Diss. ETH No. xxxxx

Design and Testing of Mixed-Range Low-Power Networks

A thesis submitted to attain the degree of

Doctor of Sciences of ETH Zurich
(Dr. sc. ETH Zurich)

presented by
ROMAN TRÜB
MSc ETH EEIT, ETH Zurich

born on 17.12.1992
citizen of
Switzerland

Prof. Dr. Lothar Thiele, examiner
Prof. Dr. Carlo Alberto Boano, co-examiner
Prof. Dr. Jan Beutel, co-examiner

2022

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

TIK-SCHRIFTENREIHE NR. xxx

ROMAN TRÜB

Design and Testing of Mixed-Range Low-Power Networks

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

A dissertation submitted to
ETH Zurich
for the degree of Doctor of Sciences

DISS. ETH NO. xxxxx

Prof. Dr. Lothar Thiele, examiner
Prof. Dr. Carlo Alberto Boano, co-examiner
Prof. Dr. Jan Beutel, co-examiner

Examination date: <Month> <day>, 2022

DOI xx.xxxx/ethz-b-xxxxxxxx

Abstract

Nowadays, more and more objects are being connected to the Internet as the resulting Internet of Things (IoT) offers a wide range of opportunities. Low-power wireless communication technology is frequently used to connect objects to the Internet because it has the advantage that no infrastructure is required at the object location. Until recently, low-power wireless technology was limited to short distances. However, in the last few years, advances in transceiver technology have enabled low-power communication over long distances which enables connecting objects wirelessly at a city scale. A fundamental difficulty of the new technology is that transmissions suitable for different distances have significantly different characteristics. Low-power long-distance transmission schemes typically offer only low data rates, resulting in relatively long transmission times and significantly higher energy consumption compared to short-distance transmissions. This makes exchanging or extending short-range with long-range communication technology difficult.

In this thesis, we focus on the design of reliable and energy-efficient schemes that provide high coverage in scenarios with a mix of long- and short-range wireless connections. First, we present infrastructure and tools for the investigation of such schemes. Then, we extend existing and design novel schemes for low-power mixed-range application scenarios. More concretely, we make the following contributions in this thesis:

- We present a platform design for wireless IoT devices that support multiple modulation schemes and therefore offers low-power short and long-range communication capabilities. The use of clear abstractions and a well-defined interface enables modular integration which allows the platform to be used for prototyping as well as for real-world deployments. A detailed analysis of relevant timing overheads form the basis for the implementation of timing-critical communication schemes.
- We introduce a testbed architecture for testing and validation of mixed-range communication protocol implementations with large link distances between testbed nodes. In addition, the testbed architecture allows simultaneous high-quality tracing of different aspects including timing, power consumption, and functional correctness. Moreover, it provides support for remote accessing the on-chip debug and trace sub-

system that is built-in in modern microcontrollers.

- Based on the introduced testbed architecture, we propose a methodology for non-intrusive and instrumentation-free tracing of the program execution on all testbed nodes simultaneously. In contrast to commonly used approaches, our methodology allows tracing with high temporal resolution and without requiring interactions from the experimenter. Furthermore, our proposed method includes a time synchronization scheme allowing to align the traces obtained from the distributed testbed nodes on a common time axis.
- We propose two communication schemes based on time-division multiple access (TDMA) channel access that are compatible with the widely used Long Range Wide Area Network (LoRaWAN) communication protocol. With an analysis that takes into account legally enforced duty-cycle limits, we show that our proposed schemes increase reliability and energy-efficiency for a wide range of application scenarios when compared to the default random channel access scheme of LoRaWAN.
- We present a protocol that makes use of synchronous transmission floods with multiple modulation settings to enable reliable and high coverage multi-hop communication in networks with inhomogeneous link characteristics. We propose a scheme based on non-destructive probing within floods that allows to collect connectivity information of the network in parallel to data transmissions without adding significant overhead. The obtained connectivity information enables a wide range of optimizations. In a extensive evaluation, we show how the use of multiple modulations and applying the proposed optimizations can reduce the power consumption significantly for certain application scenarios compared to a state-of-the-art protocol using a single modulation and no optimizations.

Zusammenfassung

Heutzutage werden immer mehr Objekte mit dem Internet verbunden, da das daraus resultierende Internet der Dinge (IoT) eine breite Palette von Möglichkeiten bietet. Für die Anbindung von Objekten an das Internet wird häufig drahtlose Kommunikationstechnologie mit geringem Stromverbrauch eingesetzt, da sie den Vorteil hat, dass am Standort des Objekts keine Infrastruktur erforderlich ist. Bis vor kurzem war die drahtlose Kommunikation mit geringem Energieverbrauch auf kurze Entfernungen beschränkt. In den letzten Jahren haben jedoch Fortschritte in der Sender-Empfänger-Technologie die Kommunikation mit geringer Leistung über grosse Entfernungen ermöglicht, so dass Objekte verteilt über ganze Städte drahtlos verbunden werden können. Eine grundlegende Schwierigkeit der neuen Technologie besteht darin, dass Übertragungen, die für unterschiedliche Entfernungen geeignet sind, sehr unterschiedliche Eigenschaften aufweisen. Stromsparende Übertragungsverfahren für grosse Entfernungen bieten in der Regel nur niedrige Datenraten, was zu relativ langen Übertragungszeiten und einem deutlich höheren Energieverbrauch im Vergleich zu Übertragungen über kurze Entfernungen führt. Dies erschwert den Austausch oder die Erweiterung von Kurzstrecken- mit Langstreckenkommunikationstechnik.

In dieser Arbeit konzentrieren wir uns auf den Entwurf zuverlässiger und energieeffizienter Systeme, die eine hohe Abdeckung in Szenarien mit einer Mischung aus drahtlosen Verbindungen mit langer und kurzer Reichweite bieten. Zunächst stellen wir Infrastruktur und Werkzeuge für die Untersuchung solcher Systeme vor. Dann erweitern wir bestehende und entwerfen neue Verfahren für Anwendungsszenarien mit unterschiedlicher Reichweite und geringem Stromverbrauch. Konkret beinhaltet die vorliegende Arbeit die folgenden Beiträge:

- Wir stellen eine Plattform für drahtlose IoT-Geräte vor, die mehrere Modulationsverfahren unterstützt und daher stromsparende Kurz- und Langstrecken-Kommunikationsmöglichkeiten bietet. Die Verwendung klarer Abstraktionen und einer klar definierten Schnittstelle ermöglicht eine modulare Integration, wodurch die Plattform sowohl für das Erstellen von Prototypen als auch für reale Einsätze verwendet werden kann. Eine detaillierte Analyse der relevanten zeitlichen Abläufe bildet

die Grundlage für die Implementierung von zeitkritischen Kommunikationsverfahren.

- Wir stellen eine Architektur für das Testen und die Validierung von Implementierungen von Mixed-Range-Kommunikationsprotokollen mit grossen Entfernungen zwischen den Knoten vor. Darüber hinaus ermöglicht die Architektur die gleichzeitige, qualitativ hochwertige Verfolgung verschiedener Aspekte wie Zeitverhalten, Stromverbrauch und funktionale Korrektheit. Ausserdem bietet sie Unterstützung für den Fernzugriff auf das On-Chip Debug- und Trace-Subsystem, das in modernen Mikrocontrollern enthalten ist.
- Basierend auf der vorgestellten Test- und Validierungs-Architektur schlagen wir eine Methodik zur nicht-invasiven und instrumentierungsfreien Ablaufverfolgung der Programmausführung auf allen Knoten gleichzeitig vor. Im Gegensatz zu gängigen Ansätzen erlaubt unsere Methode eine Verfolgung mit hoher zeitlicher Auflösung und ohne Interaktionen durch den Experimentator zu erfordern. Darüber hinaus beinhaltet die von uns vorgeschlagene Methode ein Zeitsynchronisationsschema, das es erlaubt, die von den verteilten Knoten erhobenen Zeitverläufe auf einer gemeinsamen Zeitachse auszurichten.
- Wir schlagen zwei Kommunikationsschemata vor, die auf dem TDMA-Kanalzugriffverfahren (Time-Division Multiple Access) basieren und mit dem weit verbreiteten LoRaWAN-Kommunikationsprotokoll (Long Range Wide Area Network) kompatibel sind. Mit einer Analyse, die die gesetzlich vorgeschriebenen Duty-Cycle-Grenzen berücksichtigt, zeigen wir, dass die von uns vorgeschlagenen Schemata die Zuverlässigkeit und Energieeffizienz für eine Vielzahl von Anwendungsszenarien im Vergleich zum standardmässigen zufälligen Kanalzugriffsverfahren von LoRaWAN erhöhen.
- Wir stellen ein Protokoll vor, das Fluten von synchronen Übertragungen mit mehreren Modulationseinstellungen nutzt, um eine zuverlässige Multi-Hop-Kommunikation mit hoher Abdeckung in Netzwerken mit inhomogenen Verbindungseigenschaften zu ermöglichen. Wir schlagen ein Schema vor, das auf zerstörungsfreiem Sondieren innerhalb von Fluten basiert und es ermöglicht, Konnektivitätsinformationen des Netzwerks parallel zu Datenübertragungen zu sammeln, ohne signifikanten Zusatzaufwand hinzuzufügen. Die erhaltenen Konnektivitätsinformationen ermöglichen eine breite Palette von Optimierungen. In einer umfangreichen Evaluierung zeigen wir, wie die Verwendung mehrerer Modulationen und die Anwendung der vorgeschlagenen Op-

timierungen den Stromverbrauch für bestimmte Anwendungsszenarien im Vergleich zu einem State-of-the-Art-Protokoll mit einer einzigen Modulation und ohne Optimierungen erheblich reduzieren können.

Contents

List of Figures

List of Tables

Acronyms

ADC analog-to-digital converter
API application programming interface

BLE Bluetooth Low Energy

CAD Channel Activity Detection
COTS commercial off-the-shelf
CRC cyclic redundancy check
CSMA carrier-sense multiple access
CSS chirp spread spectrum

DAC digital-to-analog converter
DevAddr device address
DPP Dual Processor Platform
DPP2 Dual Processor Platform 2
DR data rate
DSSS direct-sequence spread spectrum
DUT device under test
DWT Data Watchpoint and Trace Unit

ED end-device
ETM Embedded Trace Macrocell

FCnt frame counter
FCtrl frame control
FEC forward error correction

FHDR frame header
FOpts frame options
FPGA field-programmable gate array
FPort frame port
FRR flood reception ratio
FSK frequency-shift keying

GDB GNU Debugger
GFSK Gaussian frequency shift keying
GNSS Global Navigation Satellite System
GW gateway

HAL hardware abstraction layer

IDE integrated development environment
IoT Internet of Things
ISM industrial, scientific and medical
ITM Instrumentation Trace Macrocell

LDO low-dropout
LoRa Long Range
LoRaWAN Long Range Wide Area Network
LPWAN low-power wide-area network
LSR Long-Short-Range
LWB Low-Power Wireless Bus

MAC media access control
MHDR MAC header
MIB MAC Information Base
MIC message integrity code

NS network server
NSS SPI Slave Select
NTP Network Time Protocol

PC program counter

PDR packet delivery ratio

PMU Performance Monitoring Unit

POR power-on-reset

PPS pulse-per-second

PRU Programmable Real-Time Unit

PTM Program Trace Macrocell

PTP Precision Time Protocol

RF radio frequency

RPC remote procedure call

RTT Real Time Transfer

SBC single-board computer

SDR software-defined radio

SF spreading factor

SNR signal-to-noise ratio

SoC system on a chip

SPI Serial Peripheral Interface

SWD Serial Wire Debug

SWO Serial Wire Output

TDMA time-division multiple access

TPA Trace Port Analyzer

TPIU Trace Port Interface Unit

WSN wireless sensor network

1

Introduction

Advances in microelectronics have made systems for sensing, actuating, and communicating much smaller and less expensive, and therefore they have become ubiquitous today. In the course of digitalization, a large variety of objects are being connected to the Internet in order to automate and optimize workflows and processes, track goods and stock levels, and open up new monitoring opportunities. Consequently, the number of such Internet of Things (IoT) devices increases with a yearly growth rate of more than 20 % [Gar19, Cis20].

In the area of building automation, IoT systems can for example be deployed to measure relevant parameters such as temperature or CO_2 concentration, which can then be used to minimize the use of resources such as heating energy while ensuring a high quality of the indoor climate. In a similar way, in smart cities, IoT devices can be used to measure the power consumption of households and control large consumers, such as the charging of electric cars, to make optimal use of the available power grid transmission capacity. In the domain of environmental monitoring, IoT networks can be deployed to collect data that are important regarding weather, climate, or natural hazards. For example, the seismic activity of mountain slopes or volcanoes can be measured and used for predictions and early warning systems.

All the mentioned examples require a data exchange between the IoT device and the Internet to make use of the collected and processed data. Often, the requirement for an infrastructure with wired communication and wired power supply at the IoT node's location is avoided, because

suitable infrastructure is not available or access to it is difficult. Therefore, wireless communication and power supply based on batteries and energy harvesting solutions are mainly used. However, since frequent maintenance is not possible for a large number of devices or at very remote locations, and energy harvesting solutions cannot be arbitrarily large and expensive either, the available power is heavily limited. As a result, many IoT systems use a low-power wireless technology to connect the nodes to the Internet via a gateway.

There are various low-power wireless physical layer technologies such as Bluetooth, Bluetooth Low Energy (BLE), IEEE 802.15.4. In addition, a multitude of corresponding communication protocols for wireless sensor networks (WSNs) including multi-hop schemes exist. However, most of the systems mentioned are limited to short-range communication, which allows ranges of a few 10 meters for point-to-point connections, which does not meet the requirements of the application scenarios mentioned above. A subset of the nodes of the anticipated applications may be reachable only via connections with a large path loss due to large distances or obstacles.

Recent advances in low-power radio technology enable radios with a significantly larger link budget, e.g. 170 dB for Semtech SX1262 [Sem21] compared to 128 dB for the Texas Instruments CC430 [Tex18]. The improved link budget of modern radios is possible partially due to the ability to send with higher transmit powers but mainly because of the support of novel modulation schemes which enable very low sensitivity levels. Such modulation schemes allow bridging multiple kilometers or larger obstacles at the cost of an increased time-on-air. Consequently, modern radios permit low-power wide-area network (LPWAN) application scenarios such as the use cases mentioned above.

Many low-power long-range transceiver implementations support multiple modulation schemes, i.e. different physical layers. The option to select the modulation scheme extends the design space by a new *modulation* dimension, in addition to the *data rate* and *transmit power* configuration. Selecting different combinations of modulation scheme and modulation settings allows different trade-offs between high data rate and better receiver sensitivity. A long-range modulation configuration, which is suited for high path loss links, requires more energy to transmit the same amount of data compared to using a short-range modulation. Consequently, the share of energy cost for communication in LPWANs is increased compared to short-range only networks. In addition, networks with long-range radios usually exhibit inhomogeneous link characteristics between nodes, i.e. the path loss of the links is diverse. In such mixed-range

LPWAN scenarios, some nodes are for example placed close to each other and therefore the link exhibits a very small path loss whereas other nodes are placed at distances of multiple kilometers or with obstacles in-between and therefore the link has a large path loss. As a consequence of these new challenges, existing protocols for traditional short-range radios are in many cases ill-suited for the above mentioned LPWAN application scenarios.

1.1 Aims of This Thesis

Based on the opportunities and challenges provided by modern low-power communication technologies discussed above, we defined the following aims for this thesis:

- Investigation of how the new modulation dimension of modern LPWAN devices can be leveraged to enable reliable and energy-efficient communication in real-world IoT application scenarios.
- Design and implementation of platforms and tools which are suited to evaluate existing and investigate new schemes related to mixed long- and short-range wireless communication for the IoT.

1.2 Challenges

Related to the aims of this thesis, we identify the following classes of challenges:

Representative Platform: Multiple technologies for LPWAN communication with their advantages and disadvantages are available. In order to derive meaningful results, the investigations need to be based on a concrete representative platform. Based on the aims of this thesis, such a platform is required to support long- and short range communication, provide data processing capacity for executing a communication protocols, and support low-power operation. Furthermore, in order to investigate and realize state-of-the-art wireless communication schemes in detail, low-level observability and controllability of the system is required.

Testing and Validation: Meaningful testing of wireless communication protocols is only possible when using real hardware and a realistic environment due to non-negligible impact of non-ideal behavior. Consequently,

testing LPWAN systems requires not only long-range radios but also a setup with long-range links. Today's platforms for wireless IoT systems exhibit higher clock frequencies and a larger range of power consumption, between low-power operation and high transmit power, compared to traditional platforms. In addition, state-of-the-art protocols are timing sensitive and require as little disturbance from testing as possible to obtain meaningful results. This means high-quality tracing is required with the least possible impact on the system under test.

Imbalance of Cost for Long-Range Transmissions: As mentioned above, modern long-range radios allow a larger link budget in exchange for providing a lower data rate. As a result, transmitting the same amount of data using a long-range modulation is more energy consuming than using a short-range modulation. Communication schemes need to take into account the increased time-on-air as well as the increased power consumption when using long-range communication. As mentioned earlier as well, many real-world application scenarios include a mix of high and low path loss links. Low-power radios usually only support using a single modulation setting at a time. Consequently, if multiple modulations are used, this requires coordinated switching of modulation schemes on all nodes involved in the communication. In order to maximize the system lifetime, the increased power consumption due to long-range transmissions needs to be distributed evenly to all nodes.

1.3 State of the Art

In the following, we provide a brief overview of the available platforms and tools as well as communication protocols for IoT systems.

Platform: A number of schemes for LPWAN systems have emerged in the recent years. Examples are LoRa/LoRaWAN, Sigfox, NB-IoT, and 802.11ah (Wi-Fi HaLow). As previously discussed, we focus on low-power mixed-range LPWAN scenarios where nodes only have a restricted energy budget. Many platforms for long-range communication for commercial and academic purposes are available [TCLW19]. However, many existing platforms have limitations regarding low-power operation [MMR+08], the support for efficient short-range communication [CADH18], and the support to adapt or exchange the lower-layer communication scheme [DHB+17]. This makes it difficult to reuse existing platforms for investigating com-

petitive networking protocols for low-power mixed-range communication protocols.

Tools for Testing: Wireless communication protocols are commonly tested and verified using dedicated testbeds [LFZ+13b, HKWW06, AWCM19, SBWR17, ABF+15]. Testbeds allow to reuse the testing infrastructure and therefore facilitate repeatability. However, today's testbeds are mostly designed for testing traditional short-range communication only. There are different implementations for running a dedicated instance of an LPWAN system [The, chi]. However, they are usually designed for a specific media access control (MAC) layer and therefore provide only limited flexibility for investigating adaptations of the lower layers of the network stack. Furthermore, they do not provide accurate and low-level actuation or tracing infrastructure. In order to perform tests with networks that contain long-range links, temporary testing deployment are frequently used [BVR16, LZK+17]. These deployments are usually not equipped with accurate high-quality actuation and tracing infrastructure but rather only support executing a hard-coded variant of a networking protocol.

Debugging and tracing of the program execution on modern microcontrollers using the on-chip debug and trace hardware is widely used in industry [Arm13]. Non-intrusive hardware assisted tracing based on the same technology is partly also used in testbeds [YMZ19, THBR11, SK13]. However, state-of-the-art testbed systems only allow for a limited polling rate which is not suited for timing-critical communication protocols. In addition, methods to obtain traces that are accurately time-synchronized are not employed, even though this would be crucial when examining the distributed state of a communication protocol executed on a testbed.

Reliable and Energy-Efficient LPWAN Schemes: One of the most widely used LPWAN systems is the Long Range Wide Area Network (LoRaWAN) protocol. There are multiple works that extend or modify LoRaWAN based on coordinated channel access to increase throughput and energy efficiency [RWTP+18, PBB18, GTLN18]. Due to its star-based topology, communication with LoRaWAN is limited to single-hop transmissions.

As discussed previously, many real-world application scenarios include nodes which cannot directly reach the gateway in a single hop. For the implementation of practical systems, systems based on a star topology are extended by introducing relay nodes [ESRB19, DP19, MAG+17, PMMB18b]). Another widely used approach is employing routing based

protocols to enable multi-hop communication in LPWANs. Many approaches are available that use routing combined with a single modulation scheme [AVBMBB18, LK18, ZLS+19, MK20, JZBY+21], only few approaches incorporate multiple modulations [STB+17] and are therefore suited for mixed-range LPWAN scenarios.

Message Flooding based on synchronous transmission primitives found wide adoption in short-range multi-hop networks due to its high reliability and low complexity by eliminating state at intermediate nodes. State-of-the-art schemes apply synchronous transmission flooding also in high coverage LPWANs schemes [BVR16, LZK+17, ZLS+18]. To the best of our knowledge, the available approaches based on flooding do not use multiple modulations schemes and therefore do not exploit the corresponding opportunities to reduce the power consumption in mixed-range scenarios.

1.4 Thesis Contributions and Outline

In this thesis, we make the following contributions. An overview is provided in Figure 1.1. On the one hand, we present methods and tools that are necessary for the design and evaluation of mixed-range LPWAN systems. On the other hand, we introduce schemes that enable reliable and energy-efficient communication in mixed-range LPWANs.

Platform for Wireless Mixed-Range Communication (Chapter 2)

We present the DPP2 SX1262 platform that enables investigating mixed-range LPWAN communication schemes and allows low-level inspection and control. The platform supports a wide range of modulation settings between fast short-range frequency-shift keying (FSK) and long-range LoRa communication. Furthermore, the platform is designed to enable low-power operation and integrates an architecture template that provides clear separation between task classes and therefore supports modular design when integrating the platform into larger wireless IoT systems. Thorough analysis of the relevant timing overheads form the basis for the implementation of timing-critical state-of-the-art and novel communication schemes. We demonstrate this by implementing Gloria, a optimized variant of the Glossy [FZTS11] multi-hop communication primitive based on synchronous transmissions.

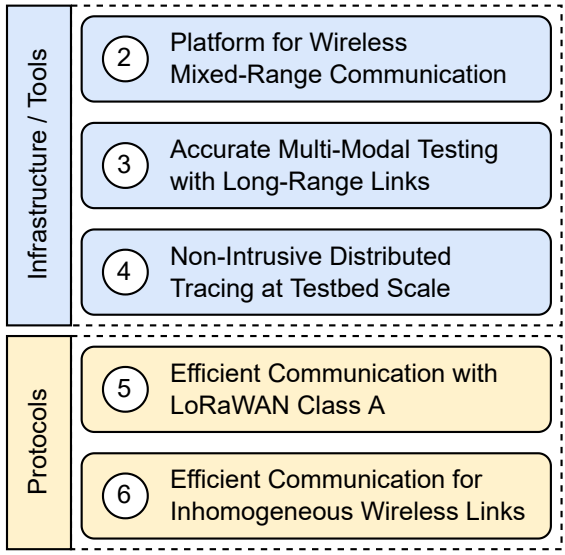


Figure 1.1: Overview of the chapters of this thesis.

Accurate Multi-Modal Testing with Long-Range Links (Chapter 3)

We introduce the FlockLab 2 testbed architecture which combines accurate and high-dynamic range actuation and tracing capabilities with support for large link distances between testbed nodes. This allows simultaneous high-quality monitoring of different aspects including timing, power consumption, and functional correctness of mixed-range LPWAN communication protocol implementations. In addition, the testbed design provides support for remotely accessing the on-chip debug and trace sub-system that is built-in in modern microcontrollers. This enables low-level debugging and tracing of the program execution of individual testbed nodes.

Non-Intrusive Distributed Tracing at Testbed Scale (Chapter 4)

We extend the FlockLab 2 testbed design by a methodology that allows non-intrusive and instrumentation-free tracing of the program execution on all testbed nodes simultaneously. For this, we make use of the FlockLab 2 architecture with the commercially available debug probes and do not need to add any additional custom hardware. The presented

time synchronization scheme makes it possible to time synchronize and therefore align the traces obtained from distributed nodes on a common time axis. This represents an essential and novel capability for accurate low-level tracing the state of timing-critical distributed algorithms.

Efficient Communication with LoRaWAN Class A (Chapter 5)

We present two schemes based on time-division multiple access (TDMA) which are compatible with the LoRaWAN communication protocol. One scheme uses periodic time synchronization whereas the other scheme makes use of on-demand time synchronization. Based on an analysis that compares the proposed schemes to the default LoRaWAN scheme that uses random channel accesses, we show that our proposed schemes provide increased reliability and energy-efficiency for a wide range of application scenarios.

Efficient Communication for Inhomogeneous Wireless Links (Chapter 6)

We present the Long-Short-Range (LSR) protocol which allows reliable multi-hop communication in network with inhomogeneous link characteristics using multiple modulations and synchronous transmission floods. With a scheme based on non-destructive probing within floods, we propose a method to determine the connectivity graph of the network in parallel to data transmissions without adding significant overhead. We show how the use of multiple modulations as well as applying optimizations can be leveraged to reduce the power consumption significantly compared to a state-of-the-art protocol using a single modulation and no optimizations.

