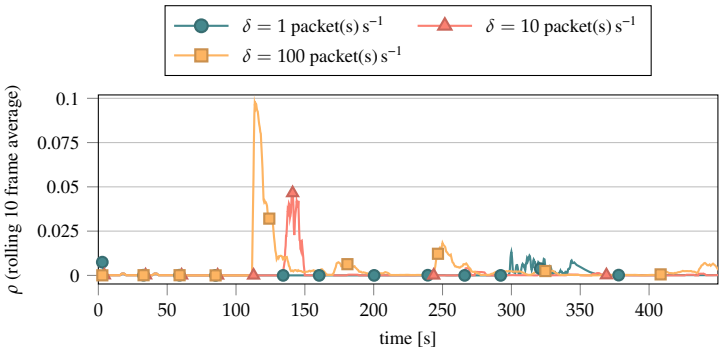


Figure 5.18: Exploration rate ρ of parameter-based exploration over time using a rolling average over ten frames for different values of δ.

To illustrate QMA's adaptability, the scenario is slightly altered to incorporate fluctuating traffic conditions. Node v_1 alternately generates packets with $\delta = 10\,\text{packet(s)}\,\text{s}^{-1}$ and $\delta = 100\,\text{packet(s)}\,\text{s}^{-1}$ for 100 s respectively. Node v_3 generates packets with a constant $\delta = 25\,\text{packet(s)}\,\text{s}^{-1}$ but joins the network 100 s after v_1. The positions of the nodes remain as shown in Fig. 5.14a. Fig. 5.19 shows the cumulative Q-values per frame. The dashed lines indicate when the traffic pattern of v_1 changes.

As one can see, node v_1 immediately reacts to changes in its packet generation pattern with increasing and decreasing Q-values. The network is oversaturated

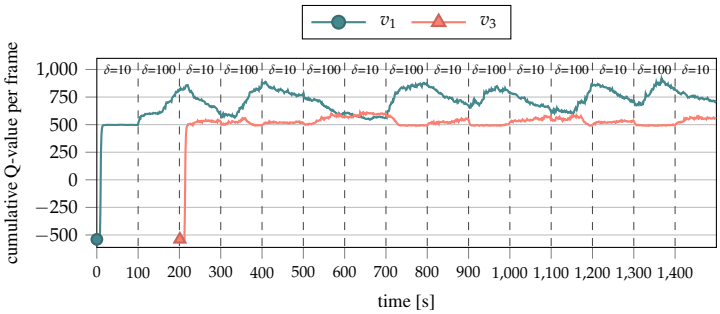


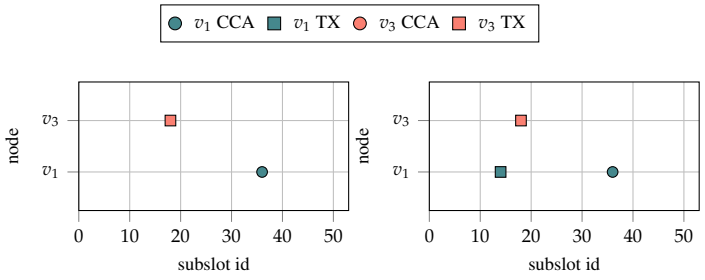
Figure 5.19: Cumulative Q-value per frame of nodes v_1 and v_3 for fluctuating traffic.

for $\delta = 100\,\text{packet(s)}\,\text{s}^{-1}$ and not saturated for $\delta = 10\,\text{packet(s)}\,\text{s}^{-1}$. Thus, node v_3 tries to acquire more transmission slots when v_1 generates packets with $\delta = 10\,\text{packet(s)}\,\text{s}^{-1}$. This is indicated by the increase of the cumulative Q-value of node v_3 for $\delta = 10\,\text{packet(s)}\,\text{s}^{-1}$ at node v_1. The change of Q-values is triggered by parameter-based exploration, as packets start to pile up in the queue of node v_3 when node v_1 transmits with $\delta = 100\,\text{packet(s)}\,\text{s}^{-1}$. Thus, QMA's reaction time is mainly dictated by the exploration rate assigned to different queue levels, as described in Sect. 5.2.4, and can be increased and decreased according to the given scenario. At last, joining the network late does not influence the performance of node v_3, since a stable policy is still found.

5.2.7.4 Subslot Utilization

Fig. 5.20 shows the subslot utilization by the hidden nodes v_1 and v_3 for varying δ. As illustrated by Fig. 5.18, exploration is triggered at different times depending on δ and hence, Figs. 5.20a to 5.20f depict the subslot utilization after the first exploration phase, i.e., at 370 s for $\delta = 1\,\text{packet(s)}\,\text{s}^{-1}$, 150 s for $\delta = 10\,\text{packet(s)}\,\text{s}^{-1}$, and 170 s for $\delta = 100\,\text{packet(s)}\,\text{s}^{-1}$. In addition, the final policy is shown, and *QBackoff* is executed if no action is depicted.

As one can see, a collision-free schedule of subslots is created for all values of δ, i.e., nodes v_1 and v_3 never select actions *QCCA* or *QSend* in the same subslot. Also, no *QSend* actions take place in adjacent subslots, which is vital because transmissions span up to three subslots. In Fig. 5.20a, v_1 chooses action *QCCA*, however, there is almost no difference between *QSend* and *QCCA* in the hidden node problem as a CCA



(a) Slot utilization after 370 seconds with $\delta = 1\,\text{packet(s)}\,\text{s}^{-1}$.

(b) Slot utilization for final policy with $\delta = 1\,\text{packet(s)}\,\text{s}^{-1}$.

(c) Slot utilization after 150 seconds with $\delta = 10\,\text{packet(s)}\,\text{s}^{-1}$.

(d) Slot utilization for final policy with $\delta = 10\,\text{packet(s)}\,\text{s}^{-1}$.

(e) Slot utilization after 170 seconds with $\delta = 100\,\text{packet(s)}\,\text{s}^{-1}$.

(f) Slot utilization for final policy with $\delta = 100\,\text{packet(s)}\,\text{s}^{-1}$.

Figure 5.20: Slot utilization after the first exploration phase and for the final policy for varying packet generation rates δ.

is almost always successful. The only exception is when an ACK by v_2 is received by chance by one of the hidden nodes. In the final policy, v_1 selects an additional transmission slot to maximize the total expected reward.

Figs. 5.20c and 5.20e show that the first exploration phase results in many transmission slots at a single node. Additionally, Figs. 5.20d and 5.20f display that v_1 and v_3 divide the medium among themselves in the final policy. Such patterns arise when the exploration rate is high, i.e., when the queue is full using parameter-based exploration. In this case, many consecutive random actions are executed as shown in Fig. 5.20e. If no more packets are available for transmission, no action is selected. In the next iteration, a node uses the newly allocated transmission subslots, potentially leaving later subslots unused because the queue is empty. This allows v_1 to allocate transmission slots in the later subslots and, eventually, v_3 gives up these later subslots because it overhears ACKs for v_1.

In contrast to that, many subslots are not utilized for $\delta = 1\,\mathrm{packet(s)}\,\mathrm{s}^{-1}$ and $\delta = 10\,\mathrm{packet(s)}\,\mathrm{s}^{-1}$, i.e., action *QBackoff* is chosen. An implication is that there is still room for scheduling additional transmissions. However, one should be aware that transmissions can last for more than one subslot and thus there is a necessity for some idle slots.

5.2.7.5 Scalability in Data-collection Scenarios

Fig. 5.21 shows the PDR of DSME's secondary traffic for an increasing number of nodes. As one can see, QMA outperforms slotted and unslotted CSMA/CA for any number of nodes, however, the performance difference is larger for a smaller number of nodes. That is because QMA can learn a collision-free policy faster if fewer nodes are within interference range because the probability of randomly selecting an unused subslots is higher. In comparison to QMA, and as shown in Sect. 5.2.7.2, CSMA/CA cannot fully solve the hidden node problem. Thus its PDR is significantly lower for all numbers of nodes. Here, the number of nodes does not significantly influence the performance of CSMA/CA since they are placed far enough from each other. QMA's increased PDR during the CAP also results in an increased PDR of primary traffic of about 10 % for any number of nodes, as GTS are (de)allocated faster and fewer packets are dropped due to a full queue.

Fig. 5.22 demonstrates QMA's capability of learning the hidden patterns during slot allocations. As shown in Sect. 5.1, slot (de)allocations through 3-way handshakes require a high PDR during the CAP. The increased PDR during the CAP significantly increases the number of successfully transmitted *GTS request* messages, which are

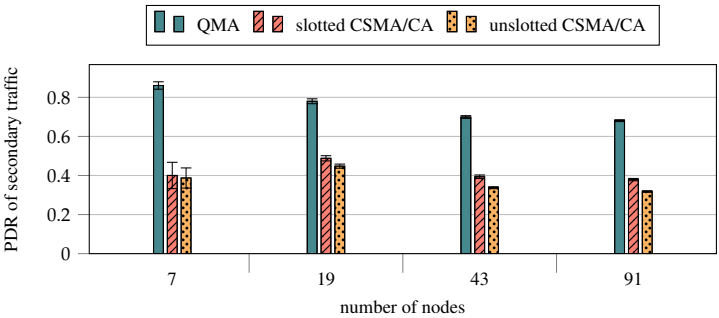


Figure 5.21: PDR of secondary traffic during the CAP for increasing numbers of nodes.

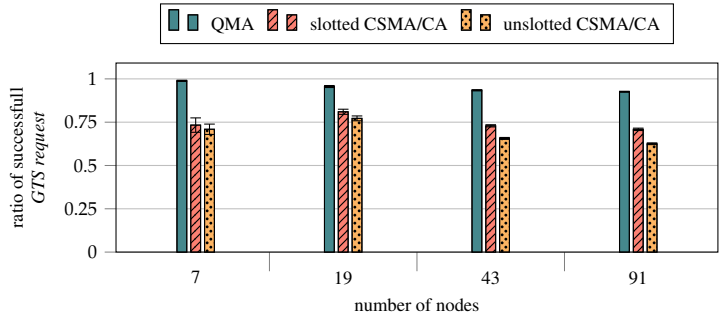


Figure 5.22: Ratio of successfully transmitted to totally transmitted *GTS request* for an increasing number of nodes.

used to initialize the 3-way handshake. QMA's advantage seems to diminish for *GTS response* and *GTS notify* messages. For both message types, QMA manages to transmit 99 % messages successfully, while CSMA/CA achieves about 93 % and 99 % for *GTS response* and *GTS notify* messages, respectively. However, in absolute numbers, CSMA/CA only transmits about 2/3 of the *GTS notify* messages that QMA transmits so that both methods result in about 1 % failed *GTS notify* messages.

5.2.8 Verification on Hardware

The following sections verify QMA on hardware by applying it in realistic data-collection scenarios. Simulations in Sect. 5.2.7.2 and initial hardware experiments

scenario	$\lvert V \rvert$	sensitivity	tx power
star	17	−90 dBm	3 dBm
tree	10	−72 dBm	−9 dBm
traffic	δ	**wake duration**	**sleep duration**
Poisson	10 packet(s) s^{-1}	61.44 ms	61.44 ms
Poisson	10 packet(s) s^{-1}	61.44 ms	61.44 ms

Table 5.6: Parameters for the evaluation of QMA on hardware

have shown that slotted and unslotted CSMA/CA exhibit almost the same performance. Thus the following sections feature only a comparison of unslotted CSMA/CA and QMA. More precisely, Sect. 5.2.8.1 covers the experimental setup in the FIT IoT-LAB [ABF+15], and Sects. 5.2.8.2 and 5.2.8.3 describe results of realistic use-cases in a dense single-hop star topology and a multi-hop tree topology, respectively.

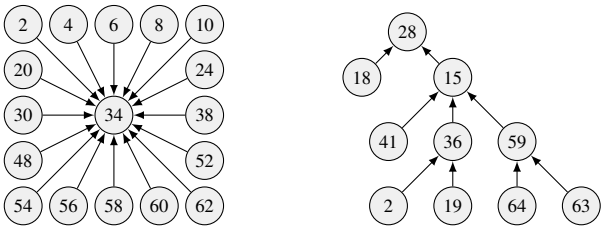
5.2.8.1 Scenario Description

To verify QMA's resource-efficiency and apply it in a scenario with physical nodes, experiments are conducted in the *Strasbourg* testbed of the FIT IoT-LAB [ABF+15]. It should be emphasized again that the targeted M3 Open Node, described in Sect. 3.5, does not feature a floating-point unit and hence an efficient implementation is required.

Two scenarios are evaluated: a star topology with 17 nodes and a tree topology with ten nodes, illustrated in Figs. 5.23a and 5.23b, respectively. The tree topology is generated using an algorithm proposed in [KKT18]. According to their description, transmission power is set to −9 dBm and sensitivity is set to −72 dBm. For the star topology, a transmission power of 3 dBm and a sensitivity of −90 dBm is used. The traffic conditions are similar to Sect. 5.2.7, i.e., 1000 packet(s) are generated according to a Poisson distribution and a MAC protocol with alternating active and sleeping phases of 61.44 ms duration is utilized. Tbl. 5.6 shows the relevant configuration parameters for the star topology and the tree topology and Tbl. A.3 shows the full configuration.

5.2.8.2 Star Topology

Fig. 5.24 shows the PDR of the star topology illustrated in Fig. 5.23a. As one can see, QMA achieves a higher PDR than unslotted CSMA/CA for all nodes. That is remarkable because most nodes in the star topology can hear one another so that



(a) Star topology. **(b)** Tree topology.

Figure 5.23: Physical node positions and routing links for evaluation of a star topology and tree topology in the Strasbourg testbed of the FIT IoT-Lab.

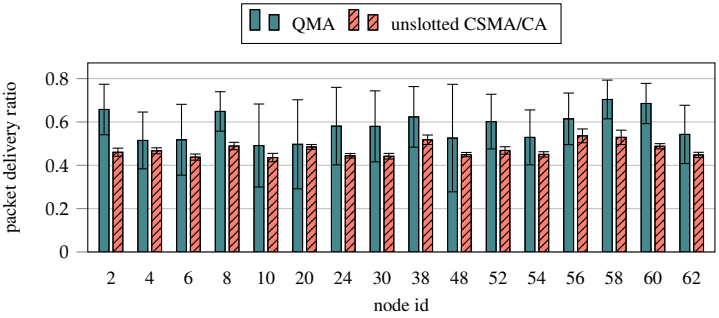


Figure 5.24: Average PDR per node for a star topology in the Strasbourg testbed of the FIT IoT-Lab with $\delta = 10\,\text{packet(s)}\,\text{s}^{-1}$.

many collisions can be prevented through CCAs. Slotted CSMA/CA performs a CCA before every transmission while QMA has to learn such behavior.

Another interesting observation is that nodes seemingly achieve a fair distribution of transmission subslots using QMA, although no fairness constraint is defined explicitly in its reward function. For example, it could have happened that a few nodes acquire all available subslots for action *QSend* while the other nodes starve. Fairness arises due to two reasons.Firstly, QMA utilizes a cooperative Q-learning approach, as discussed in Sect. 3.6.3.4. This helps build a schedule but does not fully ensure fairness. However, parameter-based exploration results in many random actions if a node possesses too few transmission subslots. Thus, if a node used all subslots for *QSend*, its transmissions would be interfered by these random actions, and it would give up some transmission slots. Another node could then acquire these subslots

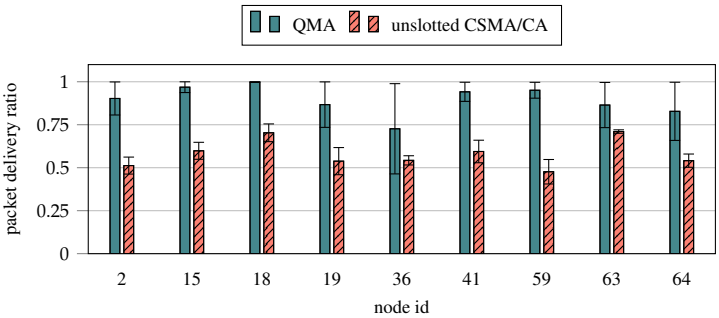


Figure 5.25: Average PDR per node for a tree topology in the Strasbourg testbed of the FIT IoT-Lab with $\delta = 10\,\mathrm{packet(s)}\,\mathrm{s}^{-1}$.

for *QSend*. This process balances itself, given that the network is not drastically oversaturated.

5.2.8.3 Tree Topology

In contrast to the star topology from Sect. 5.2.8.2, the tree topology shown in Fig. 5.23b exhibits several hidden node problems. Hence, Fig. 5.25 supports the results from Sect. 5.2.7.2 in the sense that QMA outperforms unslotted CSMA/CA on every node in terms of PDR. The performance difference between QMA and unslotted CSMA/CA is larger than in the star topology because only the transmissions of parents, children, and siblings interfere with each other. Additionally, confidence intervals are smaller because less nodes try to access the wireless medium in a given area making it easier for QMA to agree on a schedule.

At last, measurements in the FIT IoT-LAB show no difference between QMA and unslotted CSMA/CA in terms of power consumption. Thus, one can conclude that QMA increases reliability without introducing an energy-overhead. In fact, both multiple access schemes conduct about the same number of transmission attempts, however, using CSMA/CA many failed CCAs and retransmissions occur.

5.2.9 Discussion and Conclusion

We demonstrate QMA's effectiveness in three different scenarios:

- It is utilized for primary traffic in a hidden node problem with three nodes and varying packet generation rates. Simulations show that it is possible to increase

the number of transmitted packets from 10 packet(s) s^{-1} to 50 packet(s) s^{-1} in comparison to CSMA/CA while maintaining the same packet delivery ratio.

- QMA's scalability is evaluated for secondary traffic in DSME in a realistic scenario with an increasing number of nodes. QMA achieves a higher packet delivery ratio than CSMA/CA and manages to (de)allocate up to twice more TDMA-slots per second.

- A real implementation of QMA is deployed and verified on hardware.

It must be said that there is much potential to reduce QMA's memory overhead. For example, by carefully selecting rewards, it should be possible to significantly reduce the entries of the Q-table so that only 2 – 8 bit are required per entry. Simultaneously, floating-point operations could be avoided. A lot of research is designated to these specific optimizations, so that these are out of the scope of this dissertation. Additionally, the validation of QMA as contention-based multiple access scheme for secondary traffic in protocols other than DSME remains subject to future research.

All in all, we believe that QMA is a practical realization of a more reliable contention-based channel access for the IIoT. The benefits of QMA are an indicator that machine learning techniques bear a high potential for application in communication protocols.

Chapter 6

Primary Traffic in DSME

Primary traffic refers to data messages transmitted by the application layer. They are are usually delivered through exclusive channel access to ensure high reliability and strict timeliness, i.e., in DSME during the CFP. To ensure that all receivers are listening, application layer broadcasts may also be transmitted with contention-based channel access, e.g., in DSME's CAP. As shown in Chap. 5, primary and secondary traffic are subject to considerable reciprocal interactions. Therefore, the following sections analyze sending multiple packets per GTS to reduce the management overhead in secondary traffic (Sect. 6.1) and describe a way to reduce the ACK overhead using GACKs (Sect. 6.2).

More specifically, we show that sending multiple packets per GTS not only increases the throughput of primary traffic by utilizing the unused space at the end of GTS, but also significantly reduces the number of management messages for GTS (de)allocations. Group Acknowledgments have the potential to amplify this effect by reducing the ACK overhead and freeing up time for further data transmission. However, the obtained results indicate that GACKs are generally not a valid alternative to regular ACKs in DSME, and only provide benefits in a best-case star topology.

Sending Multiple Packets per GTS
Section 6.1

Sect. 5.1 analyzed the influence of bursty and fluctuating primary traffic on the slot negotiation process of DSME, demonstrating that the reliability of the secondary traffic is severely degraded when bursts in the primary traffic occur. As a consequence, also the reliability and timeliness of primary traffic is diminished because transmission slots cannot be (de)allocated sufficiently fast and queue drops occur. Therefore, it is vital to envision mechanisms that alleviate CSMA/CA traffic to provide increased responsiveness without diminishing performance of the primary traffic. We identify three possible solutions:

- a more reliable contention-based multiple access scheme (see QMA in Sect. 5.2)
- negotiating multiple GTS with one handshake
- transmitting multiple packets per GTS

The second option requires far-reaching modifications of the allocation handshake to incorporate information about additional slots to negotiate. Additionally, it can easily result in inconsistent states if only a subset of the GTS to (de)allocate is available. To the authors knowledge, there is currently no mechanism that can (de)allocate several slots in bounded time. The third option appears more promising by considerably lowering the load caused by GTS allocations since the number of required GTS is significantly reduced. We believe that such a mechanism may be the missing ingredient for DSME's adaptability to time-varying traffic. It thus enables the construction of more reliable and agile IIoT applications.

The following sections study the transmission of multiple packets per GTS. For this, Sect. 6.1.1 provides an overview of related work, and the theoretical evaluation in Sect. 6.1.2 calculates the maximum number of packets per GTS analytically. At last, Sect. 6.1.3 analyzes sending multiple packets per GTS through simulation under the above-mentioned aspects.

Previously published in "Meyer, F., Mantilla-González, I., Turau, V.: Sending Multiple Packets per Guaranteed Time Slot in IEEE 802.15.4 DSME: Analysis and Evaluation. Internet Technology Letters. 4, e167 (2020). https://doi.org/10.1002/itl2.167." [MMGT20b]. The author is responsible for the simulative evaluation and parts of the theoretical analysis.

6.1.1 Related Work

An parameter study of *SO* and *MO* in DSME is conducted by F. Kauer [Kau19]. They find that small values of *SO*, i.e., slots with short length, increase throughput for transmission of single packets. Contrary, large values of *SO* improve energy consumption in low-traffic scenarios, since nodes can turn off their transceivers while no transmission is pending. Kauer also discusses sending multiple packets per GTS but do not formally analyze this option. They contend that incrementing *SO* doubles the number of transmittable packets per GTS, yet exhibits the same throughput as a single packet transmission mechanism with the original *SO* and twice the number of GTS. Therefore, Kauer only considers the transmission of a single packet per GTS.

Simulations analyzing the GTS (de)allocation process in DSME are conducted by Vallati et al. using Cooja [VBPA17]. Their work aims to select CSMA/CA parameter that minimize the number of CAPs to complete the network setup, i.e., the association process. The DSME implementation used in their work delivers four packets per GTS. Results evidence the advantage of sending multiple packets per GTS in contrast to a single packet. However, they do not analyze the influence on secondary traffic.

Akbar et al. propose and analyze sending multiple packets per time slot in wireless body area sensor networks based on IEEE 802.15.4 [AYC17]. They utilize a frame aggregation mechanism to increase throughput and reliability in patient monitoring systems. To this end, they examine different traffic pattern and quality of service requirements but do not estimate the impact of fluctuating primary traffic on the allocation process of the TDMA slots.

6.1.2 Theoretical Evaluation

The following subsections analyze the maximum number of transmission attempts per GTS analytically. The basic calculation is explained in Sect. 6.1.2.1, while Sect. 6.1.2.2 highlights the influences of the slots length and packet payload.

6.1.2.1 Maximum Number of Transmission per GTS

For the maximum number of transmissions per GTS, a best-case and a worst-case estimation can be given. According to the IEEE 802.15.4 standard [IEE20], an ACK during the CFP is transmitted at earliest a $aTurnaroundTime$ after the last symbol of a packet is received, i.e., a bast-case. In the worst case, a receiver has to wait a $macAckWaitDuration$ for an ACK, which amounts to an $aUnitBackoffPeriod$ and

	P_{GTS}				
S	$SO = 3$	$SO = 4$	$SO = 5$	$SO = 6$	$SO = 7$
1 B	7-10	14-20	28-40	56-80	112 - 160
50 B	2	4-5	9-11	19 - 22	39 - 44
100 B	1	3	6-7	13-14	26-28
127 B	1	2	5	11	22-23

Table 6.1: Upper and lower bounds for the maximum number of transmissions per GTS with different values of SO and payloads S.

an *aTurnaroundTime* [IEE20]. Therefore, the maximum number of transmissions per GTS P_{GTS} is bounded by

$$\frac{\mathcal{S}_{GTS}}{2S + 54\,\text{symbol(s)} + IFS} \leq P_{GTS} \leq \frac{\mathcal{S}_{GTS}}{2S + 34\,\text{symbol(s)} + IFS}, \tag{6.1}$$

where $\mathcal{S}_{GTS} = 60 \cdot 2^{SO}$ is the number of symbols per GTS, S is the payload of the packet in bytes (2 symbols per byte), and $IFS = 40\,\text{symbol(s)}$ when $S > 18\,\text{B}$ and $IFS = 12\,\text{symbol(s)}$, otherwise. The constants of 34 symbol(s) and 54 symbol(s) include the overhead of the physical layer header, the size of an ACK packet, and the minimum and maximum ACK wait duration, respectively. Scheduling algorithms must take the transmission of multiple packets per GTS into account by estimating the required number of GTS based on the lower bound obtained from Eq. (6.1).

6.1.2.2 Influence of Payload Length and SO

Tbl. 6.1 shows that the maximum number of packets per GTS increases exponentially for an increasing SO regardless of S. That is because by enlarging SO, the duration of GTS doubles (see Sect. 2.4.4.4). However, this can lead to more than twice as many transmitted packets, e.g., two packets can be transmitted for $S = 127\,\text{B}$ and $SO = 4$, while up to five packets can be transmitted for $SO = 5$. The reason is that the duration of frame transmissions, including acknowledgments, does not exactly match the duration of a GTS. The unused time at the end of a GTS increases with increasing SO until it is large enough to transmit an extra packet. Consequently, the potential throughput of the network increases by transmitting multiple packets per GTS with larger SO.

Since P_{GTS} increases exponentially with increasing SO, the number of CAP messages is assumed to decrease exponentially as well, especially because fewer collisions and hence fewer retransmissions occur. On the other hand, this can result in longer GTS queues and higher packets delays since packets cannot be immediately trans-

mitted upon arrival or generation. Instead, they are transmitted in chunks of packets in a few GTS. Also, the probability of overprovisioning increases for higher *SO* because P_{GTS} is larger, and sending M additional packets could result in the allocation of a new GTS and overprovisioning for up to $P_{GTS} - (M \mod P_{GTS})$ additional transmissions. This overprovisioning reduces delays and avoids queue drops because packets are transmitted immediately when a burst happens, and queued packets can be transmitted faster.

6.1.3 Simulative Evaluation

The following sections evaluate sending multiple packets per GTS through simulation in OMNeT++ to verify the theoretical results obtained in Sects. 6.1.2.1 and 6.1.2.2 and to provide further insights into potential benefits and drawbacks. For this, Sect. 6.1.3.1 first describes the scenario. Afterward, Sect. 6.1.3.2 verifies the results of the theoretical model, Sect. 6.1.3.3 describes the influence on secondary traffic, and Sect. 6.1.3.4 analyzes additional metrics such as reliability, delay, and energy consumption.

6.1.3.1 Scenario Description

A grid topology with 4 x 4 nodes is considered to evaluate the performance of sending multiple packets per GTS under bursty traffic. It utilizes a static shortest path routing which guarantees that all outer nodes of the grid communicate with unique and fixed partners and that the path length between these nodes is at least two. Inner nodes act as relays, i.e., they receive and forward packets. Thus, six communication pairs (i.e., six routes) are active per run: Two routes with length three and four routes with length two. This setup creates constantly shifting, highly variable traffic characteristics, especially for inner nodes.

Tbl. 6.2 summarizes the simulation setup. The full configuration is given in Tbl. A.4. In this scenario, we set the queue length to $Q_{CAP} = Q_{CFP} = 40$ to ensure that DSME can cope with large bursts of up to $\epsilon = 32\,\text{packet(s)}$. The MAC payload is 127 B. *MO* and *BO* are fixed to nine because they do not influence the performance of the presented scenario, resulting in an D_{MSF} and BI of 7.8464 s. The packet generation interval γ follows a normal distribution with mean $\mu = 16\,\text{s}$ and standard deviation $\sigma = 0.5\,\text{s}$. Hence, γ is at least twice as long as D_{MSF} to allow for the (de)allocation of GTS.

scenario	$\|\mathbb{V}\|$	SO	MO	BO	traffic	δ
grid	16	3,..,6	9	9	Normal	2,..,40 packet(s) s^{-1}

Table 6.2: Parameter for the evaluation of multiple packets per GTS through simulation.

6.1.3.2 Transmissions per GTS

As predicted in Sect. 6.1.2.1, simulations yield an exponentially increasing number of transmitted packets per GTS for an increasing SO. However, the actual number of packets transmitted in a GTS can be highly unbalanced, e.g., with $\epsilon = 16\,\text{packet(s)}$, $SO = 6$ and two allocated GTS, eleven packets are transmitted in the first GTS and five in the second GTS on average, resulting in an average of eight transmissions per GTS.

Fig. 6.1 shows the average number of allocated GTS for transmitting nodes and different values of ϵ and SO. The larger the SO, the more packets can be sent per GTS and the fewer GTS must be allocated. On the other hand, a too large SO results in overprovisioning. For example, $SO = 5$ is optimal for $\epsilon = 4$ since $SO = 4$ requires two more allocated GTS and $SO = 6$ implies overprovisioning of six to seven packets. Moreover, the average number of allocated GTS is higher than expected, e.g., for $SO = 3$ and $\epsilon = 8$, one would expect an average of four allocated GTS instead of about 5.7 GTS since the number of required GTS is either zero or eight. However, TPS keeps one remaining GTS per established link, increasing the average number of allocated GTS. For large bursts with $SO = 3$ and $\epsilon \geq 32$, GTS cannot be allocated fast enough anymore so that its performance is about equal to $SO = 4$. For the evaluated scenario, this limit is not reached for configurations with $SO > 3$.

6.1.3.3 Influence on Secondary Traffic

As fewer GTS are allocated, fewer CAP messages have to be transmitted for $SO \geq 4$ when compared to a single packet transmission per GTS ($SO = 3$), as illustrated in Fig. 6.2. TPS further emphasizes this effect by deallocating GTS cautiously. For example, only one GTS is required for $SO = 6$ and $\epsilon = 10$ so that there is almost no CAP traffic. It may seem counter-intuitive at first that the number of CAP messages decreases for $SO = 3$ with an increasing ϵ, while it increases with for higher values of SO. The reason is that for $SO = 3$ contention increases with increasing ϵ as more nodes try to (de)allocate GTS. Therefore, nodes go into backoff more often and as long as not all required GTS are (de)allocated according to the incoming traffic

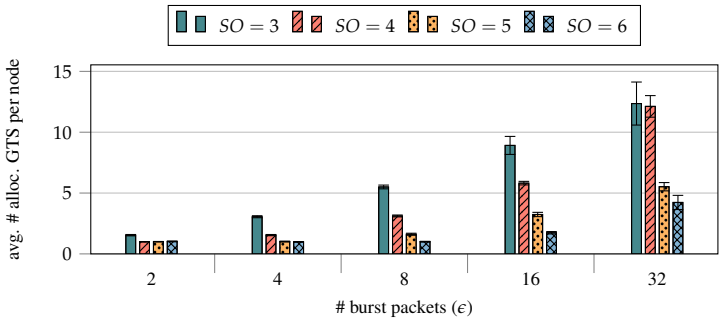


Figure 6.1: Average number of allocated GTS per node for different slot lengths (SO) and packets per burst (ϵ).

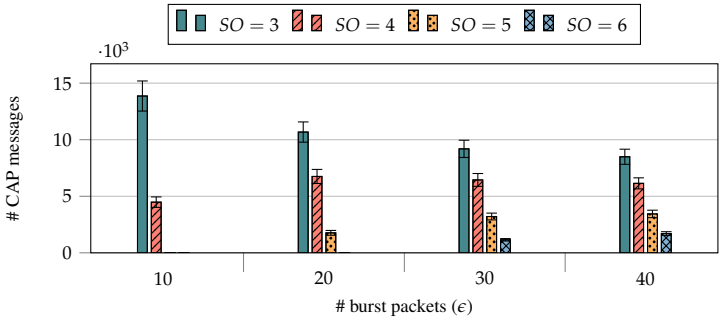


Figure 6.2: Total number of transmitted CAP messages for different slot lengths (SO) and packets per burst (ϵ).

rate, queues overflow and the number of GTS handshakes during the CAP decreases, resulting in a lower PDR. $SO = 6$ performs best for all ϵ. For example, in the case of $\epsilon = 10$, the CAP traffic is reduced by 99 % in comparison to $SO = 3$.