2

Platform for Wireless Mixed-Range Communication

A variety of different low-power wireless radio technologies such as IEEE 802.15.4, Long Range (LoRa), NB-IoT are available. To investigate and understand the opportunities of such technologies for mixed-range low-power wide-area network (LPWAN) application scenarios, a concrete platform is required. This platform needs to support multiple modulation schemes to enable long- and short-range communication. On the one hand, the goal is an experimental platform that supports fast prototyping and enables the implementation of state-of-the-art time-critical protocols. On the other hand, the platform should also be suitable for integration into complex real Internet of Things (IoT) deployments.

In this chapter, we present the DPP2 SX1262 platform that consists of a hardware design and a software framework that satisfy the above mentioned requirements. Compared to existing comparable platforms, our platform provides support to implement timing-critical protocols.

The presented platform serves as a basis for a major part of this thesis. First, it is used as a node platform in the design of a mixed-range testbed in Chapter 3 and represents the reference platform for designing a non-intrusive distributed tracing infrastructure in Chapter 4. Furthermore, the platform including the precise timing support and the presented implementation of the Gloria flooding primitive are used for the validation of the mixed-range Long-Short-Range (LSR) protocol in Chapter 6.

2.1 Introduction

Low-power wireless communication is widely used to connect various devices to the Internet via a gateway because it does not require an elaborate infrastructure, such as a wired power supply or a wired network at the node location (see Figure 2.1).

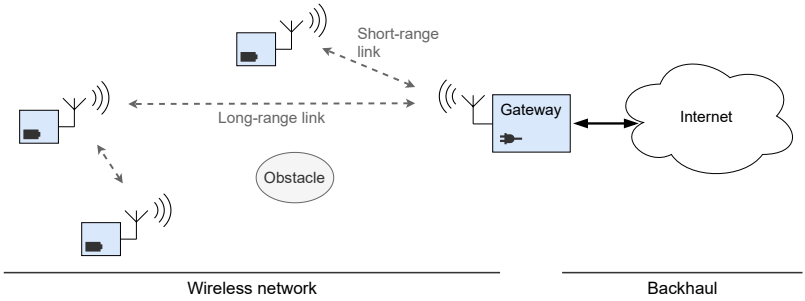


Figure 2.1: Typical setup to connect wireless IoT nodes to the Internet. The wireless IoT nodes without infrastructure and the gateway with access to infrastructure form a wireless network that may contain long- and short-range links as well as nodes that cannot directly communicate with the gateway (e.g. due to obstacles). The gateway serves as a bridge between the wireless network and the backhaul to the Internet.

Today's application scenarios for the IoT range from building automation over smart cities to smart agriculture. Traditional low-power wireless short-range communication technologies, such as IEEE 802.15.4 typically provide 10s of meters of coverage. Using single-hop communication with such technologies would require a large number of gateways to provide the required coverage for the mentioned IoT scenarios. However, deploying gateways with the corresponding backhaul is usually more expensive and difficult than deploying regular nodes because they require more infrastructure. This makes the use of a large number of gateways impractical. The use of multi-hop communication can mitigate this problem to some extent. However, if the density of nodes is low in parts of the network, short-range multi-hop connections do not help either.

Recent advances in low-power wireless radio technology have enabled radios with increased link budgets that provide ranges of multiple kilometers. This enables LPWANs, see Figure 2.2. The larger link budget comes at the cost of a reduced data rate. As a result, the time-on-air of a message

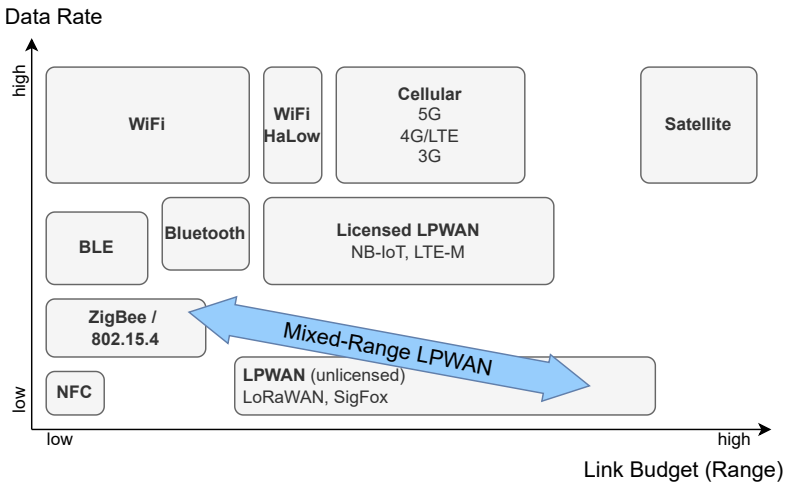


Figure 2.2: Qualitative overview of existing wireless communication technologies (non-exhaustive). The arrow represents the trade-off when using multiple low-power radio technologies. (Figure inspired by [BAR21, IoT21]).

is increased which leads to an increased power consumption for the same amount of transmitted data. For example sending a message with 20 bytes of physical layer payload requires 1.8 ms when using a common short-range scheme (frequency-shift keying (FSK) with a raw bit rate of 125 kbps) and 58.6 ms when using a common long-range scheme (LoRa with SF7). This means, for this example, the long-range communication requires a more than 30 times longer time-on-air and consequently (if using the same transmit power) also more than 30 times more energy compared to the short-range variant.

The opportunities of the new capabilities open up a class of novel research questions. It is for example not straightforward to determine whether a communication via a single-hop long-range connection or a communication via multiple hops is beneficial in terms of power consumption. Such questions involve to determine the required overhead to establish and maintain each of the communication patterns. Furthermore, many scenarios contain not only short-range or only long-range links. In such mixed-range LPWAN scenarios, some nodes are for example placed close to each other and therefore the corresponding link exhibits a very small path loss whereas other nodes are placed multiple

kilometers apart or with obstacles in between and therefore exhibit a large path loss (see Figure 2.1). Consequently, platforms supporting multiple modulations enable versatile use. Having multiple modulations schemes extends the design space for low-power communication protocols as depicted Figure 2.2. In this case, an additional question is how to combine multiple modulations to achieve reliable and energy-efficient communication between the nodes and the gateway.

To investigate the above mentioned and related questions, a concrete platform is required. On the one hand, the goal is an experimental platform that allows to investigate and understand the technical feasibility of flexible mixed-range LPWAN systems. On the other hand, the platform should also be suitable for the integration into real IoT deployments. Concretely, the platform needs to fulfill the following requirements:

- Support of multiple modulation schemes to enable long-range as well as short-range communication.
- Open hardware and software architecture to enable low-level inspection and control.
- Support for precise timing of radio operations to allow the implementation of state-of-the-art timing-critical communication schemes, such as flooding using synchronous transmissions primitives, to support multi-hop communication.
- Support for low-power operation, such that powering the platform from battery for a year or more is feasible.
- Hardware interface and software framework that enable rapid prototyping and debugging and are suitable for the use in real deployments.

In this chapter, we present the DPP2 SX1262 platform that supports long-and short-range communication using the LoRa and FSK physical layer modulation schemes, respectively. We focus on FSK and LoRa because the use of these technologies is possible with low power consumption and, compared to most cellular and licensed technologies such as 5G or NB-IoT, they allow simple stand-alone operation and are therefore independent of cellular operators and the availability of complex network infrastructure. The presented hardware design allows feature-rich debugging, enables low-power operation, and integrates an interface that enables clear separation of communication related tasks from other tasks and therefore allows the modular design of larger IoT systems. Based on extensive analysis of

the timing overheads, we designed a software framework that provides useful abstractions for accurate timing of radio operations and therefore facilitates building more complex wireless communication protocols on top of it.

In this chapter, we make the following contributions:

- We analyze the design space enabled by the selected radio technology.
- We present a hardware design that combines long-range and short-range low-power communication, enables low-power operation, supports low-level debugging, and facilitates the integration into complex systems.
- We design a software framework that takes into account timing overheads of radio operations and provides support for implementing timing-critical communication protocols that switch between different modulations.
- In a case study, we demonstrate the implementation of a timing-critical communication primitive based on synchronous transmission flooding for different long- and short-range modulation settings.

In Section 2.2, we first discuss state-of-the-art platforms and explain the advantages of our platform. Then, we explain the design decisions for our proposed platform, analyze the design space of the selected radio technology, and present a concrete hardware implementation in Section 2.3. In Section 2.4, we describe the corresponding software framework. In Section 2.5, we present a case study in which we implement a timing-critical flooding primitive using the presented platform. Finally, we summarize the chapter in Section 2.6.

2.2 Related Work

Platforms for the research related to wireless communication schemes exist for many years. However, many platform designs are of limited use for investigating low-power mixed-range IoT communication. For example CitySense represents a platform for a city-scale network but makes use of IEEE 802.11 mesh networking which requires wired power supply [MMR+08]. OpenMote uses low-power IEEE 802.15.4 communication but is therefore limited to short-range communication schemes [VTWP15].

Chung et al. built a prototyping platform based on an Arduino and a Sigfox board [CADH18]. With Sigfox, the platform features a long-range radio technology. However, it is lacking an alternative short-range modulation scheme which makes it unsuitable for investigating communication schemes for mixed-range LPWANs. In addition, the Sigfox technology to a large extent consists of closed-source proprietary system components. For example the, the gateway nodes and the backhaul part of the network are operated by the Sigfox company and are not easily replicable in a lab which would be required for investigating and designing communication schemes that require low-level control and inspection.

Other more recent platforms incorporate low-power long-range communication and provide multiple modulation schemes. For example, Signpost [AGJ+18] is a modular platform that allows to combine different sensor modules and provides a power supply infrastructure based on energy-harvesting. A communication module provides support for 3 different communication technologies: cellular, LoRa, and Bluetooth Low Energy (BLE). However, the main focus of the platform is to make use of different sensor modules and only to lesser extent to investigate communication schemes. The LoRaBug nodes from the OpenChirp system [DHB+17] combine a Texas Instruments CC2650 microcontroller with integrated IEEE 802.15.4 and BLE radio interfaces with the Semtech SX1276 LoRa radio. Even though the hardware platform would likely allow direct point-to-point LoRa communication, the platform is designed to use Long Range Wide Area Network (LoRaWAN) to exchange data via a gateway with the Internet. Similar to the Signpost platform, the LoRaBug platform provides support for prototyping higher-layer IoT services and applications and is less focused on investigating low-level communication schemes. Thoen et al. presented a LPWAN communication platform based on an ATmega328p microcontroller and the Semtech SX1276 radio which supports LoRa communication [TCLW19]. They compare their platform to four commercially available platforms and argue that those platforms are not suitable for large-scale deployment of battery-powered sensor nodes as they do not meet the low-cost and low-power requirements. Jiang et al. propose a platform based on an nRF52832 system on a chip (SoC) and the Semtech SX1262 radio chip to combine communication via ANT and LoRa [JZBY+21]. Piyare et al. present a Texas Instruments MSP430 based platform that combines the Semtech SX1276 radio with a wake-up receiver for on-demand communication [PMMB18a]. The SX1276 radio provides long- and short-range communication and acts as a wake-up transmitter.

Many of the mentioned platforms support a combination of multiple

communication interfaces. This increases the system complexity and is not necessary for investigating different modulation schemes as we show with our proposed platform. Furthermore, none of the proposed platforms provide tight timing analysis or abstractions that are required for the implementation of state-of-the-art timing-critical multi-hop protocols.

Our platform supports multiple modulation schemes, namely different LoRa spreading factors (SFs) as well as FSK communication. It is based on the open LoRa ecosystem and therefore allows for low-level inspection and control of the whole communication chain from issuing commands to the radio chip on the IoT node to the arrival of data in an Internet-based service. This enables detailed analysis and control of the timing. The hardware design allows making use of the various low-power operating modes provided by the radio and microcontroller chips. Furthermore, the integrated hardware interface and the software framework allow for straightforward integration into more complex systems used for real deployments.

2.3 Platform Design

In this section, we first motivate our choice of the radio technology and and analyze the resulting design space for mixed-range low-power wireless communication. Then, we explain the architecture of our platform and describe the hardware implementation.

2.3.1 Selection of Low-Power Wireless Technology

As previously discussed, we focus on applications of wireless IoT devices which do not need infrastructure at the node. Also frequent maintenance is not possible. Consequently, the platform needs to support battery-powered operation for longer time periods, i.e. years. Based on this requirement, cellular and licensed LPWAN technologies are not suited as they require a larger energy budget. We identify a mix of low-power short- and long-range (LPWAN) communication technologies, as depicted in Figure 2.2, as a suitable selection.

Based on related work, we find that for low-power long-range communication, the LoRa physical layer is the most promising LPWAN technology as it has an open ecosystem. This is important to investigate and realize new approaches which require low-level modifications or observations. For

example, radio drivers [lor] as well as back end server implementation (e.g. [chi, The]) are available as open-source. With LoRaWAN, a widely used higher-layer protocol with openly accessible specifications is available, too.

In order to keep the system and hence the required precise timing analysis simple, we limit ourselves to a single radio interface. We identify the SX1262 radio chip as a good candidate for the low-power mixed-range LPWAN because it allows FSK short-range high-data rate communication as well as LoRa for long-range low data rate communication.

2.3.2 Design Space for Mixed-Range Wireless Communication

With FSK and LoRa, we have two modulation schemes that provide different trade-offs and therefore cover different parts of the overall design space. With FSK, digital information is represented by a discrete set of of frequency offsets from the carrier frequency. The LoRa modulation scheme uses the chirp spread spectrum (CSS) technique and cyclic shifted chirp pulses are used to represent the digital data. In contrast to FSK, LoRa uses a forward error correction (FEC) with 4 different coding rates (4/5 to 4/8). The basic relations are the same for both schemes. To achieve long-range communication, a large link budget is required. The link budget is mainly determined by the transmit power and the receiver sensitivity. As we focus on low-power radio technologies, the transmit power is limited. However, spreading the message over a larger amount of time as well as as reducing the bandwidth of the radio frequency (RF) signal can be used to improve the sensitivity and therefore to increase the available link budget. At the same time, these parameter changes lead to a reduction of the achievable data throughput. As a result, the time which is required to transmit a fixed amount of data increases and, consequently, also the required energy for transmitting the data increases as well.

To get an overview of the design space which is covered by the SX1262 radio, we select a concrete set of configurations listed in Table 2.1 and use datasheet values, measurements, and simplified calculations in order to analyze the trade-offs.

First, we look at the time-on-air, i.e. the duration which is required to transmit a packet. For calculating the time-on-air, we use the equations provided in the datasheet [Sem21]. Based on this, we also calculate the actually achieved bit rate, i.e. the the rate with which the physical-layer payload is transmitted considering the overhead of the physical layer. The

	LoRa	FSK
Payload (PHY layer)	20 bytes	20 bytes
Bandwidth	125 kHz	according to data rate
Header	yes	yes
CRC	yes	yes
Sync word	n/a	yes
Low data rate optimization	no	n/a

Table 2.1: Used configuration parameters for the design space analysis.

results for different modulation settings are depicted in Figure 2.3. Please note that the obtained data rate values are limited by the fixed number of physical payload bytes and could be improved by increasing the payload size of the packet.

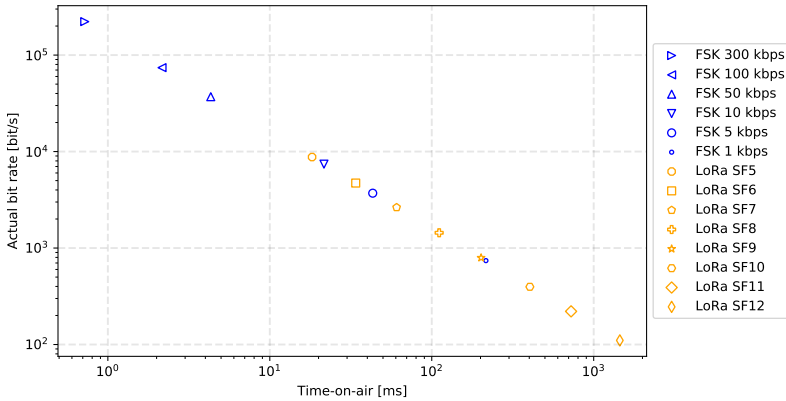


Figure 2.3: Time-on-air and actual bit rate for different modulations settings when using the parameters in Table 2.1.

Using the same radio configuration parameters and set of modulation settings, we also analyze the energy cost and the achievable range of transmissions. We obtain energy per bit values by dividing the actual energy consumption for transmitting a message by the number of physical layer payload bits. The energy consumption is derived from the calculated time-on-air values and power consumption measurements performed with the hardware platform described in Section 2.3.4 using the RocketLogger [SGL+17]. For determining the range, we first determine the link budget. For this, we use the interpolated sensitivity values from the

datasheet [Sem21] and the configured transmission power and calculate the link budget according to relation (2.1).

$$\text{LinkBudget} = \text{TxPwr} - \text{Sensitivity} \tag{2.1}$$

Then, we estimate the maximum range that can be achieved with a point-to-point communication according to the Friis free-space loss equation [Fri46] assuming an antenna gain of $G = 1$ for the antennas at the transmitter and receiver. The corresponding results are plotted in Figure 2.4.

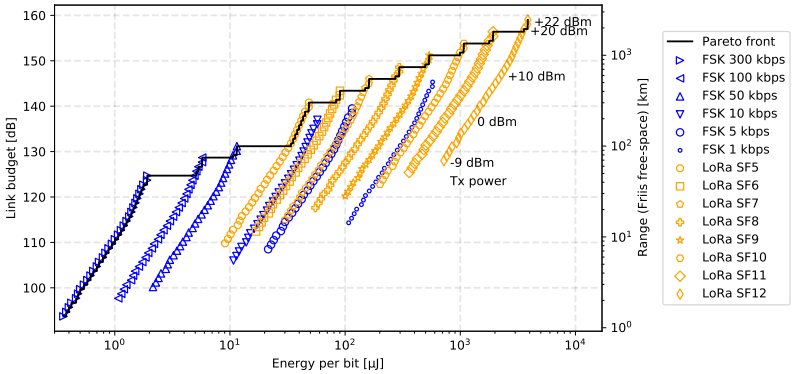


Figure 2.4: Maximum achievable link budget and range versus energy per bit for point-to-point communication when using different modulation and transmit power settings as well as the parameters in Table 2.1. Please note that the figure includes all available LoRa settings but only shows a subset of the possible settings for FSK.

From the two plots, we see that the design space of the FSK and LoRa modulation schemes overlap. In addition, from Figure 2.4 we see that for the LoRa modulation scheme, increasing the transmit power (Tx power) is cheaper in terms of energy per bit compared to switching the LoRa modulation setting (next higher SF). Only if the available transmit power levels are not sufficient to increase the link budget enough, it makes sense to switch to the next modulation setting. The values for the plotted FSK modulation settings suggest an analogous relationship for the FSK modulation scheme. However, it should be noted that the FSK modulation scheme offers a more fine-grained range of settings (relevant settings include the physical data rate and the used bandwidth) and therefore would

allow additional values between the depicted modulation settings.

2.3.3 Integration of the Dual Processor Platform Architecture

To facilitate the integration of the proposed mixed-range communication platform into more complex IoT devices, we build on the Dual Processor Platform (DPP) [SZDF+15, BTDF+19]. DPP is an architecture template for networked embedded systems based on the asynchronous processor interconnect that allows to minimize interference with a proven predictable behavior. The idea is that tasks are mapped onto two different physically separated processing elements (usually low-power microcontrollers). While the communication processor (COM) handles wireless packet transmission and reception, the application processor (APP) is dedicated to the sensor data acquisition, data processing, and actuation. This allows that each sub-system can be optimized according to its individual requirements. Such hardware partitioning is a standard approach, frequently found in more complex sensor system implementations [GBG+12]. In order to allow data exchange between the processors but keeping the strict separation, the BOLT [SZDF+15] processor interconnect is used. It enables a strict decoupling of the power, clock, and time domains of the two processing elements by allowing only asynchronous message passing between the two.

The limitation to an asynchronous interface supports a predictable run-time behavior that can be formally verified [SZDF+15]. In addition, DPP provides the advantage that each subsystem can take independent decisions on when to utilize low-power modes. Furthermore, task decoupling significantly reduces the complexity of software development and validation. Finally, due to the provided modularity, exchanging one of the sub-systems does not require a change to the other sub-system.

The latest version of the this modular system architecture is called DPP2 and is depicted in Figure 2.5 [BTDF+19]. With DPP2, the *application board* acts as a motherboard for the smaller *communication board*. It typically integrates I/O, power management, debugging and other support functions. BOLT resides on the communication board to keep this board as small and simple as possible. The interconnect between the two boards is a 2-row 26-pin high density header. This board-to-board connector includes 7 pins for the BOLT message passing interface, 3 pins for power supply lines (voltage for COM and BOLT plus ground), and multiple interfaces for

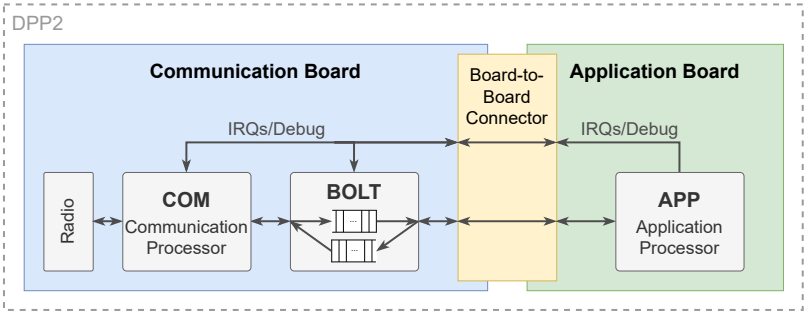


Figure 2.5: DPP2 architecture [BTDF+19] consisting of a communication and an application board connected by a board-to-board connector. The stateful BOLT interconnect decouples time, power, and clock domains of the two boards.

programming and debugging the microcontrollers.

2.3.3.1 Opportunities of Precise Timing

Time synchronization is a central component of many distributed systems and is therefore also required in some form by many wireless IoT networks. Usually, energy-constrained IoT nodes use clocks that exhibit a significant drift over time. A typical use case of precisely timed wireless operations is to apply clock drift compensation.

The presented platform provides support to achieve precise timing by using timer-based timestamping and execution of radio operations. This timer-based approach has the advantage that the timing is largely independent from influences of the regular program execution and is straightforward for the implementation of time synchronization schemes. More implementation details are provided in Section 2.4.3.

Time synchronization can help to make a network protocol more reliable and energy efficient. For example, protocols based on time-division multiple access (TDMA) reduce or eliminate collisions by giving exclusive access of the RF frequency channel to a single node. Other schemes, for example multi-hop flooding with Glossy [FZTS11], are based on synchronous transmissions where it is important that the transmissions of different nodes are accurately aligned in time.

The knowledge of timing overheads allows for predictable switching between different modulation and radio operating modes. Together with time synchronization, this enables protocols to switch modulations in a

coordinated and therefore reliable fashion network-wide.

Another use case for time synchronization is the time stamping of collected data or time accurate actuation. In the framework of the DPP2 SX1262 platform, such actions take place on the application processor. For this purpose, the application processor can be time-synchronized to the communication processor by using the interrupt (IRQ) line of the DPP2 board-to-board interface.

2.3.4 Hardware Implementation of the DPP2 SX1262 Communication Board

With the DPP2 SX1262 communication board[1] depicted in Figure 2.6, we present an implementation of the described hardware design. The board features the STMicroelectronics STM32L433CC ARM Cortex-M4 microcontroller and the Semtech SX1262 [Sem21] long-range low-power sub-1-GHz (868 MHz in our case) transceiver which allows up to 170 dB of link budget. A Serial Peripheral Interface (SPI) bus is used as a connection between the radio and the microcontroller (also see Figure 2.8). BOLT is implemented on a Texas Instruments MSP430FR5969 with 64 kB of ferroelectric RAM (FRAM). We list the relevant specifications of the platform in Table 2.2. Please note that we specify the technically correct term Gaussian frequency shift keying (GFSK) in the table, but we use FSK for it everywhere else in the thesis for the simplicity.

Both the STM32L4 microcontroller and the radio offer low-power modes with sub-microwatt power consumption. In addition, the SX1262 radio features a very efficient receive mode when using the integrated DC-DC converter. The microcontroller provides enough processing performance and storage to support the implementation of a real-time operating system and a non-trivial protocol stack as we demonstrate in Section 2.5. In addition, the ARM Cortex-M4 features the on-chip debug and trace subsystem that allows accurate low-level debugging of the program execution. The DPP2 board-to-board connector represents a well defined interface that simplifies the integration and reuse of the communication board. For debugging on the desk, we use the *Application Development Board*[1] which represents an implementation of a DPP2 application board based on a Texas Instruments MSP432 ARM Cortex-M4F processor and provides power supply and headers for attaching programming and logging probes. In another use case, the DPP2 SX1262 board was integrated into a geophone

[1] https://gitlab.ethz.ch/tec/public/dpp/dpp/-/wikis

sensor node measuring seismic events and was used in a real-world deployment [BFG+21]. In Chapter 3, we use the DPP2 board-to-board connector interface to integrate the DPP2 SX1262 nodes into a testbed infrastructure.

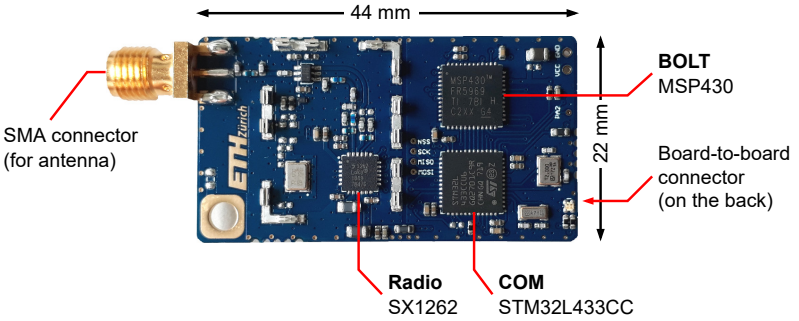


Figure 2.6: DPP2 SX1262 Communication Board featuring an STM32L4 microcontroller, the Semtech SX1262 LoRa capable radio and BOLT.

Microcontroller: STMicroelectronics STM32L433CC	Architecture	ARM Cortex-M4
	Max. clock frequency	80 MHz
	Memory	256 kB Flash, 64 kB SRAM (16 kB of them with option for retention)
	Min. power consum.	0.28 µA @1.8 V (standby with RTC)
Radio: Semtech SX1262	Min. sensitivity	−148 dBm (LoRa SF12, BW=10.4 kHz)
	Max. Tx power	+22 dBm
	Modulations schemes	LoRa (SF5 - SF12), GFSK (0.6 - 300 kbps)
	Min. Rx power consum.	4.6 mA @3.3 V (DC-DC mode, LoRa)
	Min. sleep power consum.	160 nA @3.3 V (cold sleep)

Table 2.2: Specifications of the DPP2 SX1262 communication board implementation [STM21, Sem21].

2.4 Software Framework

In this section, we explain the relevant aspects of the *flora* software framework we developed for the DPP2 SX1262 platform. It includes support for using the low-power modes and facilitates the implementation of

higher-layer protocols that require precise timing. The flora software framework is available as open-source[2] [TDFW+21].

2.4.1 Overview

First, we provide an overview of the main parts of the software framework.

Radio Driver: The SX1262 radio includes its own timers for timeouts, a packet engine for the different modulations, and a state-machine. These features allow a command-based high-level interface for controlling the radio, which represents a significant advancement over traditional radio chips. The radio driver is based on the Semtech reference implementation [lor]. We extended and modified the driver to enable precisely timed radio operations (for more details see Section 2.4.3).

Bare-Metal vs. Operating System: The timing-critical components such as the radio driver and lower-layer communication primitives (see Section 2.5) are event-driven and implemented using the hardware abstraction layer (HAL) provided by the integrated development environment (IDE) but without the use of an operating system. Nevertheless, we integrate the FreeRTOS real-time operating system to provide abstractions for the implementation of higher-layer communication protocols. Tasks, for example, provide a clean separation of different functional components of the communication software and queues serve as a commonly used abstraction for temporarily storing messages. We use cooperative scheduling to avoid unpredictable concurrency issues.

Peripherals: An overview of the used peripherals is listed in Table 2.3. We use the 32 bit general-purpose timer TIM2 as a high-speed timer for timing-critical operations such as radio interactions. For time-triggered wake-up from low-power modes, we use the 16-bit low-power timer LPTIM1.

Interrupts: There are two main sources for interrupts: (1) The previously mentioned timers of the STM32L433 microcontroller, and (2) the SX1262 radio. The most important SX1262 interrupts are listed in Table 2.4 and visualized in Figure 2.7.

BOLT Interface: The flora framework integrates methods to interface with BOLT in order to asynchronously exchange messages with an application processor connected via BOLT and the DPP2 board-to-board connector.

[2] https://gitlab.ethz.ch/tec/public/flora/wiki/-/wikis

Peripheral	Description & usage
USART1	Serial communication (e.g. logging)
SysTick	Timer for FreeRTOS
TIM1	Timer for HAL delay
TIM2	Timer for flora (incl. timing of radio)
LPTIM1	Low-power timer for wake-up from low-power mode
SPI1	SPI communication with BOLT
SPI2	SPI communication with SX1262 radio

Table 2.3: Overview of a subset of the STM32L433CC peripherals and their use in the flora software framework.

Interrupt	Modulation		Description
	LoRa	FSK	
TxDone	•	•	Packet transmission completed
RxDone	•	•	Packet reception completed
PreambleDetected	•	•	Preamble detected
SyncWordValid		•	Valid FSK sync word detected
HeaderValid	•		Valid LoRa header received
HeaderErr	•		LoRa header CRC error
CrcErr	•	•	Wrong CRC received
Timeout	•	•	Rx or Tx timeout

Table 2.4: Overview of relevant interrupts that can be triggered by the SX1262 radio [Sem20].

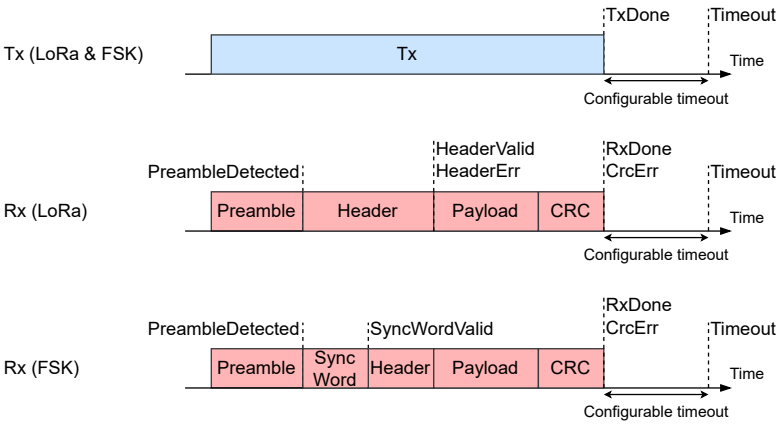


Figure 2.7: Visualization of SX1262 radio interrupts.

2.4.2 Low-Power Operation

As explained in the Section 2.1, we assume that there is no infrastructure at the node locations and that the nodes are usually battery-powered. Energy harvesting is also used in certain cases, but is limited by the available space and the cost. As a result, the nodes have only a very limited energy budget. To ensure operation over a longer period of time without external intervention, it is therefore necessary to optimize all components for energy-efficient operation.

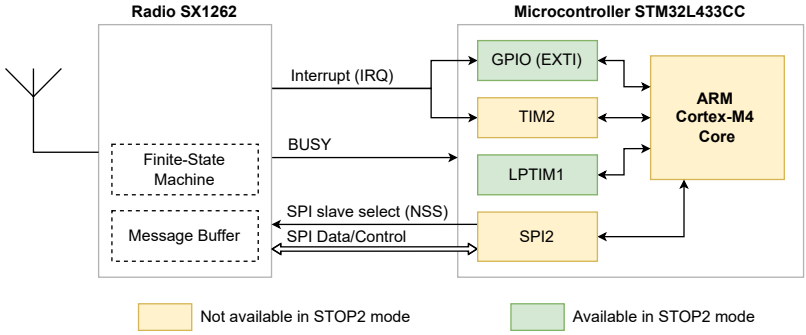


Figure 2.8: Interface between radio and microcontroller on the DPP2 SX1262 platform. In addition, the active components inside the microcontroller during low-power modes are shown.

The flora software framework uses the STOP2 mode of the microcontroller (1.3 µA @1.8 V [STM21]) for low-power operation. As depicted in Figure 2.8, this allows to keep the GPIO EXTI and LPTIM1 peripherals active in order to support event-based and time-triggered wake-up of the microcontroller, respectively. In addition, the STOP2 mode supports retention of SRAM and peripheral registers. For the radio, the warm sleep mode (0.6 µA @3.3 V) which allows configuration retention is used. When the radio is in low-power mode, the SPI interface is not available. In this case, the the microcontroller can wake up the radio by pulling the SPI Slave Select (NSS) (active low) to low. The interrupt line (IRQ) from the radio to the microcontroller is connected to two inputs: (1) to the timer TIM2 for accurate timestamping in active mode, and (2) to GPIO EXTI in order to wake up the microcontroller in low-power mode.

In order to the allow low-power operation in conjunction with using FreeRTOS, the flora library makes use of the tickless-idle mode and idle task hooks concepts of FreeRTOS. Whenever there is no task (except the

idle task) ready for execution, FreeRTOS executes a function that prepares all peripherals for low-power operation and puts the microprocessor into STOP2 mode. If an interrupt is triggered and the corresponding interrupt service routine is executed, another function is called which re-initializes all peripherals. Depending on the updated state, the task execution of FreeRTOS continues or FreeRTOS goes back to sleep again. The tick interrupts that are used to advance the FreeRTOS scheduler during normal mode, would wake-up the microcontroller from low-power mode. Configuring the tickless-idle mode, allows disabling the periodic tick interrupts during idle periods.

2.4.3 Precise Timing

In many use cases, time synchronization is an important aspect of an IoT system (also see Section 2.3.3.1). In order to accurately time-synchronize multiple nodes via wireless communication the timing behavior of the entire chain from executing the function to send a message on the transmitter until the handling of a corresponding interrupt triggered on the receiver needs to be understood and precisely timed. Concretely, this involves:

1. The delay from the function call on the microcontroller until the radio starts transmitting the beginning of the message.
2. The transmission time, i.e. the time it takes until the end of the message arrived at the radio of the receiver, which consists of the time-on-air (mathematical equation provided in datasheet) and the propagation time (physical property of the physical medium)
3. The delay between the arrival of the message at the receiver's radio until the corresponding interrupt is triggered on the microcontroller.

For the second part, a good estimate can be calculated by using the equation for calculating the time-on-air and equations for modeling the propagation time of electromagnetic waves. The first and the third part depends on the concrete hardware design of the radio. Whereas the documentation [Sem20] and an open-source driver implementation [lor] for the SX1262 are available, the hardware design is not publicly available. In this section, we therefore analyze the timing behavior of the relevant parts.

2.4.3.1 Precise Triggering of Radio Operations

Changing the configuration or putting the radio into a different operating mode can be initiated by the microcontroller. For this, the SPI bus lines plus two additional signals, the BUSY signal controlled by the radio and the NSS (slave-select active-low) signal controlled by the microcontroller are used as depicted in Figure 2.9. Via the SPI bus, the microprocessor can send commands to the radio and access the message buffer inside the radio. If the BUSY line is high, this indicates that the radio is busy, i.e. it is for example executing a command, handling an internal interrupt, or sleeping. If the BUSY line is low, the radio is ready to accept a command via the SPI bus. For initiating a change, the microcontroller asserts that the BUSY line is low, pulls the NSS low and starts sending via the SPI bus a command (CMD) followed by writing parameters and reading a status value. As a last step, the NSS line is pushed high by the microcontroller.

Analysis of the exact behavior revealed that some commands are executed right after the last clock cycle of the SPI bus, i.e. before the rising edge of NSS. However, other commands, this includes setting the radio into transmit and receive mode, are executed only after the rising edge of NSS. We use this feature to precisely trigger the transmission of a message. The transmission parameters and the payload of the message are transferred to the radio with separate commands beforehand. Then the command for initiating the mode transition from standby to Tx mode is initiated but not executed, i.e. the rising edge of the NSS line is prevented. Only at the precise moment when the message shall be transmitted, the NSS line is pushed high. This method has the advantage that the timing of putting the radio into Tx mode is independent of uncontrollable overhead and non-deterministic behavior involved with SPI bus transactions.

After the rising edge of the NSS signal, the radio ramps up the power amplifier and then starts transmitting the message. The falling edge of the BUSY line indicates that the radio finished the transition to Tx mode and started transmitting the message. The delay between t_{exec} and the time when the radio is starting the transmission, is in the order of 100 µs [Sem21]. As the hardware implementation is not open, we are unable to understand what it depends on.

2.4.3.2 Precise Timing of Received Messages

As depicted in Figure 2.8, the interrupt line from the radio is directly connected to the input of timer TIM2. This allows to accurately timestamp

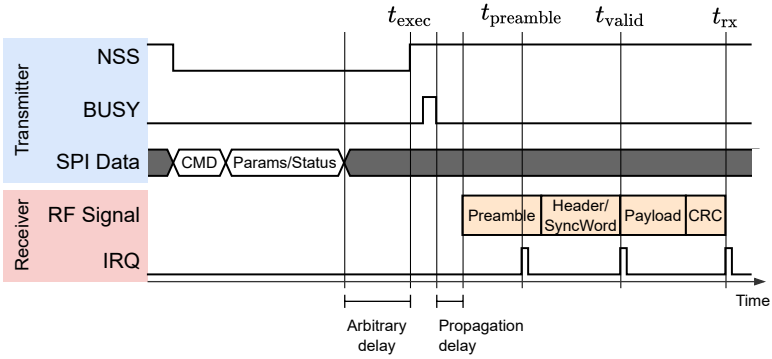


Figure 2.9: Timing diagram with the NSS-based initiation on the sender and the timing of the corresponding interrupts (IRQ line) on the receiver. In the example, we assume successful packet reception and that all radio interrupts are enabled and immediately handled on the receiving node.

radio interrupts in hardware using the input capture method.

As previously discussed, the radio can be configured to trigger multiple interrupts when different parts of the message has been successfully received. Consequently, there are 3 interrupts that can potentially be used for timestamping the reception of a message (also see Figure 2.9):

PreambleDetected: This interrupt is triggered when any preamble that matches the currently active radio settings is observed by the radio. As the minimum preamble length required for detection on the receiver is usually configured to be shorter than the transmitted preamble, preamble detection can occur at arbitrary positions towards the end of the preamble as indicated by t_{preamble}.

HeaderValid / SyncWordValid: The `HeaderValid` and `SyncWordValid` interrupts are triggered when the header of a LoRa or the synchronization word of a FSK message has been received successfully, respectively. This option corresponds to t_{valid} in Figure 2.9.

RxDone: The `RxDone` interrupt is triggered when the complete message has been received and the cyclic redundancy check (CRC) is successful. This option corresponds to t_{rx} in Figure 2.9.

Due to the large inherent jitter of the `PreambleDetected` interrupt, we do not consider this method to be useful for accurate timing of received

messages. We conduct experiments to measure the accuracy of the two remaining two options. For this we are interested in the jitter of the two delays:

$$\Delta_{\text{valid}} = t_{\text{valid}} - t_{\text{exec}}$$
$$\Delta_{\text{rx}} = t_{\text{rx}} - t_{\text{exec}}$$

We use a setup with two DPP2 SX1262 nodes, one transmitting and one receiving node. Each of the nodes is equipped with a logic tracing device that is accurately time synchronized based on Global Navigation Satellite System (GNSS) and provides a resolution of 0.1 µs. The testing setup is based on the FlockLab 2 testbed which described in more details in Chapter 3. The use of independent logic tracing has the advantage of allowing experiments with a realistic setup and environment including large physical distance between the two nodes without the requirement of long wiring for tracing.

We configure the sending node to periodically send packets with a physical payload size of 10 bytes and to toggle a GPIO pin right before executing the command instructing the radio to go into transmit mode. The logic tracing device on the sending node logs this GPIO pin toggle, i.e. we collect timestamp corresponds to t_{exec}. On the receiving node, the radio is configured to continuously receive packets and to trigger the interrupts that correspond to t_{valid} and t_{rx}. The interrupt line of the radio is traced by the logic analyzer on the receiving node.

We vary the configured modulation and for every modulation setting we perform 1000 packet transmissions. The two nodes used for this measurement are located in adjacent office rooms. The resulting range of the distributions of the two delays are listed in Table 2.5.

The results show that the timing of FSK compared to LoRa is significantly more precise. In addition, there is a trend that the measurements based on t_{rx} provide a narrower distribution compared to t_{valid}. However, this does not hold for all modulation settings. Especially, with certain FSK settings, the measurements based on t_{rx} provide very bad precision. The reason for this are outlier values. The presented data contains 39 and 1 out of 1000 samples with a deviation of more than 20 µs from the bounds of the 95 % confidence interval of the Δ_{rx} delay distribution for FSK 200 kbps and FSK 250 kbps, respectively.

We repeated the experiment with different nodes with different distances, and different payload sizes and did not observe significant differences. As expected, for node combinations that exhibit a larger distance

Modulation	Number of samples	95% Conf. int. [us]		Entire range [us]	
		Δ_{valid}	Δ_{rx}	Δ_{valid}	Δ_{rx}
LoRa SF12	1000	4.5	4.5	5.5	5.5
LoRa SF11	1000	4.9	7.7	6.6	8.5
LoRa SF10	1000	5.1	4.5	7.1	5.4
LoRa SF9	1000	4.6	4.4	6.1	8.0
LoRa SF8	1000	4.5	4.4	5.2	7.9
LoRa SF7	1000	4.7	4.6	5.2	5.1
LoRa SF6	1000	4.9	3.8	5.9	5.0
LoRa SF5	1000	4.6	4.4	5.7	5.1
FSK 125 kbps	1000	2.3	1.5	4.1	2.0
FSK 200 kbps	1000	2.5	22.7	3.6	1215.3
FSK 250 kbps	1000	1.2	0.9	1.9	255.8

Table 2.5: Measurements of the jitter of the two delays Δ_{valid} and Δ_{rx}. The numbers for the delays represent the range of the corresponding distribution.

between sender and receiver the propagation time is visible in the absolute value of both delays (for more details see Section 2.5.1).

In the performed measurement runs, we observed many outliers for FSK modulation and in rare cases also for LoRa modulations. The experiments suggest that low signal levels at the receiver additionally increase the amount of outliers. In our experiments, we registered outliers almost exclusively in measurements derived from t_{rx}. Based on these observations, we decide to use t_{valid} for the precise timing of receiving packets.

2.5 Gloria: Implementation of a Synchronous Transmission Flooding Primitive

In order to demonstrate how the presented platform can be used to implement a state-of-the-art timing-critical protocol for low-power wireless networks, we port the Glossy [FZTS11] multi-hop network flooding protocol to the DPP2 SX1262 platform. We call our implementation Gloria to highlight that we optimized and therefore adapted the original Glossy scheme for the implementation on the mixed-range DPP2 SX1262 platform. Gloria, and also Glossy, make use of the concept of synchronous transmissions to flood messages over multiple hops. According to this

concept, multiple nodes send the same message simultaneously. Thanks to the capture effect, other nodes are able to successfully receive the message. By repeated retransmission of the message by all nodes that received it, the flood propagates through the network. This enables one-to-all packet dissemination that does not need information about the network topology and requires no in-network buffering of messages at intermediate nodes after the flood has ended. Previous work shows that the concept of synchronous transmission flooding can also be used in combination with the LoRa modulation scheme [BVR16, LZK+17, ZLS+18].

Gloria includes improvements to the original Glossy scheme regarding the Rx-Tx sequence that have been introduced with Robust Flooding by Lim et al. [LDFST17] and which are for example also included in BlueFlood [ANDL19]. In contrast to Robust Flooding, we do not implement frequency-hopping or the randomized transmit power scheme. As depicted in Figure 2.10, with Gloria a single node initiates the flood by sending a message multiple times in successive time slots. All nodes that received a message synchronously retransmit it a pre-defined number of times (3 times in our example) in subsequent slots. In contrast to Glossy, in Gloria the timing of packet retransmissions within a flood is based on timer events rather than the directly preceding reception of a packet. As a consequence, a node successfully receives a packet within a Gloria flood at most once and retransmits packets multiple times back-to-back. This reduces the overhead of a flood compared to the original Glossy implementation as all but one receive slots are eliminated. In addition, timer-based execution of radio operations provides great flexibility to investigate further variations of the synchronous transmission flooding schemes, e.g. with a different sequence of Rx and Tx slots or with adding additional slot types. An example of such a variation using delay slots for implementing delayed retransmissions is proposed in Section 6.4.

For the implementation, we make use of the results from the timing analysis presented in Section 2.4.3. In order to support the different modulations offered by the platform, we measured relevant flood timing parameters for 11 representative modulation configurations and integrated them into the software framework as lookup tables. This for example includes the minimum overhead between finishing a receive and starting a transmit operation, or the time between the `TxMarker` and the corresponding `SyncWordValid/HeaderValid` interrupt.

In the following, we provide a brief overview of a subset of the interface of Gloria. The full implementation is available as open-

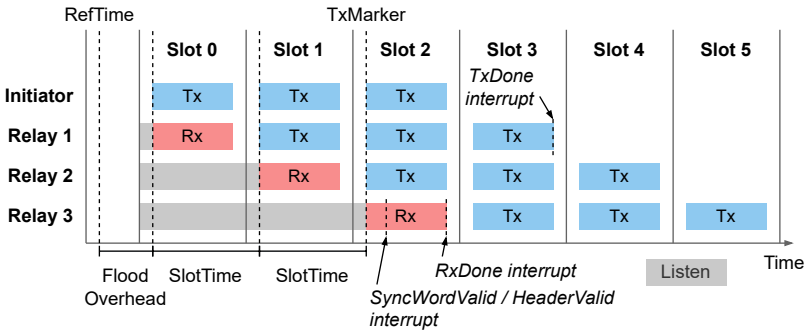


Figure 2.10: Overview of the timing of a Gloria flood. The number of retransmissions is a parameter of the Gloria primitive (here set to 3).

source[3] [TDFW+21].

`set_modulation(mod_index)`: Configure Gloria to use one of the predefined flora modulations using the modulation index `mod_index`.

`get_flood_time(payload_len, num_slots)`: Returns the duration of the flood based on the current modulation setting of Gloria as well as the provided payload length `payload_len` and number of Gloria slots `num_slots`. The obtained value is used by the calling application or network layer to determine when to stop Gloria.

`start(is_initiator, *payload, n_tx, …)`: Starts Gloria. If `is_initiator=1`, the node initiates, i.e. starts sending, a flood. Otherwise, a node starts participating in a flood, i.e. the node listens for a message and potentially receives and retransmits it. The pointer `*payload` that points to a memory buffer is used to store the message which is sent or received. `n_tx` indicates how many times the node transmits the message.

`stop()`: Stops Gloria, i.e. stops initiating or participating in a flood.

`get_rx_cnt()`: Returns the number of messages received during the last Gloria run, i.e. the return value is 1 if a flood has been received.

`get_t_ref()`: Returns the timestamp (in local time) that corresponds to the reference point in time of the last received Gloria flood (see `RefTime`

[3] https://gitlab.ethz.ch/tec/public/flora/wiki/-/wikis

in Figure 2.10). This timestamp can be used to time synchronize the receiving nodes to the flood-initiating node.

The above interface is available on all nodes. For a typical transmission of a flood, all but one node start a listening flood using `start()` with `is_initiator=0`. A single node starts Gloria as an initiator (`is_initiator=1`) thus starts transmitting the message. All listening nodes potentially receive and retransmit the message until either the number of transmissions (`n_tx`) is reached or the participation in the flood is stopped via `stop()`. At the nodes that received the flood, the received message is then available in the `payload` message buffer.

2.5.1 Evaluation of the Gloria Implementation

In this section, we assess the performance of the Gloria flooding primitive implemented on the DPP2 SX1262 platform. The goal is to measure the reliability and the time synchronization accuracy of Gloria floods. For the measurements, we use the FlockLab 2 testbed infrastructure that we describe in detail in Chapter 3 and that makes use of the presented DPP2 SX1262 hardware platform. Because we are interested in precise timing measurements, we limit ourselves to GNSS time-synchronized observer nodes. This includes 15 nodes inside an office building and 4 rooftop nodes with a link distance of up to 2 km, see Figure 3.5.

We use node 2 to initiate 1000 floods. All other nodes are listening for the floods and potentially receive and retransmit them. Following each flood, all nodes that received or initiated the flood precisely trigger a GPIO pin at time instant t_{sync} with a predefined time offset relative to `RefTime` which is determined based on the initiated or received Gloria flood (see Figure 2.10).

We use messages with physical layer payload size of 20 bytes. For Gloria, we configure all nodes such that they send 3 transmissions within a flood (`n_tx` = 3) and we use a flood duration that allows up to 7 hops (`num_slots` = 7). The radios of all nodes are configured to send with a transmit power of +6 dBm. We use two different modulation settings: (1) a short-range modulation using FSK with 250 kbps and a bandwidth of 312 kHz and (2) a long-range modulation using LoRa with spreading factor SF7, a bandwidth 125 kHz, and the coding rate 4/5.

We measure the reliability using the flood reception ratio (FRR) on each node, i.e. we divide the number of successfully received floods on a node by the total number of initiated floods. The corresponding results are depicted

in Figure 2.11. In addition, we determine the time synchronization accuracy by determining for each flood the time deviation between the t_{sync} of each node that received the flood and t_{sync} of the initiator (node 2). The resulting distributions are visualized as a boxplot in Figure 2.12.