6.1.3.4 Delay, Reliability, and Energy

The queue drops mentioned in Sect. 6.1.3.3 are also reflected in the PDR, depicted in Fig. 6.3. For transmissions of a single packet per GTS ($SO = 3$), the PDR decreases exponentially for an increasing ϵ down to 33.6 % at $\epsilon = 40$. Contrary, higher values of SO result in a PDR above 80 % for $\epsilon \leq 40$, with $SO = 6$ performing best. That is

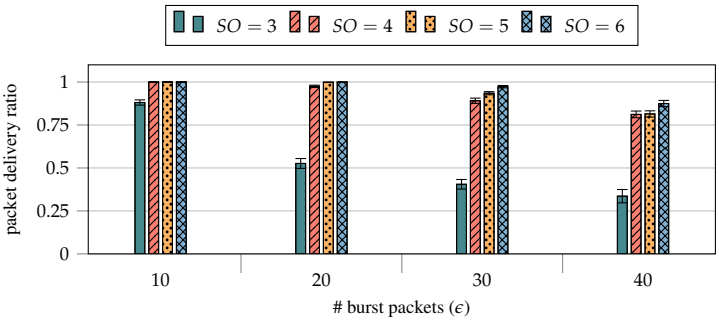


Figure 6.3: Average PDR for different slot lengths (SO) and packets per burst (ϵ).

because fewer GTS are required, and it is more likely that all GTS negotiations can be performed and no packets are dropped due to a full queue. Therefore, it is possible to transmit about 30 packet(s) more per burst while maintaining the same PDR by choosing $SO = 3$ instead of $SO = 6$. Additionally, one should note that lost packets are expected for $\epsilon = 40$ for any SO because it exactly matches the capacity of the GTS queue. For $\epsilon < 8$, no packet loss is observed.

As mentioned in Sect. 6.1.2.2, the transmission delay is expected to increase for an increasing SO as packets are delayed and sent in a single slot. That, however, is disproved by the results shown in Fig. 6.4. Here, the delay per hop for $SO = 3$ is significantly higher than the delay for other SO. That is due to two reasons:

- More GTS have to be allocated for $SO = 3$, resulting in a long time until the schedule is completed and packets can be transmitted.
- Choosing $SO \geq 4$ results in overprovisioning that is potentially larger for larger SO as indicated in Sect. 6.1.2.2.

For example, for $SO = 6$ and $\epsilon = 16$, two GTS are allocated so that five extra packets can be transmitted. This extra GTS reduces the GTS queue length and hence the delay per hop. With $SO = 5$, two GTS with $P_{GTS} = 5$ are allocated while one GTS with $P_{GTS} = 11$ is allocated for $SO = 6$. Although the overprovisioning is similar in both cases, D_{MSF} is longer for $SO = 6$, resulting in a higher expected time until the next allocated GTS and thus increased delay.

At last, Fig. 6.5 shows the average energy consumption per node over the course of the simulation. The energy consumption is highest for a single packet per GTS

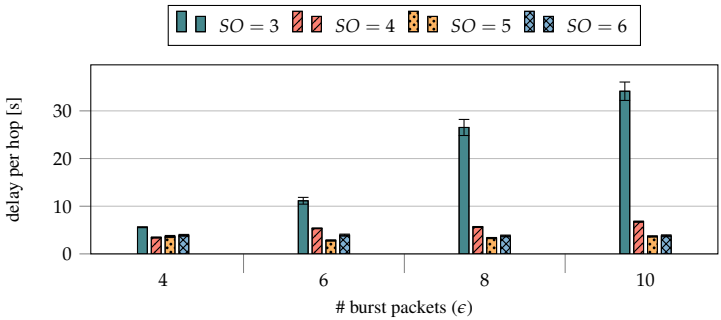


Figure 6.4: Average delay per hop for different slot lengths (SO) and increasing numbers of packets per burst (ϵ).

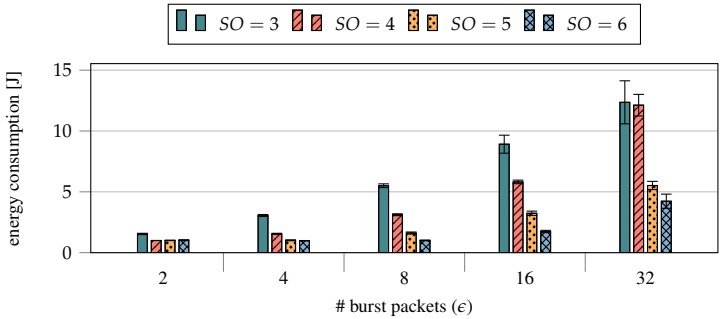


Figure 6.5: Average total energy consumption per node for different slot lengths (SO) and increasing numbers of packets per burst (ϵ).

($SO = 3$) because more GTS negotiations occur in the CAP, during which the transceiver is in a high-power state. Additionally, it can be seen that the energy consumption stays about constant for $SO = 6$ until $\epsilon = 16$. From this point, nodes need to allocate two GTS instead of one GTS, resulting in higher energy consumption. For $\epsilon = 1$, higher values of SO result in higher energy consumption because the receiver has to stay active during the whole GTS.

6.1.4 Discussion and Conclusion

Sects. 6.1.3.2 to 6.1.3.4 show that sending multiple packets per GTS reduces the load for secondary traffic during the CAP and enables higher reliability, shorter delays, and less energy consumption. Nonetheless, one should note that sending multiple packets per GTS is not beneficial for all application scenarios. CAPs are longer for larger SO, potentially resulting in a non-negligible fraction not being utilized. This idle time can be counterproductive for energy-harvesting devices as nodes have to stay active during the whole CAP. In data-collection scenarios, this additional time could be used for secondary traffic or unreliable primary traffic if the application allows it. Still, CAPs at the end of a MSF are unused most of the time. Therefore, an essential next step is to find an algorithm that tunes the number of CAPs according to the traffic demand of the network because DSME only supports using all CAPs or a single CAP per MSF with CAP-Reduction (CAP-R). Such an algorithm is proposed and further explored in Chap. 8.

The evaluation assumes that all nodes generate similar amounts of traffic. If the packet generation rates of individual nodes vary considerably, much time might be wasted during a GTS for large SO. In that sense, it is necessary to carefully define the value SO that optimally fits the traffic demand of the network. Energy consumption during idle time of a GTS can be reduced by making the transmitter announce the end of the communication to the receiver. Then, both communication partners can turn off their transceivers for the remaining part of the GTS. This idea was implemented and verified as part of a student thesis on the basis of Information Elements (IEs) and produced good results [Die20].

All in all, the transmission of multiple packets per GTS reduces the number of slot allocations and increases the reactivity of DSME with bursty and dynamically changing traffic, e.g., in 6LoWPAN applications. A theoretical estimation is given, and results are verified by simulation. They indicate that at $\epsilon = 30\,\mathrm{packet(s)}$, it is possible to maintain a PDR of 97 % with $SO = 6$ in comparison to the transmission of a single packet per GTS ($SO = 3$) with a PDR of 40 %. Also, the required number of management messages is reduced by up to 99 %.

Group Acknowledgments
Section 6.2

Sect. 6.1 proposes and analyzes sending multiple packets per GTS to minimize the overhead of required management messages for GTS (de)allocations. Every packet is acknowledged individually imposing a significant overhead in terms of throughput and energy consumption.

To stretch the time a GTS can be used for sending application data, the acknowledgment overhead can be reduced. This allows for transmission of more data packets per GTS and reduces energy consumption. In the simplest form, data packets sent in a GTS are aggregated and acknowledged in a so-called block acknowledgment at the end of a GTS. A similar variant is proposed in [BBVC14], where block acknowledgments are only issued on request of the data source. Block acknowledgments are well researched and have already proven to increase throughput in a trade-off for memory overhead, e.g., as part of the IEEE 802.11n standard [IEE21a]. They have yet to be verified in DSME, but their benefits are apparent. Block acknowledgments can, however, be further optimized if data packets from multiple nodes are confirmed in a single broadcasted Group Acknowledgment (GACK) to free up even more time for data transmission. The IEEE 802.15.4 DSME 2012 standard defines such a GACK scheme, but it was removed in later iterations of the standard [IEE12]. Still, many works deal with GACKs and their optimization in DSME [BCL+20, SPW17, GAQ+17] as they are relevant in various fields of application like vehicle-to-vehicle communication [CXQ+18], and have proven to be valuable in other protocols like LoRa [LKZK21, ZAKP20] and LLDN [IEE12].

Naturally, one question arises which we attempt to answer in the following sections: Are GACKs worth anything in DSME? More precisely, it shall be assessed whether GACKs constitute a viable alternative to regular ACKs in data collection scenarios requiring high reliability. To answer this question, three novel possibilities to broadcast GACKs are proposed and investigated:

- In network beacons (GACK-Beacon).
- As management messages during the CAP (GACK-CAP).
- In dedicated GTS (GACK-GTS).

Partly previously published in "Meyer, F., Malessa, P., Diercks, J., Turau, V.: Are Group Acknowledgements Worth Anything in IEEE 802.15.4 DSME: A Comparative Analysis. In: 2022 5th Conference on Cloud and Internet of Things (CIoT'22). Accepted for publication, Marrakech, Morocco (2022)." [MMDT22].

The following sections provide an overview of the proposed GACK schemes (Sect. 6.2.2), and state hypotheses about their expected performance (Sect. 6.2.3). Results from a theoretical (Sect. 6.2.4) and simulative evaluation (Sect. 6.2.5) follow and all schemes are validated on hardware (Sect. 6.2.6). A discussion concludes this section (Sect. 6.2.7).

6.2.1 Related Work

Gomes et al. propose a hybrid channel hopping and channel adaptation scheme for DSME called H-DSME [GAQ+17]. They identify the problem that GACKs are always transmitted on the same channel, which can potentially exhibit poor quality. Thus, H-DSME introduces a channel hopping mechanism for GACKs, where channels are selected using a round-robin mechanism. They compare H-DSME with two other techniques they propose, but their results do not indicate the performance of GACKs in comparison to regular ACKs.

DSME's reliability and end-to-end delay using GACKs according to the IEEE 802.15.4e 2012 standard are compared to TSCH by Alderisi et al. in realistic and increasingly dense process automation environments [APMB15]. DSME yields a higher delay than TSCH for networks with less than 30 nodes but outperforms TSCH at 50 nodes, where GACKs had to be disabled to increase throughput and allow every node to allocate sufficiently many GTS. Alderisi et al. conclude that DSME's GACKs, where packets are retransmitted at the end of superframes, severely increases delay of regular data transmissions. Their results provide a good indication of the inflexibility of GACKs according to the IEEE 802.15.4 2012 standard.

Sahoo et al. propose a GACK scheme for DSME where GACKs are transmitted as parts of regular beacons limiting their use to coordinators [SPW17]. Additionally, they devise a retransmission scheme where lost packets are retransmitted in the CAP without using CSMA/CA, thus, reducing delay. The details of their work are further discussed in Sect. 6.2.1.2.

Similar to [SPW17], Battaglia et al. propose a novel ACK scheme based on the periodicity of traffic flows to prevent the retransmission overhead of GACKs according to the IEEE 802.15.4 2012 standard [BCL+20]. In DSME, a GTS is repeated every MSF even if a flow has a period of n MSFs, resulting in $n-1$ unused GTS. Their idea is to utilize these unused GTS to transmit $n-1$ replicas of a data message in the flow, giving the receiver $n-1$ additional opportunities to receive a message correctly. Obviously, the proposed scheme increases reliability though redundant information but greatly increases energy-consumption.

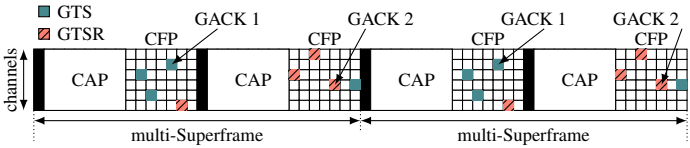


Figure 6.6: DSME's GACK scheme as defined by IEEE 802.15.4e 2012.

6.2.1.1 GACKs According to IEEE 802.15.4e 2012

In the 2012 amendment to the IEEE 802.15.4 standard [IEE12], GACKs were defined as part of DSME. Here, every coordinator allocates two GTS for the transmission of GACKs (GACK 1 and GACK 2) and announces their SF ID, slot ID, and channel ID in its enhanced beacons. Thereby, all packets received before GACK 1 but after GACK 2 are acknowledged in GACK 1 while all packets received after GACK 1 but before GACK 2 are acknowledged in GACK 2. As shown in Fig. 6.6, all GTS before GACK 1 are used for regular data transmission but for every allocated GTS another so-called *GTS for retransmission* (GTSR) must be allocated between GACK 1 and GACK 2. These allow for retransmission of lost data packets. IEEE 802.15.4e specifies the frame format for a GACK, in which the reception status of a packet is indicated by a single bit in a bitmap. The size of the bitmap can be configured in bytes using 2 bit, for a maximum size of 4 B, i.e., 32 acknowledgeable packets.

The described GACK scheme has are several shortcomings:

- Only coordinators can utilize GACKs because information about them is disseminated in their beacons.
- Many GTSR are unused if data packets are transmitted successfully, reducing potential throughput [SPW17].

6.2.1.2 GACKs According to Sahoo et al.

Due to the shortcomings of IEEE 802.15.4e's GACK scheme, Sahoo et al. propose a scheme where GACKs are transmitted as part of beacon messages [SPW17]. For this, every beacon is extended with a 7 bit bitmap, where every bit indicates the successful transmission during a respective CFP slot, of which there are 7 CFP slots without CAP-R. On reception of a GACK, every node counts the number l of lost packets which are then retransmitted in the first l CAP slots without using CSMA/CA. During

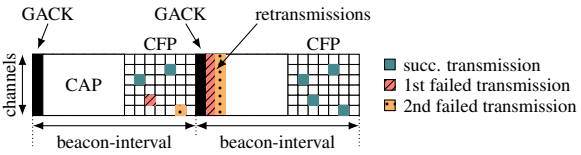


Figure 6.7: GACK scheme for DSME according to [SPW17].

this time, no other node is allowed to transmit packets. Afterwards, all nodes utilize the CAP normally. The operation of the scheme is depicted in Fig. 6.7.

While Sahoo's scheme significantly increases throughput, it comes with drawbacks, e.g., a fixed bitmap size of 7 bit only allows for $SO = MO = BO$, i.e., exactly seven CFP slots per BI, and CAP-R is unusable. Similar to the IEEE 802.15.4 standard, GACKs are limited to coordinators only, disallowing upstream traffic. Additionally, retransmitting packets in the CAP can become a problem in scenarios with fluctuating traffic where many management messages are transmitted.

6.2.2 Proposed Group Acknowledgement Schemes

In IEEE 802.15.4 DSME 2012, GACKs were defined as part of the official standard [IEE12]. They were removed in later iterations but their potential to increase DSME's throughput and energy-efficiency in reliable environments has yet to be fully determined. Several works propose new GACK schemes achieving auspicious results [SPW17, BCL+20]. The following sections provide an overview of the GACK schemes proposed in this work, i.e., GACK-Beacon, GACK-CAP, and GACK-GTS[1] (Sects. 6.2.2.1 to 6.2.2.3), and contrast them with existing schemes from literature.

In general, GACKs avoid the overhead of an acknowledgment and an $AIFS$ for every data transmission, as illustrated in Fig. 6.8. This allows for more data transmissions within a single GTS and also avoids switching the transceiver from transmitting to receiving after every data transmission.

6.2.2.1 Transmission through Beacons

Although the scheme by Sahoo et al. (see Sect. 6.2.1.2) significantly increases throughput, it yet introduces several drawbacks. The fixed bitmap size of seven GTS only allows for $SO = MO = BO$, i.e., exactly seven GTS per BI [SPW17]. Consequently,

[1] P. Malessa and J.N. Diercks provided basic implementations for GACK-GTS and GACK-CAP as part of their theses, which were extended by the author.

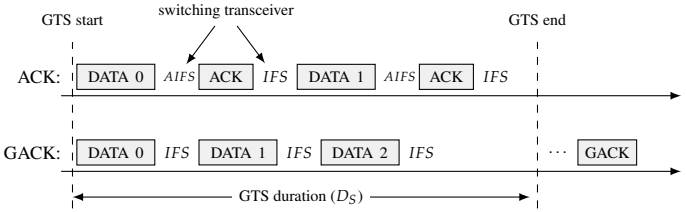


Figure 6.8: Transmission process using GACKs and regular ACKs within a single GTS with a MAC payload of $S = 1$ B.

CAP-R is not usable. Similar to IEEE 802.15.4 2012, GACK transmissions are limited to coordinators, disallowing upstream traffic to leaves of the network. At last, packet retransmission in the CAP is problematic in scenarios with fluctuating traffic where secondary traffic is oversaturated.

Nevertheless, the idea of transmitting GACKs through beacons seems promising, as it does not restrain throughput in the CFP. Thus, we adopt this idea but combine it with a more flexible bitmap format, discussed in Sect. 6.2.2.4. In addition, retransmission are handled through regular GTS instead of utilizing the CAP. This way, *SO*, *MO*, and *BO* can be selected without any restrictions. In the following, we refer to this scheme as GACK-Beacon.

6.2.2.2 Transmission during the CAP

DSME's CAP is primarily utilized for transmitting secondary traffic, i.e., management and broadcast messages. Thus, its utilization in a settled network with stable environmental conditions is usually low. The idea of GACK-CAP is simply to send GACKs as regular CAP messages. The main advantage is that no acknowledgment overhead is imposed during the CFP, maximizing potential throughput. On the other hand, the scheme suffers from the same disadvantages as CSMA/CA, i.e., low reliability and high delays in dense networks with fluctuating traffic conditions and many hidden-node problems (see Sect. 5.1). Similar to GACK-Beacon, retransmissions of data packets are performed in regular GTS.

6.2.2.3 Transmission in GTS

The GACK-GTS scheme is not only a generalization but also a simplification of the GACK-IEEE scheme, discussed in Sect. 6.2.1.1. The main idea is to allocate dedicated

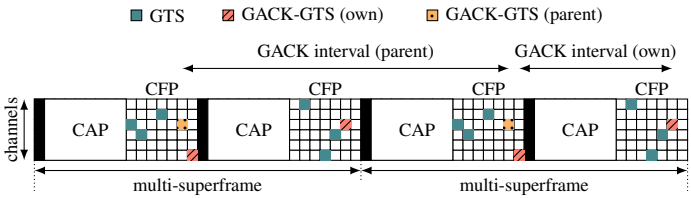


Figure 6.9: Example of the frame structure using the GACK-GTS scheme.

GTS during the CFP in which GACKs can be broadcasted to nodes in the neighborhood. The frequency of these GACK-GTS, i.e., the GACK interval is determined by the *group acknowledgment order* (GAO), where 2^{MO-GAO} gives the number of GACK-GTS per MSF. An example of the frame structure using GACK-GTS is shown in Fig. 6.9, where $GAO = MO$ applies at a parent node and $GAO - 1 = MO$ locally.

Allocation of a GACK-GTS is conducted with DSME's 3-way handshake using an additional flag, i.e., when a node sends a *GTS request* to allocate a regular GTS. The receiver of the *GTS request* checks if there is already a GACK-GTS within one GACK interval of the GTS to be allocated and either selects the next GACK-GTS or allocates a new GACK-GTS within one GACK interval. The GACK-GTS is communicated with the requesting node in the *GTS response* in a designated Information Element (IE). Thereby, GACK-GTS are randomly allocated from the tail of a SF while regular GTS are allocated randomly from the beginning to increase the odds of scheduling a GACK after the data transmission.

One should notice that during the described GACK-GTS allocation procedure, a node might return a GACK-GTS that conflicts with an already allocated GTS at the requesting node. There is no way to avoid such collisions as the GACK-GTS is communicated to several nodes and a change of the GACK-GTS would likely result in another collision. Thus, in case of a conflict, the requesting node may abort the handshake and try to allocate the GTS in another SF to retrieve another GACK-GTS. If this fails, it can relocate the conflicting GTS to make room for the GACK-GTS, e.g., using a *GTS-change* command.

6.2.2.4 GACK Bitmap Format

As described in Sect. 6.2.1.1, IEEE 802.15.4e 2012 defines a bitmap structure for acknowledgments through GACKs, which supports up to 32 ACKs. Especially in

	header	payload 1				payload 2	...
field	# payloads	node addr.	bitmap length	seq. number	bitmap	...	...
octets	1	2	1	1	*bitmap length*	...	...

Table 6.3: Generalized GACK bitmap structure utilized by GACK-Beacon, GACK-CAP and GACK-GTS.

scenarios with a large *SO*, where many packets can be transmitted per GTS (see Sect. 6.1), a larger and more flexible bitmap format is required.

The proposed bitmap format is shown in Tbl. 6.3. Unlike the GACK formats from Sects. 6.2.1.1 and 6.2.1.2, the bits in the bitmap do not directly address a GTS but acknowledge data packets relative to a specified sequence number. This has two benefits: First, the bitmap does not contain one bit per packet that is potentially sent, so it is smaller on average. Second, and more importantly, consistency problems are avoided because each bit is associated with a given sequence number. It would otherwise be impossible to determine which packet within a GTS failed. The GACK contains a *payload* for each node from which a packet was received. This payload consists of the *node address*, the *bitmap length*, the *sequence number* of the first packet to acknowledge, and a *bitmap* indicating the reception status of all packets in relation to the given sequence number.

Note that the proposed bitmap format is not yet optimal and its size can be further compressed with additional effort, e.g., by omitting the *# payloads* field [CXQ+18]. However, the format is flexible and can be easily processed by sensor nodes with limited resources, making it well suited for the evaluated scenarios.

6.2.3 Comparison and Hypotheses

Several metrics can be utilized to appraise the performance of GACKs in DSME, as also indicated in Sect. 3.2. However, due to the diversity of the presented GACK schemes, it is hardly possible to derive general statements about their performance. Therefore, Tbl. 6.4 tries to provide insights into their expected capacity for selected metrics.

In general, all GACK schemes are expected to provide higher throughput than the normal ACK scheme, however, while GACK-CAP and GACK-Beacon achieve the maximum throughput, GACK-GTS's throughput is slightly lower because it requires at least one GTS per coordinator for the transmission of GACKs. Dwell time is directly correlated to throughput because queues are reduced faster with

metric	ACK	GACK-Beacon	GACK-CAP	GACK-GTS
throughput	○	⊕⊕	⊕⊕	⊕
dwell time	○	⊕	⊕	○
ACK delay	○	⊖⊖	⊖⊖	⊖
energy consumption	○	⊕	⊕⊕	⊕
memory overhead	○	⊖	⊖	⊖

Table 6.4: Hypotheses about the performance GACK-Beacon, GACK-CAP, and GACK-GTS in DSME in relation to regular ACKs.

higher throughput. GACK-Beacon and GACK-GTS exhibit a variable ACK delay by tuning BO and GAO, respectively. GACK-CAP provides a low expected ACK delay of half the number of GTS per SF, which also constitutes the lower bound for GACK-Beacon. GACK-GTS can reach a lower ACK delay for a direct trade-off with throughput. It should be noted that the ACK delay plays a subordinate role in reliable environments and is only relevant in case of packet loss. For all GACK schemes, the energy consumption is inversely proportional, and memory overhead is proportional, to the ACK delay since more data packets are received and have to be stored in the GACK-bitmap before a GACK is transmitted. In particular, GACK-CAP requires the fewest memory while more is required for GACK-CAP and GACK-Beacon in its default configuration. All GACK schemes are expected to require less energy and more memory than regular ACKs. Another benefit of GACKs is that a time-critical task, i.e., the acknowledgment of packets is moved to a less utilized phase, e.g., beacon slots.

6.2.4 Theoretical Evaluation

Depending on the scenario, the goal of the MAC layer is to increase the possible throughput for the application - either in terms of transmittable packets or in terms of goodput. Thus, the following two sections, Sects. 6.2.4.1 and 6.2.4.2, provide theoretical estimations of the maximum packet throughput and maximum goodput for different ACK schemes. Together, they allow assessment of the performance gain of GACKs over regular ACKs.

6.2.4.1 Maximum Throughput

As mentioned in Sect. 6.2.3, GACKs increase the maximum number of transmittable packets per GTS. That is due to two reasons: First, there is no need to transmit an ACK in response to every data packet. Furthermore, the IEEE 802.15.4 standard requires an

IFS for processing after every received data packet. It includes an *aTurnaroundTime* for switching the transceiver from the RX to the TX state and vice versa. However, this is unnecessary if no ACK is transmitted.

The expected maximum throughput in packets per second with ACKs, TP_A, and with GACKs, TP_G, for a given SO and payload S in bytes is given by

$$TP_A = \frac{\mathcal{S}_{sec}}{\mathcal{S}_{SF}} * GTS_{SF} * \lfloor \frac{\mathcal{S}_{GTS}}{\mathcal{S}_A(S)} \rfloor, \tag{6.2}$$

$$TP_G = \frac{\mathcal{S}_{sec}}{\mathcal{S}_{SF}} * GTS_{SF} * \lfloor \frac{\mathcal{S}_{GTS}}{\mathcal{S}_G(S)} \rfloor, \tag{6.3}$$

where $\mathcal{S}_{sec} = 62\,500\,\text{symbol(s)}$ is the constant symbol rate, $GTS_{SF} = 7\,\text{GTS}$ is the number of GTS per SF, $\mathcal{S}_{GTS} = 60 * 2^{SO}$ symbol(s) is the number of symbols per GTS and $\mathcal{S}_{SF} = 16 * \mathcal{S}_{GTS}$ is the number of symbols per SF. Finally, the functions $\mathcal{S}_A(S)$ and $\mathcal{S}_G(S)$ compute the number of symbols per data transmission for a given S with and without ACK

$$\mathcal{S}_A(S) = \mathcal{S}_H + 2S + \mathcal{S}_{ACK} + \mathcal{S}, \tag{6.4}$$

$$\mathcal{S}_G(S) = \mathcal{S}_H + 2S + \mathcal{S} - aTurnaroundTime. \tag{6.5}$$

Here, S is the payload length in bytes, $\mathcal{S}_{ACK} = 34\,\text{symbol(s)}$ is the overhead for the ACK packet and an $AIFS$, and $\mathcal{S}_{IFS}$ is either a $SIFS$ with 12 symbol(s) or a $LIFS$ with 40 symbol(s) if $S > 18\,\text{B}$. At last, $\mathcal{S}_H = 34\,\text{symbol(s)}$ symbols is the header overhead for the data packet.

Fig. 6.10 shows the expected maximum throughput of GACKs (TP_G) in comparison to regular ACKs (TP_A) for different values of S and $MO = 8$. As one can see, the number of transmittable packets can be significantly increased for small S using the proposed GACK schemes, while for large S no significant increase is achievable. Over all SO, the expected maximum throughput for a given S is almost constant as a SF with a shorter length is repeated multiple times per second. As anticipated, the expected maximum throughput of GACK-IEEE is significantly lower than for the other techniques because twice as many GTS have to be allocated for every transmission. GACK-IEEE only outperforms regular ACKs for $S = 1\,\text{B}$, i.e., in a highly unrealistic scenario.

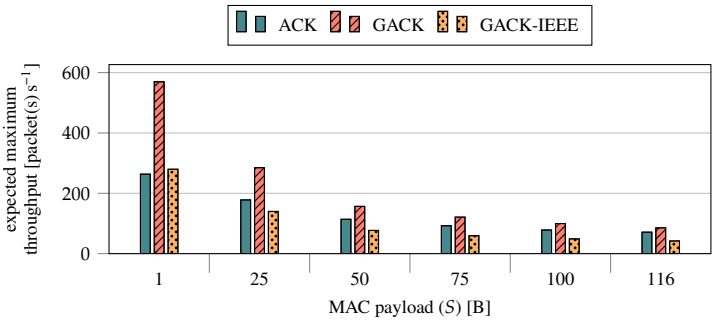


Figure 6.10: Expected maximum throughput in packet(s) s^{-1} using GACKs and regular ACKs for increasing payload lengths S.

6.2.4.2 Maximum Goodput

Contrasting Sect. 6.2.4.1, an application is usually not only interested in the raw packet throughput but rather in the goodput, i.e., the total application layer payload the MAC protocol can transmit per second. Thereby, the goodput is adjusted for header, acknowledgment, and management overhead, as discussed in Sect. 3.2.

The expected maximum goodput per second with ACKs, GP_A, and with GACKs, GP_G, in bytes is calculated similarly to Eqs. (6.2) and (6.3) in Sect. 6.2.4.1 as

$$GP_A = \frac{\mathcal{S}_{sec}}{\mathcal{S}_{SF}} * GTS_{SF} * G_A, \tag{6.6}$$

$$GP_G = \frac{\mathcal{S}_{sec}}{\mathcal{S}_{SF}} * GTS_{SF} * G_G, \tag{6.7}$$

where G_A and G_G are the goodputs per GTS with and without ACKs, respectively. They are calculated as

$$G_A = 116 * \lfloor \frac{\mathcal{S}_{GTS}}{\mathcal{S}_A(116)} \rfloor + max\{0, \frac{(\mathcal{S}_{GTS} \mod \mathcal{S}_A(116)) - \mathcal{S}_A(0)}{2}\}, \tag{6.8}$$

$$G_G = 116 * \lfloor \frac{\mathcal{S}_{GTS}}{\mathcal{S}_G(116)} \rfloor + max\{0, \frac{(\mathcal{S}_{GTS} \mod \mathcal{S}_G(116)) - \mathcal{S}_G(0)}{2}\}, \tag{6.9}$$

with the first part calculating the goodput of packets with a maximum payload of 116 B and the second part calculating the maximum payload of a packet that can be sent in the remaining free symbols of a GTS.

Fig. 6.11 shows the expected maximum goodput of GACKs (GP_G) and regular

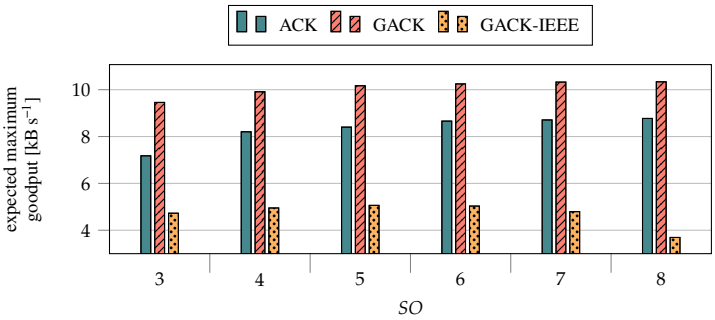


Figure 6.11: Expected maximum goodput using regular ACKs and GACKs.

ACKs (GP_A) in bytes for different SO and $MO = 8$. The expected goodput using the proposed GACK schemes is between 30 % for $SO = 3$ and 17 % for $SO = 8$ higher than for regular ACKs. For $SO > 8$, the performance difference between both schemes is constant. GACK-IEEE reaches a significantly lower goodput than the other schemes because two slots have to be allocated for every transmission. Additionally, the goodput decreases for a decreasing difference between MO and SO because two GTS are always allocated for GACK 1 and GACK 2, and hence unusable for regular data transmission.

6.2.5 Simulative Evaluation

To determine whether GACKs are feasible alternative to regular ACKs, the GACK schemes are compared in three scenarios that are described in Sect. 6.2.5.1. These constitute a worst-case, best-case, and average-case scenario for the GACKs for which results are provided in Sects. 6.2.5.2 to 6.2.5.4, respectively. Together, these scenarios create a holistic picture of the GACKs' performance and allow to identify promising application scenarios.

6.2.5.1 Scenario Description

GACKs are designed for dense networks where multiple nodes are acknowledged at once with a single GACK. Thus, a star topology, as shown in Fig. 6.12, constitutes a best-case scenario and the opposite, a line topology, constitutes a worst-case scenario because each node only communicates with its child and parent. Multiple packets of a single node can still be acknowledged at once. For the best-case scenario, a small

scenario	*SO*	*MO*	*BO*	*GAO*	topology	nodes	*S*
worst-case	3	6	8	6	line	10	116 B
best-case	4	7	7	7	star	19	1 B
average-case	4	6	8	6	tree	31	1 – 116 B

Table 6.5: Parameter for the evaluation of GACKs in a worst-case, best-case, and average case scenario.

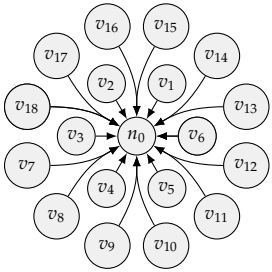


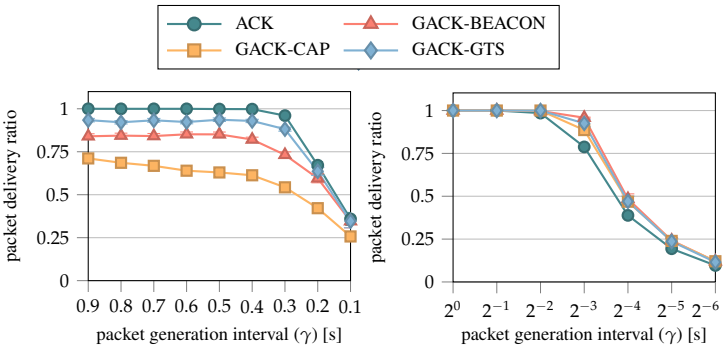
Figure 6.12: Star topology for the best-case analysis of GACKs.

payload of 1 B is chosen and *MO* and *SO* are selected in a way that as many packets as possible are acknowledged using a single GACK. For the worst-case scenario, a single packet with maximum payload is transmitted per GTS, impeding the usefulness of GACKs even further. Packets are generated according to a Poisson distribution.