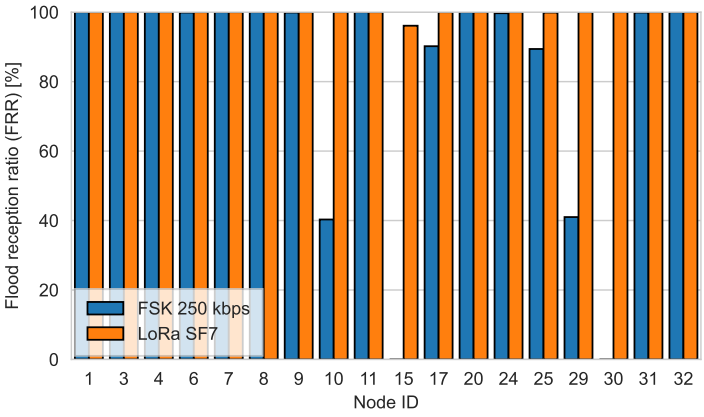


Figure 2.11: The flood reception ratio (FRR), i.e. the amount of received floods, on each node shows the reliability of flooding using Gloria. 1000 Gloria floods were initiated by node 2.

The FRR depends heavily on the network topology, i.e. the link characteristics between the nodes, and the chosen radio settings. As expected, for some nodes the FRR is lower when using the short-range modulation compared to using the long-range modulation. This is especially true for nodes, that are further away and only reachable via high path loss links. In addition, the FRR of nodes with only few connections, such as node 10, is worse compared to well connected nodes, such as node 4 for example.

The time synchronization accuracy measurements show a behavior that is consistent with the measurements from Section 2.4.3.2, i.e. the short-range FSK modulation shows better timing accuracy than the long-range LoRa modulation. The influence of the propagation time manifests itself by an increased median of the time deviation. This is clearly visible for node 15, that is located 2.3 km away from the initiator (node 2) and thus has a propagation time of about 7.7 µs. As reference, we also plot the hop distance in Figure 2.13. The hop distance corresponds to the number of hops between the initiator and the receiver of the flood. This value

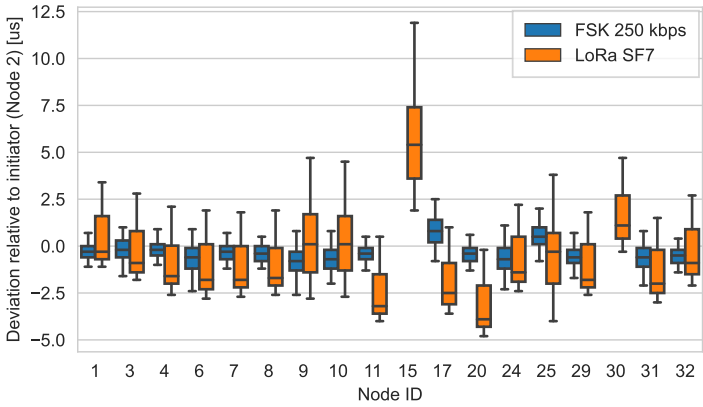


Figure 2.12: The deviation of the t_{sync} event of each node relative to the initiator (node 2) shows the time synchronization accuracy when using the Gloria flooding primitive. The box extends from the lower to the upper quartile values of the data and the line represents the median. The whiskers span the 95 % confidence interval.

can be determined based on the slot index that is included as part of the transmitted message and is incremented on every intermediate hop. The distribution span of the time deviation of the nodes increases with the number of hops as the synchronization error accumulates. Another contributing factor is that a message can take different paths with different propagation times.

The measurements contain outliers in the order of multiple 10s of microseconds (not visible in Figure 2.12 due to the 95 confidence interval). In the presented measurements there are a total of 5 outliers with a deviation larger than 20 µs. Similar to the measurements in Section 2.4.3.2, the outliers are very rare and seem to be related to low signal-to-noise ratio conditions at the receiving node. The impact of these outliers is limited since the floods still reach the destination and the small number of outliers usually does not affect typical use cases such as clock drift compensation because such schemes usually rely on multiple past synchronization points.

Overall, the implemented Gloria flooding primitive shows a good performance. Therefore it represents a useful basis for the reuse in higher-layer protocols implementations but also serves as a template for investigating further variations of the synchronous transmission flooding primitive in conjunction with different modulation schemes.

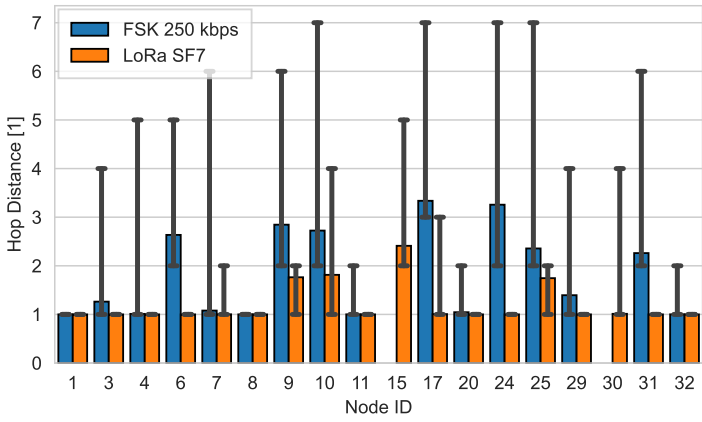


Figure 2.13: The hop distance indicates of how many segments the path between initiator (node 2) and receiver of the flood consisted. The hop distance is determined for each received flood and is based on the slot index that is transmitted as part of the message. The bars represent the mean of all considered rounds and the black vertical lines span entire range from the minimum to the maximum.

2.6 Summary

In this chapter, we presented the DPP2 SX1262 platform that supports long-range LoRa and short-range FSK communication. It represents a prototyping platform for investigating mixed-range LPWAN communication schemes but is also suited for the use in real deployments. The hardware design and the software framework provide support for low-power operation and modular design of more complex IoT systems based on the integrated the DPP abstraction. In contrast to state-of-the-art platforms, the proposed platform provides support for the precise timing of radio operations that is needed for the implementation of timing-critical networking protocols which for example make use of synchronous transmissions. For this, we performed an extensive analysis of the low-level timing overheads and integrated our findings into the software framework to allow the reuse. In a case study, we implement the Gloria flooding primitive that is an optimized variant of the Glossy flooding primitive and therefore is a representative example of a timing-critical networking protocol.

The author of this thesis designed in close collaboration with Reto

Da Forno and Jan Beutel the concrete hardware and software design of the proposed DPP2 SX1262 platform whereas the concept of the DPP architecture in general and the DPP2 platform specifically have been designed and presented earlier.

In the following chapter, we design a testbed infrastructure that allows to execute tests on multiple distributed instances of the DPP2 SX1262 platform and to trace relevant system states using a variety of interfaces.

3

Accurate Multi-Modal Testing with Long-Range Links

The design and development of networking protocols for wireless Internet of Things (IoT) networks is challenging due to the distributed state. Often, mathematical models and simulations are used. However, measurements on real hardware remain essential to reliably verify feasibility and collect accurate performance measurements. Testbeds in which a fixed infrastructure consisting of devices under test (DUTs) and a system for actuation and monitoring are a good option for efficient and high-quality testing and validation of networking protocols on real hardware and embedded in a realistic environment. Testbeds have the advantages over temporary solutions that the hardware and software setup can be reused, the infrastructure can be shared among multiple experimenters, and the fixed infrastructure facilitates repeatability significantly.

The requirements for testbeds have changed with the advancements of the technology for wireless IoT devices. Today's platforms, as the platform presented in Chapter 2, use higher clock frequencies, feature lower power consumption in low-power modes, and the radios support communication over larger distances and with higher transmit power. Consequently, there is demand for more temporal accuracy and higher dynamic range measurements. In addition, the increased communication range requires large link distances to perform realistic tests. Furthermore, debug and trace circuits, which are an integral part of modern microcontrollers, offer even higher-quality observability.

In this chapter, we present the FlockLab 2 testbed architecture. Com-

pared to existing testbed architectures for the IoT, it allows multi-modal high-dynamic range tracing, supports large link distances between testbed nodes, and enables remote use of the on-chip debug and trace circuitry built into modern IoT devices.

The presented testbed architecture serves as a basis for the non-intrusive distributed tracing system proposed in Chapter 4 and is used for the development and validation of the Long-Short-Range (LSR) protocol in Chapter 6.

3.1 Introduction

The ever-increasing complexity and care for detail that must be mastered in developing state-of-the-art distributed networked embedded applications requires modern and adequate tool support for experimentation. In scaling to large distributed applications, simulations can help but cannot replace experiments on real hardware. Simulation always implies simplifications, significant especially at the hardware level. The latest microcontrollers and radios used in wireless IoT applications feature numerous power modes that need to be accurately fine-tuned and orchestrated for efficiency. The interaction between peripherals and the system core needs to be well-understood and validated for reliable operation down to the instruction level. Timing needs to be controlled at application as well as driver level up to the speed of light, as recent work on network protocols incorporating the time-of-flight of radio signals has shown [LMT16].

The development of embedded software is commonly based on state-of-the-art debug and trace infrastructure integrated into the hardware of modern microcontroller architectures [Sto11]. This in-situ infrastructure is supported by a multitude of development tools that can be used on the user's desk and also remotely. Today, such tooling is limited to a single DUT, therefore severely limiting capabilities to develop and test algorithms and systems for distributed wireless IoT devices. It is exactly this distributed nature of many devices, coupled over variable wireless channels and directly influenced by the embedding environment, that is known to be a challenging task in designing, implementing and validating IoT and cyber-physical systems. Therefore, distributed testbeds with representative hardware deployed in a real environment are widely used. Such testbeds allow (1) the reuse of the testing infrastructure, (2) controlled and repeatable testing and validation, and (3) the comparison of different

implementations on a common platform (benchmarking). A number of testbeds exist that support a subset of the aspects mentioned above that are required for contemporary software development and evaluation for wireless IoT devices. An overview of existing testbeds and their capabilities as well as our remarks on 8+ years of testbed development and operation is provided in Section 3.2. However, none of the existing testbeds supports combined native in-situ debug and trace infrastructure, accurate timing measurements as well as the detailed assessment of power consumption over a large dynamic range.

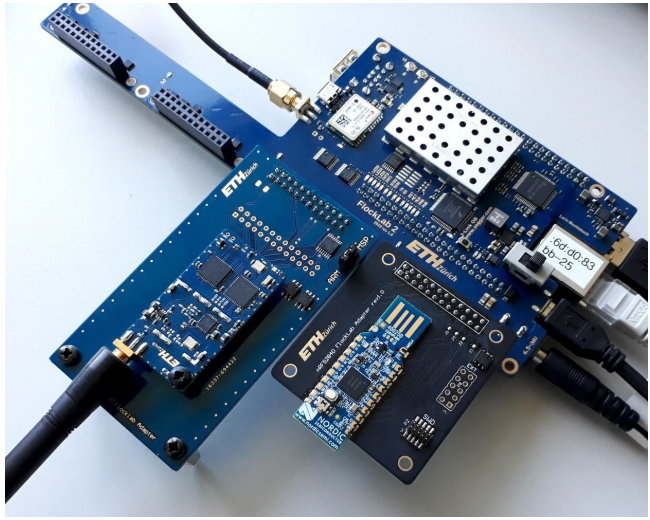


Figure 3.1: FlockLab 2 observer with 4 target slots.

In this chapter, we present a versatile testbed with capabilities addressing the aforementioned requirements. In jointly addressing challenges in power measurement, timing, functional correctness based on native, in-hardware debug and trace functionality integrated at testbed scale, this work takes the methodological aspect of developing IoT and cyber-physical systems to the next level. The integration of support for native hardware-based debug and real-time tracing into every observer node allows full testbed-wide access to the ARM real-time debug and trace collection infrastructure (CoreSight [Sto11, Arm13]) at the program execution level. Access is provided remotely in exactly the same manner as on a single developer's desk to all devices-under-test. This alleviates the need for

inflexible instrumentation of the software code run on a DUT as well as invasive run-stop debugging. In addition, this testbed integrates high fidelity power profiling at nanoampere resolution, dynamic control of the power supply and highest precision tracing and actuation of a set of DUTs based on Global Navigation Satellite System (GNSS) time synchronization. The testbed features a well-defined and open interface for test creation and test result fetching. A Python-based library and command line tool provides support for automated test management and visualization. The hardware design and software source code of FlockLab 2 is available as open source[1] [DFTW+21, DFMST21].

In summary, this chapter contains the following contributions:

- We propose a testbed architecture that combines state-of-the-art debug and trace capabilities with accurate high-dynamic range measurements and actuation.
- We provide a characterization of the system implementation.
- We demonstrate the capabilities of FlockLab 2 in a case study.

Section 3.2 gives an overview of the testbed landscape. In Section 3.3, we discuss the design of FlockLab 2 and characterize its implementation. In Section 3.4, we demonstrate the capabilities of FlockLab 2. In Section 3.5, we summarize the chapter.

3.2 Past Experience and Related Work

The design of FlockLab 2 is heavily influenced by 8+ years of experience in developing and operating the FlockLab 1 testbed [LFZ+13b]. This testbed was based on the very successful target-observer model [LWF+12, LFZ+13b] with multi-modal capabilities to monitor and influence devices-under-test at very high precision and fine-grained resolution. The FlockLab 1 testbed has been operated publicly since 2012. It ran over 70'000 tests by more than 370 users from more than 130 institutions in 30 countries. In addition, the testbed has been used by students in hands-on courses and many student projects.

Over time, a number of extensions based on the original concept of FlockLab 1 have been implemented [LMD+15, LMT16] calling for a revisit of the original concept with improved performance figures. Existing

[1] https://www.flocklab.ethz.ch

competitor testbeds each provide interesting features. However, none of them combine all three capabilities: (1) in-situ debug and trace, (2) high-dynamic range power profiling, and (3) accurate timing. In the following, we give an overview of the current testbed landscape.

TWIST [HKWW06] and Indriya2 [AWCM19] are both based on USB interconnects. Therefore they do not provide elaborate debug and trace features or accurate observations of hardware behavior like precise timing or power. On TWIST the power supply can be controlled by turning the USB interface to the targets on or off. Furthermore, a hierarchical back-channel using USB and Ethernet allows scalability.

The D-Cube [SBWR17] testbed focuses on benchmarking wireless protocols in pre-defined scenarios with a technique to embed test parameters directly in the software for the DUT, control radio frequency (RF) interference, and the automated publication of test metrics. In addition to serial logging, it supports setting and tracing GPIO pins and allows power consumption measurements. However, it does not support the use of debug and trace capabilities of modern microcontrollers.

FIT IoT-Lab [ABF+15] supports a wide range of sensor nodes (MSP430 to ARM Cortex-M8) at many different locations. Basic debug and trace based on JTAG and monitoring of power consumption is supported. Furthermore, the testbed supports injecting and sniffing radio packets and monitoring on a single frequency RF channel. To the best of our knowledge, it does not support accurate timing for control and measurements.

Shepherd [GCZ19] focuses on recording and replaying energy harvesting power traces for research in batteryless IoT devices. The architecture supports basic debugging and GPIO tracing. Power measurements are supported up to 50 mA which is limiting for modern long-range radios with high transmit power. Currently, there is no publicly available instance of the Shepherd testbed.

3.3 A Real-Time Tracing Architecture

An IoT testbed needs to support multi-modal distributed interaction and tracing. We identify the following key requirements for a state-of-the-art testbed for wireless IoT devices:

- Support for native debug and trace infrastructure.

- Accurate and high-dynamic range power measurements (sub-µA sleep current up to radio TX current of 170 mA).
- High-precision timing (sub-µs accuracy) across the distributed testbed.

The FlockLab 2 testbed architecture consists of a *testbed server* hosting data services and the web interface, a set of distributed *observers* carrying the instrumentation and providing connectivity and the DUTs which we call the *targets*. In FlockLab 2, multiple targets, typically manifested by different sensor node architectures, are supported on each observer system. Each target is connected to the observer hardware using a multiplexer crossbar allowing a user to select a distinct target hardware architecture (see Figure 3.2). The independent and stateful observer, which stores tracing data locally, allows for a strong coupling between observer and target (see Figure 3.3). This enables highest accuracy and throughput of the DUT instrumentation especially when comparing to direct out-of band back-channels of early testbed architectures [WASW05].

The basic services of FlockLab 1 are retained in the new architecture: actuation and tracing of serial port and GPIO pins, target programming, power tracing and adjustable target supply voltage. The new system supports a native in-situ debug and trace service and generally a higher fidelity of the aforementioned basic services as well as support for an extended testbed layout covering also wide-area distances. The instrumentation for measuring the power consumption has been significantly improved: nA current measurements resolution, peak power up to 500 mA, a high sampling rate, and electrical isolation of targets to not perturb low-power measurements.

3.3.1 Observer Instrumentation Platform

Each observer consists of a Linux host system, a main board and several target adapter boards hosting up to four different targets.

A standard Linux single-board computer (SBC), a BeagleBone Green, is used as host platform for the decentralized stateful observers. All tracing data is recorded on the observer in order to alleviate the inherent bottleneck to the testbed server. The BeagleBone Green SBC includes a single core ARM Cortex-A8 processor and two single-cycle Programmable Real-Time Unit (PRU) co-processors for low latency tracing. It further features an Ethernet interface with integrated hardware support for network-based time synchronization (NTP/PTP), generic IO extensions and local Flash

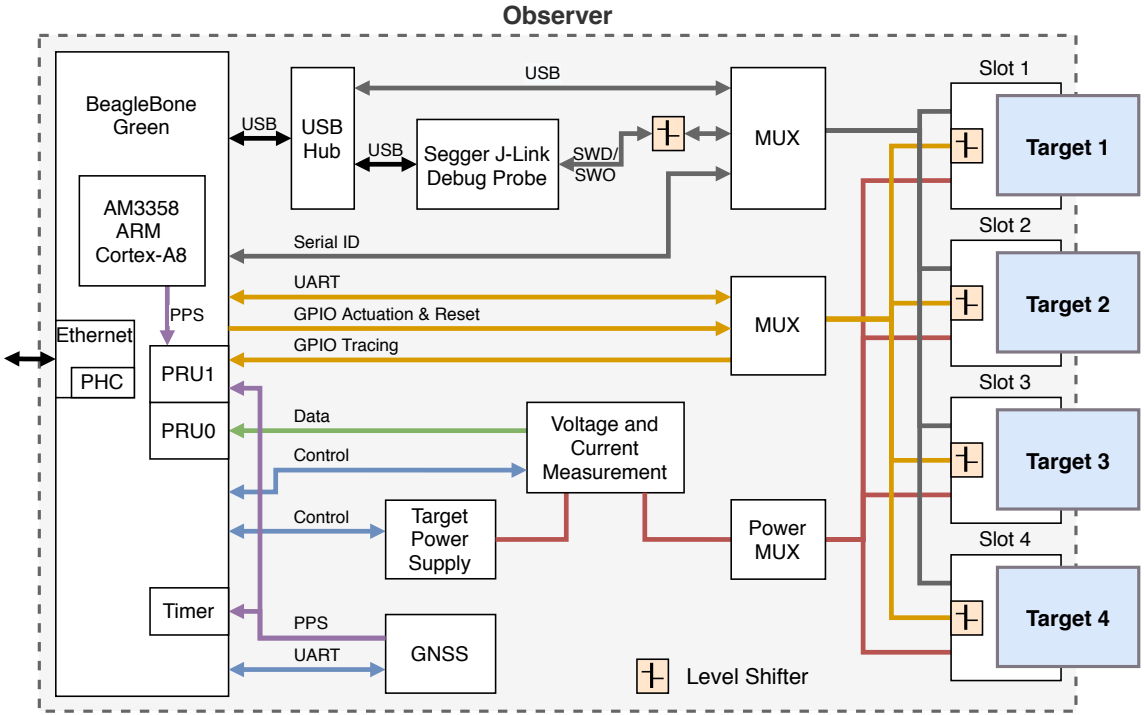


Figure 3.2: FlockLab 2 observer architecture.

memory.

3.3.1.1 Testbed Services

The proposed testbed architecture enables the following services for actuation and tracing.

Native Hardware-Based Debugging and Tracing: The key feature on the FlockLab 2 observer is an integrated Segger J-Link OB debug probe. This gives native access to state-of-the-art ARM Cortex-M CoreSight debug and trace facilities which are built into modern systems on a chip (SoCs) [Arm13]. This allows to utilize simple halting debug mode (where architectural state can be observed), single step execution, breakpoint units and Performance Monitoring Units (PMUs). CoreSight further provides an Embedded Cross Trigger mechanism to synchronize or distribute debug requests and profiling information across the SoC. Embedded Trace Macrocells (ETMs) or Program Trace Macrocells (PTMs) allow to trace program execution at runtime and without instrumentation in the code that (i) alters program behavior and (ii) needs to be adapted for every single analysis step. The trace macrocells can either be captured using an on-chip trace buffer or accessed via the generic Serial Wire Debug (SWD) connection implemented on the J-Link debug probe acting as off-chip Trace Port Analyzer (TPA) (see Figure 3.3). Dedicated synchronization points and global 64-bit timestamps across the whole SoC architecture can be enabled in the tracing architecture to gain accurate temporal context of an application and its interaction with the underlying hardware at runtime. The debug and tracing architecture is implementation specific and can be found in the respective microprocessor documentation.

The use of SWD on FlockLab 2 with the SWDCLK and SWDIO signals as well as a dedicated Serial Wire Output (SWO) trace port is a tradeoff between bandwidth and pin count. By using data buffer exchange capabilities of the debug probe, e.g. with Segger Real Time Transfer (RTT) software technology that can be easily integrated with user code, high-speed and little impact data transfer from the target to the observer is supported. For example, this allows more efficient `printf()`-style logging on the target.

Besides the native hardware support for debug and trace, the main advantage lies in the ability to connect to all standard developer tools allowing interactive debug sessions on the testbed directly from a developer's integrated development environment (IDE) or use ready made tooling for

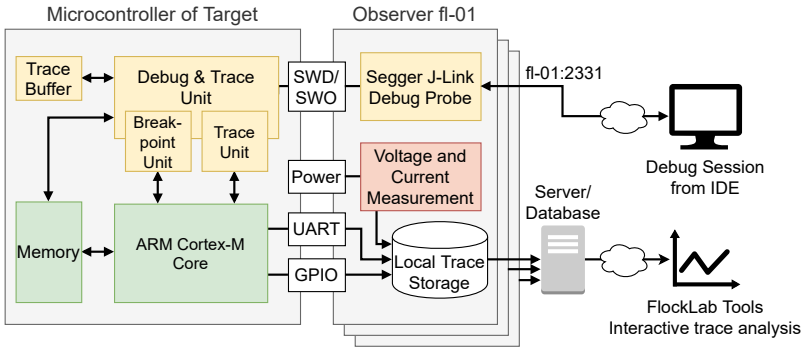


Figure 3.3: Tracing instrumentation based on the in-situ debug and trace unit and external capabilities.

automating tasks. A lot of time in sensor networks and IoT research has been spent in developing custom tooling and a host of scripts incompatible with industry standard debugging and profiling tools.

Power Tracing: Highly accurate and high-dynamic range power profiling is based on the RocketLogger embedded measurement device [SGL+17]. It combines two measurement principles using seamless auto-ranging: a shunt ammeter (high current range), as well as a feedback ammeter (low current range) provide the necessary precision and range to measure sleep currents (sub-µA and peak power consumption (100's of mA) of modern radios). The measurement circuit allows to measure the current through as well as the voltage at the target. A careful electrical isolation of all target IO lines allows to accurately measure on the order of nAs for the lowest power modes. Measurements are performed by precision analog-to-digital converters (ADCs) and periodically transferred to the single-board computer's storage via PRU0. To compensate for hardware and manufacturing variations, the voltage and current measurement circuit on each observer is calibrated before first use.

Serial Logging and Forwarding: Serial port communication via UART or USB is supported on each observer. This allows logging of simple `printf()` based console output and supports interactive communication with the target during tests by forwarding the serial port via TCP to the testbed user. To achieve high performance and accurate timing, logging is implemented in C and events are timestamped using the GNSS or Precision Time Protocol (PTP) disciplined system clock.

Logic Actuation and Tracing: On each observer, 5 target GPIO pins can be captured and 2 target GPIO pins can be actuated. In FlockLab 2, logic tracing is implemented on the programmable real-time unit PRU1, which allows to acquire highly accurate timing trace data. Logic actuation is implemented by a Linux kernel module and the actuation events are logged by the PRU based logic tracing as well.

3.3.1.2 Testbed-Wide Time Synchronization

Accurate timing across all signals for tracing and actuation on a single observer platform as well as across the whole testbed is one of the most important success factors of a distributed IoT testbed. With recent advances in higher system clock rates and ever more timing-critical behavior in advanced communication schemes [LMT16] the requirements for accurate timing is in the sub-microsecond scale. Since this testbed is designated to support long-range communication where observers might be distributed over kilometers, time synchronization needs to work independent of other observers and their location.