At last, GACKs are evaluated in an average-case scenario, which we exemplary choose as a binary-tree topology with 31 nodes. Here, packets are generated with a random payload between 1 B and 116 B to reflect common use cases. It should be noted that a binary-tree exhibits many hidden-node problems (similar to the line topology) but also offers the possibility to acknowledge two nodes using a single GACK. Tbl. 6.5 summarizes the relevant configuration parameters, and Tbls. A.5 to A.7 shows the full configuration.

6.2.5.2 Worst-case Analysis

Fig. 6.13a shows the PDR using different GACK schemes and regular ACKs for decreasing packet generation intervals γ. As one can see, regular ACKs perform best while GACK-CAP performs worst. That is because GACKs are transmitted in the CAP using CSMA/CA which suffers from hidden-node problems so that GACKs are frequently lost. GACK-Beacon requires $BO \geq 8$ to enable sufficiently many beacon



(a) Worst-case scenario.

(b) Best-case scenario.

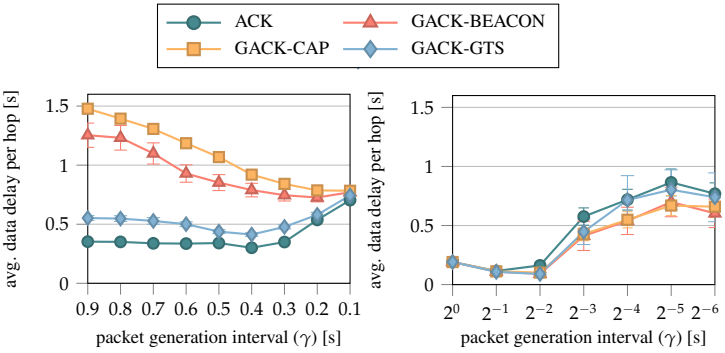
Figure 6.13: Average PDR for a decreasing packet generation interval γ with GACK-Beacon, GACK-CAP, GACK-GTS in comparison to regular ACKs.

slots for all nodes to join the network. However, with a maximum queue length of $Q_{CFP} = 22\,\text{packet(s)}$ in openDSME, many packets are dropped because packets are not acknowledged fast enough. At last, GACK-GTS requires one GTS for GACKs per node so that less GTS can be used for regular data packets.

The average data packet delay per hop (see Sect. 3.2) is shown in Fig. 6.14a. As expected, the delay of GACK-CAP and GACK-Beacon is high because GACKs frequently collide using the former scheme and the ACK interval is to high using the latter scheme. GACK-GTS performs slightly worse than regular ACKs due to the GTS overhead.

Fig. 6.15a shows the average ACK delay, i.e., the time from transmitting a data packet to receiving the according ACK. GACK-Beacon and GACK-GTS provide the worst performance due to their large acknowledgment interval, however, the delay of GACK-GTS is unstable for $0.4\,\text{s} \leq \gamma \leq 0.9\,\text{s}$. That is because a GACK-GTS is allocated through the regular GTS-handshake. Due to hidden node problems many of these handshake messages fail and thus the allocation of GACK-GTS is delayed.

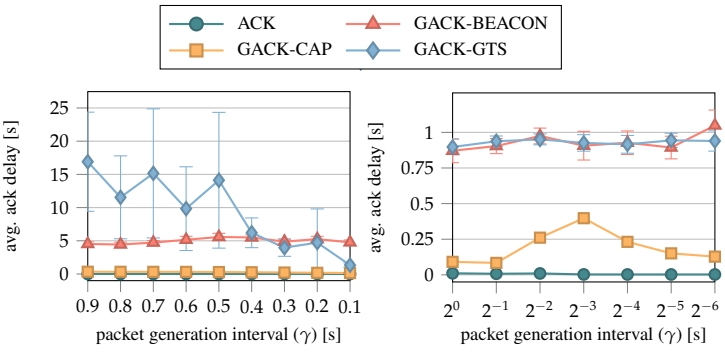
At last, Fig. 6.16a depicts the PDR of the different ACK schemes in a more realistic scenario, i.e., under external interference of a packet with maximum payload $S = 127\,\text{B}$ and varying interference interval. Interference lasts for about 5 ms. For sake of simplicity, the interference occurs on all channels simultaneously and $\gamma = 1\,\text{s}$ is fixed. The relation between the ACK schemes is similar as before with one



(a) Worst-case scenario.

(b) Best-case scenario.

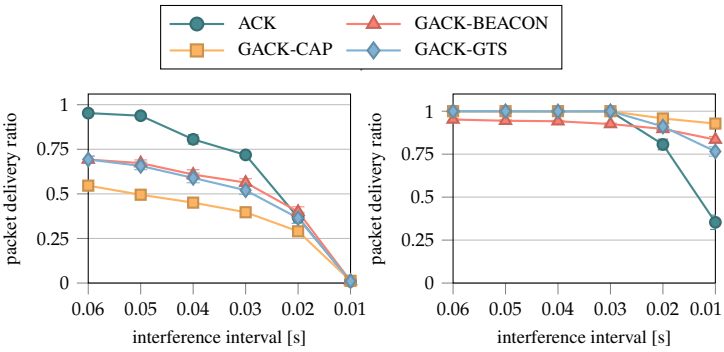
Figure 6.14: Average end-to-end delay of data packets sent during the CFP for decreasing γ. The end-to-end delay of each node is normalized by the number of hops towards the sink.



(a) Worst-case scenario.

(b) Best-case scenario.

Figure 6.15: Average ACK delay, i.e., the time between sending a data packet and receiving the according acknowledgment, for a decreasing packet generation interval γ.



(a) Worst-case scenario.

(b) Best-case scenario.

Figure 6.16: PDR of GACK-Beacon, GACK-CAP, GACK-GTS, and regular ACKs under external interference of a packet with maximum S and decreasing interval.

exception: GACK-Beacon performs slightly better than the other schemes, including regular ACKs, for an interference interval of 20 ms and lower.

6.2.5.3 Best-case Analysis

Fig. 6.13b shows the PDR in a best-case scenario. Notice the changed axis scaling compared to the worst-case scenario. The PDR of regular ACKs decreases slightly faster for decreasing γ than that of the GACK schemes. All GACK schemes perform similarly, where GACK beacon performs best because the beacon interval is low and there is no additional overhead during the CFP. The same applies to the average data packet delay, depicted in Fig. 6.14b. Regular ACKs result in the highest delay because fewer packets can be transmitted per GTS and hence packets remain in the queue for longer. GACK-CAP and GACK-Beacon achieve the lowest delay, where GACK-GTS introduces a slightly higher delay due to the GTS allocation through the 3-way handshake. At last, the acknowledgment delay in Fig. 6.15b behaves similar to the worst-case scenario. However, one should notice the different scaling of the y-axis.

Under the influence of external interference, more packets can be transmitted successfully using GACKs for small γ, as presented in Fig. 6.16b. That is because more packets can be transmitted per GTS so that a single lost packet has less influence.

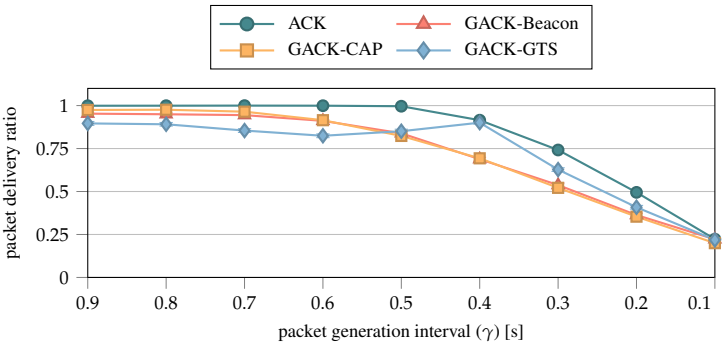


Figure 6.17: PDR for GACK-Beacon, GACK-CAP, GACK-GTS, and regular ACKs for a decreasing packet generation interval γ in a binary-tree topology.

In general, the performance difference between regular ACKs and GACKs is smaller than in the worst-case scenario, discussed in Sect. 6.2.5.2, indicating that GACKs can provide improvements in optimal scenarios but also impose great risks as their worst-case performance is significantly worse than the performance of regular ACKs.

6.2.5.4 Average-case Analysis

Fig. 6.17 shows the PDR of the different GACK schemes and regular ACKs for decreasing γ. Here, regular ACKs achieve the highest PDR while the performance of the GACK schemes highly depends on γ. GACK-GTS suffers from the initial overhead of one GTS per node for sending GACKs and hence only achieves a PDR of about 90 % for $0.4\,\text{s} < \gamma \leq 0.9\,\text{s}$. However, in comparison to GACK-CAP and GACK-Beacon which for high γ achieve PDRs of 97 % and 98 %, respectively, GACK-GTS manages to maintain the same PDR for higher γ. The performance of GACK-CAP and GACK-Beacon falls off because of increased management traffic in the CAP and a too large acknowledgment interval using beacons. All GACK schemes perform worse than regular ACKs.

On the other hand, GACK-GTS results in the lowest data delay and even outperforms regular ACKs, as shown in Fig. 6.18 because more packets can be transmitted per GTS. GACK-Beacon and GACK-CAP exhibit the highest delay for lower γ because less packets are transmitted successfully and the queues start to fill up.

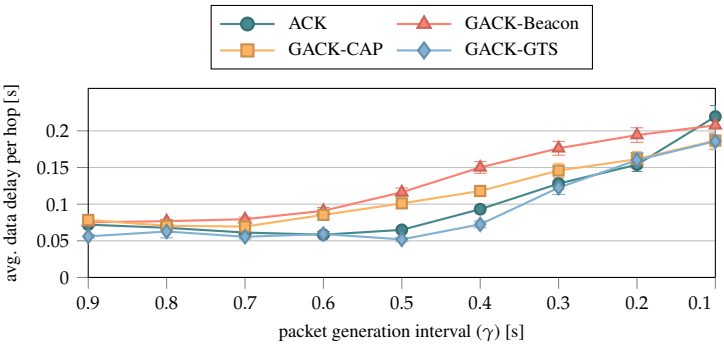


Figure 6.18: Average end-to-end delay of data packets sent during the CFP for decreasing γ. The end-to-end delay of each node is normalized by the number of hops towards the sink.

The acknowledgment delay, depicted in Fig. 6.19, behaves similar to the best-case and worst-case scenarios, where regular ACKs achieve a minimal acknowledgment delay while GACK-CAP acknowledges packets after every SF and thus also achieves a relatively low delay. For the given network a high *BO* is required so that the acknowledgment delay of GACK-Beacon is high. However, one should be aware that the acknowledgment delay is only relevant if packets are lost and retransmitted. Also, the acknowledgment delay contributes to the data packet delay, i.e., the delay of retransmitted data packets includes the acknowledgment delay.

6.2.6 Verification on Hardware

To verify the applicability of GACKs in a scenario with physical nodes, we conduct energy measurements in the *Strasbourg* testbed of the FIT IoT-LAB [ABF+15]. Sect. 6.2.6.1 explains the setup and Sect. 6.2.6.2 states the results of the energy measurements.

6.2.6.1 Scenario Description

Experiments are conducted in a tree topology with ten nodes and depth three, as shown in Fig. 6.20, generated by the tree selection algorithm proposed in [Kau19]. According to [Kau19], transmission power was configured to −3 dBm and transceiver sensitivity to −60 dbm. Tbl. 6.6 lists the relevant configuration parameter.

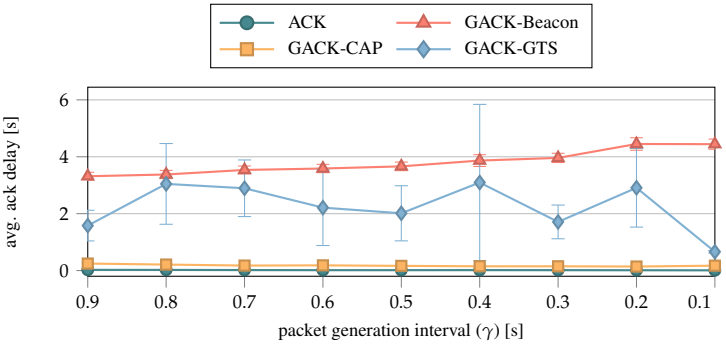


Figure 6.19: Average ACK delay, i.e., the time between sending a data packet and receiving the according acknowledgment, for a decreasing packet generation interval γ.

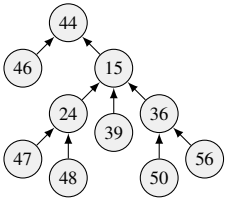


Figure 6.20: Tree topology utilized for experiments in the FIT IoT-LAB. The numbers indicate the node IDs in the Strasbourg testbed.

SO	*MO*	*BO*	*GAO*	topology	nodes
5	6	8	7	tree	10
S	**distribution**	γ	**packets**	**repetitions**	
116 B	Poisson	0.5 s	1000	10	

Table 6.6: Parameter for the evaluation of GACKs on hardware.

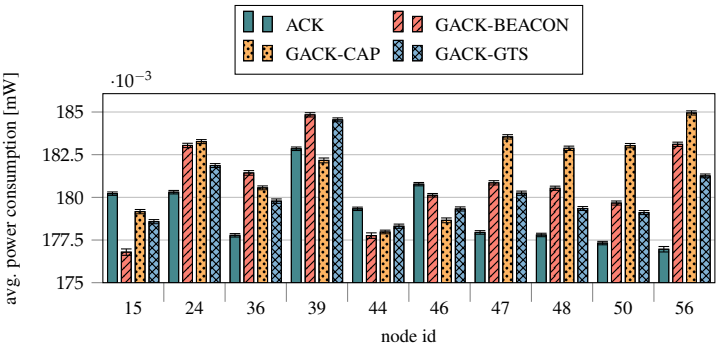


Figure 6.21: Average power consumption for nodes in a tree topology in the Strasbourg testbed of the FIT IoT-LAB.

6.2.6.2 Power Consumption

Fig. 6.21 shows the average power consumption of all nodes in the tree topology, depicted in Fig. 6.20. Opposing our initial hypothesis, there is no significant difference in the measured power consumption of GACKs and regular ACKs. This is due to two reasons:

- In high-traffic scenarios, GACKs are likely to transmit additional data packets per GTS This, however, offers no benefit over transmitting ACKs because the switching power consumption is close to the transmission power consumption.
- In low-traffic scenarios, power consumption is dominated by the CAP which makes up 56.25 % of a SF. During the CAP, the receiver has to be turned on to exchange secondary traffic. Therefore, saving only a few ACKs during a GTS does not constitute significantly to power consumption.

6.2.7 Discussion and Conclusion

In summary, one has to say that GACKs do not provide significant benefits in terms of throughput, data packet delay, and ACK delay in the tested scenarios to justify their general usage over regular acknowledgments. They can provide small performance increases in terms of throughput and reliability in optimal environments, e.g., in star topologies where many nodes can be acknowledged at once. On the other hand, they suffer from significant performance loss in comparison to regular ACKs in all but

optimal scenarios. Regular ACKs prove to be superior in environments with strong interference because packets have to be buffered longer due to retransmissions using GACKs. As a consequence, frequent packet drops occur when the queue overflows. Thus, coming back to the main question from Sect. 6.2, we could not identify GACKs as a valid alternative to regular ACKs in scenarios other than a star topology. Even by transmitting GACKs in multiple ways, no solution was able to verify results from literature [SPW17, APMB15], where GACKs are handled as a valid alternative to regular ACKs. Nevertheless, one should keep in mind that the proposed schemes only consider different ways of broadcasting GACKs, which can likely be augmented with additional techniques to a degree that they are beneficial in specific application scenarios.

Above statement is reinforced by the high implementation effort for GACKs which require additional message definitions, alteration of the transmission mechanism, and adapted GTS and transceiver management, i.e., large deviations from the standard. Furthermore, GACKs impose a large memory overhead for storing data packets until they are acknowledged. Consequently, we conclude that the achieved performance benefits of GACKs are usually not worth the additional implementation effort. At last, it has been established that GACKs work best in single-hop networks (star topologies) with many nodes. DSME is explicitly designed for scalable multi-hop networks and such scenarios keep DSME from performing best by definition, limiting the effectiveness of GACKs in DSME even further. If a GACK scheme should be used, GACK-CAP offers the best trade-off between throughput and acknowledgment delay but heavily suffers from hidden node problems. In scenarios with many hidden node problems, GACK-GTS offers a valid alternative.

The obtained results and findings from literature [BBVC14], indicate that block acknowledgments are a more promising alternative to ACKs than GACKs. Thus, future work can evaluate their feasibility in DSME.

Chapter 7

Scheduling for Primary Traffic

Besides measures to accelerate the slot allocation procedure, as detailed in Chaps. 5 and 6, the design of an efficient distributed scheduler contributes considerably to the agility of a MAC protocol. Yet the maximum achievable performance of such schedulers is often obscure. Therefore, the following two sections formalize the scheduling problem for systems with alternating primary and secondary traffic phases through Mixed-integer Linear Programs (MILPs) (Sect. 7.1) and Quadratic Unconstrained Binary Optimizations (QUBOs) (Sect. 7.2). One focus is delay-minimal scheduling, where the resulting schedules can be used as baselines for dynamic schedulers.

Specifically, two dynamic distributed schedulers - TPS and LLSF - are compared with the formal definition in terms of reliability and delay. The results demonstrate that both are suitable for scheduling in the IIoT, but still have potential for further improvement. The computation via ILPs has a high computational complexity, hence SA is tested as an alternative and achieves promising results with respect to runtime.

Scheduling through Integer Linear Programming
Section 7.1

For high agility, MAC protocols require powerful distributed schedulers that optimize a variety of predefined metrics. In this context, it can be expected that distributed scheduling algorithms exhibit lower performance than centralized scheduling, since no complete knowledge about the network is available. Incoming traffic is therefore slightly overestimated by distributed scheduling algorithms to prevent packet loss. Nevertheless, there is often no baseline for the maximum achievable performance, which makes it difficult to assess how well a scheduling algorithm performs and whether a better algorithm can be found.

While sensors in periodic monitoring record data at constant time intervals, data acquisition in event-based monitoring is triggered by external events. The exact timing of such events is rarely known, but often an immediate reaction from the application is required to avoid potential damage. Therefore, the data needs to be transmitted to the application as reliably as possible and within a given time limit. While scheduling of time slots is a well-researched topic [ASLM12, CMGL05], so far little attention has been paid to scheduling for DSME. It requires different scheduling strategies because it divides time into the CAP for secondary traffic and the CFP for primary traffic, and makes use of channel diversity.

The rest of this section is structured as follows: After an overview of related work in Sect. 7.1.1, scheduling requirements are introduced in Sect. 7.1.2, formally describing the scheduling problem. Next, Sect. 7.1.3 discusses the optimal parameter selection for DSME, and Sect. 7.1.4 describes the formulation of scheduling strategies and optimization goals as Linear Programs (LPs). For this, Sect. 7.1.4.1 formulates base constraints for collision-free communication, and latter subsections introduce different optimization goals. Sect. 7.1.8 analyzes the schedules and presents the results.

7.1.1 Related Work

A lot of research has been done in the field of time slot scheduling [IGK10, PAD+12, OGAP19, HGT17], optimizing for a variety of metrics such as throughput [KKT18], end-to-end delay, energy-consumption [CMGL05], and minimal schedule length

Extension and revision of the previously published "Meyer, F., Turau, V.: Delay-Bounded Scheduling in IEEE 802.15.4e DSME Using Linear Programming. In: 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS). pp. 659–666. IEEE (2019). https://doi.org/10.1109/dcoss.2019.00119." [MT19].

[ASLM12]. Cui et al. prove that scheduling outgoing links after incoming links yields minimal delay under certain conditions in non-periodic data collection scenarios [CMGL05]. Additionally, they discuss the energy-delay trade-off depending on the number of scheduled slots. Few works also consider linear programming for scheduling. Amdouni et al. formulate a LP that minimizes the required schedule length and delay in multi-hop networks [ASLM12]. They calculate the minimum schedule length for a selection of topologies and present the Traffic-Aware Time Slot Assignment (TRASA) algorithm for centralized scheduling with priorities. However, frequency diversity is not considered in their work.

Recently, the growing popularity of 6TiSCH has led to increased efforts to standardize scheduling algorithms to enable interoperability and accessibility, resulting in the standardization drafts SF0/SFX [DGPA17, DGPA18] and SF1 [ALZ+18] - now combined in 6P-MSF [VVDD21]. The former is based on an approach from [PWW+16] and describes a distributed algorithm that adapts the number of scheduled slots to the traffic demands of the nodes. Timeslots are randomly selected. Therefore, Chang et al. present the LLSF which is based on SF0 and daisy-chains timeslots to reduce latency [CWWV16]. A similar approach is taken by Daneels et al. which define the RESF [DSLF18]. The idea is to allocate minimal-delay paths with respect to the arrival times of periodically generated packets. The exact timings are measured in a so-called housekeeping period before the schedule is built.

TPS is inspired by SFX and aims for maximum throughput and reliability [KKT18], as described in Sect. 2.6.1. For this, they filter the number of incoming packets with an exponentially weighted moving average to predict the future traffic load and apply a hysteresis for smoothness. Slots are allocated at random slot indices. They evaluate the performance of TPS in openDSME [KKT19a], and show its effectiveness in scenarios with strongly fluctuating traffic.

7.1.2 Scheduling Requirements

This section builds on and extends the system model from Sect. 3.1. For network communication, time is divided into an infinite sequence of frames, each of them subdivided into K time slots with duration D_S. Here, D_S is obtained from the parameter selection in section Sect. 7.1.3. Every node $v_i \in \mathbb{V}$ utilizes a single half-duplex transceiver and is synchronized to this frame-based structure.

The following criteria have to be met at every node $v_i \in \mathbb{V}$ for a successful transmission:

1. In each slot, v_i can either send a single packet to its parent N_i^+, or receive a single packet from a child $v_j \in N_i^-$, not both.

2. In a slot, v_i can only receive a packet from a single child $v_j \in N_i^-$. Several packets from different children collide and are corrupted.

3. If v_i sends a packet in a slot, none of the nodes $v_j \in \mathbb{I}_i$ can send a packet in the same slot. In particular, a child and the parent of a node v_i cannot transmit a packet in the same slot.

A schedule is a mapping of the slots of a frame to the nodes $v_i \in \mathbb{V} \setminus \{v_0\}$. A schedule is valid if it satisfies criteria 1 to 3. A valid schedule is called *δ-resilient* for data rate δ when all generated packets reach the gateway v_0 without buffer overflows. A necessary condition for a *δ-resilient* schedule is $\delta \leq {D_S}^{-1}$. A schedule has delay D, if each packet arrives at the root v_0 at most D time units after it was generated, similarly to the definition in Sect. 3.2. The following sections aim to develop *δ-resilient* scheduling mechanisms with respect to above requirements.

7.1.3 Parameter Selection

In order to design efficient scheduling mechanisms for DSME, suitable configuration parameters have to be found based on the routing tree T, packet generation rate δ, and expected payload length S. Several question have to answered:

1. What is the length of a GTS (parameter SO)?

2. What is the length of a MSF (parameter MO)?

3. Is there secondary traffic, i.e., can CAP-R be used?

4. Which slots and channels are assigned to which links?

Based on S, a suitable value for SO can be found using Tbl. 7.1. If S equals the maximal packet size given in Tbl. 7.1 then there is only little time to do data processing between two slots. If this is required, a larger value for SO must be chosen. Note that with increasing SO, the CAP's length also increases.

The energy consumption caused by a schedule is minimal if all assigned slots of the schedule are utilized for transmission. If the number of packets generated per MSF is not integral then the slots assigned to a leaf will not be used in every frame. Thus, an *δ-resilient* schedule requires $\delta \cdot D_S \cdot K$ to be integer. Such a schedule

	$SO = 1$	$SO = 2$	$SO = 3$	$SO = 1$	$SO = 2$	$SO = 3$
			$MO - SO = 1$			$MO - SO = 2$
D_S [ms]	1.92	3.84	7.68	1.92	3.84	7.68
D_{MSF} [ms]	61.44	122.88	245.76	122.88	245.76	491.52
max. payload size [B]	18	66	116	18	66	116
number GTS (per second)						
without CAP-R	14 (227)	14 (113)	14 (56)	28 (227)	28 (113)	28 (56)
with CAP-R	22 (358)	22 (179)	22 (89)	52 (423)	52 (211)	52 (105)
			$MO - SO = 3$			$MO - SO = 4$
D_S [ms]	1.92	3.84	7.68	1.92	3.84	7.68
D_{MSF} [ms]	245.76	491.52	983.04	491.52	983.04	1966.08
max. payload size [B]	18	66	116	18	66	116
number GTS (per second)						
without CAP-R	56 (227)	56 (113)	56 (56)	112 (227)	112 (113)	112 (56)
with CAP-R	112 (455)	112 (227)	112 (113)	232 (472)	232 (236)	232 (118)

Table 7.1: Characteristic values for the configuration of DSME.

does not necessarily exhibit minimal delay. Moreover, in DSME frame lengths are restricted to certain values. The choice of parameter MO hence strongly depends on the optimization target.

A DSME schedule only uses the GTS and not the slots belonging to a CAP. Thus, the number of CAPs per MSF has a strong influence on the delay of a schedule. If secondary traffic is low, CAP-R can be used. This increases the number of GTS by a factor of at least two (except for the cases $MO - SO \leq 1$), and can therefore reduce delay. The answer to the fourth question is answered in the following sections.

7.1.4 Scheduling as Optimization Problem

To formulate the scheduling task as an optimization problem, we assume that time slots are numbered in ascending order beginning with 0, and form a schedule with K slots. The CAP and beacon slots of DSME are not part of this schedule. In a first step we disregard the fact that these slots exist and that GTS are not all side by side. As a consequence, the schedules from Sects. 7.1.4.1 to 7.1.4.8 are generally applicable to any TDMA system, e.g., TSCH. Only Sect. 7.1.5 accounts for the CAPs between slots.

The goal is to find a *δ-resilient* schedule optimizing a given metric. A schedule is described by $|\mathbb{V}| - 1$ tuples of K binary variables x_{ik} with $i \in [1, |\mathbb{V}|)$ and $k \in [0, K)$ - one tuple for each node $v_i \in \mathbb{V} \setminus \{v_0\}$. The binary variable x_{ik} is defined as:

$$x_{ik} = \begin{cases} 1, & k \text{ is a transmission slot from } v_i \text{ to its parent } N_i^+ \\ 0, & \text{otherwise.} \end{cases}$$

That means, a node v_i can transmit a packet to its parent in slot k if and only if $x_{ik} = 1$. Also, $x_{ik} = 0$ indicates that v_i is in the receiving or idle state in slot k. A slot can be uniquely identified as a reception slot if $(x_{ik} = 0) \wedge (\sum_{j \in N_i^-} x_{jk} = 1)$ and as idle slot if $x_{ik} = \sum_{j \in N_i^-} x_{ik} = 0$. Let $x_{0k} = 0$ for $k \in [0, K)$. Next we show how the scheduling problem can be formulated as a MILP.

7.1.4.1 Collision-free Communication

All MILPs defined in the upcoming sections rely on some basic constraints to enforce collision-free communication in TDMA systems. In particular, the optimization problem can be formulate as an MILP with $(|\mathbb{V}| - 1) \cdot K$ binary variables x_{ik}.

$$\forall_{i,k}: \quad x_{ik} \in 0,1 \tag{7.1}$$

$$\forall_{i,k}: \quad x_{ik} + x_{N_i^+ k} + \sum_{j \in N_i^-} x_{jk} \leq 1 \tag{7.2}$$

$$\forall_{i}: \quad \gamma_i \cdot \delta \leq \sum_{k} x_{ik} \tag{7.3}$$

Thereby, constraint (7.2) ensures that either v_i, one of v_i's children, or its parent N_i^+ is allowed to send in any given time slot k to guarantee that no packet collisions can occur. Additionally, constraint (7.3) guarantees that the number of allocated slots is sufficient for a node's traffic demand. In other words, it ensures that the number of allocated slots at node v_i is sufficiently large to handle all incoming and self-generated packets. Here, δ is a constant defined by the application where $\delta \leq \delta_{max}$ applies - with δ_{max} being the maximum feasible packet generation rate per node to ensure that a valid schedule can be created. The calculation of δ_{max} is covered in Sect. 7.1.4.5.

Obviously, solutions produced by the MILP with constraints (7.1) to (7.3) are not necessarily valid schedules with respect to criteria 1 to 3, since criteria 3 is not fully satisfied, i.e., nodes $v_i \in \mathbb{I}_i$ may still collide. This will be dealt with in the next sections.

7.1.4.2 Frequency Diversity

The basic constraints from Sect. 7.1.4.1 do not take into account that the interference ranges of nodes that are not neighbors in T can overlap. In a convergecast scenario it is possible that nodes in the same area, but different branches of T, interfere with each other, as depicted in Fig. 7.1. For this network, node v_4 interferes with nodes v_1 to v_7 even though it only transmits packets to v_2. Constraints (7.1) to (7.3) may assign a common sending slot to a pair of such conflicting nodes.

This problem can be solved with frequency diversity, i.e., if two nodes operate on different channels they can communicate concurrently and collisions are avoided. Let $\mathbb{C}$ be the set of available non-overlapping channels, as defined by the physical layer. As long as the number of conflicts in each slot is at most $|\mathbb{C}|$, the schedule can be realized through a careful assignment of channels.

Sect. 7.1.4.1 already considered one such conflict through constraint (7.2) by disallowing a node to use the same slot as one of its grandchildren. Assuming the availability of a sufficiently large number of channels, it is possible to increase the upper bound on the average packet generation rate by relaxing constraint (7.2) in

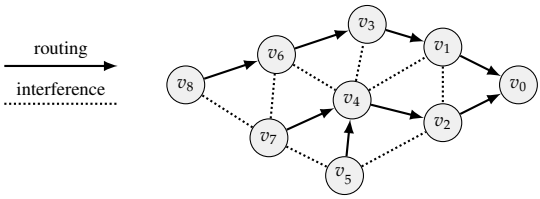


Figure 7.1: A network showing potentially interfering nodes, which are not in the same branch of a routing tree.

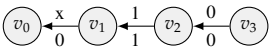


Figure 7.2: For a line topology with four nodes and $K = 2$, all solutions without frequency diversity are infeasible, i.e., $\delta_{max} = 0$ (above). The relaxed version has a solution with $\delta_{max} = 1$ (below).

a way that it does not consider this type of conflict. The relaxed constraint can be expressed as

$$\forall_{i,k}: \quad x_{ik} + \sum_{j \in N_i^-} x_{jk} \leq 1. \tag{7.4}$$

It replaces constraint (7.2) from Sect. 7.1.4.1. This constraint disallows that a node v_i or any of its children $v_j \in N_i^-$ share a common sending slot. The motivation is that v_i cannot receive and transmit a packet in the same time slot. Although the improvement brought about by this change may seem marginal at first, it can make a big difference especially for networks with a high load, as exemplified by Fig. 7.2.

7.1.4.3 Frequency Assignment

The constraints from Sect. 7.1.4.1 can be extended to also compute a channel assignment. This requires knowledge about interference. One way to extend the MILP is to replace variables x_{ik} by variables x_{ikc} that express whether node v_i uses channel c in slot k and an additional constraint that prevents conflicts between node v_i and nodes in $\mathbb{I}_i$. The LP has $(i-1)K|\mathbb{C}|+1$ variables. It has the disadvantage that it is more difficult to solve. Fortunately, we can leave the concrete channel assignment to DSME, since it provides an efficient algorithm for distributed channel negotiation. This is transparently integrated into the DSME slot allocation handshake. Also, the

selection of a specific channel has no influence on the performance of the network, assuming the absence of external interference.

This leads to a much simpler LP. We merely limit the number of simultaneous, spatially overlapping transmissions in time slot k to $|\mathbb{C}|$, e.g., in DSME $|\mathbb{C}| = 16$. This is achieved by the following constraint

$$\forall_{i,k}: \quad \sum_{j \in \mathbb{I}_i} x_{jk} \leq |\mathbb{C}|. \tag{7.5}$$

The constraint ensures that a maximum of $|\mathbb{C}|$ nodes in a neighborhood is allowed to transmit in slot k.