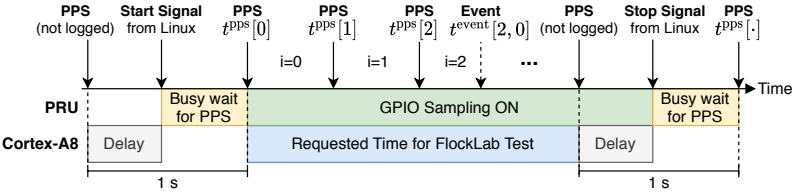


Figure 3.4: Logging of PPS signal alongside the target signals for accurate time synchronization.

With the FlockLab 2 architecture, the local Linux system time referenced to UTC and disciplined by GNSS or PTP serves as time reference for a majority of the testbed services. The integrated GNSS receiver (u-blox M8) allows to generate an accurate pulse-per-second (PPS) signal which is tracked in dedicated hardware timers on the single-board computer. This time synchronization source based on GNSS provides an accuracy of ~50 ns [LMT16]. Observers at locations with limited GNSS signal reception can use PTP as a fallback solution to discipline their Linux system clock to another computer (PTP master) equipped with a GNSS receiver. A prerequisite for this is a network infrastructure which fulfills the requirements of PTP. Using hardware assisted timestamping at the PHY

and MAC layer of the single-board computer's Ethernet interface allows to achieve synchronization accuracy in the order of ~1 µs for PTP [LBMB16]. Statistics data from the `chrony` software from 9 PTP-synchronized observers of the publicly accessible instance of FlockLab 2 (see Section 3.3.3) collected over 24 hours indicate that the system time deviation is below 1 µs for 98.9% of the time, while the maximum logged deviation is 14 µs.

For accurate timing of the logic tracing, the GNSS-based PPS pulse is logged alongside the target signals (see Figure 3.4). A linear correction factor is calculated for each epoch i and applied to timestamp of the logic tracing event (numbered by k) once a test has completed.

$$t^{\text{event}}_{\text{Global}}[i,k] = t^{\text{event}}_{\text{PRU}}[i,k] \cdot \frac{t^{\text{pps}}_{\text{Global}}[i+1] - t^{\text{pps}}_{\text{Global}}[i]}{t^{\text{pps}}_{\text{PRU}}[i+1] - t^{\text{pps}}_{\text{PRU}}[i]}$$

In case of an PTP time-synchronized observer, the PPS signal required for accurate logic tracing is based on the Linux system time and generated by a kernel module.

For the debug probe, incoming messages from the SWO trace port can be timestamped using the system time as well. In addition, the debug and trace sub-system can be configured to export local timestamps via SWO. These can be converted to system time by applying a piece-wise linear regression similar to the correction factor for logic tracing described earlier, also see Chapter 4.

3.3.1.3 Infrastructure for Targets

The presented observer architecture provides the following infrastructure for the targets.

Multiple Target Slots: Each observer features 4 slots and a multiplexer crossbar to connect the targets to the observer hardware allowing a user to switch the actively used target without physical intervention at the observer (see Figure 3.2). This allows to run tests on different target architectures physically collocated, e.g. to compare different radio architectures side-by-side. In addition, by installing multiple instances of the same target architecture equipped with different antennas (e.g. low- and high-gain), it is possible to perform tests with different antenna setups simply by selecting different targets for the individual tests.

Target Adapter: The target adapter is mainly a hardware adapter bridging form-factor and pinout. It may contain configuration options (e.g. jumpers)

and extra debug pins depending on the target platform. Additionally, it contains a serial ID chip for automated identification of every target connected to an observer.

Target Power Generation and Reset: The target supply voltage is generated using a low-dropout (LDO) regulator controlled by a digital-to-analog converter (DAC) based reference voltage. This allows to dynamically control the target supply voltage. The target can be reset either by controlling the reset pin or by a full power-on-reset (POR). These power and reset capabilities enable to test under different operating conditions and under most realistic conditions.

Target Programming: Programming of the microcontroller of the targets is performed either using a bootloader (BSL) or native SWD for ARM based devices.

3.3.1.4 FlockLab 2 Observer Key Characteristics

Based on the publicly accessible FlockLab 2 testbed described in Section 3.3.3, we provide a characterization of the FlockLab 2 observer in Table 3.1.

3.3.2 Testbed Management and User Interface

Testbed Infrastructure: The testbed is orchestrated by a server which executes the scheduled tests, provides a MySQL database, provides storage space for test results, hosts the web interface and exposes an application programming interface (API) for automated test scheduling and fetching. The database stores test scheduling information and configuration as well as the current state of the hardware infrastructure (e.g. which target is connected to which observer).

API and Visualization: FlockLab 2 focuses on fully autonomous test execution but also supports live interactions. Tests are configured and scheduled using a single XML file. This test configuration file allows to (1) select the target platform and the testbed nodes, (2) enable and configure actuation and tracing services, and (3) include one or more program images which will then be flashed to the targets. Real-time interaction during test execution is supported via the serial communication service (read/write) and via a remote debug session using the integrated Segger J-Link debug

Target Power Supply	
Voltage range	1.1 - 3.6 V
Voltage resolution	13.5 mV
Max. current	500 mA
Logic Actuation	
Timing accuracy	<100 µs (typ.)
Power Tracing (see [SGL+17] for details)	
Max resolution / sampling rate	15.625 µs / 64 kHz
Voltage accuracy	0.37% + 4 mV
Current accuracy (low range 0 - 2 mA)	0.01% + 60 nA
Current accuracy (high range 2 - 500 mA)	0.02% + 48 µA
Logic Tracing	
Max resolution / sampling rate	0.1 µs / 10 MHz
Max burst event rate (≤ 2000 edges)	10 MHz
Max continuous event rate (typ.)	900 kHz
Timing Accuracy (GNSS)	<0.25 µs (typ.)
Serial Tracing	
Max continuous throughput	460 kbaud
Timing accuracy	<10 ms (typ.)

Table 3.1: Characterization of the FlockLab 2 observer.

probe which allows to set breakpoints, halt the execution, read processor state and to retrieve data via SWO.

The results are stored on a server and can be downloaded as an archive file. Test management (creating and stopping tests, retrieving status information, downloading results) as well as an intuitive visualization of results is supported via web interface or the `flocklab-tools` command-line tool executed on the user's computer.

3.3.3 Publicly Available Testbed

We implemented an instance of the FlockLab 2 testbed with currently 30 active observers and offer it as a public service (Figure 3.1). As depicted in Figure 3.5, the network topology consists of 26 observers distributed on a floor of an office building, 1 observer is installed on the rooftop of the same office building, and 3 observers are installed at remote long-distance rooftop locations. The testbed can be publicly accessed via the website[2]

[2] https://www.flocklab.ethz.ch

where we also published documentation, examples, the hardware design, and software source code.

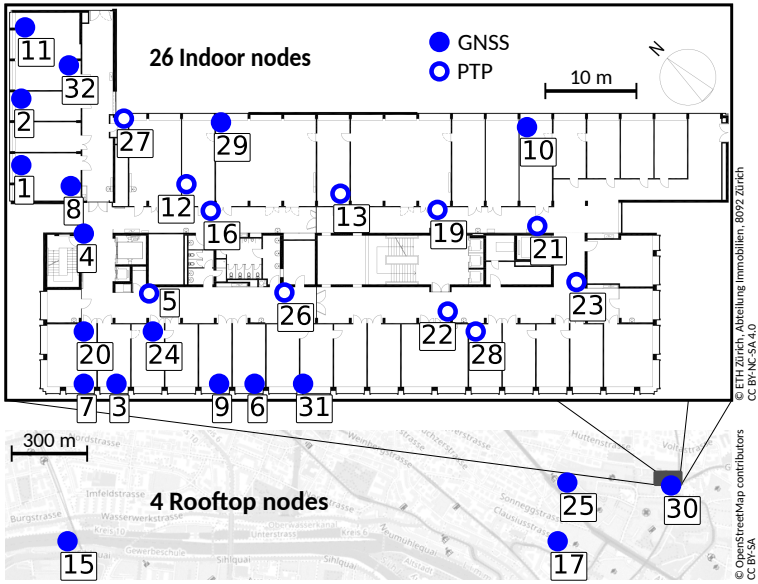


Figure 3.5: Network topology of the publicly accessible instance of FlockLab 2 at ETH Zurich. Depending on the location, the observers are time-synchronized based on GNSS or PTP.

Currently, the four target platforms listed in Table 3.2 are available. Additional target platforms can easily be added thanks to the generic target interface.

Target	MCU	Arch.	Radio
Tmote Sky / TelosB	MSP430F1611	MSP430	CC2420, 802.15.4, 2.4 GHz
DPP2 CC430 [BTDF+19]	CC430F5147	MSP430	CC430 SoC, CC1101-based, 868 MHz
DPP2 SX1262 (Chapter 2)	STM32L433	ARM M4	SX1262, LoRa/FSK, 868 MHz
nRF52840 Dongle	nRF52840	ARM M4	nRF52 SoC, 802.15.4/BLE, 2.4 GHz

Table 3.2: Target platforms supported on the publicly accessible instance of FlockLab 2.

Since the installation of the first FlockLab 2 observers in March 2020, more than 7800 tests have been executed on the testbed (as of March 2022).

Currently, there are 38 active users from 8 different countries who have logged in during the last 12 months.

3.4 Using FlockLab 2 in Practice

In order to demonstrate the capabilities of FlockLab 2 regarding features and performance, we discuss an example using Gloria network flooding based on synchronous transmissions, also see Section 2.5. Building and scaling-up a communication protocol based on synchronous transmissions requires a very careful arbitration of all radio activities and extensive debugging as transmissions and corresponding interrupts need to be correctly aligned. Gloria is therefore a suitable example to showcase the capabilities of FlockLab 2. In this example, we use the DPP2 SX1262 target which is based on the DPP2 SX1262 platform presented in Chapter 2 and consists of a STM32L433 Cortex-M4 microcontroller and a Semtech SX1262 radio.

3.4.1 Simple Synchronous Transmission Protocol

Gloria is an optimized variant of the Glossy [FZTS11] multi-hop network flooding protocol which we ported to the DPP2 SX1262 platform, also see Section 2.5. As depicted in Figure 2.10, all nodes synchronously retransmit a received message a pre-defined number of times (3 times in our example) in subsequent transmission slots. Contrary to Glossy, in Gloria nodes listen to a message only once. For the setting used in this example, the relevant values are `FloodOverhead` = 1.783 ms and SlotTime = 4.548 ms (see Figure 2.10). Both values are fixed for each specific radio configuration (in this case FSK 250 kbps) and have been determined using the datasheet and measurements.

3.4.2 Testing Workflow

Creating a Test: First, the software for the target platform is compiled using the standard toolchain/IDE. Then, an XML test configuration file is created containing the nodes, images and platform to be used, test duration, actuation, tracing and debugging services configuration. A minimalist example is shown in Listing 3.1. This is then uploaded to the FlockLab 2

```
<testConf xmlns="http://www.flocklab.ethz.ch">
   <generalConf>
      <name>FlockLab XML template</name>
      <schedule><duration>60</duration></schedule>
   </generalConf>
   <targetConf>
      <obsIds>2 4 6 7 9</obsIds>
      <voltage>3.3</voltage>
      <embeddedImageId>Image_1</embeddedImageId>
   </targetConf>
   <serialConf>
      <obsIds>2 4 6 7 9</obsIds>
      <baudrate>115200</baudrate>
   </serialConf>
   <powerProfilingConf>
      <obsIds>2 4</obsIds>
      <samplingRate>1000</samplingRate>
   </powerProfilingConf>
</testConf>
```

Listing 3.1: FlockLab 2 test configuration example.

server using the web interface or the `flocklab-tools`. On the server, the test is then scheduled. The server initiates the start of the test at the time specified, distributes the target images and configures all testbed services.

Interaction During Test Execution: Progress is monitored on the web interface where information on configuration and status is available. If the serial forwarding service is used, it is possible to connect to an individual observer for the duration of the test using a TCP connection, e.g. by using `netcat`. Likewise an interactive debug session can be opened from the IDE on the user's computer to a GNU Debugger (GDB) server running on the observer.

Analyzing Test Results: After the test completes, the server fetches all results from the observers and combines them into a single test result archive file that can be used for custom post-processing or can be visualized using the `flocklab-tools` (an example is depicted in Figure 3.7).

3.4.3 Debugging and Analysis of the Protocol

Native Hardware-Based Debugging at Testbed Scale: In this example, 8 nodes are performing a Gloria network flood. To validate the correct

protocol implementation, we use the debugger functionality which allows to extract internal variables. Concretely, we want to verify the correct calculation of the time of the next transmission (`TxMarker` in Figure 2.10). In Figure 3.6, for each node, radio activity is shown in the first row (orange bars), the radio interrupts are shown in the second row (black bars). For node 4, the power trace (black line) is shown as well. A breakpoint has been set on the first `TxDone` interrupt of node 9 that can be inspected using a remote debug session to the target (see Section 3.3.1.1). Since the breakpoint halts this specific microprocessor, the captured traces show no more GPIO events after the node reached the breakpoint. The variables inspected at the breakpoint show e.g. `slot_index` = 2; `message_size` = 30 as expected. The variables `reconstructed_marker` and `current_tx_marker` correspond to `RefTime` and `TxMarker` in Figure 2.10, respectively. The extracted time difference value of 10.879 ms conforms with the expected value for `FloodOverhead` + 2 · `SlotTime`. This confirms the correct calculation of `TxMarker` in the synchronous Gloria flood.

Timing Validation Using Logic Tracing: In synchronous protocols, transmissions and corresponding interrupts need to be correctly aligned. This is traditionally done by instrumenting code and tracing GPIO pins with the logic tracing service (see Section 3.3.1.1). In Figure 3.6, radio activity with two interrupts (`SyncWordValid` and `RxDone`) corresponds to a message reception and radio activity with a single interrupt (`TxDone`) corresponds to a transmission. Retransmissions are scheduled based on the timing of received messages. For this, the `SyncWordValid` timestamp is used to calculate individual start times on each node. For this, the exact time offset between the start of the transmission and the `SyncWordValid` interrupt needs to be calibrated. In the example in Figure 3.7, this offset is not set correctly and consequently synchronous transmissions are not aligned. Using logic tracing, this malfunction can be detected (green lower bars) and the correct value can be determined.

Optimizing for Low Power Consumption: To maximize the lifetime of a battery-powered cyber-physical system, careful optimization and orchestration of the low-power modes is required. In this example, we validate the low-power behavior of the Gloria flood implementation by using the power tracing service together with the logic tracing capabilities, see Section 3.3.1.1.

In a simple implementation communication is executed in a fixed-length active window (see Figure 3.8). In the optimized case a node will transit to low-power sleep mode immediately after completing a required

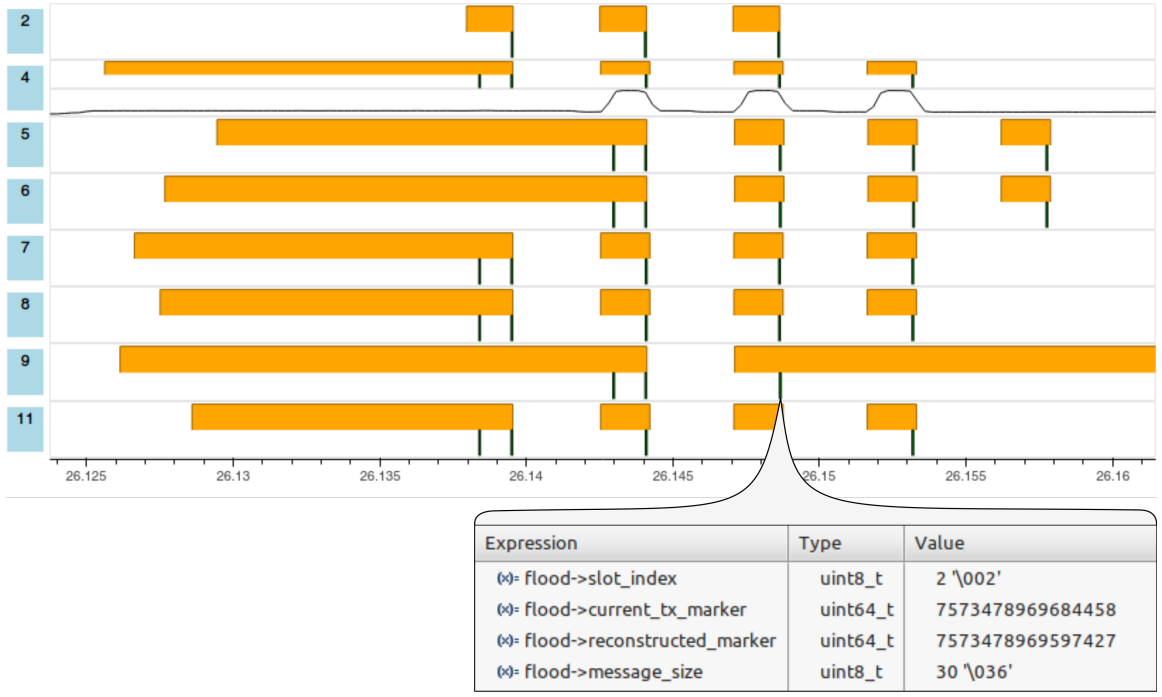


Figure 3.6: With the debug service, a breakpoint is set on the first `TxDone` interrupt on node 9. This allows to extract the values of the internal variables at that point in time.

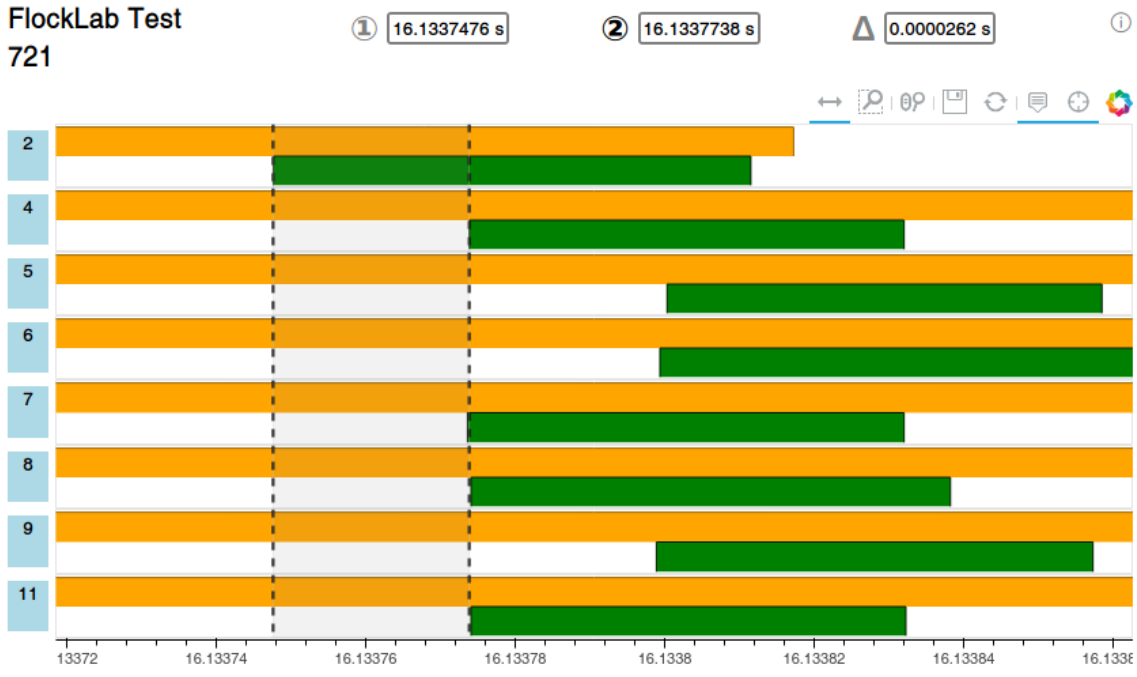


Figure 3.7: The transmissions (`TxDone` interrupts; green bars) of different nodes are not aligned because an offset timing parameter is not set correctly. The logic tracing service allows to detect and correct this erroneous behavior at interrupt-level granularity.

action, e.g. sending and receiving data. The difference between fixed-length and dynamic active window sizes can be seen in Figure 3.8 together with the respective radio interrupts.

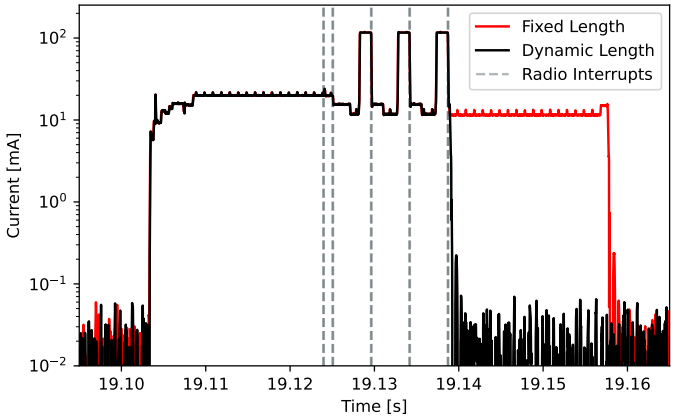


Figure 3.8: High-dynamic range power tracing is used to validate and optimize low-power behavior.

3.5 Summary

In this chapter, we presented the novel FlockLab 2 testbed architecture. Compared to existing testbeds, we integrated higher quality tracing in order to ensure that state-of-the-art and therefore higher-performance IoT platforms can be adequately traced. The integration of a current measuring device with range switching enables accurate and high-dynamic power measurements covering the entire range of the system's power consumption, from the highly optimized low-power operation to the high load peaks that can be caused by modern radios in transmit mode. In contrast to existing testbeds, the FlockLab 2 testbed has been specifically designed to provide high-quality actuation and tracing when the testbed nodes are placed far away from each other. This is an important capability for testing modern long-range platforms in a realistic environment. The on-chip debug and trace support which is widely used in industry and allows

non-intrusive low-level inspection of the program execution on modern microprocessors has been made accessible by equipping each observer with a dedicated debug probe. By using the established interfaces, existing debugging software tools can be remotely connected for interactive testbed-wide debugging and tracing.

Based on the publicly available instance of the testbed and with three usage examples, we highlighted how the set of services of the FlockLab 2 testbed architecture allow high-quality, accurate, and high-dynamic range multi-modal actuation and tracing. The hardware and software of the publicly accessible FlockLab 2 testbed implementation are available as open source[3] [DFTW+21, DFMST21].

The following chapter presents how the on-chip debug and trace hardware of the presented testbed architecture can be exploited to perform non-intrusive tracing and obtain time-synchronized tracing data from all distributed testbed nodes simultaneously without requiring online interactions.

[3] https://www.flocklab.ethz.ch

4

Non-Intrusive Distributed Tracing at Testbed Scale

To verify the correct behavior of wireless Internet of Things (IoT) networks, it is important to trace the distributed state during development and especially before deployment of systems with as little impact on the system as possible. Non-intrusive and instrumentation-free tracing is especially important for timing-critical software, e.g. synchronous transmission based wireless communication schemes. Modern platforms for wireless IoT networks, such as the one presented in Chapter 2, often include on-chip debug and trace hardware which enable the non-intrusive tracing. However, support for debugging and tracing is typically limited to a single device and usually requires online interaction. Many testbeds for wireless IoT devices provide no or only limited support for non-intrusive tracing that does not require code instrumentation. In Chapter 3, we presented the FlockLab 2 testbed architecture which allows remote use of the debug and trace features on every node. Nevertheless, distributed tracing of a large number of devices in the testbed is challenging because tracing requires online interaction with each device and the data obtained from different devices is not time-synchronized.

In this chapter, we use the FlockLab 2 architecture presented in Chapter 3 and make the following contributions: First, we present a method to perform non-intrusive tracing without interaction which facilitates tracing a large number of nodes simultaneously and improves repeatability significantly. In addition, we present a time synchronization scheme that makes it possible to time synchronize and therefore align the traces

obtained from distributed nodes on a common time axis.

We make use of the presented non-intrusive tracing infrastructure for the development of the Long-Short-Range (LSR) protocol in Chapter 6 which includes timing-critical communication using synchronous transmissions.

4.1 Introduction

The development and validation of networking protocol implementations for distributed IoT devices is challenging. Each IoT device contains its own local state which is influenced by interactions with its environment through peripheral interfaces such as the wireless radio and sensors/actuators attached to the device. In a distributed context, these state changes are additionally based on interactions between the individual nodes via a wireless communication network.

In many cases, the design of software for IoT nodes is first supported by a simulator or a handful of devices on the developer's desk. Simulations are valuable as a first step but cannot replace testing on actual hardware since many details of real-world systems have a significant impact on the system operation. Tests with devices on the developer's desk often simplify important aspects as only a limited number of devices under test (DUTs) is feasible and as they are co-located and hence offer limited spacial diversity. These simplifications limit the validation of the concurrent aspects of distributed sensing systems under realistic conditions. For this reason, testbeds for wireless sensor networks are used to debug and validate hardware and software of wireless IoT nodes before the deployment in a targeted environment.