7.1.4.4 Heterogeneous Packet Generation Rates

So far we assumed that all nodes except the sink generate packets according to a Poisson distribution with equal mean $\delta \leq \delta_{max}$. In applications, events may also occur at different rates at different nodes. Also, some nodes may have been introduced to work as relays only, i.e., they generate no packets themselves. The constraints from Sect. 7.1.4.1 can be adopted to this situation by changing constraint (7.3) as follows

$$\forall_i: \quad \sum_{j \in T_i} \delta_j \leq \sum_k x_{ik}, \tag{7.6}$$

where $\delta_j \geq 0$ is the fixed average packet generation rate of node v_j. The modified LP will have a solution if $\delta_i \leq \delta_{max}$ for all nodes v_i. This is a sufficient, but not a necessary condition.

7.1.4.5 Optimizing for Throughput

Each routing tree T has a fixed capacity which can be expressed by the maximum value of δ for which a *δ-resilient* schedule exists. First, we show how to compute the maximal value for δ given T and the length K of a schedule. As discussed in Sect. 7.1.4.1, the optimization problem can be formulated as a MILP with $(i-1)K$ binary variables x_{ik}. Here, we additionally utilize one real variable δ instead of a constant δ. This MILP is called $\mathcal{LP}_1$. We denote the solution of $\mathcal{LP}_1$ by δ_{max}. The objective function is

$$\mathbf{O_1}: \quad \max\ \delta. \tag{7.7}$$

An upper bound for δ_{max} can be derived from $\mathcal{LP}_1$. Let d_T be the height of T, and $\delta_T = \max\{\delta_i \mid v_i \in N_0^-\}$. Furthermore, let

$$\Delta_T = \min\{(i-1)^{-1}, (2\delta_T - 1)^{-1}, (3(d_T - 1))^{-1}\}. \tag{7.8}$$

The following lemma is easy to prove (see also [ASLM12]).

Lemma 1. *δ_{max} is for any tree T and $K > 0$ at most Δ_T.*

It can be shown that $\delta_{max} = \Delta_T$ for $K = 1/\Delta_T$. Thus, in this case each node generates one packet per frame. Note that $\delta_{max} \geq \Delta_T$ for $K \geq 1/\Delta_T$.

7.1.4.6 Optimizing for Energy

After calculating the upper bound for the packet generation rate δ_{max} in Sect. 7.1.4.5, this section provides a way to calculate schedules with minimum energy consumption. The LP from 7.1.4.1 serves as a basis, where

$$\mathbf{O_2}: \quad \min \quad \sum_{v_i \in \mathbb{V}, k} x_{ik} \tag{7.9}$$

is used as an optimization goal. This minimizes the total number of transmission slots in the network while ensuring that each node has enough slots to handle its traffic. The solution of this LP is called $\mathcal{LP}_2$. Minimizing the number of transmission slots increases energy efficiency of the network in a trade-off with delay: the more slots allocated, the lower the delay, but the higher the power consumption. An ideal solution in terms of delay would therefore be to allocate as many slots as possible at all nodes. However, this is usually not intended, and so the following sections describe three approaches for calculating delay-optimized schedules for the minimum number of slots required. The approaches are listed according to increasing complexity. As input to the system, an average packet generation rate δ_i for every node $v_i \in \mathbb{V} \setminus \{v_0\}$ has to be provided, as well as the routing tree T and the number of slots in the schedule K.

7.1.4.7 Heuristically Optimizing Delay

To minimize packet delay we need to minimize the dwell time of a packet at each node. We therefore disallow that two consecutive slots of a node are used as sending

slots. The rationale of this constraint is to enable a node to transmit a packet received in slot k in slot $k+1$. This is enforced by

$$\forall_{i,k}: \quad x_{ik} + x_{i(k+1)\%K} \leq 1. \tag{7.10}$$

The extension of $\mathcal{LP}_1$ by (7.10) is denoted $\mathcal{LP}_3$. Fig. 7.3a depicts a schedule resulting from this approach. Note that constraint (7.10) often enforces that each reception slot of a node is immediately followed by a transmission slot, i.e., reception and transmission slots alternate.

7.1.4.8 Optimizing for Delay

Fig. 7.3a illustrates that constraint (7.10) does not lead to a minimal dwell time at every node (e.g., at node v_4). The reason is that constraint (7.10) does not interrelate the slots of a node with those of its parent. For non-periodic data collection scenarios, Cui et al. show that for certain conditions scheduling all incoming links before outgoing links yields minimal delay [CMGL05] but their solution may result in large queues.

We use a relaxed version of this approach by enforcing that a node must have a transmission slot directly after a reception slot. This leads to a kind of *hot-potato routing*. It can be achieved as follows

$$\forall_{i,k}: \quad x_{ik+1} \geq \sum_{j \in N_i^-} x_{jk}. \tag{7.11}$$

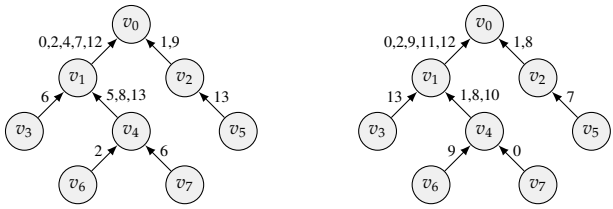
$\mathcal{LP}_1$ together with constraint (7.11) is called $\mathcal{LP}_4$. The next lemma shows that non-trivial solutions of $\mathcal{LP}_4$ are delay optimal. The proof of the lemma is omitted.

Lemma 2. *The following holds for all solutions of $\mathcal{LP}_4$.*

1. *All nodes $v_i \in \mathbb{V} \setminus \{v_0\}$ satisfy constraint* (7.11)*.*
2. *The delay of a packet is equal to the sum of the initial waiting time before transmission at the generating node and the product of D_S and the depth of the node in T.*

$\mathcal{LP}_4$ may have only solutions with very small values of δ or even $\delta = 0$. In order to find solutions of $\mathcal{LP}_1$ that come close to satisfying constraint (7.11), we introduce binary variables u_i for all $v_i \neq v_0$ and use the following constraint

$$\forall_{i,k}: \quad u_i + x_{ik+1} \geq \sum_{j \in N_i^-} x_{jk}. \tag{7.12}$$



(a) Solution of $\mathcal{LP}_3$.

(b) Solution of $\mathcal{LP}_5$.

Figure 7.3: Schedules for a network with $|\mathbb{V}| = 8$, $K = 14$, and $\delta_i = 1$ for all $v_i \in \mathbb{V}$. Edges show the absolute transmission slot indices.

Moreover, we replace variable δ by a constant $\delta_c \leq \delta_{max}$ (the solution $\mathcal{LP}_1$) and use the following objective function

$$\mathbf{O_3}: \quad \min \sum_i u_i \tag{7.13}$$

Thus, the goal is to find a schedule for the data generation rate δ_c minimizing the number of consecutive slot violations on all paths. We refer to this LP as $\mathcal{LP}_5$. Fig. 7.3b shows a schedule obtained from $\mathcal{LP}_5$.

7.1.5 Extension to Interrupted Transmission Phases

In this section, we extend the formulation of $\mathcal{LP}_5$ from Sect. 7.1.4.8 to TDMA-based system with interrupted transmission phases, i.e., with periods where no slots can be scheduled and hence no packets can be transmitted. These phases could be sleep phases or, in the case of DSME, the CAP and increase delay if a packet was received before this phase but can only be transmitted afterwards.

Consider a packet p generated a node v_i and forwarded via path ρ_i to v_0. The delay of packet p along path ρ_i is given by the sum of dwell times Dt_j for $v_j \in \rho_i$. The dwell time in unit of slots is defined as

$$Dt_i = TX_i - RX_i + \theta_i K. \tag{7.14}$$

In this expression TX_i denotes the absolute slot index of the transmission slot of v_i in which p is sent and RX_i is the absolute slot index of the reception slot of v_0 in which p is received. Additionally, it might happen that the transmission slot is scheduled before the reception slot, and thus p is transmitted in the next frame, as illustrated in

Fig. 7.4. In this case θ_i is 1, and 0 otherwise, accounting for the additional delay by adding the duration of K slots. It remains to express (7.14) as an LP.

For formulation of the objective function according to (7.14), absolute slot indices are required. Therefore, we define the binary variable x_{ikl} as

$$x_{ikl} = \begin{cases} 1, & \text{if the } l^{th} \text{ transmission of } v_i \text{ occurs in slot } k \\ 0, & \text{otherwise} \end{cases}, \tag{7.15}$$

so that there are K rows with K transmission slots at every node v_i. In other words: a $K \times K$ matrix of binary optimization variables is assigned to every node v_i, where each row indicates the position of a single scheduled transmission slot. The variables x_{ikl} are subject to the constraints

$$\forall_{i,l}: \quad \sum_k x_{ikl} \leq 1, \tag{7.16}$$

$$\forall_{i,k}: \quad \sum_l x_{ikl} + \sum_{j \in N_i^-, l} x_{jkl} \leq 1. \tag{7.17}$$

Here, constraint (7.16) ensures that at most one slot is allocated per row, while constraint (7.17) is equivalent to constraint (7.4) and ensures that a node v_i and its children cannot send at the same time. Therefore, the absolute index of the l^{th} transmission slot at a node v_i is given by $\sum_k x_{ikl} k$. Similarly to constraint (7.5), the maximum number concurrent transmissions within interference range of a node v_i can be limited to $|\mathbb{C}|$ by

$$\forall_{i,k}: \quad \sum_{j \in \mathbb{I}_i, l} x_{jkl} \leq |\mathbb{C}|. \tag{7.18}$$

At last, one has to ensure that every node schedules sufficiently many slots to support its traffic, i.e., packets generated by v_i and nodes in T_i, which are then retransmitted by v_i. The constraint can be expressed as

$$\forall_{i \neq 0}: \quad \lceil \delta \gamma_i \rceil = \sum_{k,l} x_{ikl}. \tag{7.19}$$

Let $\mathbb{P} = \{\rho_i \mid v_i \in \mathbb{V} \setminus \{v_0\}\}$ be the set of all paths in the network, where the optimization objective is to minimize dwell time of all nodes on all path $\rho_i \in \mathbb{P}$ only if such path exists. Such conditional statements can be easily expressed using Quadratic Programs (QPs), e.g., if a node v_i schedules a transmission slot in slot 0,

and its parent $v_{N_i^+}$ forwards the packet in slot 4, the resulting dwell time would be $4 \cdot \sum_l x_{i0l} \sum_l x_{N_i^+ 4l}$. However, solving QPs is computationally expensive, and thus we formulate the optimization objective as an ILP using the *Big M* method:

$$\forall_{i,l}: \quad \sum_k x_{ikl} + (1 - \alpha_{il}) = 1, \tag{7.20}$$

$$\forall_{i \notin N_0^-, l}: \quad \sum_k x_{N_i^+ kl} f(k) - \sum_k x_{ikl} f(k) - \theta_{il} K - \phi(1 - \alpha_{il}) \leq Dt_{il}, \tag{7.21}$$

$$\forall_{i \notin N_0^-, l}: \quad \sum_k x_{N_i^+ kl} f(k) - \sum_k x_{ikl} f(k) - \theta_{il} K + \phi(1 - \alpha_{il}) \geq Dt_{il}, \tag{7.22}$$

$$\forall_{i \notin N_0^-, l}: \quad -\phi\alpha_{il} \leq Dt_{il}, \tag{7.23}$$

$$\forall_{i \notin N_0^-, l}: \quad \phi\alpha_{il} \geq Dt_{il}, \tag{7.24}$$

$$\mathbf{O_4}: \quad \textit{minimize} \quad \sum_{i \neq 0, l} Dt_{il} + \theta_{il}. \tag{7.25}$$

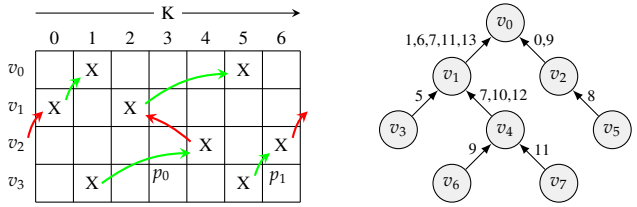
Here, f(k) is a mapping of slot indices to the number of elapsed slots since the start of the MSF, including CAP and beacon slots. For example, without CAP-R, $f(8) = 26\,\text{slot(s)}$ and with CAP-R, $f(8) = 17\,\text{slot(s)}$. ϕ is a large constant, 100 in our case, and α_{il} are slack variables. Consequently, constraints (7.21) and (7.22) apply when $\alpha_{il} = 1$, i.e., when node v_i transmits at least l packets. In this case, $Dt_{il} = \sum_k x_{N_i^+ kl} k - \sum_k x_{ikl} k - \theta_{il} K$. One the other hand, constraints (7.23) and (7.24) apply for $\alpha_{il} = 0$ so that the dwell time $Dt_i = 0$. Constraint (7.20) ensures that $\alpha_{il} = 1$ if there is a packet is forwarded on the l^{th} path. This formulation introduces $3K$ additional variables per node for a total of $\mathbb{V}(K^2 + 3K)$ optimization variables in the ILP.

We denote constraints (7.16) to (7.25) as $\mathcal{LP}_6$. Fig. 7.4b shows a solution of $\mathcal{LP}_6$ where paths are scheduled between the CAPs, which occur after slot 6 and 13.

In addition to $\mathcal{LP}_6$, we provide an equivalent, more efficient description of the LP in Appendix A.3. However, the description is less comprehensible and requires an algorithm to precalculate transmission paths.

7.1.6 Theoretical Evaluation

This section considers three data collection scenarios, for which Tbl. 7.2 depicts the configuration parameter. The MO, was chosen so that the number of packets per MSF is as small as possible, but integer. For evaluation, we utilize the theoretical multi-hop model proposed in [Kau19]. The scenarios are evaluated with periodic



(a) Exemplary slot placement. **(b)** Solution of $\mathcal{LP}_6$.

Figure 7.4: Illustration of backwards steps on paths (left) and exemplary schedule for $|\mathbb{V}| = 8$, $K = 14$, and $\delta_c = 1$ (right).

scenario	sampling rate	sampling resolution	*SO*	*MO*	*K*
1	20 Hz	10 B	1	4	56
2	8 Hz	50 B	2	4	28
3	1 Hz	100 B	3	6	56

Table 7.2: Parameter for three scenarios of the theoretical evaluation.

traffic regarding their worst-case end-to-end delay. We assume that packets are generated at the beginning of a slot so that there is no initial waiting time. The analysis uses the network depicted in Fig. 7.3. The queue length is considered infinite. As an example, the schedules in Figs. 7.3a and 7.3b yield a delay of 829.44 ms and 337.92 ms, respectively. The former includes two CAPs for a packet sent on a path over slot 6 at v_7, slot 8 at v_4, and slot 0 at v_1.

Fig. 7.5 shows that the delay strongly varies based on the scenario and scheduling strategy. $\mathcal{LP}_6$ achieves minimum delay in all scenarios with a path length of 3 slots, varying based on the increasing slot length (*SO*). In contrast to $\mathcal{LP}_5$, $\mathcal{LP}_6$ consistently schedules paths between CAPs, which happens for $\mathcal{LP}_5$ only by chance in scenario 3. Therefore, the delay of $\mathcal{LP}_6$ is bounded by the maximum number of hops in the network and includes the duration of a number of CAPs, equal to the integer quotient of the maximum number of hops and the number of slots in the CFP. The same applies to $\mathcal{LP}_5$ but it may introduce a delay of an additional CAP. For the other LPs, the maximum delay depends on the scenario. Since the network is asymmetric, the other strategies assign more slots to the subtree T_1 rooted in v_1. Between v_2 and v_0, few slots are scheduled which may not be next to each other, resulting in large delays. The delay of $\mathcal{LP}_6$ and $\mathcal{LP}_5$ is independent of δ_c, while it increases for a decreasing δ_c for the other LPs. Therefore, $\mathcal{LP}_6$ always yields

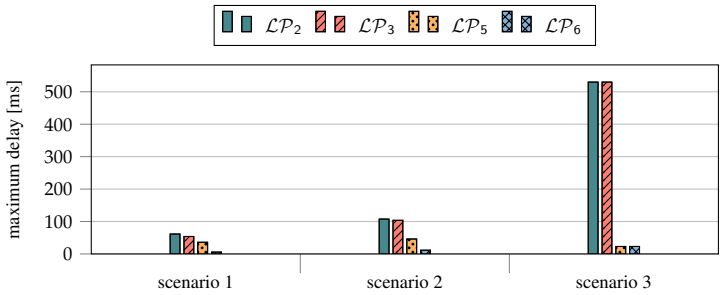


Figure 7.5: Maximum end-to-end delay for different scheduling strategies and scenarios.

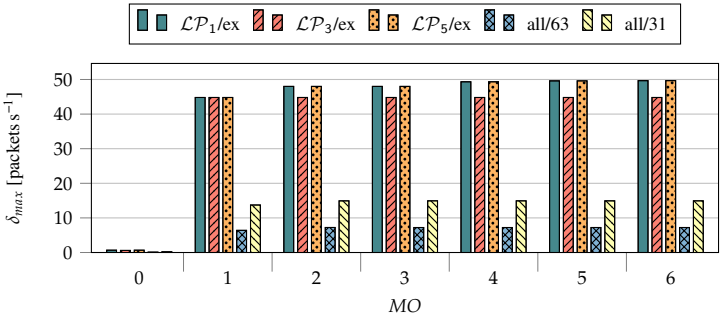


Figure 7.6: Maximum packet generation rate δ_{max} for different topologies and scheduling strategies.

minimal delay for periodic traffic, significantly outperforming the other scheduling strategies. The number of scheduled transmission slots is the same for all scheduling strategies with 280 slot(s) s^{-1}, 112 slot(s) s^{-1}, and 14 slot(s) s^{-1} for scenarios 1, 2, and 3, respectively.

Fig. 7.6 shows δ_{max} for different scheduling strategies for the topology from Fig. 7.3 and binary trees with $|\mathbb{V}| = 63$ and $|\mathbb{V}| = 31$. The scheduling strategies achieve the same δ_{max} in symmetric networks but vary slightly for asymmetric networks. Here, $\mathcal{LP}_2$ achieves a higher δ_{max} than $\mathcal{LP}_1$ because channel diversity is considered. Thereby, δ_{max} is independent of MO.

δ	topology	$\lvert\mathbb{V}\rvert$	S	SO	MO	K
0.83 packets s^{-1}	multi-line	56	100 B	3	6	56

Table 7.3: Parameter for the simulative evaluation of the proposed LPs.

7.1.7 Simulative Evaluation

The following sections analyze the presented scheduling strategies in realistic scenarios. In particular, we consider a multi-line topology, where we assess the effectiveness of the LPs in comparison to LLSF and TPS. The goal is to find out how well the latter algorithms perform and whether the LPs can act as a baseline in scenarios with irregular traffic.

7.1.7.1 Scenario Description

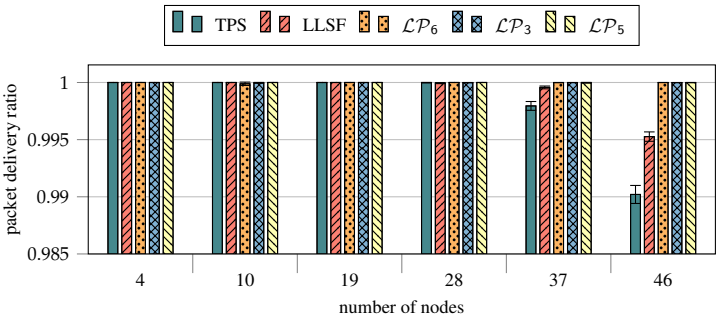
We consider scenario three from Sect. 7.1.6 with an altered sampling rate of 5/6 Hz. Schedules are calculated for a sampling rate of 1 Hz to ensure that no packets are lost due to queue drops. LLSF and TPS have similar overprovisioning built-in to increase reliability. We consider a multi-line topology with three lines and an increasing number of nodes $\lvert\mathbb{V}\rvert$. Sect. 7.1.7.1 summarizes the most important parameters, while Tbl. A.8 shows the whole configuration.

7.1.7.2 Reliability and Queue Length

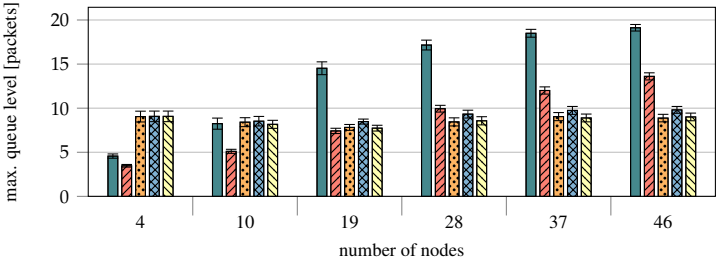
Fig. 7.7a shows the PDR of different scheduling strategies. The LPs achieve a lossless transmission, while the PDR of TPS and LLSF starts to decline at 28 nodes. That is because they cannot find a fully optimized schedule in a distributed manner, resulting in frequent slot relocations. Their maximum queue length increases for an increasing number of nodes, as illustrated in Fig. 7.7b. Over all nodes, about 19 packet(s) are in the queue at the same time for TPS, increasing up to 22 packet(s) at children of the sink. For the LPs, it stays between 7 – 10 packet(s), with $\mathcal{LP}_6$ and $\mathcal{LP}_5$ performing better than $\mathcal{LP}_3$ because packets arrive to an empty queue and are directly forwarded. Only bursts of self-generated packets increase the queue length.

7.1.7.3 Allocated GTS and Delay

Fig. 7.8a shows the number of scheduled slots. For four nodes, TPS and LLSF allocate about twice as many slots as the LPs because they dynamically adapt to changing traffic. During a burst, both protocols allocate many slots while the LPs



(a) PDR for all scheduling strategies.



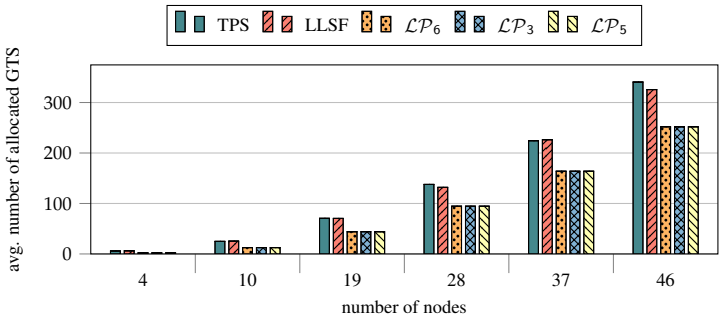
(b) Maximum queue level for all scheduling strategies.

Figure 7.7: PDR and queue length for a multi-line topology with 3 lines and increasing depth for $SO = 3$, $MO = 6$, and $K = 56$.

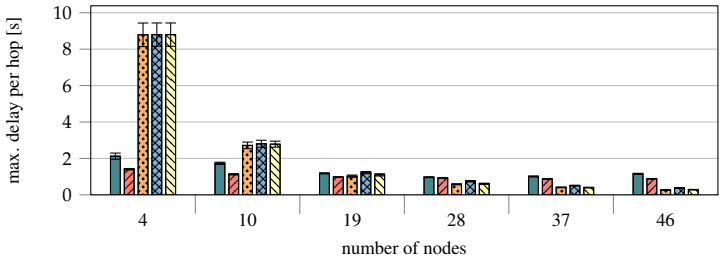
always allocate slots according to the optimal delay-energy trade-off. Beyond 28 nodes, TPS and LLSF underestimate the required number of GTS and are unable to fully meet the traffic requirements of the nodes. This also benefits queue lengths. Fig. 7.8b shows the resulting maximum delay per hop of all nodes. All LPs perform best for deep networks. $\mathcal{LP}_6$ performs better than the other LPs since it schedules paths between CAPs, however, the improvement compared to $\mathcal{LP}_5$ is not significant enough to justify its high computational complexity.

7.1.7.4 Influence of CAP-Reduction

At last, Fig. 7.9 shows the influence of CAP-Reduction (CAP-R) on the maximum delay per hop of the protocols. For this, the multi-line topology with 19 nodes was



(a) Number of allocated TX slots.



(b) Maximum end-to-end delay.

Figure 7.8: Number of allocated GTS and delay for a multi-line topology with 3 lines and increasing depth for $SO = 3$, $MO = 6$, and $K = 56$.

evaluated, since the delay of all scheduling strategies is almost equal here. Using CAP-R changes $K = 56\,\text{slot(s)}\,\text{s}^{-1}$ to $K = 112\,\text{slot(s)}\,\text{s}^{-1}$ for $MO = 6$, and $K = 104\,\text{slot(s)}\,\text{s}^{-1}$ for $MO = 5$. CAP-R with $MO = 6$ increases the delay for all strategies. That is because the number of scheduled slots per node is small in comparison to K. Therefore, all slots could be scheduled consecutively at the start of a 6P-MSF, resulting in large delays for packets that arrive afterwards. TPS schedules time slots randomly, mitigating this effect for a larger number of scheduled slots. The problem can be counteracted by calculating a schedule for a smaller MO, as illustrated in Fig. 7.9 for $MO = 5$. This yields almost the same number of slots but also decreases the maximum delay because slots are more evenly distributed and packets are not delayed by the CAP. $\mathcal{LP}_6$ already schedules paths between CAPs so that it does not benefit from CAP-R as much as the other strategies.

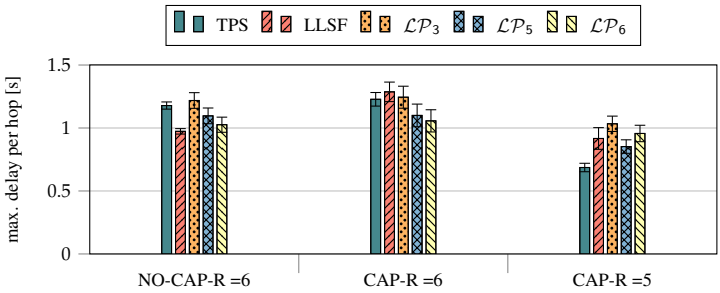


Figure 7.9: Influence of CAP-R on the delay in a multi-line topology with 3 lines for $SO = 3$, $MO = 6$, and $K = 56$.

7.1.8 Discussion and Conclusion

All in all, the preceding sections have shown that the proposed LPs can be used as a baseline for delay-minimal and energy-optimized scheduling. However, for scenarios with dynamically changing traffic, the LPs cannot adapt to changes in traffic so that they might perform worse when all GTS are allocated. Here, TPS and LLSF have the advantage of shifting GTS from one node to another when queues impend to overflow. In terms of energy, both protocols allocate more GTS than required. It should be noted that the LPs only provide minimal delay for packets with an unknown random generation time. If the generation times are known, they can be extended to schedule slots directly after the generation. However, this assumption is highly unrealistic because it requires the packet generation interval to be an exact multiple of the $aBaseSuperframeDuration$.

It became apparent that the computation time of the LPs overwhelms even more powerful gateways for $K \geq 56$ and $\mathbb{V} \geq 20$, especially if multiple updates are required. Consequently, we conceive three use cases for the LPs:

- As a baseline for dynamic schedulers.
- As a starting configuration, similar to an *Orchestra* schedule or a *Minimal 6TiSCH Configuration*. Dynamic schedulers can adapt these schedules to variations in traffic.
- As a fallback configuration to ensure that every node has a minimum number of slots for communication in case of emergency.

The high computation time demands for heuristic approaches like Simulated Annealing (SA) or Particle Swarm Optimization (PSO) to calculate the schedules. This approach is considered in Sect. 7.2.

Finally, simulations show that the LPs decrease the maximum delay by up to 80% for deep networks while also increasing reliability and reducing the required buffer size. As an example, our analysis shows that for a sampling frequency of 5/6 Hz and a sampling resolution of 100 B, packets are guaranteed to be delivered within 10.97 s for a multi-line topology with three lines and 46 nodes. This indicates that TPS and LLSF do not provide near-optimal scheduling and that there is room for the development of a more powerful dynamic scheduler. We believe that the introduced techniques are a valuable tool for the practical realization of data collection tasks requiring delay bounds.

Scheduling through Quantum-inspired Annealing
Section 7.2

Solving ILPs is a computationally complex task, especially if many optimization or slack variables are required. For example, this is the case with $\mathcal{LP}_6$, defined in Sect. 7.1.5. A variety of heuristic methods exist that promise a faster, though often not exact, solution of these combinatorial optimization problems. They include Genetic Algorithm (GA) [Sar19], Particle Swarm Optimization (PSO) [EAK+19], and Simulated Annealing (SA) [DCM18]. With the emergence of quantum computers and hardware that emulates their functionality, SA is a promising solution that can be directly applied in Quantum Annealing (QA) [RK19].

One should be aware that there is still some way to go before quantum computers are widely available. However, a few hardware platforms have been presented that offer *quantum-inspired annealing*, i.e., a duplication of every computation core per quantum bit to reflect the parallelism of quantum computing [ARV+19, RK19]. Since such multiplication entails various physical side effects, the available platforms are usually limited to only a few (e.g., up to 1024 [ARV+19]) quantum bits. Nevertheless, they are suitable as hardware accelerators for smaller problems. Additionally, larger problems can be subdivided into several smaller problems and computed sequentially.

For quantum-inspired annealing, the LPs from Sect. 7.1 must be reformulated as Quadratic Unconstrained Binary Optimizations (QUBOs) [ARV+19]. The following sections show this exemplary for $\mathcal{LP}_6$, which provides delay-minimal scheduling. Until the general availability of quantum computers or special hardware for quantum-inspired annealing, the QUBOs can be used for regular simulated annealing.

7.2.1 Related Work

Quantum-annealing can be applied to various types of combinatorial optimization problems. In [SLL21], a quantum-inspired Digital Annealer (DA) developed by Fujitsu [ARV+19] is utilized for solving the vehicle sharing problem in car pooling. The goal is to minimize the total travel distance for a fleet of vehicles that transports passengers from starting their starting position to their destinations. Such optimization minimizes the maintenance cost of the fleet. They compare the DA's performance with conventional optimization through the CPLEX optimizer and demonstrate that it scales better. However, the solution is not as accurate. Suen et al. list the limited number of available bits as the DA's biggest drawback.

Another scheduling problem is analyzed in [INH19], where the working schedule of nurses is calculated. Similar to the problem stated in Sect. 7.1, working hours are interrupted by breaks, and collisions in the working schedules must be avoided. They perform experiments on the D-Wave 2000Q quantum annealer to show the feasibility of calculating scheduling problems through QA.

7.2.2 Scheduling as QUBO Problem

SA provides a probabilistic method for approximating the global optimum of an optimization problem. If SA is conducted on binary optimization variables, quantum computing can be applied, reducing the required solving time from days to a few seconds.

QA problems can be expressed using QUBOs, where the optimization problem is given by

$$\text{QUBO:} \qquad \text{minimize / maximize} \qquad x^t Q x.$$

Here, Q is a $M \times M$ square matrix of constants and x is a vector of binary decision variables of length M. As discussed by Glover et al., traditional ILP constraints can be formalized as penalties, which have a negative influence in the optimization function. An excellent tutorial on how to reformulate these constraints as penalties is provided by them in [GKD19]. The idea is that a penalty is greater 0 if a constraint is violated, and equal 0 otherwise. The optimization function is a linear combination of different penalties and the optimization goal. In the following we provide penalties corresponding to the constraints and the optimization goal from Sect. 7.1:

$$(7.16) \Rightarrow \quad \forall_{i,l}: \qquad P\Big(\sum_{k=0}^{K-2} \sum_{m=k+1}^{K-1} x_{ikl}\, x_{iml}\Big) \tag{7.26}$$

$$(7.17) \Rightarrow \quad \forall_{i,k}: \qquad P\Big(\sum_{h \in N_i^- \cup \{i\}} \sum_{l} \sum_{j \in N_i^- \cup \{i\}} \sum_{\substack{n=0 \\ l<n, \tilde{h}=\tilde{j} \\ \textbf{OR } \tilde{h}<\tilde{j}}}^{K-1} x_{hkl}\, x_{jkn}\Big) \tag{7.27}$$

$$(7.19) \Rightarrow \quad \forall_{i \neq 0}: \qquad P\Big(\big\langle \lceil \delta T_i \rceil - \sum_{k} \sum_{l} x_{ikl} \big\rangle^2\Big) \tag{7.28}$$

$$(7.20) \Rightarrow \quad \forall_{i,N_i^+ \neq 0,l}: \qquad P(\sum_k x_{ikl} - \sum_k \sum_{m=0}^{K-1} x_{ikl} x_{N_i^+ ml}) \tag{7.29}$$

$$(7.21)\text{-}(7.24) \Rightarrow \quad \forall_{i,N_i^+ \neq 0,l}: \qquad Q(\sum_k \sum_{\substack{m=0 \\ m \neq k}}^{K-1} \langle [f(m) - f(k) + K] \% K \rangle x_{ikl} x_{iml}) \tag{7.30}$$

Here $\bar{h}$ and $\bar{j}$ denote the index to the set $N_i^- \cup \{v_i\}$ of h and j. Additionally, P and Q represent weight functions for the constraints and the optimization goal respectively, where the optimization goal is weighted lower than the constraints. More specifically, we utilize $P(x) = 10x$ and $Q(x) = x$. The QUBO formulation does not require additional variables α_{il}, θ_{il}, or Dt_{il} as the according constraints can be expressed as quadratic terms (see Eqs. (7.29) and (7.30)). Thus, $\mathbb{V}K^2$ binary decision variables are required. These penalties can be used to calculate the matrix Q, as described in Eq. (7.26). Alternatively, they can be directly entered into an optimization framework, which then generates the according matrix and optimizes for delay.

7.2.3 Theoretical Evaluation

Since the availability and access to quantum annealers or quantum-inspired annealing hardware is not yet feasible, the following subsection focus on the theoretical evaluation of the described QUBO through SA. For this, the open-source library *Ocean dimod* developed by D-Wave Systems is utilized, which (in theory) also allows access to their quantum annealer with up to 5000 quantum bits. One should note that SA ideally attains the same solution as $\mathcal{LP}_6$ from Sect. 7.1. Therefore, all of $\mathcal{LP}_6$'s results also apply to the QUBO and simulated annealing, and the following subsections focus on how reliably the meta-heuristic can find optimal solutions and how scalable it is compared to the ILPs.

7.2.3.1 Energy Levels

Simulated annealing is a meta-heuristic for solving minimization and maximization problems with a fixed number of execution steps. If the number of steps is selected too small, a sub-optimal solution is found; if it is too large, execution time is unnecessarily prolonged. Fig. 7.10 shows the Cumulative Distribution Function (CDF) for different numbers of execution steps and the resulting energies of the simulated annealing process for the example network from Fig. 7.3a. Experiments are repeated 1000 times per number of steps to ensure sufficient accuracy.

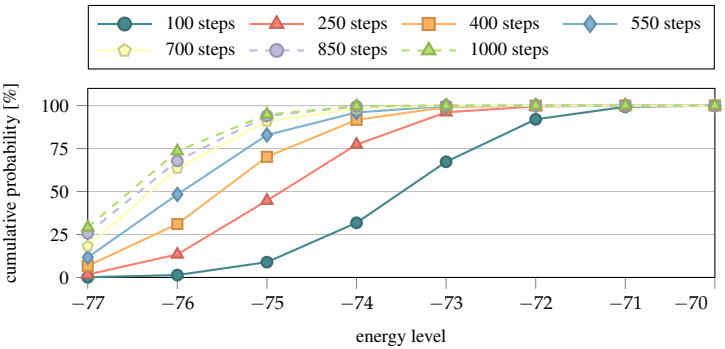
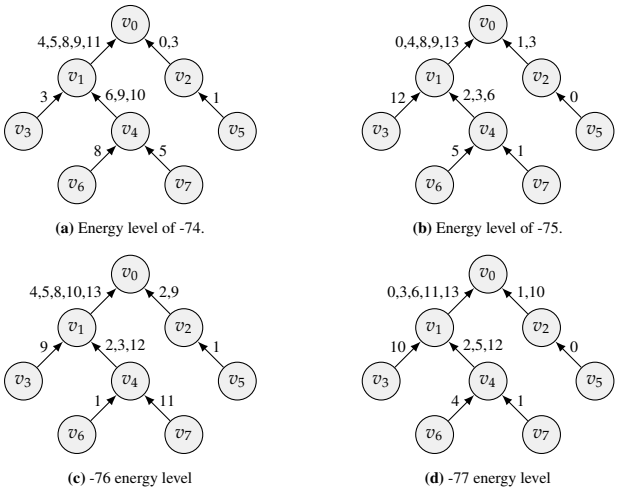


Figure 7.10: CDFs for different numbers of execution steps of the simulated annealing process and the resulting energy levels.

As shown in Fig. 7.10, it is difficult to find an optimal solution with low numbers of steps (e.g. 100 steps). The exact probability for 100 steps is 0.1 % which would require a high number of repetitions to obtain an optimal solution. With 1000 steps, however, the optimal solution is found in 29.1 % of the cases so that on average only about four repetitions are needed. In addition, only solutions with an energy less than or equal to -73 are found. Overall, a higher number of steps provides an effectively faster solution.

Above observations raise the question of the quality of schedules with different energy levels. Therefore, Fig. 7.11 shows schedules for the lowest four resulting energy levels, where schedules with an energy of -77 are optimal and identical to those discussed in Sect. 7.1.5. For an energy level of -74 (Fig. 7.11a), no valid schedule is found as node v_4 and v_1 both utilize slot nine for transmission. For energy levels -75 and -76 (Figs. 7.11b and 7.11c) feasible schedules are created, and the only difference is the resulting delay. For the former, v_4 allocates slot three which would be required be v_1 to form an optimal path. Additionally, v_1 does not forward the packet received in slot six directly in slot seven but only in slot eight. With an energy level of -76, the only violation happens at v_4 similarly to the one described above. This leads to the conclusion that schedules of non-optimal energy levels are also suitable (especially in larger networks) but should be verified before use. That way, execution times can potentially be considerably reduced.



(a) Energy level of -74.

(b) Energy level of -75.

(c) -76 energy level

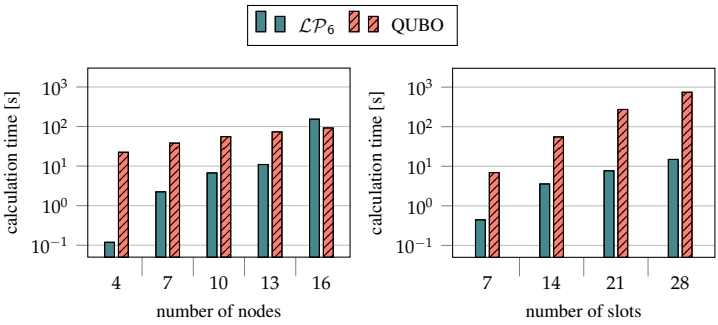
(d) -77 energy level

Figure 7.11: Schedules created through Simulated Annealing for different resulting energy levels of the annealing process.

7.2.3.2 Scalability Considerations

At last, we consider the scalability of the QUBO and $\mathcal{LP}_6$ concerning the number of nodes, number of slots, and packet generation rate δ. These are the three key parameters for the optimization problem. Fig. 7.12a shows the scalability for an increasing number of nodes in a multi-line topology. We use $K = 28$ and the respective δ_{max}. As one can see, the execution time for both methods increases exponentially with an increasing number of nodes, whereas $\mathcal{LP}_6$ shows a significantly lower execution time for small numbers of nodes since the computation of the QUBO takes a higher preparatory calculation. From 16 nodes on, $\mathcal{LP}_6$ exhibits a higher execution time. The number of optimization variables grows by K^2 for each additional node.

Similar behavior can be observed for the scaling over an increasing number of slots in Fig. 7.12b. Here, $\mathcal{LP}_6$ performs better, not only for small K but also for large K. The number of optimization variables increases by $(2K-1)i$ per additional slot, so the scaling for the 10-node case shown in Fig. 7.12b is worse than in Fig. 7.12a. For increasing δ, the execution time remains almost constant for both techniques, presumably because no additional optimization variables are required.



(a) Scaling with number of nodes.

(b) Scaling with number of slots.

Figure 7.12: Scaling behavior of the QUBO and $\mathcal{LP}_6$ over an increasing number of nodes and slots.

7.2.4 Discussion and Conclusion

The Quantum Annealing problem was also evaluated in an optimization framework by Fujitsu [ARV+19] and provided solutions equivalent to the ILP. One should notice, that QA can also find sub-optimal solutions if the optimization time is not sufficient but for the conducted experiments in random tree topologies, an optimal schedule was found in all cases. One severe limitation of the framework is its restriction to 1024 quantum bits. However, there are specialized quantum annealing hardware platforms that support up to 8192 quantum bits [SLL21], allowing for large problems that cannot be calculated by LPs within a reasonable time. Furthermore, generalized quantum computers with more and more quantum bits are developed at the time of writing [RK19] so that the provided formulation offers an interesting approach that promises to achieve remarkable results as soon as the hardware catches up.

Chapter 8

Trade-off Between Primary and Secondary Traffic

The preceding chapters have examined the complex interactions between secondary (see Chap. 5) and primary traffic (see Chap. 6) regarding their adaptability to scenarios with dynamically changing traffic. In particular, it became clear that the CAP is heavily utilized during fluctuations of primary traffic, but hardly used for slot (de)allocations in stable conditions. This suggests that potential throughput can be further increased. The objectives of a high degree of adaptability and a high throughput are conflicting. A high throughput demands that the CFP is larger than the CAP to transmit more primary traffic, while adaptability requires CAPs to be sufficiently long to accommodate the necessary (de)allocations of GTS. Hence, the fraction τ_{CFP} of CFP's time slots per dataframe is a critical value and each application demands its dedicated fraction.

As discussed in Chap. 2, DSME defines two standard operating modes: CAP-R and no CAP-R (NO-CAP-R) [IEE20]. The fraction τ_{CFP} in NO-CAP-R mode corresponds to $7/16$ and in CAP-R mode increases to $(15 - 8/(2^{SO-MO}))/16$. From these values, it is clear that the two operating modes offer extremely diverging fractions. Consequently, they do not allow an optimal use of resources for a wide spectrum of applications, and time-varying primary traffic is not well supported. To solve this issue, the following sections present Dynamic CAP-Reduction (D-CAP-R)

Excerpt and slight extension of the previously published "Meyer F., Mantilla-González I., Turau V.: New CAP Reduction Mechanisms for IEEE 802.15.4 DSME to Support Fluctuating Traffic in IoT Systems. In: Grieco L.A., Boggia G., Piro G., Jararweh Y., Campolo C. (eds) Ad-Hoc, Mobile, and Wireless Networks. ADHOC-NOW 2020. Lecture Notes in Computer Science, vol 12338. Springer, Cham. https://doi.org/10.1007/978-3-030-61746-2_13." [MMGT20a]. The author proposed the D-CAP-R algorithm and was further responsible for the theoretical (especially Sect. 8.3.3) and simulative evaluation.

- an extension of DSME that changes τ_{CFP} with a fine granularity and dynamically, i.e., after the deployment of the network. The main idea is to gradually reduce CAPs when primary traffic is stable and utilize the freed slots as GTS.

The following sections first provide an overview of related work in Sect. 8.1 and then describe D-CAP-R in detail in Sect. 8.2. At last, Sects. 8.3 and 8.4 explicate a fine-grained theoretical and simulative evaluation, respectively. Here, D-CAP-R is not only compared to CAP-R and NO-CAP-R but also to Alternating CAP-Reduction (A-CAP-R), which is described in Sect. 2.4.4.4. It offers increased reliability in contrast to the other schemes.

8.1 Related Work

A comparative performance analysis of IEEE 802.15.4 and DSME has been carried out in several works [CMML20, JL12, KSKT17, LJ12]. They show that CAP-R improves latency and throughput in applications with strict demands. At the same time the energy consumption is higher than with NO-CAP-R. Furthermore, simulation results demonstrate an increased network throughput of about 7 % with CAP-R compared to NO-CAP-R. Therefore, CAP-R raises bandwidth and enhances the scalability of the system.

A simulative evaluation of DSME in openDSME [KKT19a] is made in [Kau19]. Results evidence the influence of the parameter MO in conjunction with CAP-R on the overall throughput. It increases with increasing MO. However, higher values of MO also imply a severe reduction of the CAP so that the network is unable to handle secondary traffic.

Regarding the impact of CAP-R in DSME, Vallati et al. analyze it from the perspective of network formation [VBPA17]. The authors propose an active back-off mechanism and appropriate selection of configuration parameters that along with CAP-R reduce setup time up to 60 %. In the same direction, a dynamic MSF tuning technique (DynaMO) in conjunction with CAP-R is proposed in [KSKT20]. DynaMO was evaluated in openDSME showing a latency reduction of 15 – 30 % in a large scale networks.

At last, Lee et al. propose TaCFPext [LKZK21], which shows striking similarities to the algorithm presented in this chapter and previously published work [MMGT20a]. The main difference is that TaCFPext converts whole CAPs into allocatable GTS and then requires additional messages for the actual allocation. For this, they do not utilize regular GTS-handshake messages but introduce additional messages for allocating GTS during the CAP which slightly lowers memory overhead. Similarly,

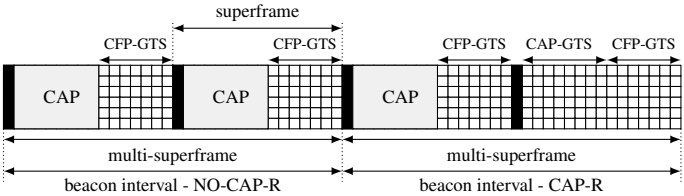


Figure 8.1: Frame structure using A-CAP-R for $BO = MO$ and $MO - SO = 1$ [MMGT20a].

they introduce an additional SAB and ACT for GTS during the CAP which is a simple renaming of the extended data structures required by D-CAP-R. TaCFPext is evaluated in Matlab and indicates similar results to D-CAP-R.

8.1.1 Alternating CAP-Reduction

A-CAP-R is a CAP-R mode proposed in [MMGT20a]. It alternates between NO-CAP-R and CAP-R every beacon interval, first initialized by a coordinator announcing A-CAP-R and the current phase in its beacon. Thereafter, no other beacons are required but nodes alternate autonomously between both modes every BI. In order to avoid self-interference all nodes start in NO-CAP-R mode until a beacon from a coordinator is heard and A-CAP-R is started. This allows for higher throughput with minimum effect on secondary traffic, and requires no additional control messages.

A-CAP-R's key features are its standards compliance and simple implementation. The frame structure of A-CAP-R is illustrated in Fig. 8.1 for the case $MO = SO + 1$ and $MO = BO$, i.e., a MSF consists of two SFs. A-CAP-R does not increase the potential GTS bandwidth provided by CAP-R. In fact, the fraction τ_{CFP} is incremented by the factor $2.14 \cdot ((2^{MO-SO} - 1)/(2^{MO-SO} + 1))$ compared to the fraction τ_{CFP} of NO-CAP-R. The following sections utilize A-CAP-R for comparison with D-CAP-R.

8.2 Dynamic CAP-Reduction

D-CAP-R starts in NO-CAP-R mode and is triggered when all GTS in the CFPs are depleted, i.e., allocated. At this point, D-CAP-R allocates additional GTS during CAPs through DSME's standardized 3-way handshake. For this, two allocating nodes, v_0 and v_1, negotiate a GTS during the last time slot of a pseudo-random SF's CAP,

shrinking it from that moment on. The last slot is chosen because GTS negotiations are triggered at the start of CAPs in openDSME and thus the first slots of a CAP are usually busier than the last slots. Choosing a pseudo-random CAP ensures that CAPs are reduced evenly and about the same time is available for GTS negotiations during all portions of a MSF. Thereby, D-CAP-R does not affect the first CAP of a MSF. The channel c, is chosen so that $c \neq c_{CAP}$. Therefore, communication during the new GTS does not interfere with regular CAP traffic.

All nodes, except v_0 and v_1, can use the CAP normally with the restriction that they cannot communicate with v_0 or v_1 during the allocated GTS. This means, D-CAP-R reduces CAPs locally. After an allocation of D-CAP-R, v_0 and v_1 have reduced their CAP by one slot, while a third node v_2 can use one less CAP slot for communication with them but all CAP slots for communication with other nodes. The rest of the network remains unaffected. Optionally, v_2 can hold back messages to v_0 and v_1 during the allocated GTS and send those during the next CAP to reduce traffic. The whole allocation process using D-CAP-R is detailed in Alg. 2.

Based on the traffic demand of the network, D-CAP-R continues to allocate additional GTS during CAPs until CAP-R mode is reached. That means, the first CAP of every MSF is not affected to ensure that GTS can be deallocated again. This way, up to $8(2^{MO-SO}-1)$ additional GTS can be allocated. The resulting frame structure of D-CAP-R is illustrated in Fig. 8.2, where the first CAP of a MSF is immutable, while GTS can be allocated in the other CAPs, starting with the last time slot of a CAP. The deallocation of GTS works exactly the opposite way until all CAPs are restored. However, as illustrated in Fig. 8.2, CAPs can become fragmented during GTS deallocation, e.g., if the first and third GTS allocation was done by one node and the second GTS allocation was done by another node. Then, the deallocation of the second GTS results in a split CAP. A solution for this is the relocation of the third GTS to a later time slot. This behavior is currently not implemented, but also not mandatory as timers are stopped outside of CAP slots, and therefore no timeouts for CAP messages occur.

8.2.1 Scheduling Requirements

D-CAP-R provides opportunities for more sophisticated scheduling algorithms. For example, a scheduler could decide to allocate additional GTS during the CAP if the traffic demand is high but deallocate GTS again if the PDR of the CAP traffic falls below a certain threshold, i.e., when primary traffic fluctuations appear. Additionally, dwell time can be greatly reduced if transmission slots are scheduled on a path as,

Algorithm 2: Algorithmic description of Dynamic CAP-Reduction.

initialize $F \leftarrow \emptyset$
initialize $ACT(slot, msf) \leftarrow 0 \quad for\ 0 \leq slot < 15,\ 0 \leq msf < SM$
on *GTS allocation* **do**
  $\mathcal{S} \leftarrow \{(slot, msf) \mid ACT(slot, msf) \neq 1\ for\ 8 \leq slot < 15,\ 0 \leq msf < SM\}$
  $\mathcal{T} \leftarrow \{(slot, msf) \mid ACT(slot, msf) \neq 1\ for\ 0 \leq slot < 15,\ 0 \leq msf < SM\}$
  if $\mathcal{S} \neq \emptyset$ **then**
    `// allocate a GTS in a CFP`
    $slot, msf \leftarrow$ random $(i, j) \in \mathcal{S}$, $channel \leftarrow$ random $c \in \mathbb{C}$
    $ACT(slot, msf) \leftarrow 1$
    return $GTS(slot, msf, channel)$
  else if $\mathcal{T} \neq \emptyset$ **then**
    `// allocate a GTS in a CAP`
    if $F = \emptyset$ **then**
      $F \leftarrow \{msf \mid 0 \leq msf < SM\}$
    $msf \leftarrow$ random $msf \in F$, $F \leftarrow F \setminus \{msf\}$
    $slot \leftarrow \max\{slot \mid ACT(slot, msf) \neq 1\}$
    $channel \leftarrow$ random $c \in \mathbb{C} \setminus \{c_{CAP}\}$
    $ACT(slot, msf) \leftarrow 1$
    return $GTS(slot, msf, channel)$
  else
    `// all GTS are allocated`
    return none

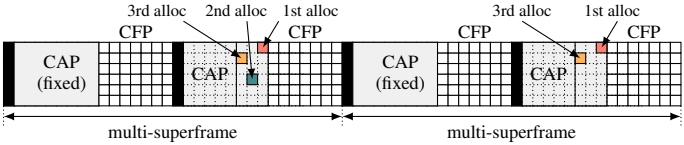


Figure 8.2: Frame structure using D-CAP-R for $MO - SO = 1$. The allocation order of GTS during the CAP is shown.

described in Sect. 7.1.4.8. D-CAP-R potentially enables longer paths by scheduling part of them in the CAP.

Generally, a modification of the scheduler has to be done in order for it to work with D-CAP-R- it has to be ensured that GTS during the CFP and CAP are managed according to the description above. That means, the scheduler first allocates GTS

during the CFP until they are depleted and then starts allocating additional GTS during the CAP.

8.2.2 Specifics in openDSME

It must be mentioned that there is a storage overhead for D-CAP-R because it requires larger data structures (SAB and ACT) for storing information about the $8 \cdot (2^{MO-SO} - 1)$ additional GTS. Even if no CAP-R is active, nodes must use the SAB and the ACT of the CAP-R mode for storing their GTS information. That is because the size of the structures in openDSME cannot be changed dynamically but they are determined during network configuration. There is no way to the decision is purely based on the current traffic demand of the network. At last, it has to be said that D-CAP-R is not conform with the IEEE 802.15.4 standard and openDSME requires larger code modifications to allow the allocation of GTS during the CAP. For example the transceiver cannot be turned on and set to a single frequency for the duration of the whole CAP anymore but it has to be checked if the next time slot is a GTS and the frequency has to be adapted accordingly. This changes how GTS are handled. The default behavior at the end of a GTS is to turn off the transceiver. Now, if the GTS was inside a CAP, the frequency has to be switched back to c_{CAP} and the transceiver has to stay on. This does not increase energy-consumption though as the transceiver of nodes must be turned on during the whole CAP anyways.

8.3 Theoretical Evaluation

The goal of the proposed CAP-R mechanism is to increase throughput while maintaining adaptability to time varying traffic. We mainly consider two metrics: the fraction τ_{CFP} and the adaptability of DSME expressed by the expected time CM to send a CAP message. Both metrics strongly depend on the average number, CM, of CAPs per MSF and the average number of the CAP's time slots, CTM, per MSF. For example, CAP-R is characterized by $CM = 1$ and $CTM = 8$ - a single CAP per MSF with eight time slots. Thus, CAP-R heavily sacrifices adaptability for a higher throughput, which is especially problematic if not all GTS are utilized because they could have been used for CAP traffic. D-CAP-R attempt to overcome this problem by modifying CM and CTM in an appropriate manner.

	Fraction of CFP's time slots per dataframe			
	$MO = 4$	$MO = 5$	$MO = 6$	$MO = 7$
NO-CAP-R	43.75 %	43.75 %	43.75 %	43.75 %
CAP-R	68.75 %	81.25 %	87.5 %	90.06 %
A-CAP-R	56.25 %	62.5 %	65.63 %	67.19 %
D-CAP-R	43.75 – 68.75 %	43.75 – 81.25 %	43.75 – 87.5 %	43.75 – 90.06 %

Table 8.1: Values of τ_{CFP}, i.e., the fraction of the CFP's time slots per dataframe, with NO-CAP-R, CAP-R, A-CAP-R, and D-CAP-R for $4 \leq MO \leq 7$.

8.3.1 Fraction of CFP's Time Slots per Dataframe

The parameter τ_{CFP}, can be calculated based on CM and CTM as

$$\tau_{CFP} = 1 - \frac{CTM + SM}{TM}, \tag{8.1}$$

where $TM = 16SM$ is the total number of time slots per MSF. Incorporation of SM is required to account for SM beacon slots per MSF. Therefore, the fraction τ_{CFP} increases for decreasing values of CTM, as illustrated in Tbl. 8.1. NO-CAP-R is independent of MO, while CAP-R converges towards 93.75 % as MO increases. A-CAP-R offers a compromise between NO-CAP-R and CAP-R. Similarly, D-CAP-R dynamically adapts the time for sending packets with the lower bound equal to NO-CAP-R and the upper bound equal to CAP-R.

8.3.2 Expected Channel Access Time on Slot Level

The adaptability of DSME is characterized by the expected time to allocate a new GTS, i.e., the expected time until a GTS-handshake can be conducted in a CAP. In contrast to τ_{CFP}, which indicates the maximum throughput of a system, adaptability expresses how long it would take to allocate all required GTS. Additionally, it provides an insight into how well the system adapts to dynamically changing traffic. Since a time slot is a baseline unit in DSME's frame structure, we consider it to estimate adaptability in terms of the expected number of time slots that a node must wait to send a CAP message, i.e., N_{CAP}. This value is calculated per MSF as follows

$$N_{CAP}(CM, CTM) = \frac{1}{TM \cdot CM} \sum_{i=1}^{\frac{TM-CTM}{CM}} i \tag{8.2}$$

	Expected waiting time to send a CAP message [slot(s)]			
	$MO = 4$	$MO = 5$	$MO = 6$	$MO = 7$
NO-CAP-R	2.25 slot(s)	2.25 slot(s)	2.25 slot(s)	2.25 slot(s)
CAP-R	9.38 slot(s)	24.94 slot(s)	56.72 slot(s)	120.61 slot(s)
A-CAP-R	5.81 slot(s)	14.10 slot(s)	29.49 slot(s)	61.43 slot(s)
D-CAP-R	2.25 – 9.38 slot(s)	2.25 – 24.94 slot(s)	2.25 – 56.72 slot(s)	2.25 – 120.61 slot(s)

Table 8.2: Expected waiting time to send a CAP message on time slot level with NO-CAP-R, CAP-R, A-CAP-R, and D-CAP-R for $4 \leq MO \leq 7$.

The expected number of time slots to send a CAP message in NO-CAP-R is characterized by $N_{CAP}(SM, 8SM)$, which is independent of MO with a value of 2.25 slot(s). For CAP-R this value is given by $N_{CAP}(1,8)$. Hence, it depends on the difference between MO and SO. As shown in Tbl. 8.2, the expected waiting time to send a CAP message increase exponentially for an increasing MO. For A-CAP-R, N_{CAP} is calculated as an average of the expected times for CAP-R and NO-CAP-R, since the first phase of A-CAP-R is CAP-R and the second phase is NO-CAP-R. The value amounts to $0.5(N_{CAP}(SM, 8SM) + N_{CAP}(1,8))$. D-CAP-R dynamically adjusts the number of GTS between NO-CAP-R and CAP-R so that the expected times are bounded by the expected times of these mechanisms.

8.3.3 Expected Channel Access Time on Symbol Level

In the following, a single channel access without contention is considered where the corresponding packet is generated according to a uniform distribution on the interval $[0, D_{MSF})$. Two cases have to be considered:

- Packet generation during the CAP.
- Packet generation during the CFP.

For calculation, beacon slots are treated as CFP slots in which no data is sent.

Packet generation during the CAP: The probability for generating a packet during a CAP, p_{CAP}, is given by

$$p_{CAP} = \frac{CTM}{TM}. \tag{8.3}$$

The system can be discretized into time steps, $\mathcal{S}_{UB}$, of *aUnitBackoffPeriod*, the base duration of the CSMA/CA algorithm. If a packet is generated at the end of the

CAP, its transmission is more likely to be shifted to the next CAP. Therefore, the expected CSMA/CA backoff duration, $T_b(s)$, for a packet generated during the kth CSMA/CA slot is given by

$$T_b(s) = (2^{BE} - 1)^{-1} \sum_{i=0}^{i \leq 2^{BE}-1} B(s,i) \tag{8.4}$$

$$B(s,i) = \begin{cases} i \cdot \mathcal{S}_{UB} & \text{if } \mathcal{S}_{UB} \cdot (s+i) < \mathcal{S}_{CAP} \\ \eta \cdot \mathcal{S}_{SF} - \mathcal{S}_{UB} \cdot s & \\ +\mathcal{S}_{UB}(i+s) \; mod \; \mathcal{S}_{CAP} & \text{otherwise} \\ +\eta \cdot \mathcal{S}_{SF} \lfloor \frac{\mathcal{S}_{UB}(s+i)}{\mathcal{S}_{CAP}} \rfloor & \end{cases}, \tag{8.5}$$

where BE is the backoff exponent of the CSMA/CA algorithm, $\mathcal{S}_{SF} = 16 \cdot 60 \cdot 2^{SO}$ is the number of symbols per SF, $\mathcal{S}_{CAP} = \frac{CTM}{CM} \cdot 60 \cdot 2^{SO}$ is the number of symbols per CAP, and $\eta = \frac{SM}{CM}$ is a stretching factor if $CM < SM$ under the assumption that CAPs are evenly distributed in the MSF. In other words: Eq. (8.4) calculates the expected backoff from a given slot for all possible CSMA/CA backoff values. The total backoff, $B(s,i)$, equals to the CSMA/CA backoff ($i \cdot \mathcal{S}_{UB}$) if there is enough space in the CAP. Otherwise, it includes the wait duration of the CFPs. In this case (Eq. (8.5), case otherwise), the first summand is the backoff until the next CAP from the time of packet generation, the second summand is the remaining backoff time in a latter CAP, and the third term adds the backoff for multiple SFs if $i \cdot \mathcal{S}_{UB}$ spans over multiple CAPs. The expected channel access time, t_{ChCAP}, for a packet generation during the CAP can then be calculated as

$$t_{ChCAP} = \frac{\sum_{s=0}^{s<CTM/(\mathcal{S}_{UB} \cdot CM)} T_b(s)}{CTM/(\mathcal{S}_{UB} \cdot CM)}. \tag{8.6}$$

Packet generation during the CFP: When a packet is generated during the CFP, the expected channel access time is the combination of the expected time to send a CAP message, i.e. t_{CAP}, and the expected CSMA/CA backoff time from slot 0. The expected time to send a CAP message on symbol level is given by $t_{CAP} = 0.5 \cdot (t_{CAPmin} + t_{CAPmax})$, where $t_{CAPmin} = 1$ is the minimum time to send a CAP message and t_{CAPmax} is the maximum time to send a CAP message, and given by

$$t_{CAPmax} = \eta \cdot \mathcal{S}_{SF} - \mathcal{S}_{CAP}. \tag{8.7}$$

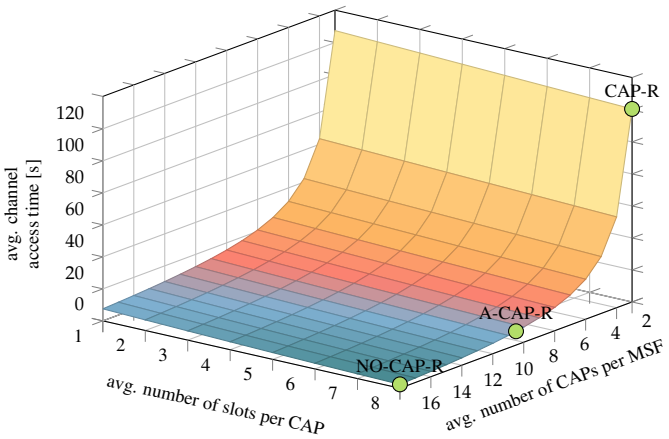


Figure 8.3: Expected channel access time of D-CAP-R for different values of *CM* and average number of slots per CAP for $SO = 3$ and $MO = 7$. The performance of the static CAP-R mechanisms is marked by the green dots for comparison.

The expected channel access time, t_{ChCFP}, for a packet generated during the CFP is $t_{ChCFP} = t_{CAP} + T_b(0)$.

Total Channel Access Time The total expected channel access time, t_{Ch}, is given by

$$t_{Ch} = p_{CAP} \cdot t_{ChCAP} + (1 - p_{CAP}) \cdot t_{ChCFP}. \tag{8.8}$$

The expected channel excess time for different values of *CM* and *CTM* is shown in Fig. 8.3. Here, one can see that A-CAP-R achieves a much lower expected channel access time than CAP-R. Therefore, the adaptability with A-CAP-R is higher. It can be seen that the expected channel access time scales exponentially with a decreasing CAP frequency. In contrast to that, D-CAP-R reduces the number of slots in individual CAPs. Therefore, the expected time to send a CAP message is always quite low. However, in extreme cases all CAP slots might be allocated, and D-CAP-R performs as CAP-R in terms of expected channel access time.

	SO	MO	BO	Q_{CAP}	Q_{CFP}	δ
Values	3	$\{4,\ldots,7\}$	7	8	22	$\{1,\ldots,4\}$

Table 8.3: Setup parameter for the simulative evaluation of D-CAP-R.

8.4 Simulative Evaluation

The following subsections present a performance assessment of D-CAP-R through simulation by comparing it with A-CAP-R, NO-CAP-R, and CAP-R. The following subsection, Sect. 8.4.1, describes the simulation setup. Sect. 8.4.2 first analyses DSME's reliability in scenarios with dynamically varying primary traffic, i.e., bursts with increasing number of packets. Afterward, Sect. 8.4.3 describes results for a scenario without bursts but decreasing packet generation interval. Here, especially the adaptability of the system is evaluated, e.g., by means of dwell time for GTS-negotiation messages in Sect. 8.4.3.1.

8.4.1 Scenario Description

For evaluation, a rooted binary tree with 31 nodes is utilized to reenact common data-collection scenarios, where data is gathered by all nodes and routed towards the sink for evaluation. We abstained from using an unbalanced tree to eliminate the influence of heavily skewed traffic.

Packet generation follows a Poisson distribution, where two scenarios are considered - the generation of one burst per second with ϵ packets, and the generation of 1 packet(s) in γ second intervals. The second scenario allows the assessment of the network close to saturation because traffic is stable. In the first scenario, the proposed CAP-R mechanisms have to (de)allocate GTS continuously and adjust to the current traffic demand of the network.