A basic structure for testbeds that uses a separate set of stateful observer nodes implemented as an infrastructure service on top of the actual devices under test (DUTs), i.e., the network of IoT devices or sensor nodes, has proven to be useful [SBWR17, ABF+15, GCZ19, LFZ+13b]. The FlockLab 2 testbed presented in Chapter 3 is a representative example of such a testbed architecture. A high-level overview is depicted in Figure 4.1. The distributed *targets* are resource constrained devices that usually consist of a microcontroller for collecting and processing information and a low-power radio which is used to communicate with other targets over wireless channels. Each target is supplied, controlled and monitored by a dedicated device which we refer to as *observer*. Nowadays, supporting

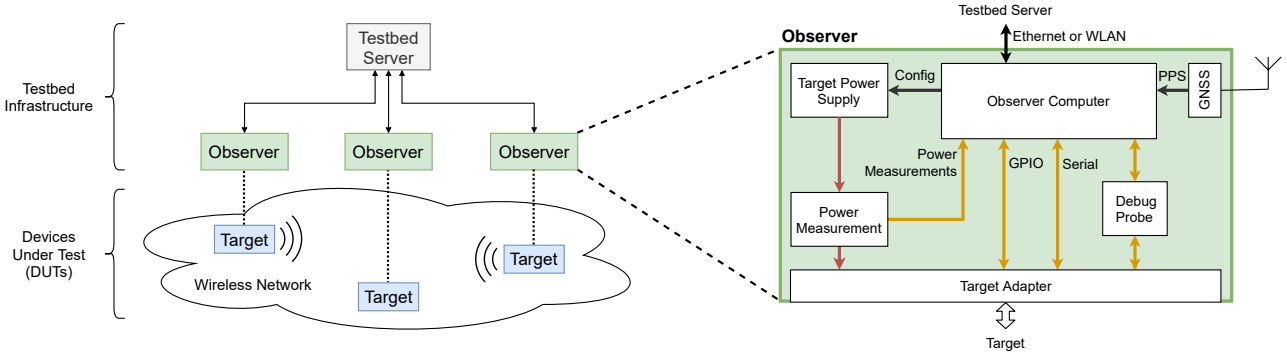


Figure 4.1: A common testbed architecture includes observers and a testbed server which represent the testbed infrastructure. This infrastructure allows to control and observe a set of targets which form a wireless network. The exemplary observer, based on the FlockLab 2 testbed presented in Chapter 3, supports multi-modal actuation and tracing.

multiple different modes of actuation and observation in a single testbed is common best-practice. The observers of the FlockLab 2 testbed provide: support for programming the targets, a configurable power supply, capabilities for logic actuation (writing GPIO signals), support for monitoring of power consumption, logic tracing (reading GPIO signals) as well as interactions via a serial or debug port. Observer nodes are connected to a central testbed server using an out-of-band channel, for example Ethernet. During the execution of a test, the stateful observers collect measurements and traces locally in order to avoid limitations by the shared network infrastructure. At the end of a test, the testbed server collects and processes all results which are then made available to the developer for analysis.

In order to monitor the state of software running on distributed targets, a variety of methods have been developed and applied, see for example [LT17, SK13, THBR11, TSBE15]. The four most important classes of methods are:

- **Serial Interface**: Methods that make use of a serial interface permit to use `printf`-style logging to capture a diverse set of system states, see for example [HKWW06]. Such methods are easy to use since human-readable text is logged to inform about the state of the system. However, the timing accuracy is usually low due to slow UART interfaces and the non-deterministic behavior of further components in the transmission chain. Furthermore, such methods require code instrumentation and are intrusive, thereby influencing the tight timing constraint of wireless protocols.
- **Buffer transfer**: Methods that use buffer transfers, e.g., via CoreSight [Arm13], are a faster variant of `printf`-style logging, see for example [ABS+19] where Segger RTT is used. Similar to approaches using a serial interface, these methods are highly versatile but require code instrumentation and are intrusive. The timing accuracy is limited by delays in the debug channels.
- **Logic tracing** : Logging GPIO pin signals with a logic analyzer or similar hardware provides high timing accuracy and is minimally intrusive, see for example [SBWR17] or Chapter 3. At the same time, such methods are limited to very few states (1 bit per GPIO pin). Of course, schemes to serialize more state information and send it over one or multiple GPIO lines can be applied. However, this would increase the code instrumentation and the intrusiveness of such methods significantly.

- **Hardware assisted debugging and tracing**: Tracing with dedicated debug and trace hardware (including built-in subsystems) allows for non-intrusive tracing and does not require code instrumentation, see for example [THBR11, LML20]. However, this is usually only supported for a single device under test. Without sub-millisecond time synchronization, such methods are not suited for the tracing or debugging of distributed devices.

All these methods help to validate and confirm the correct behavior of the software running on distributed targets in a testbed. However, to the best of our knowledge, no testbed architecture provides a truly non-intrusive distributed tracing method without the need to instrument the code. However, this would be necessary in order to validate the software in the last stage before deployment [WLT13]. As a consequence, the software running on a testbed is in almost all cases different from the software running in an actual deployment, rendering it almost impossible to draw conclusions from testbed tests which are also valid for a deployed system.

Adding or adapting code in order to trace the system state leads to changes in the functional behavior of the system and most importantly, it potentially changes the timing behavior. Communication protocol software for distributed wireless sensing devices is in many cases highly delay sensitive and even small changes in the code can significantly impact the overall system operation in terms of correctness, throughput, reaction time, reliability and energy consumption. Furthermore, additional instrumentation for tracing potentially influences the timing of the traced events. As a consequence, the correct ordering of events that are traced in the distributed system becomes even more challenging.

Based on our own needs to test timing-critical networking protocols in setups which are as close to the real-world deployment as possible, we worked out the following requirements which should be met by a testbed when debugging and validating target software before deploying it:

- State and state transitions of the targets need to be observable network-wide. State transitions on different nodes should be chronologically ordered with an accuracy of at least the duration of a single radio message transmission, typically in the order of one millisecond or more.
- The supported event rate should be sufficiently high in order to detect state changes which only last for a short period of time, e.g., for a few tens of processor clock cycles.

- The method to collect observations should be non-intrusive, i.e., the program execution should not be influenced by the collection of observations. This is important for time-critical parts of the program code such as radio operations in wireless networking protocols.
- No code instrumentation should be required as code for verification is typically not included in the software for a deployed system in order to reduce power consumption and increase operational stability.

To fulfill these requirements, hardware support inside the microcontrollers is required. To observe the distributed state of all targets in a testbed, such hardware support and the corresponding required infrastructure is needed not only at one or a few central locations, but at all targets. Such a distributed tracing infrastructure requires testbed-wide orchestration and the traces collected at different nodes need to be merged in a way that they allow statements about the network-wide state.

In this chapter, we present a system for testbed-wide non-intrusive tracing of targets without requiring code instrumentation. To enable this, we propose the use of a dedicated debug and trace probe for each target. This permits us to make use of the on-chip debug and trace hardware available in modern microcontrollers (see Figure 4.2 and Section 4.3.2) which enables data logging in parallel to the normal program execution without influencing the program execution in any way. The proposed tracing system records memory accesses (read/write) to existing variables, thus there is no need for manual code instrumentation or re-instrumentation if a different aspect of the software is investigated. In addition, the proposed tracing system does not require any debug information which can optionally be generated when compiling the executable binary. The same program image can be used in different test runs to trace different memory locations without the need to recompile the program code. In contrast to commonly used stop-start debugging, the proposed tracing method is completely free of any online interaction with the targets during the test execution. This facilitates the repeatability of tests and allows for unattended test execution. However, as described in Chapter 3, the same testbed architecture allows the implementation of a stop-start based online debugging service.

To order and align the traces from different observers on a common time base, all traced events are timestamped twice: (1) a local time is logged by the debug and trace hardware which is part of the microcontroller, and (2) the synchronized testbed time is logged on the observer. In post-

processing on the testbed server, the collected data is used to apply time correction, i.e., to add an accurate global timestamp to each traced event. This also permits us to compare the traces with events from other tracing services such as logic or power tracing.

For every event, the resulting tracing data contains a synchronized timestamp, an identifier of the target, accessed variable (or memory location), mode of access, value of the accessed variable and optionally the instruction memory address of the instruction responsible for the access. The data can be visualized for manual analysis and interpretation (see Section 4.6.3). Furthermore, a model of the system can be used to predict the output of the tracing. The prediction can then be automatically compared with the actual output of a test on the testbed. We demonstrate the applicability of this approach with a simple example including a networking protocol in Section 4.6.3.

In summary, this chapter contains the following contributions:

- A novel concept, design, and implementation of a non-intrusive distributed tracing system for a network of wireless IoT devices without the requirement for code instrumentation.
- Methodology to time synchronize traces from different independently collected data of the proposed tracing system with a sub-millisecond accuracy.
- Case-study which demonstrates how the proposed tracing system can be used for the validation of software used in distributed wireless IoT devices.

In Section 4.2, we discuss related tracing systems and compare their advantages and disadvantages. Section 4.3 introduces the proposed tracing system architecture and Section 4.4 describes the novel time synchronization methodology. In Section 4.5, we present a concrete implementation of the proposed tracing system based on the FlockLab 2 testbed and provide characterization values. In Section 4.6, we apply the tracing method to a timing-critical flooding based communication protocol and show how the resulting tracing data can be used to automatically detect deviations from the design and verify its correct behavior. Finally, we summarize this chapter in Section 4.7.

4.2 Related Work

Multiple works on testbeds for the IoT and wireless sensor networks (WSNs) integrate debugging and tracing methods such as halting on break- or watchpoints and inspecting the internal state of the microcontroller when the program is halted. For example in Clairvoyant [YSSW07], this set of methods is implemented by instrumenting the code and by transmitting the data via the wireless radio link. Marionette [WTT+06] also uses code instrumentation to integrate remote procedure calls (RPCs) which can be used for debugging. HATBED [YMZ19] makes use of the on-chip debug infrastructure for tracing. It supports low latency `printf` logging via the Instrumentation Trace Macrocell (ITM) as well as Data Watchpoint and Trace Unit (DWT) based watchpoints which are logged using a debug probe and collected via an out-of-band channel. Li et al. [LML20] also propose a testbed that supports collecting traces using the on-chip debug hardware. However, for both works, it is not clear how the traces of different nodes can be compared or aligned on a common time axis. Lim et al. [LT17] apply code instrumentation and uses `printf`-style logging combined with logic analyzer traces to reconstruct the program control flow. All of the above approaches require code instrumentation which influences the program execution and are therefore not well suited for tracing delay sensitive software.

Other work is focused on distributed debugging. For example in CDTRS [GJH+17], Gong et al. propose to use code instrumentation and intrusive `printf`-style logging of events and device status to reconstruct a network-wide trace of the network's state over time. Events are sorted using the time slot numbers of the wireless protocol. However, this work assumes that the targets run a real-time wireless networking protocol where all nodes are fully-synchronized. In many cases, full time synchronization is not assumed to be given but is part of the functionality to be tested on a testbed.

The Aveksha [THBR11] system uses on-chip debug and trace hardware to trace internal state such as variables and the program counter (PC) during program execution in a non-intrusive way, i.e., without the need to halt the execution. The monitoring of the internal state is implemented by periodic memory read accesses (polling) via the JTAG interface. The smallest polling period is larger than 30 µs which limits the the observation of state changes lasting a shorter period of time. Furthermore, the presented system is not suited for debugging the distributed program execution on a set of devices in a network, since Aveksha does not provide synchronized

timestamps for the traced events.

The closest related work for this chapter is Minerva [SK13] by Sommer et al. which proposes a system similar to Aveksha. The presented testbed architecture equips each observer with a debug probe and uses an on-chip debug and trace circuit of modern microcontrollers. Similar to Aveksha, Minerva supports non-intrusive tracing of variables and the program counter (PC) by memory polling as well as stop and start debugging. In addition, it allows collecting memory snapshots without halting the program execution and supports network-wide assertions which are based on the traced data. Sommer et al. analyze the overhead for a single memory poll (more than 600 µs on average) and the ordering of traced events and state that their method is not suited for high data rates and precise timing which is required according to Section 4.1.

In contrast to the discussed approaches, our solution does not use polling but uses the *data trace* feature of the Data Watchpoint and Trace Unit (DWT) hardware where data is only generated if a change is detected by the on-chip debug hardware. This event-driven approach supports the tracing of higher event rates for a short period of time, e.g., during a burst of events. In addition, with our proposed method, traces of different observers are time synchronized by leveraging timestamps from the on-chip debug hardware during post-processing. This allows for distributed tracing, i.e., the tracing data of different targets can be aligned and compared on a single time axis.

4.3 System Architecture

In this section, we describe the system architecture for the proposed distributed tracing method as depicted in Figure 4.2. We explain the components that enable hardware assisted tracing and discuss the infrastructure that is necessary in a testbed with distributed nodes.

4.3.1 Overview

The proposed tracing system requires an underlying testbed architecture that is similar to the one of FlockLab 2 presented in Chapter 3. As depicted in Figure 4.1, the testbed consists of three layers: (1) the targets, (2) the observer, and (3) a testbed server. The relevant components and the

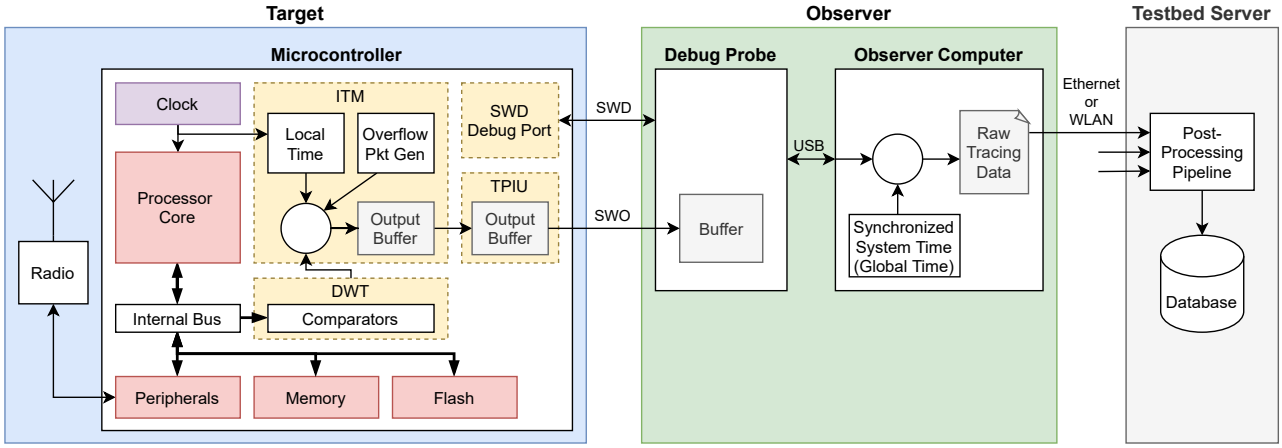


Figure 4.2: Architecture for the distributed non-intrusive tracing system proposed in this chapter. Each observer has a dedicated debug probe to interface with the on-chip debug and trace sub-system (dashed yellow boxes). Other tracing and actuation functionalities, for example available in the FlockLab 2 testbed presented in Chapter 3, are omitted in this figure.

workflow for collecting tracing data with the proposed scheme are depicted in Figure 4.2.

The targets are resource constrained devices containing a microcontroller and a low-power radio for wireless communication. For the proposed tracing system, microcontrollers on the target must feature an ARM CoreSight [Arm13] on-chip debug and trace sub-system (dashed yellow blocks in Figure 4.2) with Serial Wire Debug (SWD) and Serial Wire Output (SWO) ports. This is prevalent in modern microcontrollers. Of course, microcontrollers or processors with a similar interface and comparable services could be integrated as well.

The observer consists of an observer computer with connections to the testbed server and a debug probe (see Section 4.3.2.2) which connects the observer computer to the target via the SWD and SWO debug ports. By using the debug probe, the observer controls the target and collects tracing data from the target. The observer computer requires enough processing and storage resources to perform the necessary tracing activities independently from the testbed server during the test execution.

All observers are globally time synchronized in order to accurately timestamp the traced data close to the target where the data is generated. For observers with satellite reception, the time synchronization is achieved by synchronizing the system clock of the observer computer to the accurately time synchronized pulse-per-second (PPS) signal generated by a dedicated Global Navigation Satellite System (GNSS) receiver on the observer. For observers without GNSS reception, accurate time synchronization is achieved via the Precision Time Protocol (PTP) protocol and a PTP master device equipped with a GNSS receiver which is accessible by all observers via an Ethernet connection. In addition, the observer computer provides local buffering of traced data which ensures that enough logging capacity is available for each node, independent of potential congestion of the network between observers and testbed server. The testbed server consists of a computer which is connected to all observers via Ethernet or WLAN.

When a test is run on the testbed, the program is flashed onto the microcontroller of the distributed targets using the debug probe and the SWD port. In addition, the debug and trace hardware which is built into the target's microcontroller is configured for collecting tracing data. During the test execution, the target application is executed on the microcontrollers, and the targets may interact with each other via wireless links. In parallel, the debug and trace sub-systems on the microcontrollers generate tracing events and output tracing data via the

SWO ports. Due to the dedicated on-chip debug and trace hardware inside the microcontrollers, the tracing does not influence the program execution and is therefore non-intrusive. The traced data is forwarded to the observer computer via the debug probe. Once the test has ended, the testbed server collects the tracing data from all observers and performs the post-processing steps. Finally, the aggregated and processed tracing data can be accessed by downloading it from the testbed server.

In this chapter, we present the proposed non-intrusive distributed tracing system as an extension of the FlockLab 2 testbed architecture. However, the basic scheme of the tracing system is generally applicable and could be integrated in any testbed which satisfies the following requirements:

- The targets in the testbed feature the required support for hardware-assisted tracing as described in Section 4.3.2.1.
- The testbed infrastructure contains an accurately time synchronized and stateful observer node which includes a debug probe such that the traced data can be accurately timestamped and logged close to the target.
- The testbed infrastructure contains a computing device which can collect and post-process all tracing data.

4.3.2 Hardware Background

The proposed tracing system builds on top of the ARM CoreSight debug and trace hardware which is integrated in many commercially available modern microcontrollers. This on-chip debug and trace sub-system provides hardware assisted tracing functionalities. A debug probe is required to transfer the generated data to the observer computer. In this section, we describe aspects of the ARM CoreSight system and the debug probe which are relevant for the proposed system.

4.3.2.1 Target

For our tracing system, the targets are required to include the ARM CoreSight debug and trace system (or system with comparable functionalities and a compatible interface). The ARM CoreSight debug and trace hardware [Arm13] is provided by ARM as modular intellectual property (IP) cores which are integrated by many manufacturers in modern

microcontrollers. A minimal set of functionalities of the CoreSight system is guaranteed to be available, with many more additional or higher-performance variants and features which are optional. Table 4.1 contains a non-exhaustive list of commercially available microcontrollers which fulfill the requirements for the proposed distributed tracing system.

The important blocks of the proposed tracing system are the DWT, the ITM, and the TPIU:

- **Data Watchpoint and Trace Unit (DWT)**: The DWT is a debug block that provides data tracing among other functionalities. It includes a fixed number of hardware comparators which can be configured to trigger events based on data on the internal bus or the program counter (PC) of the microcontroller. In the proposed system, we use the DWT to collect the so called *data trace*, i.e., recording the accessed memory address, the data value, and the address of the instruction that caused the memory access (PC value) [Arm21]. Traceable memory addresses cover the whole memory address space which is accessible by load and store instructions, including addresses of memory-mapped peripheral devices. Apart from the PC value, no other register values from inside the processor core are included in the data trace.
- **Instrumentation Trace Macrocell (ITM)**: Among other functionalities, the ITM arbitrates debug and trace packets from different sources of the debug and trace sub-system. For the tracing system proposed in this chapter, the ITM is only used to forward the data stream from the DWT to the Trace Port Interface Unit (TPIU) output buffer and to add local timestamp packets to this stream.
- **Trace Port Interface Unit (TPIU)**: The TPIU provides an interface to output the generated debug and trace stream to an external device, the so-called Trace Port Analyzer (TPA). For the proposed tracing system, we require it to support the Serial Wire Output (SWO) which is a serial port similar to traditional UART.

4.3.2.2 Observer

In order to (1) allow the observer to instruct the debug and trace sub-system inside the microcontroller which events to trace and (2) to transfer generated tracing data to the observer computer, a commercially available

Manufacturer	Product	Core Architecture	Num. DWT Comparators
Atmel	SAM4	ARM Cortex-M4	4
Nordic Semiconductor	nRF52840	ARM Cortex-M4	4
NXP Semiconductors	LPC176x	ARM Cortex-M3	4
Renesas	RA6M1/M4	ARM Cortex-M33/M4	4
Silicon Labs	EFM32WG	ARM Cortex-M4	4
Silicon Labs	EFM32PG12/22	ARM Cortex-M33/M4	4
STMicroelectronics	STM32L1	ARM Cortex-M3	4
STMicroelectronics	STM32L4/F4	ARM Cortex-M4	4
STMicroelectronics	STM32L5	ARM Cortex-M33	4
STMicroelectronics	STM32F7/H7	ARM Cortex-M7	4
STMicroelectronics	STM32WB/WL	ARM Cortex-M4	4
Texas Instruments	MSP432P4xx	ARM Cortex-M4	4

Table 4.1: Overview of a selection of commercially available microcontrollers and systems on a chip (SoCs) which are suitable for the proposed distributed tracing system (non-exhaustive list).

debug probe is used. On ARM-based microcontrollers, a debug probe allows to interact and access the CoreSight components in the microarchitecture.

To control and configure the microcontroller, the debug probe connects to a debug port on the target, in our case an SWD debug port. To receive tracing data output from the microcontroller, the debug probe acts as a TPA and reads from the SWO port. As depicted in Figure 4.2, in our testbed architecture, the debug probe is integrated into the observer.

4.4 Tracing and Time Synchronization

The idea of our proposed tracing system architecture is to provide an out-of-band infrastructure to allow for observations which do not influence the system under test. Tracing events on distributed targets is meaningful only if the traces from different nodes have the same time base such that the events can be compared and ordered testbed-wide. The targets, on which the events are triggered and logged, have independent clock systems. The targets are resource constrained and their clocks usually exhibit non-negligible drift over time. But according to the requirements, we are not allowed to change the target implementation and add time synchronization just for the purpose of testing. Finally, the traced data passes through

different buffers on the microcontroller and on the debug probe, adding non-deterministic and non-observable delays.

In the remainder of this section, we explain our approach for synchronizing the time of events traced on different targets and discuss the potential use of the obtained data.

4.4.1 Overall Time Synchronization Architecture

There are different methods to time synchronize event observations in a network of distributed wireless devices. One approach is to use a wireless time synchronization protocol, e.g., PulseSync [LSW15] or Glossy [FZTS11], which distributes radio messages precisely at known time instants such that the nodes can reconstruct the reference time. Another approach is the use of dedicated hardware at the target such as a GNSS-receiver [LMT16] or a wired synchronization network based on PTP [LBMB16], which provides an accurate time reference at the target. However, these methods do not meet the requirements of our proposed tracing system (see Section 4.1) since they are intrusive. In addition, we do not assume that the system under test provides the infrastructure for accurate time synchronization, but rather we assume that the synchronization of the system is potentially unreliable and thus itself represents a reason for testing and validation. Therefore, we propose to use an out-of-band time synchronization based on two layers. In a first layer, the system time of each observer computer is synchronized to the accurate absolute GNSS-derived time as described in Section 4.3.1. In a second layer, the tracing data obtained from the target is synchronized to the system time of the observer computer. In the following, we explain the method for the second-layer synchronization which is applied offline using a regression.

Let us denote the time derived from the microcontroller's clock in the Instrumentation Trace Macrocell (ITM) as *local time*. For example, an event i on a target s happens at local time $t_s[i]$. Such an event i as logged on target s is then automatically timestamped in the ITM. Note that the timestamp may not be taken at the very moment of the event i as it may be added some time later to the data trace due to the necessary processing and buffering. We refer to this timestamp as $t_s^{\mathrm{ITM}}[i]$ as it is derived from the local clock of target s.

Due to the high precision of the synchronization of the system time on the observer computer, we can assume that all observers use the same time base and refer to this testbed-wide synchronized time as *global time*. When a traced event i arrives on the observer computer, a global timestamp

$t^{\mathrm{OBS}}[i]$ derived from the global time is added. More details on the collection of events in the data trace as well as on the format of the data trace are provided in Section 4.4.2 and Section 4.4.3, respectively.

In summary, an event i that happens at target s at global time $t[i]$ and local time $t_s[i]$, gets a local timestamp $t_s^{\mathrm{ITM}}[i]$ in the ITM of the microcontroller and a global timestamp $t^{\mathrm{OBS}}[i]$ on the observer. We are now interested to get an estimate of $t[i]$ from $t_s^{\mathrm{ITM}}[i]$ and $t^{\mathrm{OBS}}[i]$. This is necessary to at least partially order the distributed events occurring at the targets.