As discussed in Sect. 8.3, all CAP-R mechanisms only depend on the difference $MO - SO$. Therefore, either SO or MO can remain constant for simulation. We set $SO = 3$ so that a single packet with 127 B can be sent per GTS. Furthermore, it insures that all CAPs have the same length for all configurations. If not stated otherwise, we also fix $BO = 7$ to guarantee sufficiently many beacon slots in the network for all nodes to associate. The scenario is evaluated for $4 \leq MO \leq 7$. The configuration parameter relevant for this chapter are summarized in Tbl. 8.3 while Tbl. A.9 shows the full configuration.

8.4.2 Varying Burst Sizes

Fig. 8.4 shows the average PDR for an decreasing γ and different values of MO with one burst per second. For all methods, the PDR decreases for an increasing number of packets per burst. This is mainly due to the following reason: especially for smaller values of MO, there are not enough GTS per second to accommodate all packets generated during a burst, leading to dropped packets as the queues fill up.

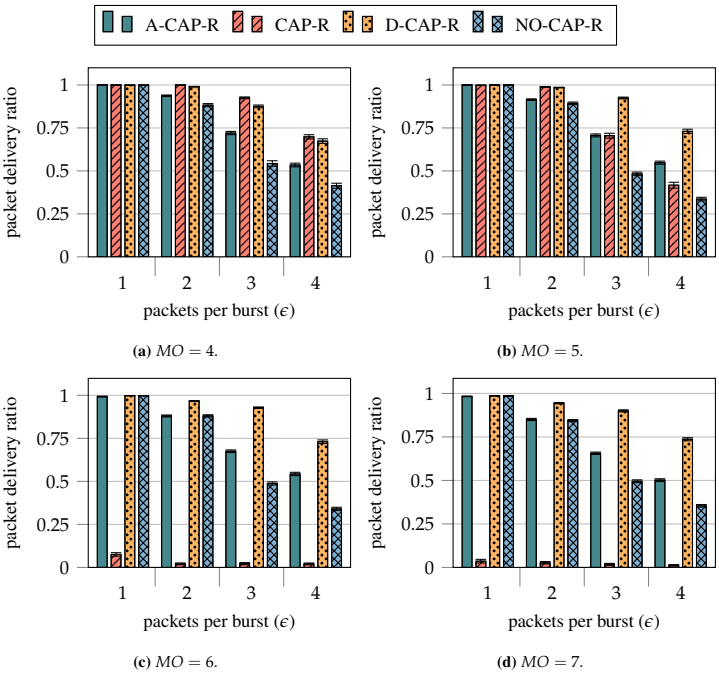
For CAP-R, the frequency of CAPs decreases for an increasing MO, resulting in more contention during the remaining CAPs. Thus, the required GTS cannot be allocated in time. This is the case for $MO = 5$, where $\tau_{CFP} = 81.25\,\%$. From this theoretical value, a high PDR is expected. However, the results show that for an increasing MO, CAP-R struggles to allocate slots in time.

The performance of NO-CAP-R is independent of MO, as already indicated by the theoretical analysis in Sect. 8.3.2. Therefore, the choice of CAP-R or NO-CAP-R depends largely on MO, as CAP-R performs better than NO-CAP-R for $MO = 4$ and $MO = 5$, but NO-CAP-R performs better for $MO = 6$ and $MO = 7$. If $SO = MO = 3$, all mechanisms perform equally well, since there is only a single SF per MSF, which cannot be reduced by D-CAP-R or CAP-R.

On the other hand, D-CAP-R performs similarly to CAP-R for $MO = 4$ and outperforms CAP-R and NO-CAP-R for $MO \geq 5$. That is because D-CAP-R starts with NO-CAP-R mode and therefore allocates GTS as fast as possible (e.g., as fast as NO-CAP-R) for larger values of MO, where there is only a small number of CAPs using CAP-R. In addition, it reduces CAPs when all regular GTS are already allocated so that it converges towards CAP-R. However, the allocation of these additional resources has a slight time overhead, resulting in the performance gap between D-CAP-R and CAP-R for $MO = 4$.

Similarly to CAP-R, the performance of A-CAP-R depends on MO but its influence is not as strong as for CAP-R. For smaller values of MO, A-CAP-R's PDR is about the average between the values for CAP-R and NO-CAP-R. It corresponds to the theoretical analysis regarding τ_{CFP}, with boundaries delimited by CAP-R and NO-CAP-R (e.g., $\tau_{CFP} = 56.25\,\%$ for $MO = 4$). For larger values of MO (i.e., $MO \geq 6$), the PDR is less sensible to this parameter, performing better than NO-CAP-R. This is because of the alternating behavior every BI, which provides enough CAPs to perform the required GTS negotiations.

All in all, D-CAP-R seems to be an attractive alternative to A-CAP-R and DSME's native operation modes under varying primary traffic patterns. It dynamically adapts



(a) $MO = 4$. **(b)** $MO = 5$.

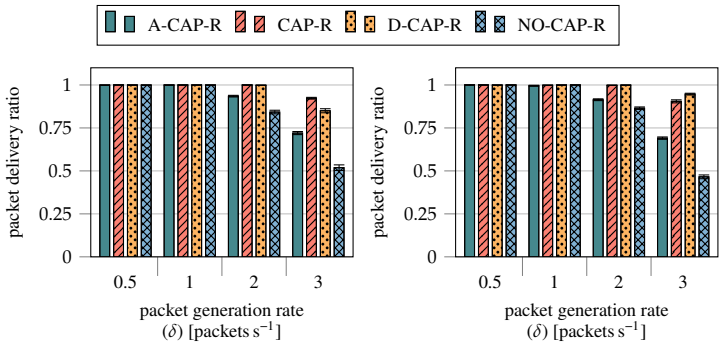
(c) $MO = 6$. **(d)** $MO = 7$.

Figure 8.4: PDR for varying numbers of packets per burst (ϵ) and different values of MO.

to the traffic demands of the network and (de)allocates additional GTS during CAPs to increase reliability.

8.4.3 Varying Packet Generation Intervals

The following figures show a performance assessment for different packet generation intervals. Particularly, Fig. 8.5 depicts the PDR for different values of MO. As explained in Sect. 8.4.2, the choice of CAP-R and NO-CAP-R largely depends on MO, with CAP-R performing better for $MO \leq 5$, and NO-CAP-R performing better for $MO \geq 6$. NO-CAP-R, A-CAP-R and D-CAP-R provide a good performance for $\delta \leq 1\,\text{packet(s)}\,\text{s}^{-1}$ and all values of MO. For $\delta > 1\,\text{packet(s)}\,\text{s}^{-1}$, NO-CAP-R is operating beyond the maximum capacity, and A-CAP-R starts to reach the limits of



(a) $MO = 4$.

(b) $MO = 5$.

(c) $MO = 6$.

(d) $MO = 7$.

Figure 8.5: PDR of D-CAP-R, A-CAP-R, CAP-R, and NO-CAP-R for different values of MO and δ.

available GTS. Therefore, the performance of NO-CAP-R and A-CAP-R diminishes. Contrary to NO-CAP-R, A-CAP-R's PDR is also slightly affected by increasing MO. D-CAP-R performs significantly better than the other CAP-R mechanisms, especially for larger values of MO. For example, it achieves a PDR of 95 % for $MO = 7$ and $\delta = 3\,\text{packet(s)}\,\text{s}^{-1}$, while NO-CAP-R achieves only about 48 %, A-CAP-R about 67 % and CAP-R less than 2 %.

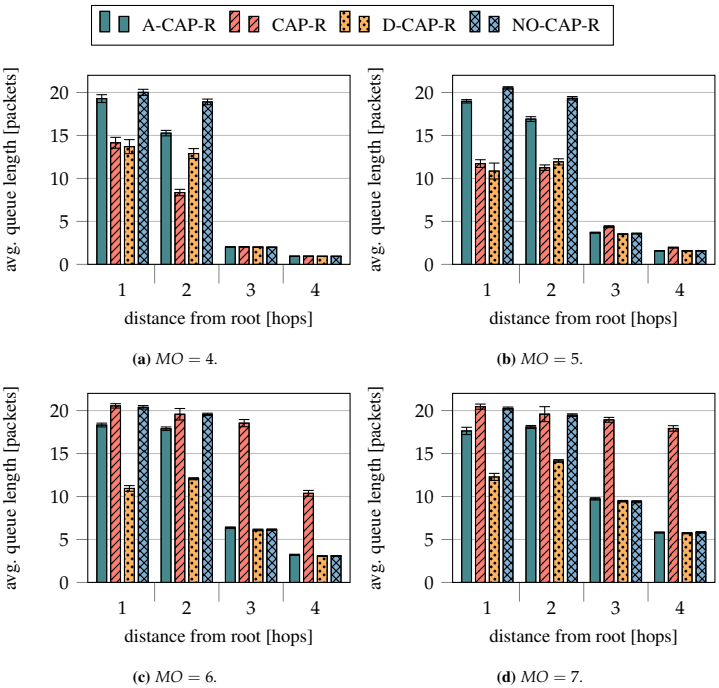
The main reason for lost packets are queue drops. This is also reflected in the average queue length, as shown in Fig. 8.6. Here, nodes with the same distance from the sink are grouped together, as nodes closer to the sink experience more traffic than nodes further away from the sink. With a maximum queue capacity

of $Q_{CFP} = 22\,\text{packet(s)}$, it is clearly visible that NO-CAP-R operates close to Q_{CFP} at nodes close to the sink for all values of MO. That is because the network is oversaturated for a packet generation interval of $\delta = 3\,\text{packet(s)}\,\text{s}^{-1}$, and no more GTS are available for allocation. For A-CAP-R, the network is close to the maximum capacity and therefore the average queue length for nodes close to the sink is significantly higher and close to Q_{CFP} (i.e., nodes up to two hops away from the sink node). This funneling effect intensifies for increasing MO, which reduces the frequency of CAPs per MSF in BIs in which A-CAP-R operates in CAP-R mode. On the other hand, the average queue length for CAP-R increases with an increasing MO because there are fewer CAPs per MSF. Consequently, the remaining CAPs are more congested and the required GTS are not allocated in time. The average queue length for D-CAP-R is lower than 68 % of the maximum queue capacity for all values of MO.

8.4.3.1 GTS Utilization

The maximum number of allocated RX- and TX-GTS by the different CAP-R mechanisms for $\delta = 3\,\text{packet(s)}\,\text{s}^{-1}$ and different values of MO is illustrated in Fig. 8.7. Nodes with the same distance to the sink are grouped together, as more GTS are allocated closer to the sink. For $MO = 4$, less slots need to be allocated because the MSF - and thus the schedule of allocated GTS - is repeated multiple times per packet generation interval. Theoretically, for $MO = 7$ and $\delta = 3\,\text{packet(s)}\,\text{s}^{-1}$, the required number of GTS at the sink node to successfully receive all packets generated in the network is about 180 GTS per BI (i.e., assuming that no packet was dropped from any queue). For NO-CAP-R and CAP-R, the highest number of available GTS per BI amounts to 112 GTS and 232 GTS, respectively. For A-CAP-R this value corresponds about 172 GTS per BI. However, given the funneling effect that is inherent to data collection scenarios this assumption does not hold. Many packets are lost at nodes closer the sink node and therefore the number of allocated GTS is slightly less than the theoretical value. That is the case for NO-CAP-R, in which nodes one hop further require about 174 GTS. However, only 112 GTS can be negotiated. Therefore, about 90 GTS are allocated at the sink node, which supports the obtained PDR with a value of 47 %.

Although the average number of 172 allocatable GTS using A-CAP-R should be sufficient to support primary traffic, and almost all slots are allocated, many queue drops occur at nodes close to the sink. That is because the number of usable GTS



(a) $MO = 4$. **(b)** $MO = 5$.

(c) $MO = 6$. **(d)** $MO = 7$.

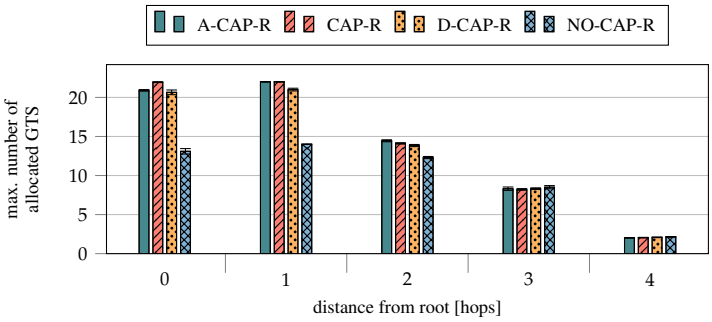
Figure 8.6: Average queue length per node for different values of MO and $\delta = 3$. Nodes with the same number of hops to the root are grouped together.

varies over a BI, i.e., 112 GTS in the first BI and 232 GTS in the other. During the first BI queues fill up and packets are lost.

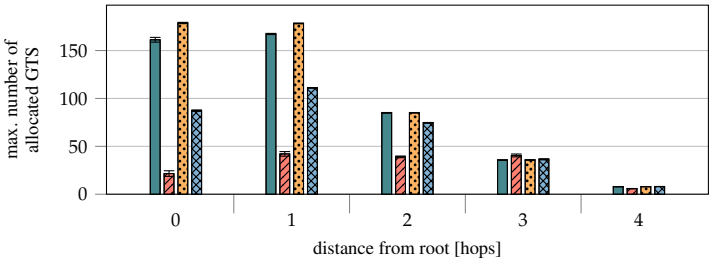
For D-CAP-R, the largest number of GTS is allocated. Theoretically, CAP-R could reach the same performance as D-CAP-R but it fails to allocate the GTS in time, as already explained. Still, D-CAP-R performs well for $MO \leq 7$ because it can allocate a large number of GTS at the start and then it successively reduces the CAPs to allocate the last remaining GTS.

8.4.3.2 Dwell Time

Evaluation of the average dwell time, as illustrated in Fig. 8.8, $\delta = 1\,\text{packet(s)}\,\text{s}^{-1}$ for messages sent during the GTS negotiation, i.e., *GTS request*, *GTS response*, and



(a) $MO = 4$.



(b) $MO = 7$.

Figure 8.7: Maximum number of allocated GTS per node for $MO = 4$, $MO = 7$, and $\delta = 3$. Nodes with the same number of hops to the root are grouped together.

GTS notify show that CAP-R has the highest dwell time. That is because there is only a single CAP per MSF, and GTS commands have to be queued for a long time. Dwell time exponentially increases for an increasing MO, as the duration of a MSF doubles when incrementing MO. Although A-CAP-R's dwell time is influenced by the length of the MSF, this effect is diminished during the BIs in which A-CAP-R operates in NO-CAP-R. It should be noted that A-CAP-R's waiting time is higher than D-CAP-R's and NO-CAP-R's. The validation of A-CAP-R's dwell time by simulation confirms the value estimated theoretically in Sect. 8.3. For $MO = 4$ it exhibits an average dwell time of 42.80 ms against 44.64 ms from the theoretical model. In this case, the difference between both values comes from the way the estimation in the theoretical model is performed (i.e., average between NO-CAP-R and CAP-R), i.e., scheduling is not considered.

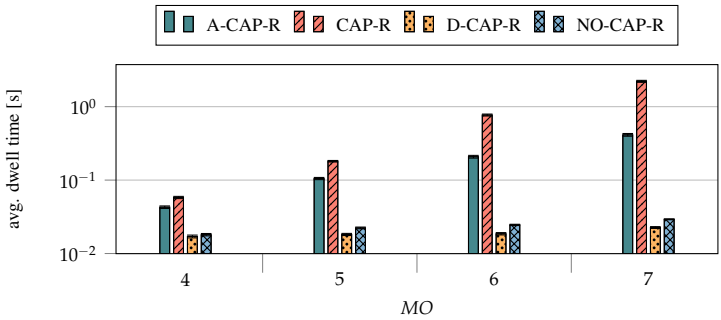


Figure 8.8: Average dwell time for GTS-negotiation messages using CAP-R, NO-CAP-R, A-CAP-R and D-CAP-R for different values of *MO* and a packet generation rate of 1 packet(s) s^{-1}.

D-CAP-R and NO-CAP-R perform equally well with average dwell times of 17.08 ms and 18.06 ms, respectively. For NO-CAP-R this matches with the value of about 17.28 ms from the theoretical models in 8.3, which do not consider packet collisions. NO-CAP-R has a higher dwell time than D-CAP-R because nodes are unable to allocate enough GTS but still can send GTS commands for the required GTS. This results in more congestion during the CAPs and the nodes backing off more frequently during the CSMA/CA algorithm. D-CAP-R and A-CAP-R are independent of *MO* - with the dwell time only marginally increasing for an increasing *MO* because more GTS are required initially for larger values of *MO*, resulting in more contention. For $MO = 3$ all mechanisms perform the same.

8.5 Discussion and Conclusion

It should be mentioned that D-CAP-R is expected to work exceptionally well with some of the previously described optimizations, in particular QMA (see Sect. 5.2), sending multiple packets per GTS (see Sect. 6.1), and scheduling through ILPs (see Sect. 7.1). QMA automatically learns which nodes have allocated GTS in the CAP through D-CAP-R and avoids transmission to them. At the same time, sending multiple packets per GTS reduces the number of required management messages, and thus allows higher adaptability when D-CAP-R is close to CAP-R mode. At last, longer paths can be scheduled using the proposed LPs decreasing dwell times even further. The results for all operation modes can be summarized as follows:

- **NO-CAP-R** provides good adaptability for all values of $MO - SO$. However, for a high number of packets per second ($\delta > 2$), NO-CAP-R cannot provide a sufficient number of GTS for sending all packets. This results in a large number of dropped packets due to full queues.

- **CAP-R** performs well for $MO - SO \leq 2$. Here, a single CAP per MSF is sufficient to allocate all required GTS and CAP-R still offers higher throughput, which increases the overall PDR. For $MO - SO \geq 3$, the performance of CAP-R diminishes since it is unable to allocate all GTS in time, resulting in large queues and packet loss.

- **A-CAP-R** offers a compromise between CAP-R and NO-CAP-R. For $MO - SO \leq 1$, it performs as a middle ground between CAP-R and NO-CAP-R. For $MO - SO \geq 2$, A-CAP-R starts to outperform the standard modes of DSME because it offers a higher adaptability than CAP-R and a higher throughput than NO-CAP-R.

- **D-CAP-R** achieves the best overall performance in terms of PDR, dwell time, and queue utilization for all values of $MO - SO$ and different packet generation rates. That is because D-CAP-R starts in NO-CAP-R mode which offers high adaptability. Then it gradually and dynamically reduces the CAP to allow for a higher throughput. Notably, it manages to achieve a PDR of over 80 % regardless of the value of $MO - SO$ in a scenario without bursts, and therefore outperforms the standard modes of DSME and A-CAP-R.

Finally, it should be said that D-CAP-R requires a powerful dynamic scheduler to fully exploit the possibilities it provides. Such scheduler is out of the scope of this work but a simple adaptation of the default scheduler, i.e., TPS, also provides outstanding results. We believe that D-CAP-R can be leveraged by future implementation of schedulers to make DSME more adaptable to bursty and dynamically changing primary traffic.

Chapter 9

Conclusion and Outlook

This dissertation explores and analyses techniques for adaptive multi-hop networks in the future IIoT with respect to time-varying and dynamically changing traffic. In this context, wireless communication between network participants is categorized as either primary or secondary traffic, whereby the former comprises data messages from the application layer, and the latter is composed of management messages to support primary traffic, e.g., for slot allocation or routing. Research is conducted from the perspective of the MAC layer IEEE 802.15.4, and evaluations are performed mostly through its sub-specification DSME, which offers a solid foundation for high agility by employing distributed scheduling and slot negotiation, and frequency diversity. Most of the proposed techniques, however, are easily transferable to other MAC protocols with alternating phases for primary and secondary traffic, e.g., TSCH.

The obtained results indicate that the adaptability of a network can be increased significantly by selecting appropriate MAC parameters; yet this is generally not sufficient to ensure a high reliability of primary traffic in the presence of time-varying traffic patterns, where transmission slots frequently have to be (de)allocated. This work demonstrates that additional techniques such as Machine Learning (ML)-based channel access or sending multiple packets per slot can close this performance gap by increasing throughput and reliability, and thus help operating networks closer to their capacity cap - especially with time-varying primary traffic.

In particular, we assess slot negotiation through a distributed 3-way handshake in the context of irregular primary traffic and identify it as a central bottleneck for high adaptivity in DSME. That is because slot (de)allocations cannot be performed in a timely manner when secondary traffic, which is usually realized through contention-based channel access, is oversaturated. Consequently, queues for primary traffic

grow and eventually overflow, eventuating in lost data packets and high delays. The reduction of the management message overhead for slot allocations is thus pivotal for good adaptability. We conceive three potential options: (de)allocating multiple slots through a single handshake, sending several packets per slot to attenuate the number of required slots, and developing a more reliable contention-based multiple access scheme for secondary traffic. The latter two options are further explored.

For more reliable channel access, QMA is developed - a contention-based multiple access scheme for secondary traffic relying on ML. Its core idea is to divide the secondary traffic phase into logical time slots and then dynamically learn at which times transmission is likely to be successful and when it is expected to lead to a collision. This way, QMA is capable of solving the hidden-node problem without additional overhead for RTS / CTS messages. Additionally, we demonstrate its effectiveness and scalability in comparison to slotted and unslotted CSMA/CA in topologies with an increasing number of nodes and on hardware, where it significantly outperforms the other two methods.

We also investigate sending multiple packets per slot with primary traffic, which proves to significantly reduce the number of conducted 3-way handshakes and hence contention during secondary traffic. Additionally, it maximizes potential throughput by minimizing the unutilized time at the end of each transmission slot. We believe that every DSME application should consider sending multiple packets per slot to increase adaptability in a simple way. The same should apply to TSCH. To strengthen this effect we further investigate GACKs and scheduling through ILPs and QUBOs.

Finally, D-CAP-R is proposed that allows using secondary traffic phases for primary traffic and therefore enables a flexible trade-off between periods of high primary traffic and high secondary traffic. Secondary traffic of other nodes takes place regularly and is not affected.

For future work, one can envision various techniques that increase adaptability; and application scenarios that require such techniques. One particularly interesting trend potentially constituting the next phase of the IoT is the tactile IoT, where touch is transmitted instead of text, audio, or video data. For example, it could enable engineers to intervene in industrial plants remotely using robotics. Obviously, the tactile IoT demands high reactivity, i.e., low latency and high throughput, and hence supplies an optimal playground for the techniques proposed in this work.

Another example is fog computing, where short-term tasks are directly performed on edge devices, i.e., IIoT nodes, and long-term analytics are performed in the cloud. Communication between both parts exhibits time-varying traffic patterns due to varying analysis times. It also requires strict timeliness if the results are used in a

control loop. These concepts will also enable the sustainable use of ML techniques in the IIoT, which can, e.g., be used for powerful schedulers that quickly react to changes in the primary traffic and, in the best case, even predict bursts.

In general, also the burgeoning pool of protocols in the 2.4 GHz frequency band poses novel challenges for existing MAC protocols due to mutual interference. Their coexistence is a hot topic in research and will presumably accompany us for years to come.

Overall, we believe that the techniques proposed and analyzed in this dissertation constitute a vital building block for future IIoT applications that are more resilient to external interference and time-varying and dynamically changing traffic.

Appendix

A.1 Markov Chain Transition Matrix for Transmitting Multiple Packets per GTS

The transition matrix Q in canonical form, corresponding to the Markov chain for GTS negotiation through the 3-way handshake, as introduced in Sect. 5.1.2.1, describes the probabilities for transitioning from one transient state to another. It is defined as

$$Q = \begin{pmatrix}
0 & p & 0 & 1-p & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & p & 0 & 0 & 0 & 1-p & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-p & 0 & 0 \\
0 & p & 0 & 0 & 1-p & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & p & 0 & 0 & 0 & 1-p & 0 & 0 & 0 & 0 & 0 & 0 \\
1-p & p & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & p & 0 & 0 & 0 & 0 & 1-p & 0 & 0 & 0 & 0 \\
0 & 0 & p & 0 & 0 & 0 & 0 & 0 & 1-p & 0 & 0 & 0 \\
1-p & 0 & p & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-p & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-p \\
1-p & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}. \tag{A.1}$$

A.2 Convergence Example for QMA

		timestep 0				timestep 1000				timestep 2000			
		S_0	S_1	S_2	S_3	S_0	S_1	S_2	S_3	S_0	S_1	S_2	S_3
v_1	B	-10	-10	-10	-10	6.498	7.203	7.986	0.103	6.554	18.603	20.677	0.103
	C	-10	-10	-10	-10	5.539	2.709	-10	8.857	19.736	2.709	0.5	20.759
	S	-10	-10	-10	-10	-10	-10	-10	-2.946	-10	0.56	14.293	9.391
v_2	B	-10	-10	-10	-10	4.289	4.759	5.282	5.862	17.002	16.684	5.259	17.295
	C	-10	-10	-10	-10	2.997	-10	2.417	-10	7.973	-1.141	18.552	-10.0
	S	-10	-10	-10	-10	-2.289	-10	-10	-10	-2.289	-10.0	-0.364	4.689

		timestep 3000				timestep 4000				timestep 5000			
		S_0	S_1	S_2	S_3	S_0	S_1	S_2	S_3	S_0	S_1	S_2	S_3
v_1	B	13.797	24.404	24.902	8.959	13.797	24.475	24.972	8.959	18.923	26.764	27.524	8.959
	C	24.955	12.175	12.688	25.455	25.028	20.65	12.688	25.525	27.08	21.964	12.688	26.393
	S	5.371	11.607	14.293	18.708	5.371	11.607	20.627	22.465	5.371	16.586	24.377	28.368
v_2	B	27.502	16.519	5.259	26.747	27.589	16.519	5.259	26.83	27.589	16.519	14.638	26.83
	C	18.196	-1.141	27.062	7.414	20.816	-1.141	27.147	7.414	20.816	13.143	27.147	23.488
	S	-2.289	28.346	9.641	4.689	-2.289	28.433	18.894	4.689	21.619	28.432	18.894	4.689

		timestep 7000				timestep 10000				timestep 20000			
		S_0	S_1	S_2	S_3	S_0	S_1	S_2	S_3	S_0	S_1	S_2	S_3
v_1	B	21.478	26.618	27.38	8.959	21.478	28.95	29.944	8.959	26.455	28.711	29.722	27.427
	C	26.932	23.665	18.083	25.982	28.049	23.665	22.587	26.258	28.021	27.504	28.147	29.198
	S	16.67	16.586	23.482	28.227	30.055	20.236	25.866	31.049	29.826	23.397	24.952	30.83
v_2	B	29.942	25.702	24.386	28.948	29.942	25.702	24.3862	28.948	29.597	27.017	25.891	28.617
	C	22.203	23.86	25.147	24.659	26.311	28.502	28.428	27.122	26.129	27.868	26.97	25.216
	S	21.619	31.032	30.038	24.053	25.949	31.047	30.053	23.95	24.881	30.707	29.718	23.184

Table A.1: Example for the convergence of QMA over 20000 time steps with a small exploration rate of $\rho = 0.01$ for two nodes v_1 and v_2, four subslots, and actions *QBackoff* (B), *QCCA* (C), and *QSend* (S). The policy for each timestep is highlighted. QMA finds a feasible policy after 2000 timesteps, albeit finding an optimal policy requires 7000 timesteps. The Q-values converge to a value of about 31 and do not become higher than 32 even after 100000 timesteps.

A.3 An Alternative Formulation of $\mathcal{LP}_6$

As discussed in Sect. 7.1.5, $\mathcal{LP}_6$ yields a delay-minimal schedule for traffic with random arrival or generation time. However, it requires $\mathbb{V}(K^2+3K)$ binary optimization variables and is hence not scalable to high values of K as discussed in Sect. 7.2. Yet, the number of required optimization variables can be reduced to $\sum_{v_i \in \mathbb{V}\setminus\{v_0\}} \lceil \delta_i \gamma_i \rceil (K+1)$ by precalculating paths through the network. We assume that the binary optimization variables x_{ikl} are defined as in Eq. (7.15).

Let $\mathbb{P} = \{\rho_i \mid v_i \in \mathbb{V} \setminus \{v_0\}\}$ be the set of all paths in the network, where each node $v_i \in \mathbb{V} \setminus \{v_0\}$ requires $\tau_i = \lceil \delta_i \gamma_i \rceil$ transmission slots. The objective function minimizes the total delay over all paths

$$\mathbf{O_3}: \quad \min \quad D_S\Big(\sum_{\rho_i \in \mathbb{P}} f(RX_i) - f(TX_i) + K \sum_{(v_m,v_n)\in\rho_i} \theta_{im}\Big), \tag{A.2}$$

where f(x), RX_i, TX_i, and θ_{im} are defined as in Sect. 7.1.5. That means for each hop $v_m \mid (v_m, v_n) \in \rho_i \in \mathbb{P}$, $\theta_{im} = 1$ indicates that a packet is not forwarded in the same MSF. Otherwise $\theta_{im} = 0$ so that $\theta_i = \sum_{(v_m,v_n)\in\rho_i} \theta_{im}$.

In contrast to Sect. 7.1.5, every node has $\tau_{|\mathbb{V}|}$ rows with K binary optimization variables instead of K rows with K binary optimization variables. They are subject to the constraints

$$\forall_{i\neq 0, l}: \quad \sum_k x_{ikl} = 1, \tag{A.3}$$

$$\forall_{i,k}: \quad \sum_l x_{ikl} + \sum_{j\in N_i^-, l} x_{jkl} \leq 1 \tag{A.4}$$

Here, Eq. (A.3) ensures that exactly one slots is allocated per row, while Eq. (A.4) is equivalents to 7.4 and ensures that a node v_i and its children cannot transmit at the same time. Therefore, the absolute index TX_i of the l^{th} transmission slot at a node v_i is given by

$$TX_i = \sum_k x_{ikl} k.$$

This enables the calculation of θ_{im} for every v_m on the path ρ_i using the constraint

$$\forall_{\rho_i \in P, (v_m,v_n)\in\rho_i}: \quad \sum_k x_{mk\mu(\rho_i,v_m)} k < \theta_{im} K + \sum_k x_{nk\mu(\rho_i,v_n)} k. \tag{A.5}$$

Algorithm 3: Round-robin assignment of transmission slot indices $\mu(\rho_i, v_j)$ for all paths and creation of a set of path $\mathbb{P}$ for a given routing tree T.

Input: T, δ_i
Output: $\mathbb{P}, \mu(\rho_i, v_j)$

initialize $T\text{'} \leftarrow T, \mathbb{P} \leftarrow \emptyset$
initialize $\forall_{v_i \in \mathbb{V}}: \quad S(v_i) \leftarrow 0$
initialize $\forall_{v_j \in \mathbb{V}, \rho}: \quad \mu(\rho_i, v_j) \leftarrow 0$
for $0 \leq y < |\mathbb{V}| - 1$ **do**
  $v_j \leftarrow$ random from $\{v_m \in \mathbb{V} \mid N_m^- = \emptyset \wedge v_m \neq v_0\}$
  for $0 \leq z < \lceil \delta_j \gamma_j \rceil$ **do**
    $\mu(\rho_j, v_j) \leftarrow S(v_j)$
    $S(v_j) \leftarrow (S(v_j) + 1) \mod \lceil \delta_j \gamma_j \rceil$
    $\rho_j \leftarrow \emptyset, v \leftarrow v_j$
    while $v \neq v_0$ **do**
      $\rho_j \leftarrow \rho_j \cup \{(v, N_v^+)\}$
      $\mu(\rho_j, N_v^+) \leftarrow S(N_v^+)$
      $S(N_v^+) \leftarrow (S(N_v^+) + 1) \mod \lceil \delta_{N_v^+} \gamma_{N_v^+} \rceil$
      $v \leftarrow N_v^+$
    end
    $\mathbb{P} \leftarrow \mathbb{P} \cup \{\rho_j\}$
  end
  $T\text{'} \leftarrow T\text{'} \setminus \{v_j\}$
end

Here, $\mu(\rho_i, v_m)$ is a mapping from a hop v_m on a path ρ_i to the index of the row the transmission slot is used in, i.e., the l^{th} transmission slot that is used for the transmission on that hop. It guarantees that $\theta_{im} = 1$ if $TX_m > TX_n$ for every $(v_m, v_n) \in \rho_i$.

The paths and the mapping$\mu(\rho_i, v_m)$ are not formed by the LP but are pre-calculated, as exemplified in Alg. 3. For this let T' be a copy of T. Starting with any leaf $v_i \in T\text{'} \mid N_i^- = \emptyset$, paths are created by cyclically selecting a transmission slot from a list of available transmission slots at every node $v_m \mid (v_m, v_n) \in \rho_i$ that retransmits δ_m on the way to v_0. This is repeated for every transmission slot of v_i that is not incorporated in a path yet. Then v_i is removed from T'. Next, another leaf is selected until $T\text{'} = \{v_0\}$. The solution of this formulation is equivalent to $\mathcal{LP}_6$ but slightly more efficient.

A.4 Detailed Evaluation Parameter Description

Parameter	Value	Parameter	Value	Parameter	Value
Network parameters					
topology	line/grid	# nodes ($\|\mathbb{V}\|$)	2...49	repetitions	20
Traffic parameters					
distribution	normal	# packets	1000	payload (S)	127 B
interval (γ)	7.864 s	burst (ϵ)	1...32	start duration	15 s
warmup duration	0 s	cooldown duration	100 s		
Protocol parameters					
SO	3...7	*MO*	7...9	*BO*	9
primary queue	8 packet(s)	secondary queue	22 packet(s)	CAP-R	no
CSMA/CA parameters					
minBE	3	*maxBE*	5		

Table A.2: Detailed evaluation parameter for the evaluation of DSME's slot (de)allocation handshake.

Parameter	Value	Parameter	Value	Parameter	Value
Network parameters					
topology	hidden node/concentric	# nodes ($\|\mathbb{V}\|$)	3/7...91	repetitions	15
Traffic parameters					
distribution	Poisson	# packets	1000	payload (S)	127 B
interval (γ)	1...0.01/1 and 0.1 s	burst (ϵ)	1	start duration	100...100.2/ s
warmup duration	0/300 s	cooldown duration	0/100 s		
Protocol parameters					
SO	3	*MO*	6	*BO*	9
primary queue	8 packet(s)	secondary queue	22 packet(s)	CAP-R	no
CSMA/CA parameters					
minBE	3	*maxBE*	5		

Table A.3: Detailed evaluation parameter for the evaluation of QMA.

Parameter	Value	Parameter	Value	Parameter	Value
Network parameters					
topology	grid	# nodes ($\|\mathbb{V}\|$)	16	repetitions	20
Traffic parameters					
distribution	normal	# packets	-	payload (S)	127 B
interval (γ)	16 s	burst (ϵ)	2...40	start duration	300...316 s
warmup duration	0 s	cooldown duration	0 s		
Protocol parameters					
SO	3...6	*MO*	9	*BO*	9
primary queue	40 packet(s)	secondary queue	40 packet(s)	CAP-R	no
CSMA/CA parameters					
minBE	3	*maxBE*	5		

Table A.4: Detailed evaluation parameters for evaluation of sending multiple packets per GTS.

Parameter	Value	Parameter	Value	Parameter	Value
Network parameters					
topology	star	# nodes ($\|\mathbb{V}\|$)	10	repetitions	20
Traffic parameters					
distribution	normal	# packets	1000	payload (S)	1 B
interval (γ)	1...0.015625 s	burst (ϵ)	1	start duration	100 s
warmup duration	100 s	cooldown duration	100 s		
Protocol parameters					
SO	4	*MO*	7	*BO*	7
primary queue	8 packet(s)	secondary queue	22 packet(s)	CAP-R	no
CSMA/CA parameters					
minBE	3	*maxBE*	5		
Other					
GAO	7				

Table A.5: Detailed evaluation parameters for best-case evaluation of GACKs.

Parameter	Value	Parameter	Value	Parameter	Value
Network parameters					
topology	binary tree	# nodes ($\|\mathbb{V}\|$)	31	repetitions	20
Traffic parameters					
distribution	Poisson	# packets	1000	payload (S)	rand 53…127 B
interval (γ)	0.9…0.05 s	burst (ϵ)	1	start duration	100 s
warmup duration	100 s	cooldown duration	100 s		
Protocol parameters					
SO	4	*MO*	6	*BO*	8
primary queue	8 packet(s)	secondary queue	22 packet(s)	CAP-R	no
CSMA/CA parameters					
minBE	3	*maxBE*	5		
Other					
GAO	6				

Table A.6: Detailed evaluation parameters for average-case evaluation of GACKs.

Parameter	Value	Parameter	Value	Parameter	Value
Network parameters					
topology	line	# nodes ($\|\mathbb{V}\|$)	10	repetitions	20
Traffic parameters					
distribution	Poisson	# packets	1000	payload (S)	127 B
interval (γ)	0.9…0.05 s	burst (ϵ)	1	start duration	100 s
warmup duration	100 s	cooldown duration	100 s		
Protocol parameters					
SO	3	*MO*	6	*BO*	8
primary queue	8 packet(s)	secondary queue	22 packet(s)	CAP-R	no
CSMA/CA parameters					
minBE	3	*maxBE*	5		
Other					
GAO	6				

Table A.7: Detailed evaluation parameters for worst-case evaluation of GACKs.

Parameter	Value	Parameter	Value	Parameter	Value
Network parameters					
topology	multi-line	# nodes ($\|\mathbb{V}\|$)	4…46	repetitions	20
Traffic parameters					
distribution	Poisson	# packets	1000	payload (S)	127 B
interval (γ)	1 s	burst (ϵ)	1	start duration	100 s
warmup duration	100 s	cooldown duration	100 s		
Protocol parameters					
SO	3	MO	6	BO	8
primary queue	8 packet(s)	secondary queue	22 packet(s)	CAP-R	no/yes
CSMA/CA parameters					
$minBE$	3	$maxBE$	5		
Other					
GAO	6				

Table A.8: Detailed evaluation parameters for scheduling through Integer Linear Program (ILP).

Parameter	Value	Parameter	Value	Parameter	Value
Network parameters					
topology	binary tree	# nodes ($\|\mathbb{V}\|$)	31	repetitions	20
Traffic parameters					
distribution	Poisson	# packets	1000	payload (S)	127 B
interval (γ)	1/2…0.25 s	burst (ϵ)	1…4/1	start duration	rand. 450…452 s
warmup duration	200 s	cooldown duration	200 s		
Protocol parameters					
SO	3	MO	4…7	BO	7
primary queue	8 packet(s)	secondary queue	22 packet(s)	CAP-R	no/yes
CSMA/CA parameters					
$minBE$	3	$maxBE$	5		
Other					
GAO	6				

Table A.9: Detailed evaluation parameter for the trade-off of primary and secondary traffic.

Bibliography

[AAB+15] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. *TensorFlow: Large-scale Machine Learning on Heterogeneous Systems*, Nov. 2015. Software available from tensorflow.org.

[AAG+21] A. Ahmadzadeh, B. Aydin, M. K. Georgoulis, D. J. Kempton, S. S. Mahajan, and R. A. Angryk. *How to Train Your Flare Prediction Model: Revisiting Robust Sampling of Rare Events*. In: *The Astrophysical Journal Supplement Series*, vol. 254(2), p. 23, May 2021. http://dx.doi.org/10.3847/1538-4365/abec88.

[AAT+19] N. Aihara, K. Adachi, O. Takyu, M. Ohta, and T. Fujii. *Q-learning Aided Resource Allocation and Environment Recognition in LoRaWAN with CSMA/CA*. In: *IEEE Access*, vol. 7, pp. 152126–152137, Oct. 2019. http://dx.doi.org/10.1109/access.2019.2948111.

[ABF+15] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, J. Vandaele, and T. Watteyne. *FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed*. In: *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pp. 459–464. IEEE, Milan, Italy, Dec. 2015. http://dx.doi.org/10.1109/wf-iot.2015.7389098.

[ABM20] L. Alkama and L. Bouallouche-Medjkoune. *IEEE 802.15.4 Historical Revolution Versions: A Survey*. In: *Computing*, vol. 103(1), pp. 99–131, Oct. 2020. http://dx.doi.org/10.1007/s00607-020-00844-3.

[AGS+21] S. Amin, M. Gomrokchi, H. Satija, H. van Hoof, and D. Precup. *A Survey of Exploration Methods in Reinforcement Learning*. In:

arXiv:2109.00157, Sep. 2021. http://dx.doi.org/10.48550/ARXIV.2109.00157.

[AI19] A. Ariza and V. Inzillo. *INETMANET Framework*. In: A. Virdis and M. Kirsche (eds.), *Recent Advances in Network Simulation*, pp. 107–138. Springer International Publishing, Cham, May 2019. http://dx.doi.org/10.1007/978-3-030-12842-5_3.

[AJO+18] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad. *State-of-the-art in Artificial Neural Network Applications: A Survey*. In: *Heliyon*, vol. 4(11), p. e00938, Nov. 2018. http://dx.doi.org/10.1016/j.heliyon.2018.e00938.

[ÁKSW21a] J. Álamos, P. Kietzmann, T. C. Schmidt, and M. Wählisch. *Poster: DSME-LoRa – a Flexible MAC for LoRa*. In: *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. IEEE, Nov. 2021. http://dx.doi.org/10.1109/icnp52444.2021.9651945.

[ÁKSW21b] J. Álamos, P. Kietzmann, T. C. Schmidt, and M. Wählisch. *Wip: Exploring DSME MAC for LoRa – a System Integration and First Evaluation*. In: *arXiv:2112.09403*, Dec. 2021. http://dx.doi.org/10.48550/ARXIV.2112.09403.

[ALNT14] M. A. Alsheikh, S. Lin, D. Niyato, and H.-P. Tan. *Machine Learning in Wireless Sensor Networks: Algorithms, Strategies, and Applications*. In: *IEEE Communications Surveys & Tutorials*, vol. 16(4), pp. 1996–2018, Apr. 2014. http://dx.doi.org/10.1109/COMST.2014.2320099.

[ALZ+18] S. Anamalamudi, B. Liu, M. Zhang, A. Sangi, C. Perkins, and S. V. R. Anand. *Scheduling Function One (SF1): Hop-by-Hop Scheduling with RSVP-TE in 6TiSCH Networks*. In: *Draft, IETF*, Apr. 2018.

[AM08] F. Abtahi and M. R. Meybodi. *Solving Multi-agent Markov Decision Processes using Learning Automata*. In: *2008 6th International Symposium on Intelligent Systems and Informatics*, pp. 1–6. IEEE, Nov. 2008. http://dx.doi.org/10.1109/sisy.2008.4664909.

[APMB15] G. Alderisi, G. Patti, O. Mirabella, and L. L. Bello. *Simulative Assessments of the IEEE 802.15.4e DSME and TSCH in Realistic Process Automation Scenarios*. In: *2015 IEEE 13th International Conference on Industrial Informatics (INDIN)*, pp. 948–955. IEEE, Oct. 2015. http://dx.doi.org/10.1109/indin.2015.7281863.

[ARV+19] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. G. Katzgraber. *Physics-inspired Optimization for Quadratic Unconstrained Problems Using a Digital Annealer*. In: *Frontiers*

in Physics, vol. 7, Apr. 2019. http://dx.doi.org/10.3389/fphy.2019.00048.

[ASHAM18] G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz. *A Survey on 5G Networks for the Internet of Things: Communication Technologies and Challenges*. In: *IEEE Access*, vol. 6, pp. 3619–3647, Dec. 2018. http://dx.doi.org/10.1109/access.2017.2779844.

[ASLM12] I. Amdouni, R. Soua, E. Livolant, and P. Minet. *Delay Optimized Time Slot Assignment for Data Gathering Applications in Wireless Sensor Networks*. In: *International Conference on Wireless Communications in Unusual and Confined Areas (ICWCUCA)*, pp. 1–6. IEEE, Jan. 2012. http://dx.doi.org/10.1109/icwcuca.2012.6402475.

[ATHD13] N. Abdeddaim, F. Theoleyre, M. Heusse, and A. Duda. *Adaptive IEEE 802.15.4 MAC for Throughput and Energy Optimization*. In: *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, pp. 223–230. IEEE, May 2013. http://dx.doi.org/10.1109/dcoss.2013.44.

[AYC17] M. S. Akbar, H. Yu, and S. Cang. *TMP: Tele-Medicine Protocol for Slotted 802.15.4 With Duty-Cycle Optimization in Wireless Body Area Sensor Networks*. In: *IEEE Sensors Journal*, vol. 17(6), pp. 1925–1936, mar 2017. http://dx.doi.org/10.1109/jsen.2016.2645612.

[BBVC14] N. Barroca, L. M. Borges, F. J. Velez, and P. Chatzimisios. *Block Acknowledgment in IEEE 802.15.4 by Employing DSSS and CSS PHY Layers*. In: *2014 IEEE Globecom Workshops (GC Wkshps)*. IEEE, Dec. 2014. http://dx.doi.org/10.1109/glocomw.2014.7063524.

[BCL+20] F. Battaglia, M. Collotta, L. Leonardi, L. Lo Bello, and G. Patti. *Novel Extensions to Enhance Scalability and Reliability of the IEEE 802.15.4-DSME Protocol*. In: *Electronics*, vol. 9(1), p. 126, Jan. 2020. http://dx.doi.org/10.3390/electronics9010126.

[BGH+18] E. Baccelli, C. Gündoğan, O. Hahm, P. Kietzmann, M. S. Lenders, H. Petersen, K. Schleiser, T. C. Schmidt, and M. Wählisch. *RIOT: An Open Source Operating System for Low-end Embedded Devices in the IoT*. In: *IEEE Internet of Things Journal*, vol. 5(6), pp. 4428–4440, Mar. 2018. http://dx.doi.org/10.1109/JIOT.2018.2815038.

[BK16] S. Backhauß and M. Köstler. *Development of a TDMA MAC Layer*. Master's thesis, Hamburg University of Technology, Hamburg, Feb. 2016.

[BYSMK18] H. Bayat-Yeganeh, V. Shah-Mansouri, and H. Kebriaei. *A Multi-state Q-learning Based CSMA MAC Protocol for Wireless Networks*.

In: *Wireless Networks*, vol. 24(4), pp. 1251–1264, Nov. 2018. http://dx.doi.org/10.1007/s11276-016-1402-0.

[CCD08] P. Cunningham, M. Cord, and S. J. Delany. *Supervised Learning*. In: M. Cord and P. Cunningham (eds.), *Machine Learning Techniques for Multimedia*, pp. 21–49. Springer Berlin Heidelberg, Berlin, Heidelberg, Feb. 2008. http://dx.doi.org/10.1007/978-3-540-75171-7_2.

[CKM+15] Y. Chu, S. Kosunalp, P. D. Mitchell, D. Grace, and T. Clarke. *Application of Reinforcement Learning to Medium Access Control for Wireless Sensor Networks*. In: *Engineering Applications of Artificial Intelligence*, vol. 46, pp. 23–32, Nov. 2015. http://dx.doi.org/10.1016/j.engappai.2015.08.004.

[CMG12] Y. Chu, P. D. Mitchell, and D. Grace. *ALOHA and Q-learning Based Medium Access Control for Wireless Sensor Networks*. In: *2012 International Symposium on Wireless Communication Systems (ISWCS)*, pp. 511–515. IEEE, IEEE, Aug. 2012. http://dx.doi.org/10.1109/iswcs.2012.6328420.

[CMGL05] S. Cui, R. Madan, A. Goldsmith, and S. Lall. *Energy-delay Tradeoffs for Data Collection in TDMA-based Sensor Networks*. In: *IEEE International Conference on Communications (ICC)*, vol. 5, pp. 3278–3284. IEEE, IEEE, Aug. 2005. http://dx.doi.org/10.1109/icc.2005.1495029.

[CMML20] N. Choudhury, R. Matam, M. Mukherjee, and J. Lloret. *A Performance-to-cost Analysis of IEEE 802.15.4 MAC with 802.15.4e MAC Modes*. In: *IEEE Access*, vol. 8, pp. 41936–41950, Feb. 2020. http://dx.doi.org/10.1109/access.2020.2976654.

[CNB12] S. Chakrabarti, E. Nordmark, and C. Bormann. *Neighbor Discovery Optimization for IPv6 Over Low-power Wireless Personal Area Networks (6LoWPANs). Tech. rep.*, IETF, Nov. 2012. RFC 6775, http://dx.doi.org/10.17487/rfc6775.

[CNM06] R. Caruana and A. Niculescu-Mizil. *An Empirical Comparison of Supervised Learning Algorithms*. In: *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 161–168. ACM Press, New York, NY, USA, Jun. 2006. http://dx.doi.org/10.1145/1143844.1143865.

[CVV+21] T. Chang, M. Vucinic, X. Vilajosana, S. Duquennoy, and D. Dujovne. *6TiSCH Minimal Scheduling Function (MSF)*. IETF, May 2021. RFC 9033, http://dx.doi.org/10.17487/RFC9033.

[CWWV16] T. Chang, T. Watteyne, Q. Wang, and X. Vilajosana. *LLSF: Low Latency Scheduling Function for 6TiSCH Networks*. In: *2016 International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 93–95. IEEE, May 2016. http://dx.doi.org/10.1109/dcoss.2016.10.

[CXQ+18] C. Chen, T. Xiao, T. Qiu, H. Zhao, L. Liu, and J. Lv. *GAS: A Group Acknowledgement Strategey in Internet of Vehicles*. In: *2018 IEEE International Conference on Smart Internet of Things (SmartIoT)*, pp. 1–8. IEEE, Aug. 2018. http://dx.doi.org/10.1109/smartiot.2018.00011.

[DAMH21] G. Dulac-Arnold, D. Mankowitz, and T. Hester. *Challenges of Real-world Reinforcement Learning: Definitions, Benchmarks and Analysis*. In: *arXiv:1904.12901*, vol. 110(9), pp. 2419–2468, Apr. 2021. http://dx.doi.org/10.1007/s10994-021-05961-4.

[DANLW15] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne. *Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH*. In: *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, pp. 337–350. ACM, New York, NY, USA, Nov. 2015. http://dx.doi.org/10.1145/2809695.2809714.

[DCM18] D. Delahaye, S. Chaimatanan, and M. Mongeau. *Simulated Annealing: From Basics to Applications*. In: *Handbook of Metaheuristics*, pp. 1–35. Springer International Publishing, Sep. 2018. http://dx.doi.org/10.1007/978-3-319-91086-4_1.

[DEA06] I. Demirkol, C. Ersoy, and F. Alagoz. *MAC Protocols for Wireless Sensor Networks: A Survey*. In: *IEEE Communications Magazine*, vol. 44(4), pp. 115–121, Apr. 2006. http://dx.doi.org/10.1109/MCOM.2006.1632658.

[DGAND+17] D. De Guglielmo, B. Al Nahas, S. Duquennoy, T. Voigt, and G. Anastasi. *Analysis and Experimental Evaluation of IEEE 802.15.4e TSCH CSMA-CA Algorithm*. In: *IEEE Transactions on Vehicular Technology*, vol. 66(2), pp. 1573–1588, Feb. 2017. http://dx.doi.org/10.1109/tvt.2016.2553176.

[DGPA17] D. R. Dujovne, L. A. Grieco, M. R. Palattella, and N. Accettura. *6TiSCH 6top Scheduling Function Zero (SF0). Internet-Draft draft-ietf-6tisch-6top-sf0-05*, IETF, Jul. 2017.

[DGPA18] D. R. Dujovne, L. A. Grieco, M. R. Palattella, and N. Accettura. *6TiSCH Experimental Scheduling Function (SFX). Internet-Draft draft-ietf-6tisch-6top-sfx-01*, IETF, Mar. 2018.

[Die20] J. N. Diercks. *Energy Efficient Acknowledgements in IEEE 802.15.4 DSME*. Bachelor's thesis, Hamburg University of Technology, Sep. 2020. Bachelor's thesis.

[DLL+20] D.-J. Deng, S.-Y. Lien, C.-C. Lin, M. Gan, and H.-C. Chen. *IEEE 802.11ba Wake-up Radio: Performance Evaluation and Practical Designs*. In: *IEEE Access*, vol. 8, pp. 141547–141557, Jul. 2020. http://dx.doi.org/10.1109/access.2020.3013023.

[DNRC20] J. Ding, M. Nemati, C. Ranaweera, and J. Choi. *IoT Connectivity Technologies and Applications: A Survey*. In: *IEEE Access*, vol. 8, pp. 67646–67673, Apr. 2020. http://dx.doi.org/10.1109/access.2020.2985932.

[DSLF18] G. Daneels, B. Spinnewyn, S. Latré, and J. Famaey. *ReSF: Recurrent Low-latency Scheduling in IEEE 802.15.4e TSCH Networks*. In: *Ad-Hoc Networks*, vol. 69, pp. 100–114, Feb. 2018. http://dx.doi.org/10.1016/j.adhoc.2017.11.002.

[DVW18] S. Duquennoy, X. Vilajosana, and T. Watteyne. *6TiSCH Autonomous Scheduling Function (ASF). Internet-Draft draft-duquennoy-6tisch-asf-01*, IETF, Mar. 2018.

[EAK+19] M. Elbes, S. Alzubi, T. Kanan, A. Al-Fuqaha, and B. Hawashin. *A Survey on Particle Swarm Optimization with Emphasis on Engineering and Network Applications*. In: *Evolutionary Intelligence*, vol. 12(2), pp. 113–129, Feb. 2019. http://dx.doi.org/10.1007/s12065-019-00210-z.

[FAC+11] M. D. Francesco, G. Anastasi, M. Conti, S. K. Das, and V. Neri. *Reliability and Energy-efficiency in IEEE 802.15.4 / ZigBee Sensor Networks: An Adaptive and Cross-layer Approach*. In: *IEEE Journal on Selected Areas in Communications*, vol. 29(8), pp. 1508–1524, Sep. 2011. http://dx.doi.org/10.1109/jsac.2011.110902.

[FLAA21] S. Figueroa-Lorenzo, J. Añorga, and S. Arrizabalaga. *A Survey of IIoT Protocols*. In: *ACM Computing Surveys*, vol. 53(2), pp. 1–53, Mar. 2021. http://dx.doi.org/10.1145/3381038.

[GAQ+17] R. D. Gomes, M. S. Alencar, D. V. Queiroz, I. E. Fonseca, and C. Benavente-Peces. *Comparison between Channel Hopping and Channel Adaptation for Industrial Wireless Sensor Networks*. In: *Proceedings of the 6th International Conference on Sensor Networks*, pp. 87–98. SCITEPRESS - Science and Technology Publications, Feb. 2017. http://dx.doi.org/10.5220/0006206800870098.

[GKD19] F. Glover, G. Kochenberger, and Y. Du. *Quantum Bridge Analytics I: A Tutorial on Formulating and Using QUBO Models*. In: *4OR*, vol. 17(4), pp. 335–371, Nov. 2019. http://dx.doi.org/10.1007/s10288-019-00424-y.

[HCC09] J. Hui, D. Culler, and S. Chakrabarti. *6LoWPAN: Incorporating IEEE 802.15.4 into the IP Architecture. Tech. Rep. 3*, Internet Protocol for Smart Object (IPSO) Alliance, Jan. 2009.

[HGT17] R. T. Hermeto, A. Gallais, and F. Theoleyre. *Scheduling for IEEE802.15.4-TSCH and Slow Channel Hopping MAC in Low Power Industrial Wireless Networks: A Survey*. In: *Computer Communications*, vol. 114, pp. 84–105, Dec. 2017. http://dx.doi.org/10.1016/j.comcom.2017.10.004.

[HMD15] S. Han, H. Mao, and W. J. Dally. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. In: *arXiv:1510.00149*, Oct. 2015. http://dx.doi.org/10.48550/ARXIV.1510.00149.

[HN14] K.-i. Hwang and S.-w. Nam. *Analysis and Enhancement of IEEE 802.15.4e DSME Beacon Scheduling Model*. In: *Advanced Mathematics and Numerical Modeling of IoT (Internet of Things)*, vol. 2014, pp. 1–15, May 2014. http://dx.doi.org/10.1155/2014/934610.

[HSER14] O. Hussein, N. M. Sadek, S. Elnoubi, and M. R. M. Rizk. *Analytical Model of Multihop IEEE 802.15.4 with Unslotted CA/CSMA*. In: *International Journal of Computer and Communication Engineering (IJCCE)*, vol. 3(3), pp. 226–230, May 2014. http://dx.doi.org/10.7763/ijcce.2014.v3.325.

[HSLZ21] S. C. H. Hoi, D. Sahoo, J. Lu, and P. Zhao. *Online Learning: A Comprehensive Survey*. In: *Neurocomputing*, vol. 459, pp. 249–289, Oct. 2021. http://dx.doi.org/10.1016/j.neucom.2021.04.112.

[HTF08] T. Hastie, R. Tibshirani, and J. Friedman. *Overview of Supervised Learning*. In: *The Elements of Statistical Learning*, pp. 9–41. Springer New York, Dec. 2008. http://dx.doi.org/10.1007/978-0-387-84858-7_2.

[IEE11] IEEE. *IEEE Standard for Information Technology–telecommunications and Information Exchange between Systems–local and Metropolitan Area Networks–specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 10: Mesh Networking*. In: *EEE Std 802.11s-2011 (Amendment to IEEE Std 802.11-2007)*, pp. 1–372, Sep. 2011. http://dx.doi.org/10.1109/ieeestd.2011.6018236.

[IEE12] IEEE. *IEEE Standard for Local and Metropolitan Area Networks–part 15.4: Low-rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC Sublayer*. In: *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)*, pp. 1–225, Apr. 2012. http://dx.doi.org/10.1109/ieeestd.2012.6185525.

[IEE17] IEEE. *IEEE Standard for Information Technology–telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 2: Sub 1 GHz License Exempt Operation*. In: *IEEE Std 802.11ah-2016 (Amendment to IEEE Std 802.11-2016)*, pp. 1–594, May 2017. http://dx.doi.org/10.1109/ieeestd.2017.7920364.

[IEE20] IEEE. *IEEE Standard for Low-rate Wireless Networks*. In: *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*, pp. 1–800, Jul. 2020. http://dx.doi.org/10.1109/ieeestd.2020.9144691.

[IEE21a] IEEE. *IEEE Standard for Information Technology–telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. In: *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pp. 1–4379, Feb. 2021. http://dx.doi.org/10.1109/ieeestd.2021.9363693.

[IEE21b] IEEE. *IEEE Standard for Information Technology–telecommunications and Information Exchange between Systems–local and Metropolitan Area Networks-specific Requirements–part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Amendment 3: Wake-up Radio Operation*. In: *IEEE Std 802.11ba-2021 (Amendment to IEEE Std 802.11-2020)*, pp. 1–180, Oct. 2021. http://dx.doi.org/10.1109/ieeestd.2021.9570110.

[IGK10] O. D. Incel, A. Ghosh, and B. Krishnamachari. *Scheduling Algorithms for Tree-based Data Collection in Wireless Sensor Networks*. In: *Theoretical Aspects of Distributed Computing in Sensor Networks*, pp. 407–445. Springer Berlin Heidelberg, Nov. 2010. http://dx.doi.org/10.1007/978-3-642-14849-1_14.

[IL22a] F. IoT-LAB. *FIT IoT-LAB Website*. GitHub repository, Mar. 2022. License: CC BY SA 4.0.

[IL22b] F. I. T. IoT-LAB. *FIT IoT-LAB Website*, Feb. 2022. License: CC BY SA 4.0.

[INH19] K. Ikeda, Y. Nakamura, and T. S. Humble. *Application of Quantum Annealing to Nurse Scheduling Problem*. In: *Scientific Reports*, vol. 9(1), Sep. 2019. http://dx.doi.org/10.1038/s41598-019-49172-3.

[JAG+16] I. Juc, O. Alphand, R. Guizzetti, M. Favre, and A. Duda. *Energy Consumption and Performance of IEEE 802.15.4e TSCH and DSME*. In: *2016 IEEE Wireless Communications and Networking Conference*, pp. 1–7. IEEE, IEEE, Apr. 2016. http://dx.doi.org/10.1109/wcnc.2016.7565006.

[JL12] W.-C. Jeong and J. Lee. *Performance Evaluation of IEEE 802.15.4e DSME MAC Protocol for Wireless Sensor Networks*. In: *2012 The First IEEE Workshop on Enabling Technologies for Smartphone and Internet of Things (ETSIoT)*, pp. 7–12. IEEE, IEEE, Jun. 2012. http://dx.doi.org/10.1109/etsiot.2012.6311258.

[KAT06] A. Koubaa, M. Alves, and E. Tovar. *A Comprehensive Simulation Study of Slotted CSMA/CA for IEEE 802.15.4 Wireless Sensor Networks*. In: *2006 IEEE International Workshop on Factory Communication Systems*, pp. 183–192. IEEE, Sep. 2006. http://dx.doi.org/10.1109/wfcs.2006.1704149.

[Kau19] F. Kauer. *Scalable Wireless Multi-hop Networks for Industrial Applications*. Ph.D. thesis, Hamburg University of Technology, Hamburg, Germany, May 2019. http://dx.doi.org/10.15480/882.2259.

[KCM+16] S. Kosunalp, Y. Chu, P. D. Mitchell, D. Grace, and T. Clarke. *Use of Q-learning Approaches for Practical Medium Access Control in Wireless Sensor Networks*. In: *Engineering Applications of Artificial Intelligence*, vol. 55, pp. 146–154, Oct. 2016. http://dx.doi.org/10.1016/j.engappai.2016.06.012.

[KKLT16a] F. Kauer, M. Köstler, T. Lübkert, and V. Turau. *Formal Analysis and Verification of the IEEE 802.15.4 DSME Slot Allocation*. In: *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '16, pp. 140–147. ACM, New York, NY, USA, Nov. 2016. http://dx.doi.org/10.1145/2988287.2989148.

[KKLT16b] M. Köstler, F. Kauer, T. Lübkert, and V. Turau. *Towards an Open Source Implementation of the IEEE 802.15.4 DSME Link Layer*. In: J. Scholz and A. von Bodisco (eds.), *Proceedings of the 15. GI/ITG KuVS Fachgespräch Sensornetze*. University of Applied Sciences Augsburg, Augsburg, Germany, Sep. 2016.

[KKT18] F. Kauer, M. Köstler, and V. Turau. *Reliable Wireless Multi-hop Networks with Decentralized Slot Management: An Analysis of IEEE*

802.15.4 DSME. In: *arXiv:1806.10521*, Jun. 2018. http://dx.doi.org/10.48550/ARXIV.1806.10521.

[KKT19a] F. Kauer, M. Köstler, and V. Turau. *openDSME: Reliable Time-slotted Multi-hop Communication for IEEE 802.15.4*. In: A. Virdis and M. Kirsche (eds.), *Recent Advances in Network Simulation: The OMNeT++ Environment and its Ecosystem*, pp. 451–467. Springer International Publishing, Cham, May 2019. http://dx.doi.org/10.1007/978-3-030-12842-5_15.

[KKT19b] F. Kauer, M. Köstler, and V. Turau. *openDSME: Reliable Time-slotted Multi-hop Communication for IEEE 802.15.4*. In: *Recent Advances in Network Simulation*, pp. 451–467. Springer International Publishing, 2019. http://dx.doi.org/10.1007/978-3-030-12842-5_15.

[KMH+19] D. Kim, S. Moon, D. Hostallero, W. J. Kang, T. Lee, K. Son, and Y. Yi. *Learning to Schedule Communication in Multi-agent Reinforcement Learning*. In: *arXiv:1902.01554*, Feb. 2019. http://dx.doi.org/10.48550/ARXIV.1902.01554.

[KRZ+20] W. Z. Khan, M. H. Rehman, H. M. Zangoti, M. K. Afzal, N. Armi, and K. Salah. *Industrial Internet of Things: Recent Advances, Enabling Technologies and Open Challenges*. In: *Computers & Electrical Engineering*, vol. 81, p. 106522, Jan. 2020. http://dx.doi.org/10.1016/j.compeleceng.2019.106522.

[KSKT17] H. Kurunathan, R. Severino, A. Koubâa, and E. Tovar. *Worst-case Bound Analysis for the Time-critical MAC Behaviors of IEEE 802.15.4e*. In: *2017 IEEE 13th International Workshop on Factory Communication Systems (WFCS)*, pp. 1–9. IEEE, IEEE, May 2017. http://dx.doi.org/10.1109/wfcs.2017.7991967.

[KSKT18] H. Kurunathan, R. Severino, A. Koubaa, and E. Tovar. *IEEE 802.15.4e in a Nutshell: Survey and Performance Evaluation*. In: *IEEE Communications Surveys & Tutorials*, vol. 20(3), pp. 1989–2010, Feb. 2018. http://dx.doi.org/10.1109/comst.2018.2800898.

[KSKT20] H. Kurunathan, R. Severino, A. Koubâa, and E. Tovar. *DynaMO: Dynamically Tuning DSME Networks*. In: *ACM SIGBED Review*, vol. 16(4), pp. 8–13, Jan. 2020. http://dx.doi.org/10.1145/3378408.3378409.

[Kur18] A. Kurniawan. *Practical Contiki-NG*. Springer International Publishing, Jun. 2018. http://dx.doi.org/10.1007/978-1-4842-3408-2.

[KVN16] L. Krupka, L. Vojtech, and M. Neruda. *The Issue of LPWAN Technology Coexistence in IoT Environment*. In: *2016 17th International*

Conference on Mechatronics - Mechatronika (ME), pp. 1–8. IEEE, Jan. 2016.