To simplify the explanation, we first assume that the relation between the local time on the microcontroller and the global time on the observer can be approximated well by a linear relation. This holds if the drift of the two time bases is constant for the duration of the test. More general schemes, e.g., for cases with clock drifts due to changing temperatures, are discussed at the end of this section. With the assumption of a linear relation, the relation can be expressed as

$$t[i] = t_s[i] \cdot m_s + q_s \tag{4.1}$$

In the relation, m_s denotes the slope and q_s denotes the intercept of node s. The idea of this approach is to compensate the drift and offset between the local and global time. The parameters m_s and q_s will be determined by linear regression as described below. Since the reference of the local time changes with a reset of the target, the regression parameters need to be re-calculated. Special synchronization packets allow us to partition the test time into time periods between resets (*synchronization epochs*) as described in Section 4.4.3. To further simplify the discussion, we will at first neglect all possible error terms in the relations.

We now describe how we can determine the global time of an event occurrence $t[i]$ from the timestamp $t_s^{\mathrm{ITM}}[i]$ contained in the data trace. We suppose that we empirically determined an estimate $E(\cdot)$ of the delay $E(\Delta^{\mathrm{ITM}})$ between the occurrence $t[i]$ of event i and the added timestamp $t_s^{\mathrm{ITM}}[i]$. Therefore, we can write $t_s^{\mathrm{ITM}}[i] = t_s[i] + E(\Delta_s^{\mathrm{ITM}})$ where $E(\Delta^{\mathrm{ITM}}) = E(\Delta_s^{\mathrm{ITM}}) \cdot m_s$ relates the timestamping delay in local and global time. We can now determine $t[i] = (t_s^{\mathrm{ITM}}[i] - E(\Delta_s^{\mathrm{ITM}})) \cdot m_s + q_s$, and therefore

$$t[i] = t_s^{\mathrm{ITM}}[i] \cdot m_s + (q_s - E(\Delta^{\mathrm{ITM}})) \tag{4.2}$$

In other words, given the timestamp $t_s^{\mathrm{ITM}}[i]$ of an event i, we can determine the global time of its occurrence, provided that we know an estimate of the delay $E(\Delta^{\mathrm{ITM}})$ as well as the parameters m_s and q_s. As

discussed in Section 4.4.2, the delay $E(\Delta^{\mathrm{ITM}})$ can be neglected in many cases.

These parameters will be determined using linear regression based on the set of pairs of timestamps $(t_s^{\mathrm{ITM}}[i], t^{\mathrm{OBS}}[i])$ in the data trace for the received events i. To this end, we suppose that we also have available an estimate of the delay $E(\Delta^{\mathrm{TRF}})$ between the timestamp of an event by the ITM and the timestamp of the observer (transfer (TRF) time). Then we can derive $t^{\mathrm{OBS}}[i] = t^{\mathrm{ITM}}[i] + E(\Delta^{\mathrm{TRF}})$ where $t^{\mathrm{ITM}}[i]$ denotes the global time of the timestamp added to event i by the ITM with $t^{\mathrm{ITM}}[i] = t_s^{\mathrm{ITM}}[i] \cdot m_s + q_s$. As a result, we obtain

$$t^{\mathrm{OBS}}[i] - E(\Delta^{\mathrm{TRF}}) = t_s^{\mathrm{ITM}}[i] \cdot m_s + q_s \tag{4.3}$$

Therefore, given the pairs of timestamps $(t_s^{\mathrm{ITM}}[i], t^{\mathrm{OBS}}[i])$ as well as an estimate $E(\Delta^{\mathrm{TRF}})$, estimations of the relation between local time at target s and the global time according to (4.1) can be determined by regression of $t^{\mathrm{OBS}}[i] - E(\Delta^{\mathrm{TRF}})$ on $t_s^{\mathrm{ITM}}[i]$.

There are multiple delays between adding the local and adding the global timestamp to the tracing data which need to be considered. A detailed list and discussion of the aggregated delays Δ^{ITM} and Δ^{TRF} is given at the end of Section 4.4.2. Typically, the expected values of the delays need to be determined experimentally since the exact models of all delay components are not known in general. In an actual implementation, Δ^{TRF} can reach values in the order of 200 ms.

Due to the uncertainty in the delays Δ^{ITM}, Δ^{TRF} as well as in the stability of the local and global clocks, (4.2) cannot be satisfied with equality. To take this into account, we introduce an error term $\epsilon[i]$ which leads to

$$t[i] = t_s^{\mathrm{ITM}}[i] \cdot m_s + (q_s - E(\Delta^{\mathrm{ITM}})) + \epsilon[i] \tag{4.4}$$

In Section 4.5.2.1, we experimentally determine an estimate of the error $\epsilon[i]$ for a concrete implementation of the tracing system.

So far, we assumed that the clock source drifts are constant and therefore a single linear regression is used to estimate the relation between local and global timestamps for the time period between two target resets, i.e., the test duration if there is no user-triggered reset. The drift variations of clocks in microcontrollers are highly dependent on the quality of the clock source and temperature influence [TAA19, EFD+18]. In the following, we determine an estimate of the timestamping error of the data trace system which is caused by the drift of the clock source. For the worst-case, we assume that there are only two clock source frequencies, the

minimum and the maximum, and that one is active in the first half and the other in the second half of the test duration. If we assume the target uses a high-quality clock source such as an external crystal oscillator with a deviation of ±25 ppm over the operating temperature range on the target and a test duration of 10 minutes, we estimate the worst-case deviation of the global timestamp to be 15 ms. In case of a low-quality clock source, such as an internal RC oscillator with a deviation in the order of ±1% over the operating temperature range and the same test duration, we estimate a deviation of 6 s. In order to reduce the duration during which the clock source can drift away from global time, we apply a piecewise linear regression. Of course more sophisticated regression models, such as cubic regression splines could be applied analogously. For regression intervals of 30 s, the worst-case estimation of the previous example would result in a deviation of 0.75 ms for the external crystal and 300 ms for the internal RC oscillator.

4.4.2 Collection of Data Trace and Time Synchronization Data

To collect data trace data from the microcontroller on the target, the debug and trace sub-system needs to be configured and the generated data need to be fetched and forwarded to the observer computer. To reconstruct the accurate global time of the events, appropriate timestamps need to be collected as well, see Section 4.4.1. In the following, we will describe the corresponding configurations and mechanisms as far as they are relevant for the tracing architecture.

The different blocks of the debug and tracing system are configured using the SWD debug port, see Figure 4.2. The debug probe translates instructions from the observer computer into SWD protocol operations. For every global variable or data memory location which shall be traced, we configure one DWT comparator to trigger whenever the corresponding address in the data memory is accessed. We use the symbol table generated by the compiler to translate global variable names into memory addresses. The type of access (read, write, or both) is configured as a condition for the triggering of events.

Whenever the comparator triggers an event, a data trace packet that contains the comparator identification (ID), the value at the address location in the data memory, and the access type is generated. Optionally, if requested by configuring the debug sub-system, a second data trace packet

is generated which contains the current program counter (PC). An example trace of generated packets and timestamps is depicted in Figure 4.3. The data trace packets are then forwarded to the Instrumentation Trace Macrocell (ITM).

The ITM adds additional packets to the output packet stream and forwards the stream to the Trace Port Interface Unit (TPIU), see Figure 4.2. After every reset of the microcontroller, the ITM adds a synchronization packet. In addition, the ITM adds local timestamp packets containing the current value of the local timestamp counter. In order to save output channel bandwidth, the ITM only adds a local timestamp packet if the value in the timestamp counter differs from the timestamp generated for the previous data trace packet, or if the timestamp counter reaches its maximum value. The local timestamps are differential, i.e., the timestamp counter is reset to zero whenever a timestamp packet has been generated. The timestamp counter is incremented based on the clock signal configured for the processor of the microcontroller. In addition, a prescaler which is specific for the local timestamp counter can be configured. As a result, for a prescaler larger than 1, events generated by different load or store instructions (at different processor clock cycles) can fall into the same local timestamp epoch.

The TPIU outputs the data as a byte stream via the SWO port, see Figure 4.2. The data is then received and buffered locally by the debug probe. The observer computer periodically polls the debug probe and fetches the data from the buffer in the debug probe. For each fetch action, the observer computer takes a global timestamp from the synchronized system time and adds it to the fetched data trace. At the end of the testbed test, the testbed server fetches all raw tracing data files from all observers.

The chain from generating data trace packets in the DWT unit to storing the traced data in a file on the observer computer contains different buffers and additional data can increase the amount of data packets in the stream on the way. Whenever one of the buffers overflows, tracing data is delayed or discarded. In order to detect such incidents, the ITM can add corresponding information to the tracing stream. The local timestamp packets contain delay markers which indicate that either the timestamp or the data or both have been delayed relative to the actual event triggered in the DWT. In addition, if packets are discarded due to a buffer overflow, the ITM can generate and insert a special overflow packet which consists of a header only. We consider these special situations and omit delayed data when selecting the pairs of timestamps that are used for the regression in (4.3).

In the following, we list the relevant components of the delay between the occurrence $t[i]$ of an event i and the local timestamp $t_s^{\mathrm{ITM}}[i]$ denoted as $\Delta^{\mathrm{ITM}} = \Delta^{\mathrm{cap}} + \Delta^{\mathrm{local}}$ as well as the delay between the timestamp of an event by the ITM and the timestamping by the observer $\Delta^{\mathrm{TRF}} = \Delta^{\mathrm{MCU}} + \Delta^{\mathrm{SWO}} + \Delta^{\mathrm{FETCH}} + \Delta^{\mathrm{USB}} + \Delta^{\mathrm{GT}}$.

Δ^{cap} The delay for capturing the data trace event in the DWT and generating data trace packet. The actual delay is implementation defined. However, based on the specification of the hardware tracing system [Arm21], we know that the delay for this action is negligible (a few clock cycles at most) compared to the delays in later stages of the trace pipeline.

Δ^{local} The delay for generating the local timestamp packet in the ITM. Again, according to the specifications [Arm21], the delay for this action is negligible compared to the delays in later stages of the trace pipeline.

Δ^{MCU} The delay for adding packets to the ITM output buffer and to forward them to the TPIU output buffer. Again, based on the specifications [Arm21], the delay for this action is negligible compared to the delays in later stages of the trace pipeline in case the delay marker in the corresponding local timestamp does not indicate a delay.

Δ^{SWO} The time for transferring all previous packets up to and including the local timestamp packet from the TPIU to the buffer of the debug probe via SWO. This delay depends on the state of the queues and is directly related to the speed of the SWO connection f_{SWO}. A typical maximum value for the SWO speed is $f_{\mathrm{SWO}} = 4\,\mathrm{Mbaud}$.

Δ^{FETCH} The delay until the observer computer initiates the next fetching action to transfer data from the buffer of the debug probe to the observer computer. This delay is bounded by the configured loop period for fetching data. Feasible loop periods are in the order of 10 ms.

Δ^{USB} The time for transferring the data from the buffer of the debug probe to the observer computer. The variance of this delay is expected to be in the order of milliseconds due to the non-deterministic behavior of the USB protocol implementation.

Δ^{GT} The delay for obtaining a global timestamp on the observer computer. The variance (jitter) of this delay is expected to be in the order of 200 µs since the timestamp is taken in software in the user space of the operating system.

As mentioned before, an estimated average of the accumulated delays $E(\Delta^{ITM})$ and $E(\Delta^{TRF})$ are determined as part of the system characterization (see Section 4.5.2.1). These values are used to determine the global time of event occurrences according to (4.2) via the regression in (4.3).

4.4.3 Time-Aware Parsing of Raw Tracing Data

After the execution of a test, the raw tracing data of each observer is available on the testbed server. In this section, we explain in detail how this data is parsed and how the global testbed time for each traced event is reconstructed in post-processing.

The collected raw tracing data of an observer contains a stream of bytes received via the Serial Wire Output (SWO) port, interleaved with global timestamps $t^{OBS}[i]$ derived from synchronized system time on the observer computer. An example is depicted in Figure 4.3. The SWO byte stream contains packets which consist of a one-byte header and a payload of 0 to 4 bytes. Since the observer computer fetches the available number of bytes from the debug probe buffer irrespective of SWO packet boundaries, the global timestamp values can be interleaved by individual bytes of a packet.

The server uses the multi-stage pipeline depicted in Figure 4.4 to parse and decode the raw tracing data of each observer. First, the SWO bytes are parsed into SWO packets. Each packet is annotated with the global timestamp corresponding to the first SWO byte from which the packet has been constructed. Then, the stream of packets is split into batches of packets that correspond to the same synchronization packet (SYNC), i.e., the period of time (synchronization epoch) between two microcontroller resets. If there was no microcontroller reset, all packets belong to the same synchronization epoch. Packets in resulting synchronization epochs are further split into batches such that all packets in one batch correspond to the same local timestamp packet (LOCAL TS). This means that all data trace packets (DATA) in the same local timestamp epoch were triggered while the local timestamp counter in the Instrumentation Trace Macrocell (ITM) (see Figure 4.2) contained the same value. In a next step, different types of data trace packets which were triggered by the same event, e.g., a packet containing data and a packet containing the program counter (PC), are

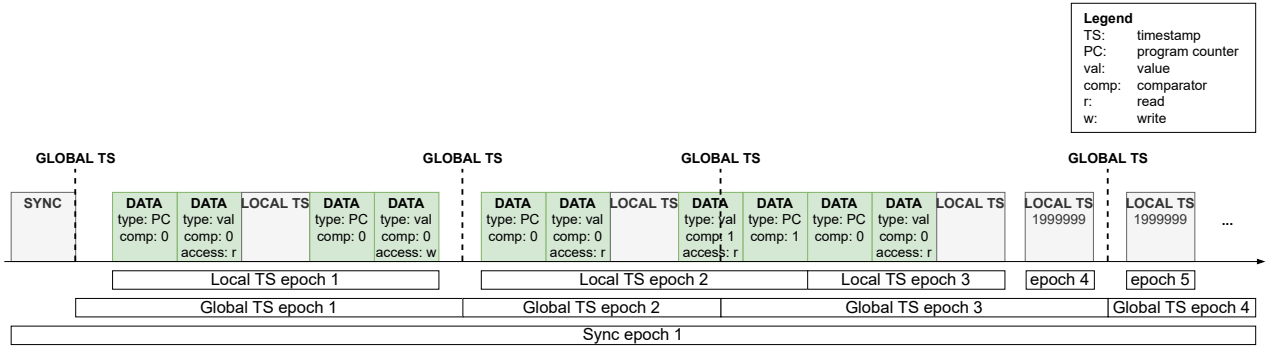


Figure 4.3: Example trace of SWO packets with local and global timestamps.

combined and the corresponding local timestamp from the local timestamp epoch is added. Each resulting data record contains a global timestamp, a local timestamp, and the traced values.

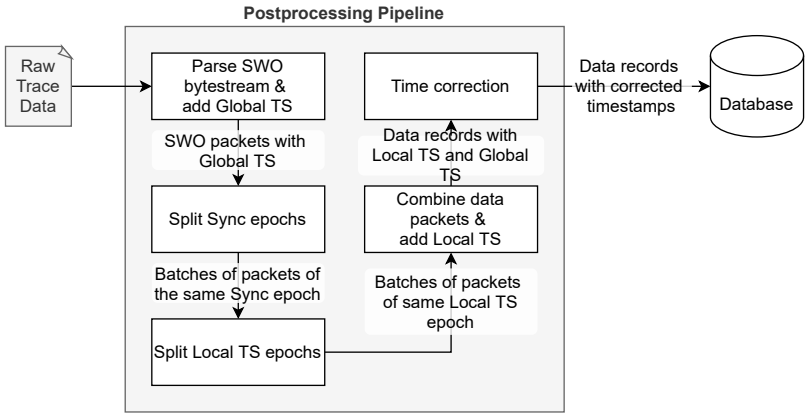


Figure 4.4: Pipeline for processing raw tracing data into records of data trace events with time synchronized timestamps on the testbed server.

In the final step, the time correction is applied to all data records, see Section 4.4.1. For this, high-quality timestamps, i.e., timestamps from local timestamp packets where the delay marker does not indicate a delay, are used to determine the regression parameters using (4.3). Since the global timestamps taken on the observer $t^{\mathrm{OBS}}[i]$ contain significant outliers due to the non-deterministic delays Δ^{TRF}, we use two stages for the regression according to (4.3). After a first regression, we remove all data points which deviate from the fitted line by more than 2 times the resulting standard-deviation. With the remaining synchronization points $(t_s^{\mathrm{ITM}}[i], t^{\mathrm{OBS}}[i])$, we perform another least-squares fit according to (4.3). According to (4.2), we use the resulting regression parameters to map the local timestamps $t_s^{\mathrm{ITM}}[i]$ to the synchronized global time $t[i]$ of the events i.

In the very last step, the comparator IDs are mapped back to the name of the variable for which the tracing was originally requested.

When the comparators in the DWT unit do not trigger the generation of data trace packets before the local timestamp counter reaches its maximum value (1999999 in the example in Figure 4.3) the ITM outputs a local timestamp packet with the maximum value. These packets form their own local timestamp epoch without data trace packets. This provides time

synchronization points for the regression even for time periods where there are no data trace packets generated.

4.4.4 Potential of the Proposed Tracing System

In this section, we describe the resulting tracing data, highlight use cases such as validation using a simulation, and discuss trade-offs and limitations of the proposed tracing system.

4.4.4.1 Resulting Trace Data

The resulting tracing data from a test execution with the proposed tracing system contains one record for every memory access of the chosen variables and access types configured on each observer. As in the example depicted in Table 4.2, the tracing data contain the corrected timestamp according to (4.2), ID of the target which generated the event, the variable name, the value in the data memory after the access, the access mode (read (r) or write (w)), the program counter (PC) (only if enabled) at the time of the memory access, and the delay marker. The delay marker is contained in the local timestamps and indicates whether a delay occurred when adding the data trace or local timestamp packets to the output buffer inside the debug and trace sub-system on the microcontroller, as discussed in Section 4.4.3.

4.4.4.2 Use Cases

The obtained tracing data can be used to visualize and manually analyze the evolution of state changes of the distributed system over time. Examples of such visualizations are provided in Section 4.6.3. We provide a visualization tool for data collected with FlockLab 2 as open source[1].

The tracing data also allow the systematic and automated validation of the behavior of the software running on a set of distributed wireless targets. The basic concept is to implement the distributed system as a simulation (e.g., [SLA+10]) or as timed automata (e.g., [JSY+18]) and also as software running on the targets on the testbed. The output of the simulation or timed automata and output of the testbed can then be analyzed and compared. Related work has shown different approaches to perform such validations.

[1] https://flocklab.ethz.ch

Timestamp	Target ID	Variable	Value	Access mode	PC	Delayed
1602661351.1832957	11	tx_len	0	r	0x800bbfc	no
1602661351.1833105	11	tx_len	0	r	0x800b8ea	yes
1602661351.1833405	11	tx_len	1	w	0x800b8fa	yes
1602661351.1834369	4	tx_len	0	r	0x800bbfc	no
1602661351.1834520	4	tx_len	0	r	0x800b8ea	yes
1602661351.1834820	4	tx_len	1	w	0x800b8fa	yes
1602661366.1017075	4	tx_len	1	r	0x800be6c	no
1602661366.1017935	11	tx_len	1	r	0x800be6c	no
1602661366.1824489	11	tx_len	1	r	0x800be6c	no
1602661366.1825450	4	tx_len	1	r	0x800be6c	no
1602661366.1826136	11	tx_len	1	r	0x800bbfc	no
1602661366.1826262	11	tx_len	1	r	0x800b8ea	yes
1602661366.1826563	11	tx_len	2	w	0x800b8fa	yes
1602661366.1827123	4	tx_len	1	r	0x800bbfc	no

Table 4.2: Example output from the data trace service on FlockLab 2. Timestamps are synchronized using the method described in Section 4.4.1.

Examples are conformance tests [WLT13, ROWA19], unit tests [OW10, IHGS14], or distributed assertions [SK13, RM09].

The proposed tracing system provides an out-of-band infrastructure to time synchronize the traced events. However, it is possible that the system under test implements a reliable time synchronization with even better temporal accuracy, e.g., based on Glossy [FZTS11]. For this case, it is possible to combine the more accurate time synchronization of the wireless protocol with the non-intrusive distributed tracing. The only requirement would be memory accesses at the critical time instants, namely at the synchronization reference points and when an event is to be logged. By using the tracing data, the offset of the event occurrence relative to the synchronization reference points can be determined with high accuracy as shown in Section 4.5.2.2. In many cases, memory accesses are required at the corresponding locations in the target code anyway and are therefore already present.

In addition, the tracing system proposed in this chapter allows to trace the program counter (PC) values, i.e., the instruction memory addresses at which the traced memory accesses are generated. One possibility to leverage this information is to reconstruct a partial program flow trace.

There is extensive previous work available that investigates methods to assess the chronological ordering of observed events. For example, Römer et al. [RM09] introduce flags to indicate whether the ordering can

be determined and Hahner et al. [HRB04] use a graph to determine the temporal order of events.

Detailed investigations of methods which make use of the traced data are not the focus of this chapter, as extensive related work that can be applied to analyze and visualize the timed traces from the distributed targets is available.

4.4.4.3 Limitations

Because the proposed tracing system is based on on-chip debug hardware, it is non-intrusive and efficient, but at the same time it has some limitations that need to be considered when using the testbed infrastructure. In the following, we list and discuss some noteworthy limitations of the proposed distributed tracing system.

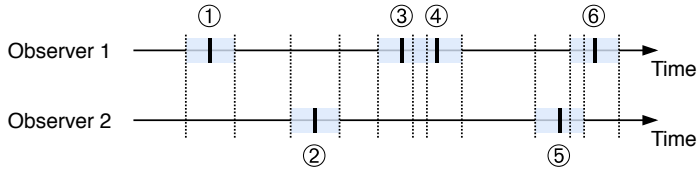


Figure 4.5: The ordering of events for the same observer (e.g., events 3 and 4) is guaranteed by the on-chip debug and trace hardware, but statements about the ordering of events of different observers (e.g., events 5 and 6) are limited by the time synchronization accuracy of the data trace service.

Timestamping Accuracy and Event Ordering: In general, total ordering with distributed independent loggers and non-zero time synchronization accuracy is not possible. The amount of events that can be ordered depends on the time synchronization accuracy and the time distance between event occurrences. Figure 4.5 depicts an example with two observers with data trace events (black bars) and their temporal uncertainties (transparent boxes mark the upper and lower bound of the timestamp). The uncertainty regions directly relate to the range of $\epsilon[i]$ from (4.4). The ordering of events for the same observer is guaranteed by the on-chip debug and trace hardware. In the example this means that the ordering of events 3 and 4 is unambiguous even though the uncertainty regions overlap. The ordering of events of different observers is limited by the time synchronization accuracy of the tracing system, see (4.4). The ordering is

unambiguous if the events are further apart than twice the upper bound of the uncertainty of a single timestamp. In the example, this means that the ordering of events 1 and 2 can be clearly determined whereas no statement on the ordering of events 5 and 6 can be made. By using an upper bound of the temporal error based on the characterization in Section 4.5.2.1 as well as the timestamps and target IDs from the output of the data trace service, a user can identify pairs of timestamps on different targets without sufficient spacing for which the ordering is ambiguous.

The use of additional hardware, such as a field-programmable gate array (FPGA) or a dedicated microcontroller, for timestamping the SWO output could potentially improve the timestamping accuracy as the delays Δ^{FETCH} and Δ^{USB} could be eliminated and the relatively large variance of Δ^{GT} could potentially be reduced. We did not investigate such an improvement as the obtained timestamping accuracy is sufficiently high as shown in Section 4.5.2.1 and Section 4.5.2.2 and additional hardware would increase the observer complexity. However, it could be an interesting approach for future work.

Maximum Event Rate: The maximum supported frequency with which a variable can be accessed such that the data trace service can still trace the accesses without issues, i.e., without delay markers indicating a delay or overflow packets being generated, is limited by the following factors:

- The maximum supported event repetition rate of the event triggering hardware inside the DWT unit.
- The size of the buffers in the debug and trace sub-system and the debug probe.
- The method and data transfer protocol that is used to offload the tracing data from the target to an external device (debug probe and observer computer).
- The amount of data that is generated by every event.

For example, enabling the tracing of program counter (PC) values increases the number of data trace packets which are generated. Configuring the service to only trace write accesses instead of read and write accesses reduces the number of packets in general.

Amount of State Which Can Be Monitored: The number of variables/memory addresses which can be monitored is limited by the number of available comparators in the on-chip debugging hardware. The feature set of the debug sub-system is determined by the manufacturer of the

microcontroller and cannot be extended. An overview of a selection of microcontrollers and SoCs with the number of comparators is provided in Table 4.1.