[LE06] Z. Liu and I. Elhanany. *RL-MAC: A QoS-aware Reinforcement Learning based MAC Protocol for Wireless Sensor Networks*. In: *2006 IEEE International Conference on Networking, Sensing and Control*, pp. 768–773. IEEE, IEEE, Aug. 2006. http://dx.doi.org/10.1109/icnsc.2006.1673243.

[Liu11] B. Liu. *Unsupervised Learning*. In: *Web Data Mining*, pp. 133–169. Springer, Jun. 2011. http://dx.doi.org/10.1007/978-3-642-19460-3_4.

[LJ12] J. Lee and W.-C. Jeong. *Performance Analysis of IEEE 802.15.4e DSME MAC Protocol under WLAN Interference*. In: *2012 International Conference on ICT Convergence (ICTC)*, pp. 741–746. IEEE, Oct. 2012. http://dx.doi.org/10.1109/ictc.2012.6387133.

[LJS20] T. Lee, O. Jo, and K. Shin. *CoRL: Collaborative Reinforcement Learning-based MAC Protocol for IoT Networks*. In: *Electronics*, vol. 9(1), p. 143, Jan. 2020. http://dx.doi.org/10.3390/electronics9010143.

[LKZK21] S.-W. Lee, J.-H. Kwon, X. Zhang, and E.-J. Kim. *Traffic-adaptive CFP Extension for IEEE 802.15.4 DSME MAC in Industrial Wireless Sensor Networks*. In: *IEEE Access*, vol. 9, pp. 94454–94469, Jul. 2021. http://dx.doi.org/10.1109/access.2021.3093893.

[LLS+13] X. Liu, X. Li, S. Su, Z. Fan, and G. Wang. *Enhanced Fast Association for 802.15.4e-2012 DSME MAC Protocol*. In: *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*, pp. 987–990. Atlantis Press, Mar. 2013. http://dx.doi.org/10.2991/iccsee.2013.248.

[LR00] M. Lauer and M. Riedmiller. *An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-agent Systems*. In: *In Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pp. 535–542. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Jun. 2000.

[LR04] M. Lauer and M. Riedmiller. *Reinforcement Learning for Stochastic Cooperative Multi-agent-systems*. In: *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.*, pp. 1516–1517. IEEE Computer Society, New York, NY, USA, May 2004.

[LSH08] T. Lennvall, S. Svensson, and F. Hekland. *A Comparison of WirelessHART and ZigBee for Industrial Applications*. In: *2008 IEEE International Workshop on Factory Communication Systems*. IEEE, May 2008. http://dx.doi.org/10.1109/wfcs.2008.4638746.

[MBCM19] K. Mekki, E. Bajic, F. Chaxel, and F. Meyer. *A Comparative Study of LPWAN Technologies for Large-scale IoT Deployment*. In: *ICT Express*, vol. 5(1), pp. 1–7, Mar. 2019. http://dx.doi.org/10.1016/j.icte.2017.12.005.

[Mel01] F. S. Melo. *Convergence of Q-learning: A Simple Proof*. In: *Institute Of Systems and Robotics, Tech. Rep*, pp. 1–4, 2001.

[MHKH19] J. Ma, S. Hasegawa, S.-J. Kim, and M. Hasegawa. *A Reinforcement-learning-based Distributed Resource Selection Algorithm for Massive IoT*. In: *Applied Sciences*, vol. 9(18), p. 3730, Sep. 2019. http://dx.doi.org/10.3390/app9183730.

[MKHC07] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. *Transmission of IPv6 Packets Over IEEE 802.15.4 Networks. Tech. rep.*, IETF, Sep. 2007. RFC 4944, http://dx.doi.org/10.17487/rfc4944.

[MKS+15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. *Human-level Control through Deep Reinforcement Learning*. In: *Nature*, vol. 518(7540), pp. 529–533, Feb. 2015. http://dx.doi.org/10.1038/nature14236.

[MMDT22] F. Meyer, P. Malessa, J. Diercks, and V. Turau. *Are Group Acknowledgements Worth Anything in IEEE 802.15.4 DSME: a Comparative Analysis*. In: *2022 5th Conference on Cloud and Internet of Things (CIoT'22)*. Marrakech, Morocco, Mar. 2022. Accepted for publication, http://dx.doi.org/https://doi.org/10.48550/arXiv.2109.06267.

[MMGKT19] F. Meyer, I. Mantilla-González, F. Kauer, and V. Turau. *Performance Analysis of the Slot Allocation Handshake in IEEE 802.15.4 DSME*. In: *Ad-Hoc, Mobile, and Wireless Networks*, pp. 102–117. Springer International Publishing, Sep. 2019. http://dx.doi.org/10.1007/978-3-030-31831-4_8.

[MMGT20a] F. Meyer, I. Mantilla-González, and V. Turau. *New CAP Reduction Mechanisms for IEEE 802.15.4 DSME to Support Fluctuating Traffic in IoT Systems*. In: L. A. Grieco, G. Boggia, G. Piro, Y. Jararweh, and C. Campolo (eds.), *Ad-Hoc, Mobile, and Wireless Networks*, vol. 12338, pp. 159–179. Springer International Publishing, Oct. 2020. http://dx.doi.org/10.1007/978-3-030-61746-2_13.

[MMGT20b] F. Meyer, I. Mantilla-González, and V. Turau. *Sending Multiple Packets Per Guaranteed Time Slot in IEEE 802.15.4 DSME : Analysis and Evaluation*. In: *Internet Technology Letters (ITL)*, vol. 4(4), p. e167, Jun. 2020. http://dx.doi.org/10.1002/itl2.167.

[MPSZ18] J. Mocnej, A. Pekar, W. K. G. Seah, and I. Zolotova. *Network Traffic Characteristics of the IoT Application Use Cases*. School of Engineering and Computer Science, Victoria University of Wellington, 2018.

[MT19] F. Meyer and V. Turau. *Delay-bounded Scheduling in IEEE 802.15.4e DSME Using Linear Programming*. In: *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pp. 659–666. IEEE, May 2019. http://dx.doi.org/10.1109/dcoss.2019.00119.

[MT21] F. Meyer and V. Turau. *QMA: A Resource-efficient, Q-learning-based Multiple Access Scheme for the IIoT*. In: *IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, Jul. 2021. http://dx.doi.org/10.1109/icdcs51616.2021.00087.

[MVK19] L. Mészáros, A. Varga, and M. Kirsche. *INET Framework*. In: *Recent Advances in Network Simulation*, pp. 55–106. Springer International Publishing, May 2019. http://dx.doi.org/10.1007/978-3-030-12842-5_2.

[MYH+19] X. Ma, T. Yao, M. Hu, Y. Dong, W. Liu, F. Wang, and J. Liu. *A Survey on Deep Learning Empowered IoT Applications*. In: *IEEE Access*, vol. 7, pp. 181721–181732, Dec. 2019. http://dx.doi.org/10.1109/access.2019.2958962.

[MZC+21] W. Mao, Z. Zhao, Z. Chang, G. Min, and W. Gao. *Energy-efficient Industrial Internet of Things: Overview and Open Issues*. In: *IEEE Transactions on Industrial Informatics*, vol. 17(11), pp. 7225–7237, Nov. 2021. http://dx.doi.org/10.1109/tii.2021.3067026.

[NSI+15] J. Nieminen, T. Savolainen, M. Isomaki, B. Patil, Z. Shelby, and C. Gomez. *IPv6 Over BLUETOOTH(r) Low Energy*. *Tech. rep.*, IETF, Oct. 2015. RFC 7668, http://dx.doi.org/10.17487/rfc7668.

[OGAP19] M. O. Ojo, S. Giordano, D. Adami, and M. Pagano. *Throughput Maximizing and Fair Scheduling Algorithms in Industrial Internet of Things Networks*. In: *IEEE Transactions on Industrial Informatics*, vol. 15(6), pp. 3400–3410, Jun. 2019. http://dx.doi.org/10.1109/tii.2018.2873974.

[PAD+12] M. R. Palattella, N. Accettura, M. Dohler, L. A. Grieco, and G. Boggia. *Traffic Aware Scheduling Algorithm for Reliable Low-power Multi-hop IEEE 802.15.4e Networks*. In: *2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications - (PIMRC)*, pp. 327–332. IEEE, IEEE, Sep. 2012. http://dx.doi.org/10.1109/pimrc.2012.6362805.

[PC09] S. Petersen and S. Carlsen. *Performance Evaluation of WirelessHART for Factory Automation*. In: *2009 IEEE Conference on Emerging Technologies & Factory Automation*. IEEE, Sep. 2009. http://dx.doi.org/10.1109/etfa.2009.5346996.

[PCN21] D. Prasad, N. N. Chiplunkar, and K. P. Nayak. *Performance Comparison of IEEE 802.15.4 and IEEE 802.15.4e Based MAC Algorithm in Wireless Body Sensor Networks*. In: *IOP Conference Series: Materials Science and Engineering*, vol. 1119(1), p. 012020, Mar. 2021. http://dx.doi.org/10.1088/1757-899x/1119/1/012020.

[PMS+09] P. Park, P. D. Marco, P. Soldati, C. Fischione, and K. H. Johansson. *A Generalized Markov Chain Model for Effective Analysis of Slotted IEEE 802.15.4*. In: *2009 IEEE 6th International Conference on Mobile Adhoc and Sensor Systems*, pp. 130–139. IEEE, Oct. 2009. http://dx.doi.org/10.1109/mobhoc.2009.5337007.

[PMSZ21] A. Pekar, J. Mocnej, W. K. G. Seah, and I. Zolotova. *Application Domain-based Overview of IoT Network Traffic Characteristics*. In: *ACM Computing Surveys*, vol. 53(4), pp. 1–33, Jul. 2021. http://dx.doi.org/10.1145/3399669.

[PWW+16] M. R. Palattella, T. Watteyne, Q. Wang, K. Muraoka, N. Accettura, D. Dujovne, L. A. Grieco, and T. Engel. *On-the-fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks*. In: *IEEE Sensors Journal*, vol. 16(2), pp. 550–560, Jan. 2016. http://dx.doi.org/10.1109/jsen.2015.2480886.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Representations by Back-propagating Errors*. In: *Nature*, vol. 323(6088), pp. 533–536, Oct. 1986. http://dx.doi.org/10.1038/323533a0.

[RK19] S. Resch and U. R. Karpuzcu. *Quantum Computing: An Overview across the System Stack*. In: *arXiv:1905.07240*, May 2019. http://dx.doi.org/10.48550/ARXIV.1905.07240.

[RN20] A. G. Ramonet and T. Noguchi. *IEEE 802.15.4 Now and Then: Evolution of the LR-WPAN Standard*. In: *2020 22nd International Conference on Advanced Communication Technology (ICACT)*. IEEE, Feb. 2020. http://dx.doi.org/10.23919/icact48636.2020.9061514.

[Sar19] D. K. Sarmah. *A Survey on the Latest Development of Machine Learning in Genetic Algorithm and Particle Swarm Optimization*. In: *Algorithms for Intelligent Systems*, pp. 91–112. Springer Singapore, Nov. 2019. http://dx.doi.org/10.1007/978-981-15-0994-0_6.

[SB11] Z. Shelby and C. Bormann. *6LoWPAN: The Wireless Embedded Internet*, vol. 43. WILEY, Aug. 2011.

[SFDPH20] A. Seferagić, J. Famaey, E. De Poorter, and J. Hoebeke. *Survey on Wireless Technology Trade-offs for the Industrial Internet of Things*. In: *Sensors*, vol. 20(2), p. 488, Jan. 2020. http://dx.doi.org/10.3390/s20020488.

[SGD11] C. Sommer, R. German, and F. Dressler. *Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis*. In: *IEEE Transactions on Mobile Computing*, vol. 10(1), pp. 3–15, Jan. 2011. http://dx.doi.org/10.1109/tmc.2010.133.

[SHB14] Z. Shelby, K. Hartke, and C. Bormann. *The Constrained Application Protocol (CoAP). Tech. rep.*, IETF, Jun. 2014. RFC7252, http://dx.doi.org/10.17487/rfc7252.

[SHG19] P. C. Sen, M. Hajra, and M. Ghosh. *Supervised Classification Algorithms in Machine Learning: A Survey and Review*. In: *Advances in Intelligent Systems and Computing*, pp. 99–111. Springer Singapore, Jul. 2019. http://dx.doi.org/10.1007/978-981-13-7403-6_11.

[SLL21] W. Y. Suen, C. Y. Lee, and H. C. Lau. *Quantum-inspired Algorithm for Vehicle Sharing Problem*. In: *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, Oct. 2021. http://dx.doi.org/10.1109/qce52317.2021.00017.

[SOM14] F. Shrouf, J. Ordieres, and G. Miragliotta. *Smart Factories in Industry 4.0: A Review of the Concept and of Energy Management Approached in Production Based on the Internet of Things Paradigm*. In: *2014 IEEE International Conference on Industrial Engineering and Engineering Management*, pp. 697–701. IEEE, IEEE, Dec. 2014. http://dx.doi.org/10.1109/ieem.2014.7058728.

[SP17] I. J. Sledge and J. C. Principe. *Balancing Exploration and Exploitation in Reinforcement Learning Using a Value of Information Criterion*. In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Mar. 2017. http://dx.doi.org/10.1109/icassp.2017.7952670.

[SPW17] P. Sahoo, S. Pattanaik, and S.-L. Wu. *A Novel IEEE 802.15.4e DSME MAC for Wireless Sensor Networks*. In: *Sensors*, vol. 17(12), p. 168, Jan. 2017. http://dx.doi.org/10.3390/s17010168.

[SSH+18] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund. *Industrial Internet of Things: Challenges, Opportunities, and Directions*. In: *IEEE Transactions on Industrial Informatics*, vol. 14(11), pp. 4724–4734, Nov. 2018. http://dx.doi.org/10.1109/tii.2018.2852491.

[Thu11] P. Thubert. *Compression Format for IPv6 Datagrams Over IEEE 802.15.4-based Networks. Tech. rep.*, IETF, Sep. 2011. RFC 6282, http://dx.doi.org/10.17487/rfc6282.

[Thu20] P. Thubert. *IPv6 Over Low-power Wireless Personal Area Network (6lowpan) Selective Fragment Recovery. Tech. rep.*, IETF, Nov. 2020. RFC 8931, http://dx.doi.org/10.17487/rfc8931.

[TMV+21] Y. Tanaka, P. Minet, M. Vucinic, X. Vilajosana, and T. Watteyne. *YSF: A 6TiSCH Scheduling Function Minimizing Latency of Data Gathering in IIoT*. In: *IEEE Internet of Things Journal*, pp. 1–1, Oct. 2021. http://dx.doi.org/10.1109/jiot.2021.3118017.

[TS21] S. F. Tan and A. Samsudin. *Recent Technologies, Security Countermeasure and Ongoing Challenges of Industrial Internet of Things (IIoT): A Survey*. In: *Sensors*, vol. 21(19), p. 6647, Oct. 2021. http://dx.doi.org/10.3390/s21196647.

[TSS+21] L. Tian, S. Santi, A. Seferagić, J. Lan, and J. Famaey. *Wi-Fi HaLow for the Internet of Things: An Up-to-date Survey on IEEE 802.11ah Research*. In: *Journal of Network and Computer Applications*, vol. 182, p. 103036, May 2021. http://dx.doi.org/10.1016/j.jnca.2021.103036.

[Unt14] S. Unterschütz. *Methodologies and Protocols for Wireless Communication in Large-scale, Dense Mesh Networks*. Ph.D. thesis, Hamburg University of Technology, Hamburg, Germany, Apr. 2014.

[UQR+19] M. Usama, J. Qadir, A. Raza, H. Arif, K.-l. A. Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha. *Unsupervised Machine Learning for Networking: Techniques, Applications and Research Challenges*. In: *IEEE Access*, vol. 7, pp. 65579–65615, May 2019. http://dx.doi.org/10.1109/access.2019.2916648.

[UWT12] S. Unterschütz, A. Weigel, and V. Turau. *Cross-platform Protocol Development Based on OMNeT++*. In: *Proceedings of the Fifth International Conference on Simulation Tools and Techniques*, pp. 278–282. ICST (Institute for Computer Sciences, Social-Informatics and, ACM, Jun. 2012. http://dx.doi.org/10.4108/icst.simutools.2012.247711.

[Var19] A. Varga. *Recent Advances in Network Simulation*. In: *Recent Advances in Network Simulation*, pp. 3–51. Springer International Publishing, Cham, May 2019. http://dx.doi.org/10.1007/978-3-030-12842-5_1.

[VBPA17] C. Vallati, S. Brienza, M. Palmieri, and G. Anastasi. *Improving Network Formation in IEEE 802.15.4e DSME*. In: *Computer Communications*, vol. 114, pp. 1–9, Dec. 2017. http://dx.doi.org/10.1016/j.comcom.2017.09.016.

[vEH19] J. E. van Engelen and H. H. Hoos. *A Survey on Semi-supervised Learning*. In: *Machine Learning*, vol. 109(2), pp. 373–440, Nov. 2019. http://dx.doi.org/10.1007/s10994-019-05855-6.

[VH08] A. Varga and R. Hornig. *An Overview of the OMNeT++ Simulation Environment*. In: *Proceedings of the First International ICST Conference on Simulation Tools and Techniques for Communications Networks and Systems*, Simutools '08. ICST, Brussels, BEL, 2008. http://dx.doi.org/10.4108/icst.simutools2008.3027.

[VSN15] A. Virdis, G. Stea, and G. Nardini. *Simulating LTE/LTE-advanced Networks with SimuLTE*. In: M. S. Obaidat, T. Ören, J. Kacprzyk, and J. Filipe (eds.), *Simulation and Modeling Methodologies, Technologies and Applications*, pp. 83–105. Springer International Publishing, Cham, Jan. 2015. http://dx.doi.org/10.1007/978-3-319-26470-7_5.

[VVDD21] M. Vučinić, X. Vilajosana, S. Duquennoy, and D. Dujovne. *6TiSCH Minimal Scheduling Function (MSF). Tech. rep.*, IETF, May 2021. RFC 9033, http://dx.doi.org/10.17487/RFC9033.

[VW18] X. Vilajosana and T. Watteyne. *6TiSCH Operation Sublayer (6top) Protocol (6P). Tech. rep.*, IETF, Nov. 2018. RFC 8480, http://dx.doi.org/10.17487/rfc8480.

[WBCJ17] F. Wilhelmi, B. Bellalta, C. Cano, and A. Jonsson. *Implications of Decentralized Q-learning Resource Allocation in Wireless Networks*. In: *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pp. 1–5. IEEE, Oct. 2017. http://dx.doi.org/10.1109/pimrc.2017.8292321.

[WD92] C. J. C. H. Watkins and P. Dayan. *Q-learning*. In: *Machine Learning*, vol. 8(3-4), pp. 279–292, May 1992. http://dx.doi.org/10.1007/bf00992698.

[Wei17] A. Weigel. *Forwarding Strategies for 6LoWPAN-fragmented IPv6 Datagrams*. Ph.D. thesis, Hamburg University of Technology, Hamburg, Germany, Nov. 2017. http://dx.doi.org/10.15480/882.1515.

[WSM+21] C. Wei, K. Sohn, C. Mellina, A. Yuille, and F. Yang. *CReST: A Class-rebalancing Self-training Framework for Imbalanced Semi-supervised Learning*. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10857–10866. IEEE, Jun. 2021. http://dx.doi.org/10.1109/cvpr46437.2021.01071.

[XZY+18] Z. Xiao, J. Zhou, J. Yan, C. He, L. Jiang, and N. Trigoni. *Performance Evaluation of IEEE 802.15.4 with Real Time Queueing Analysis*. In: *Ad Hoc Networks*, vol. 73, pp. 80–94, May 2018. http://dx.doi.org/10.1016/j.adhoc.2018.01.006.

[YKH17] D. Yuan, S. S. Kanhere, and M. Hollick. *Instrumenting Wireless Sensor Networks — a Survey on the Metrics That Matter*. In: *Pervasive and Mobile Computing*, vol. 37, pp. 45–62, Jun. 2017. http://dx.doi.org/10.1016/j.pmcj.2016.10.001.

[YTB18] S. B. Yaala, F. Theoleyre, and R. Bouallegue. *Performance Modeling of IEEE 802.15.4-TSCH with Shared Access and ON-OFF Traffic*. In: *2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC)*, pp. 352–357. IEEE, Jun. 2018. http://dx.doi.org/10.1109/iwcmc.2018.8450358.

[ZAKP20] D. Zorbas, K. Abdelfadeel, P. Kotzanikolaou, and D. Pesch. *TS-LoRa: Time-slotted LoRaWAN for the Industrial Internet of Things*. In: *Computer Communications*, vol. 153, pp. 1–10, Mar. 2020. http://dx.doi.org/10.1016/j.comcom.2020.01.056.

[Zha19] Q. Zhao. *Presents the Technology, Protocols, and New Innovations in Industrial Internet of Things (IIoT)*. In: *Internet of Things for Industry 4.0*, pp. 39–56. Springer International Publishing, Dec. 2019. http://dx.doi.org/10.1007/978-3-030-32530-5_3.

[Zha20] X.-D. Zhang. *Machine Learning*, pp. 223–440. Springer Singapore, Singapore, 2020. http://dx.doi.org/10.1007/978-981-15-2770-8_6.

[ZNLW19] R. Zhang, F. Nie, X. Li, and X. Wei. *Feature Selection with Multi-view Data: A Survey*. In: *Information Fusion*, vol. 50, pp. 158–167, Oct. 2019. http://dx.doi.org/10.1016/j.inffus.2018.11.019.

[ZVKB07] S. Zeng, B. Volz, K. Kinnear, and J. Brzozowski. *DHCPv6 Relay Agent Echo Request Option*. *Tech. rep.*, IETF, Sep. 2007. RFC 4994, http://dx.doi.org/10.17487/rfc4994.

List of Acronyms

3GPP	3rd Generation Partnership Project
5G	5th Generation Cellular
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
6P	6top Protocol
6P-MSF	Minimal Scheduling Function
6P-SF	Scheduling Function
6TiSCH	IPv6 over the TSCH mode of IEEE 802.15.4e
6top	6TiSCH Operation Sublayer
A-CAP-R	Alternating CAP-Reduction
ACK	Acknowledgment
ACT	Allocation Counter Table
ALOHA	ALOHA
ALOHA-Q	ALOHA and Q-Learning based MAC
ALOHA-QIR	ALOHA and Q-Learning based MAC with Informed Receiving
AMCA	Asynchronous Multi-channel Adaptation
AMQP	Advanced Message Queuing Protocol
ANN	Artificial Neural Network
AODV	Ad-hoc On-demand Distance Vector
ASF	Autonomous Scheduling Function
BLE	Bluetooth Low Energy
BPSK	Binary Phase Shift Keying
CAP	Contention Access Period
CAP-R	CAP-Reduction
CCA	Clear Channel Assessment
CDF	Cumulative Distribution Function
CDMA	Code Division Multiple Access
CFP	Contention Free Period
COAP	Constrained Application Protocol
CoRL	Collaborative Reinforcement Learning-based MAC
CS	Common Shared Orchestra Slot
CSMA/CA	Carrier Sense Multiple Access / Collision Avoidance
CSS	Chirp Spread Spectrum
CTS	Clear To Send
D	Delay
D-CAP-R	Dynamic CAP-Reduction
DCF	Distributed Coordination Function
DDS	Data Distribution Service
DODAG	Destination Oriented Directed Acyclic Graph
DSME	Deterministic and Synchronous Multi-channel Extension
DT	Decision Tree
DwT	Dwell time
eMBB	enhanced Mobile Broadband
eMTC	enhanced Machine Type Communication
EWMA	Exponentially Weighted Moving Average
FDMA	Frequency Division Multiple Access
FFD	Full-Function Device
FHSS	Frequency-hopping Spread Spectrum
FIT	Future Internet Testing
FSK	Frequency Shift Keying
GA	Genetic Algorithm

GACK	Group Acknowledgment
GP	Goodput
GPS	Global Positioning System
GPSR	Greedy Perimeter Stateless Routing
GSM	Global System for Mobile Communications
GTS	Guaranteed Time Slot
HART	Highway Addressable Remote Transducer
HCF	HART Communication Foundation
IE	Information Element
IETF	Internet Engineering Task Force
IIoT	Industrial Internet of Things
ILP	Integer Linear Program
IoT	Internet of Things
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IR	Information Receiving
ISM	Industrial, Scientific, Medical
LLDN	Low Latency Deterministic Network
LLSF	Low Latency Scheduling Function
LoRa	Long Range
LoRaWAN	Long Range Wide Area Network
LP	Linear Program
LPWA	Low Power Wide Area
LPWAN	Low Power Wide Area Network
LR-WPAN	Low-Rate Wireless Personal Area Network
LTE	Long Term Evolution
LTE+	Long Term Evolution Advanced
MAC	Medium Access Control
MANET	Mobile Adhoc Network
MCPS-SAP	MAC Common Part Sublayer Service Access Point
MCU	Microcontroller Unit
MDP	Markov Decision Process
MILP	Mixed-integer Linear Program
ML	Machine Learning
MLME-SAP	MAC Sublayer Management Entity Service Access Point
mMTC	massive Machine Type Communication
MQTT	Message Queuing Telemetry Transport
MSF	multi-superframe
MTU	Maximum Transmission Unity
MU-MIMO	Multi-User Multiple Input Multiple Output
NB	Naive Bayes
NB-IoT	Narrowband-Internet of Things
NED	NEtwork Description
NFC	Near Field Communication
NO-CAP-R	no CAP-R
OFDMA	Orthogonal Frequency Division Multiple Access
OLSR	Optimized Link State Routing
OQPSK	Offset Quadrature Phase Shift Keying
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
OTF	On-the-fly Bandwith Reservation
PAL	Platform Abstraction Layer
PAN	Personal Area Network
PCA	Principal Component Analysis
PCF	Point Coordination Function
PDR	Packet Delivery Ratio
PSO	Particle Swarm Optimization
QA	Quantum Annealing
QAM	Quadrature Amplitude Modulation
QL	Q-learning
QMA	Q-learning-based Multiple Access
QoS	Quality of Service
QP	Quadratic Program
QPSK	Quadrature Phase Shift Keying
QUBO	Quadratic Unconstrained Binary Optimization
RAM	Random-access Memory
RAW	Restricted Access Window
RBS	Receiver-based Shared Orchestra Slot
RESF	Recurrent Scheduling Function
REST	Representational State Transfer

RF	Random Forest
RFC	Request For Comments
RFD	Reduced-Function Device
RFID	Radio Frequency Identification Blink
RL	Reinforcement Learning
RL-MAC	Reinforcement Learning based MAC
ROM	Read-only Memory
RPL	Routing Protocol for Low Power and Lossy Networks
RTOS	Real-time Operating System
RTS	Ready To Send
RX	Reception
SA	Simulated Annealing
SAB	Slot Allocation Bitmap
SARSA	State-action-reward-state-action
SBD	Sender-based Dedicated Orchestra Slot
SBS	Sender-based Shared Orchestra Slot
SF	superframe
SF0	Scheduling Function 0
SFX	Experimental Scheduling Function
SL	Supervised Learning
SSH	Secure Shell
ST	Start time
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TIM	Traffic Indication Map
TL	Transfer Learning
TP	Throughput
TPS	Traffic-aware and Prediction-based Slot Scheduling
TRASA	Traffic-Aware Time Slot Assignment
TSCH	Time-slotted Channel Hopping
TWT	Target Wake Time
TX	Transmission
UDP	User Datagram Protocol
UL	Unsupervised Learning
UNB	Ultra Narrowband
URLLC	Ultra-reliable Low Latency Communication
WD	Warmup Duration
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network
WUR	Wake-up Radio

List of Symbols

$aBaseSuperframeDuration$	base duration of a superframe
$a_t^{(i)}$	reinforcement learning action taken by node v_i at time t
a_t	reinforcement learning action taken at time t
$aTurnaroundTime$	transceiver turnaround time
$aUnitBackoffPeriod$	number of symbols forming the basic time period in CSMA/CA
$AIFS$	acknowledgment interframe space
$\mathbb{A}_t$	set of RL action available at time t
BE	CSMA/CA backoff exponent
BI	beacon interval
BO	beacon order
c	index for the channel using MILPs
CM	CAPs per multisuperframe
$\mathbb{C}$	set of available channels
CTM	CAP's time slots per multisuperframe
D	delay
D_{MSF}	multisuperframe duration
D_{SF}	superframe duration
D_S	slot duration
Dt_i	dwell time at node i
$\mathbb{E}$	set of links in a network
GAO	group acknowledgment order
$\mathbb{G}$	network formally represented as a graph
i	index for the node using MILPs
IFS	interframe space
$\mathbb{I}_i$	set of nodes in interference range of node i
k	index for the slot using MILPs
K	number of time slots in a schedule
l	index for the line using MILPs
$LIFS$	long interframe space
$macAckWaitDuration$	maximum duration to wait for an acknowledgment
$macMaxCsmaBackoffs$	CSMA/CA maximum number of backoffs
$macMaxFrameRetries$	CSMA/CA maximum number of frame retransmissions
$maxBE$	CSMA/CA maximum backoff exponent
$minBE$	CSMA/CA minimum backoff exponent
m_i	subslot of QMA
m_{SL}	supervised learning model
M	number of subslots of QMA
MO	multisuperframe order

N_i^+	parent of node i
$\mathbb{N}_i^-$	set of children of node i
N_{CAP}	expected number of time slots to send a CAP message
$\mathbb{N}_i$	set of neighbors of node i
p_{CAP}	probability of generating a packet during the CAP
P_{GTS}	maximum number of transmissions per GTS
P_i	packet generate by node i
$\mathbb{P}$	set of paths
P_t	incoming packets per MSF
Q_{CAP}	queue length for CAP messages
Q_{CFP}	queue length for CFP messages
R_B	reinforcement learning reward for action QBACKOFF
R_C	reinforcement learning reward for action QCCA
R_S	reinforcement learning reward for action QSEND
R_t	reinforcement learning reward at time t
RX_i	reception slot id at node j along path i
Scd_t	incoming packets per MSF
$\mathcal{S}_{CAP}$	symbols per CAP
$SIFS$	short interframe space
SM	superframes per multisuperframe
SO	superframe order
S	size of data packets
$\mathcal{S}$	symbol
$\mathcal{S}_{SF}$	symbols per superframe
S_t	RL state at time t
$\mathcal{S}_{UB}$	number of symbols forming the basic time period in CSMA/CA
t_{ACK}	transmission time for an acknowledgment
t_B	expected backoff time using CSMA/CA
t	current time(step)
t_{CAP}	expected time to send a CAP message
t_{CCA}	duration of a CCA
t_{ChCAP}	expected channel access time for a packet generated during the CAP
t_{ChCFP}	expected channel access time for a packet generated during the CFP
t_{Ch}	expected channel access time
t_{CSMA}	expected time until channel access using CSMA/CA
t_{NTF}	expected transmission time for a GTS-notify
t_{RQ}	expected transmission time for a GTS-request
t_{RSP}	expected transmission time for a GTS-response
t_{setup}	expected setup time for a GTS
ttx_{ACK}	transmission time for an ACK frame
ttx_{NTF}	transmission time for a GTS-notify
ttx_{RQ}	transmission time for a GTS-request
ttx_{RSP}	transmission time for a GTS-response
t_{TT}	aTurnaroundTime
t_{UB}	aUnitBackoffPeriod
T_i	subtree rooted in node i
TM	time slots per multisuperframe
T	routing tree used for the ILPs
TX_i^j	transmission slot id at node j along path i

v_i	*i*th network participant
$\|\mathbb{V}\|$	number of network participants
$\mathbb{V}$	set of network participants
x_i	input data in SL
x_{ik}	binary variable that indicates a transmission slot
$\mathbb{X}_{SL}$	set of structured input data in SL
y_i	output data in SL
$\mathbb{Y}_{SL}$	set of labeled output data in SL
Δ	duration of cautious startup in subslots
α_{RL}	learning rate in RL
β	weighting factor of TPS
γ	discount factor in QL
γ_i	number of children of subtree T_i routed in node i
γ	packet generation interval
δ_c	constant packet generation rate
δ_{max}	maximum feasible number of generated packets per node
δ	packet generation rate
η	ratio of superframes to CAPs per multisuperframe
θ_i	backstep along the path i
λ_t	packets to send per MSF at time t
μ	mapping from hop on a path to line
ξ	constant for diminishing Q-values over time
π	policy in RL
ρ	exploration rate in RL
ρ_i	path from node i to the sink
τ_{CFP}	fraction of CFP's time slots a dataframe
τ_i	number of TX slots a node i
ψ	decision variable in the big M method
ϵ	packet burst size
ϕ	big M constant of the big M method