Complete Sequence of Local Timestamp Packets: The timestamps in local timestamp packets are differential to save tracing bandwidth. As a consequence, the timing is only correct if all local timestamp packets can be successfully forwarded to the observer computer. In case a buffer inside the on-chip debug and trace sub-system is full, an overflow of the buffer can occur and local timestamp packets may be discarded. As a result the synchronized timestamps cannot be reconstructed correctly in post-processing. However, such incidents can be detected based on the overflow packets in the SWO byte stream. System resets on the other hand do not lead to problems with reconstructing the synchronized timestamps since synchronization packets indicate a restart of the SWO stream.

Power Profiling and Low-Power Modes: The proposed non-intrusive distributed tracing system focuses on the tracing and validation of the functional behavior of the system under test. Testbeds usually feature separate services dedicated to perform detailed power profiling. As shown in Chapter 3, the FlockLab 2 testbed supports accurate high-dynamic range power profiling to accurately measure the power consumption during radio transmissions as well as during low-power modes. It is possible to use both services, the distributed data tracing and the power profiling, simultaneously. However, in this case the significance of the power measurement is limited, since the power consumption of the debug and trace sub-system is also measured, as it cannot be isolated and thus supplied with power separately. In the case of a pure power measurement, the debug and trace sub-system can be switched off and therefore does not influence the measurement.

Modern microcontrollers feature many different power modes and allow to turn on/off separate components individually. The available options of course depend on the actual implementation. Usually, the debug and trace sub-system requires a high-frequency clock source. For ultra-low-power modes, usually only low-frequency clock sources are enabled. By default, a microcontroller with data trace output enabled enters a low-power mode with a low-frequency clock source when instructed to do so, but does not disable the high-frequency clock source as it is still required by the debug and trace sub-system. As a consequence, the power consumption is increased but the functional behavior, including timing, is very close to the behavior without debug output enabled. Certain microcontroller

implementations support the automated disabling of the debug and trace sub-system in case a low-power mode with low-frequency clock source is entered. However, the ARM CoreSight architecture does not specify this mode of operation and some microcontroller implementations do no generate synchronization packets upon wake-up from low-power mode. As a consequence, the proposed distributed tracing system does not support time correction of such traces.

4.5 Implementation

In order to characterize the performance and to show the use of the system for validation of wireless IoT devices, we implemented the proposed tracing system for STM32L4 (STMicroelectronics) based targets as part of the FlockLab 2 testbed presented in Chapter 3. The implementation is open-source and the FlockLab 2 testbed with the proposed tracing service is publicly accessible[2].

4.5.1 Implemented System

In this section, we briefly describe the specifications of the implemented system and explain the workflow for using the data trace service.

4.5.1.1 Target

We use the DPP2 SX1262 target architecture described in Chapter 3. This target consists of the DPP2 SX1262 communication board presented in Chapter 2 and a simple target adapter that connects the pins of the communication board to the FlockLab 2 target interface. The DPP2 SX1262 communication board consists of an STMicroelectronics STM32L433CC microcontroller and a Semtech SX1262 low-power long-range radio transceiver. The microcontroller features an ARM Cortex-M4 core which supports a clock frequency of up to 80 MHz. For our implementation, we use the external crystal as a clock source and run the microcontroller at 48 MHz.

In the ARM Cortex-M4 architecture, the intellectual property (IP) cores for supporting CoreSight debug and trace functionalities are optional. The

[2] https://flocklab.ethz.ch

STM32L433 microcontroller implements a combined SWD and JTAG debug port and integrates the DWT, ITM, and TPIU blocks, see also Section 4.3.1 and Figure 4.2. The key specifications are listed in Table 4.3. The DWT contains 4 comparators which allow the parallel tracing of up to 4 variables or data memory addresses, respectively. The TPIU provides the SWO output port with a maximum speed of 4 Mbaud.

Component	Value
Number of DWT comparators	4
ITM output buffer size	10 bytes
TPIU output buffer size	128 bytes
Max SWO transfer speed	4 Mbaud
Debug probe buffer size	4 MB

Table 4.3: Specification of the example implementation of the proposed tracing system consisting of an STM32L433 microcontroller and the FlockLab 2 testbed infrastructure.

4.5.1.2 Testbed Infrastructure

The implemented example system is part of the publicly available FlockLab 2 testbed presented in Chapter 3. The observers are equipped with a Segger J-Link OB (on-board) debug probe which features a 4 MB buffer. A BeagleBone Green single-board computer running a Linux operating system serves as the observer computer. It interfaces with the debug probe via USB. The time synchronization of the BeagleBone observer computer is based on the PPS signal generated by a u-blox M8 GNSS receiver, providing an accuracy of 60 ns according to the datasheet. Observers without GNSS signal reception are time synchronized with PTP through the BeagleBone's Ethernet interface to connect to another BeagleBone (PTP master) with a GNSS receiver. The `chrony` software is used to adjust the BeagleBone's system time based on the PPS signal or the PTP protocol. On the observer computer, the J-Link library from Segger and the Python library `pylink` are used to interact with the debug probe. A Python script on the observer computer is used to configure the microcontroller on the target as well as to periodically check the buffer on the debug probe and to log available tracing data together with the global timestamps derived from the observer computer's system time to a file. The implemented system uses a poll period of 10 ms and a local timestamp prescaler value

of 16. The chosen values represent a trade-off between minimizing the delay jitter for forwarding the traced data and keeping the computational overhead on the observer computer low. For the SWO connection between the microcontroller and the debug probe, we use the maximum supported SWO speed of 4 Mbaud.

On the testbed server, a set of Python scripts and libraries are used to schedule a test, distribute configuration information to the different observers, fetch tracing data collected by the observers, and post-process the tracing data as described in Section 4.4.3.

4.5.1.3 Workflow of the Data Trace Service on FlockLab 2

Along with other configurations, the experimenter uses an XML-formatted test configuration file to specify the variables and the corresponding access modes which shall be traced on each observer of the testbed. The mapping of variables to addresses is performed automatically by the testbed using the symbol table provided in the program image for the microcontroller. Alternatively, it is possible to directly specify an address of the data memory which shall be monitored. The interface for the experimenter is interaction-free during test execution which facilitates repeatability.

The testbed executes the test according to the configuration specified in the XML file, i.e., during the test it collects the data trace output and processes it at the end of the test as described in Section 4.4.3. After the completion of the test, the experimenter can download the resulting tracing data or inspect a web-browser based visualization of the collected tracing data.

4.5.2 Characterization

In this section, we investigate and measure the performance of the implemented tracing system. Important aspects which do not directly follow from the specifications of the used system components are the accuracy of the corrected timestamps and the maximum supported event rate.

4.5.2.1 Absolute Time Synchronization Accuracy

In order to measure the absolute time accuracy of the implemented tracing system, we use the logic tracing service available on FlockLab 2 to obtain

the ground truth. The logic tracing provides a high temporal accuracy of typically below 0.25 µs, see Section 3.3.1.4. Logic tracing requires code instrumentation which is not a problem for this analysis since the microcontroller does not perform time-critical tasks. We run a test program on the DPP2 SX1262 target to generate events. Such an event consists of toggling a GPIO pin directly followed by writing to a global variable which is traced using the data trace service. For this test, we solely use FlockLab 2 observers that are synchronized using GNSS since this provides the necessary accuracy and stability.

We perform the test on 20 targets with 20 observers. On each target, more than 40 000 events are generated with a randomized interval between 20 µs and 71 980 µs and distributed over a test duration of 24 minutes. For every event i, we calculate the difference of the timestamps provided by the two tracing services as $\Delta_{\text{abs}}[i] = t_{\text{datatrace}}[i] - t_{\text{gpio}}[i]$ where $t_{\text{datatrace}}[i]$ is derived from data trace data and therefore equal to $t[i]$ in (4.4) in Section 4.4.1.

Figure 4.6 shows the resulting distribution of the differences for each target. These measurements allow us to estimate the magnitude of the error $\epsilon[i]$ of the time correction in (4.4) in Section 4.4.1. The maximum offsets of different targets are within ±0.45 ms. The results of this test indicate that the implemented system achieves a time accuracy below 0.45 ms which is significantly lower than typical packet on-air times and satisfies our system requirements in Section 4.1.

4.5.2.2 Relative Time Accuracy

In this subsection, we measure the relative time accuracy of the implemented tracing system, i.e., the accuracy of measuring time intervals. This allows to compare the relative time accuracy of synchronized global timestamps to the unsynchronized local timestamps. As in Section 4.5.2.1, we use a test program on the DPP2 SX1262 target to generate events and we use the logic tracing service available on FlockLab 2 to obtain the ground truth. Again, an event consists of toggling a GPIO pin directly followed by writing to a global variable which is traced using the data trace service.

On the target, 60 000 events are generated with a fixed interval of 10 ms distributed over a test duration of 10 minutes. The events are used multiple times to form 50 000 different intervals with a length between 10 ms and 400 ms. For each pair of events (event i and $i + k$, with k corresponding to the interval length), we determine the interval I based on three different sources:

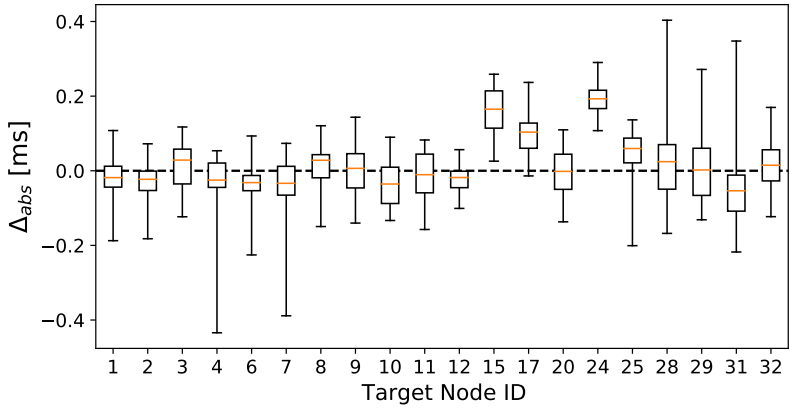


Figure 4.6: Distribution of time difference between timestamps from implemented data trace systems and the logic tracing of the FlockLab 2 testbed. For each target, the test contains more than 40 000 events with random intervals between 20 µs and 71 980 µs distributed over 24 minutes. The box extends from the lower to the upper quartile values of the time difference values, with a line at the median. The whiskers indicate the minimum and the maximum of the time differences. The IDs of the targets are not sequential because not all node IDs exist in the FlockLab 2 testbed and we only use observers with GNSS time synchronization for this measurement. The results show that the implemented system achieves an absolute time accuracy below 0.45 ms.

- Timestamps from the GPIO events from logic tracing (ground truth):
  $I_{\text{gpio}}[i,k] = t_{\text{gpio}}[i+k] - t_{\text{gpio}}[i]$
- Unsynchronized local timestamps from data tracing:
  $I_{\text{local}}[i,k] = t_s^{\text{ITM}}[i+k] \cdot m - t_s^{\text{ITM}}[i] \cdot m$
  As in Section 4.4.1, timestamp $t_s^{\text{ITM}}[i]$ is derived from the local clock on target s. Here, factor m is not obtained from regression but directly determined from the configuration (microcontroller clock frequency and prescaler of local timestamp counter) and is used to convert timer ticks to seconds.
- Synchronized global timestamps from data tracing:
  $I_{\text{global}}[i,k] = t_{\text{datatrace}}[i+k] - t_{\text{datatrace}}[i]$
  Again, timestamp $t_{\text{datatrace}}[i]$ is equal to $t[i]$ in (4.4) in Section 4.4.1.

Then the deviations of the interval based on the local timestamps $\Delta_{\text{rel,local}}[i,k] = I_{\text{local}}[i,k] - I_{\text{gpio}}[i,k]$ and the global timestamps

$\Delta_{\text{rel,global}}[i,k] = I_{\text{global}}[i,k] - I_{\text{gpio}}[i,k]$ are calculated.

Figure 4.7 shows the resulting distribution of the deviations. For this experiment, we show the data from a single observer. We performed the same experiment on multiple GNSS synchronized observers and obtained comparable results. The results show that the relative time accuracy when using the synchronized global timestamps instead of local timestamps is comparable for short intervals but significantly better for long intervals. The measurements of the interval using the local timestamps deviate by 16.5 ppm (median). This can be explained by the fact that local timestamps are not corrected for clock drift. As a consequence, the absolute value of the deviation of local timestamps increases linearly with increasing interval length.

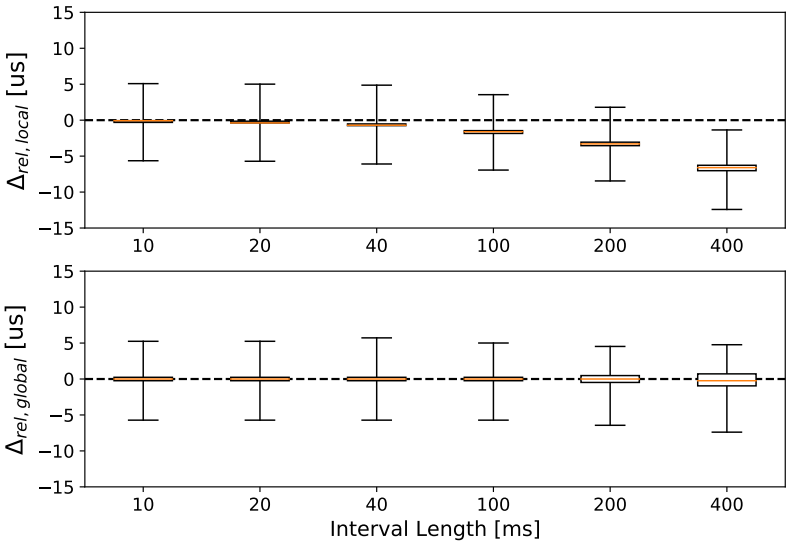


Figure 4.7: Deviation of time intervals measured with local and global timestamps from data tracing compared to measurements from the logic tracing service of the FlockLab 2 testbed. The distribution for each interval length contains 50 000 samples. The box extends from the lower to the upper quartile values of the time deviation values, with a line at the median. The whiskers indicate the minimum and the maximum of the time deviation. The results show that the relative time accuracy when using the synchronized global timestamps instead of local timestamps is comparable for short intervals and significantly better for long intervals.

4.5.2.3 Maximum Event Rate

As discussed in Section 4.4.4.3, different factors can limit the maximum event rate supported by the data tracing system. In this section, we experimentally determine the limit for the implementation of the tracing system in FlockLab 2 with the DPP2 SX1262 target. We differentiate two corner cases for the maximum supported event rate: (1) continuous generation of periodic events and (2) a burst of consecutive events without delay between individual events.

Continuous Periodic Events: For measuring the maximum event rate for continuous generation of periodic events, we use the same program for the target as in Section 4.5.2.1 but provide a set of fixed periods in the range from 10 to 20 µs to generate the events. For each period, we generate 10 000 events where each event consists of writing a single variable into memory. Data tracing is configured not to trace the program counter (PC). With logic tracing, we measure the actual period between events. From the obtained data trace data, i.e., the SWO packets, we count the number of generated SWO bytes and determine whether a delay or overflow was indicated. The results for one execution on a single observer are listed in Table 4.4. The test was repeated multiple times on different observers and the result did not differ significantly.

The results show that buffer overflows occur for event rates of 81.4 kHz and higher (period below 13 µs). For event rates between 58.1 kHz and 75.3 kHz, no buffer overflows occur but delays are indicated. This means that all tracing data could successfully be forwarded but timing information might not be sufficiently accurate. For event rates of 54.9 kHz and lower, continuous data tracing works without issues. In this case, 7 bytes per event are generated. For every event, a full data trace packet (5 bytes) and a medium-sized local timestamp packet (2 bytes) are required. For the cases where no overflows but delays are present, the number of bytes per event is lower. This can be explained with delayed local timestamp packets, which lead to cases where multiple data trace packets share the same timestamp packet. The SWO data rate appears to be limited to 3.2 Mbit/s. This represents 80 % of the configured SWO baud rate of 4 Mbaud. This is consistent with the overhead of a UART connection with one start, eight data, and one stop bit. Therefore, it seems likely that the SWO connection represents the bottleneck for the maximum event rate in case of continuous periodic events.

Burst of Events: To measure the maximum event rate in case of a burst of

Configured period [us]	10	11	12	13	14	15	16	17	18	19	20
Actual average period (logic tracing) [us]	10.3	11.3	12.3	13.3	14.3	15.2	16.2	17.2	18.2	19.2	20.2
Event rate [kHz]	97.0	88.5	81.4	75.3	70.1	65.6	61.6	58.1	54.9	52.1	49.6
Actual bytes per period (average)	4.09	4.52	4.86	5.31	5.71	6.10	6.49	6.73	7.00	7.00	7.00
Resulting SWO data rate [Mbit/s]	3.17	3.20	3.16	3.20	3.20	3.20	3.20	3.13	3.08	2.92	2.78
Buffer overflow indicated	yes	yes	yes	no	no	no	no	no	no	no	no
Delay indicated	yes	yes	yes	yes	yes	yes	yes	yes	no	no	no

Table 4.4: Results of the measurements to determine the maximum supported continuous event rate. The implemented system allows continuous tracing of periodic events with event rates up to 54.9 kHz.

events, we use a program that performs a specified number of consecutive write accesses. No explicit delay between write accesses is used, the delay is caused solely by the time it takes the microcontroller to execute the memory write instruction. The number of events within each burst is varied between 1 and 8. The data trace service is configured not to trace the program counter (PC). Again, the test was repeated multiple times on different observers and the results did not differ significantly.

The results, listed in Table 4.5, suggest that up to 3 consecutive events are supported without issues. 4 consecutive events are supported but delays occur when packets are forwarded inside the debug and trace sub-system. For 5 and more events, buffer overflows occurred and data is missing in the tracing data. The number of SWO bytes per burst in case of buffer overflows is not constant, which can be explained by a slightly varying timing behavior of different runs as the interval between events has an influence on the length of the local timestamp packet size. It can be estimated that the buffer which represents the bottleneck for the maximum event rate in case of bursts has a size of around 28 bytes.

Number of consecutive events	1	2	3	4	5	6	7	8
Data packets	1	2	3	4	4	4	4	4
Local TS packets	1	1	1	3	3	3	2	2
Number SWO bytes	9	14	19	27	28	28	27	27
Buffer overflow indicated	no	no	no	no	yes	yes	yes	yes
Delay indicated	no	no	no	yes	yes	yes	yes	yes

Table 4.5: Results of the measurements to determine the maximum supported burst event rate. The implemented tracing system allows to trace up to 3 consecutive events in a burst without delay between the events.

4.6 Case Study

We demonstrate the use of the proposed non-intrusive tracing system by applying it to an implementation of the eLWB networking protocol [SDFG+17] which is based on synchronous transmissions. In the first part (Section 4.6.2), we show the advantages of non-intrusive hardware assisted tracing by comparing the impact of the data tracing to traditional tracing via the serial interface. In the second part (Section 4.6.3), we demonstrate how our tracing system can be used for validating the eLWB

networking protocol running on a set of distributed wireless IoT devices. Before presenting the two parts of the case study, we briefly discuss the relevant aspects of the eLWB protocol and the used system setup.

4.6.1 Data Collection with eLWB

In this section, we describe the scenario used for the case study and we discuss the used networking protocol layers.

4.6.1.1 Data Collection Scenario and Realization with the FlockLab 2 Testbed

The scenario comprises a set of battery-powered wireless sensor nodes and a single sink node. In contrast to the sensor nodes, the sink node has access to an unrestricted power supply (e.g., connected to mains or supported by energy harvesting). The sensor nodes periodically sense a physical property of the environment (e.g., temperature, acoustic or seismic emissions, etc.) and determine whether the measurement is relevant, i.e., whether an event has been detected. Multi-hop wireless transmissions are used to transfer the detected events from the sensor nodes to the sink node, as not all sensor nodes have a direct link to the sink node. The sink node forwards the data to a database via an Internet connection. The goal of the system is to transfer as many of the detected events to the database as possible while minimizing energy consumption on the sensor nodes such that the system's lifetime is maximized. As a secondary objective, the latency of transferring the events should be minimized.

To realize the scenario with the FlockLab 2 testbed (see Section 4.5.1.2), we use 20 DPP2 SX1262 targets as used and described in Section 4.5.1.1. One of the targets (target 2) represents the sink, all other targets represent sensor nodes. Event value generation is emulated by obtaining a 2 byte value from a random number generator. Following each eLWB communication round, each node generates a random number of events that is uniformly distributed in the interval $[0, 2]$. Based on the FlockLab 2 topology (see Figure 3.5), the network topology consists of 17 nodes located in an office building and 3 outdoor remote rooftop nodes. The sink (node 2) is one of the nodes inside the office building. As a result, the link distance is inhomogeneous and in the range of 4 m to 2 km. The time synchronization of the wireless protocol is considered as part of the system under test and is therefore not used to timestamp the traced events. Instead, the out-of-band time synchronization provided by the FlockLab 2 testbed infrastructure is

used.

4.6.1.2 eLWB

For the case study, we use eLWB [SDFG+17] which represents a state-of-the-art wireless networking protocol designed for the energy-efficient data collection of event-based sensor data at a central sink. With eLWB, the sink organizes the data transfers in a time-division multiple access (TDMA) fashion. The state diagram of the eLWB protocol implementation used for the case study is depicted in Figure 4.8. Due to the TDMA based coordinated channel accesses, all participating nodes are supposed to be in the same state at the same time.

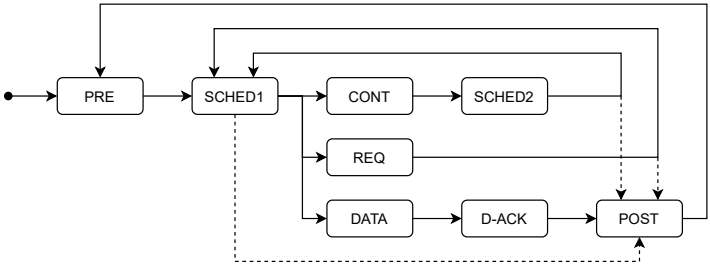


Figure 4.8: The state diagram of the eLWB implementation used for the case study. Solid lines represent normal state transitions, dashed lines correspond to transitions where nodes did not successfully receive a message or do not want to further participate in the eLWB round.

The communication is organized in communication rounds. In every eLWB round, the nodes prepare data to send in the PRE state and process potentially received information in the POST state. Each round consists of one or more sub-rounds which have a varying duration. Each sub-round consists of slots and is initiated by a schedule flood sent by the sink (SCHED1 state). A single Gloria flood (see Section 4.6.1.3) is used as the underlying communication primitive to send and/or receive a message in a slot. All nodes which receive the schedule flood follow the sequence of states as defined in the schedule sent by the sink. An eLWB round starts with a contention sub-round. During the contention slot (CONT state), the sink listens and sensor nodes can indicate their demand to transfer data. Nodes that are not yet registered use the message sent during the contention slot to register themselves. In our example, we circumvent this bootstrapping

mechanism and use a static registration for all sensor nodes. If the sink receives a transmission during the contention slot, it announces the request sub-round in a second schedule flood (SCHED2 state). In the request sub-round, a slot is exclusively assigned to each registered sensor node (REQ state). In every request slot, the corresponding sensor node can transfer its communication demand for the current eLWB round. In the following data sub-round, the sink node announces the data slots and the assigned sensor nodes (DATA state). Then, the sensor nodes use the assigned slots to transfer the data. Following the data sub-round, the sink confirms the receipt of the messages by sending an acknowledge flood (D-ACK state). If the sink does not receive any message during the contention slot or no request during the request sub-round, the following sub-rounds are skipped (dashed arrows in Figure 4.8). Analogously, the sensor nodes skip sub-rounds if they do not receive schedules from the sink node.

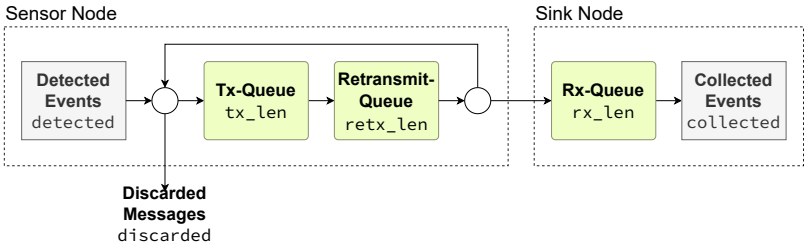


Figure 4.9: The flow of messages between event generation on the sensor node and collection on the sink node as implemented by the used eLWB data collection protocol variant.

The message flow from the sensor node to the collection on the sink node is depicted in Figure 4.9. Each sensor node has its own Tx- and Retransmit-queues. Data which needs to be forwarded to the sink is first added to the Tx-queue. When the data has been transmitted in a data slot, the data is moved from the Tx-queue to the Retransmit-queue. Based on the acknowledgement information received during the D-ACK state, the sensor node checks whether the data successfully arrived at the sink node. If this is the case, it removes the corresponding packets from the Retransmit-queue. Otherwise, it moves the packets from the Retransmit-queue back to the transmit queue. The sink node has an Rx-queue which temporarily stores all received messages before they are processed and forwarded.

For the case study, we set the eLWB a round period to 10 s and the maximum number of data slots in one round to 20 slots.

4.6.1.3 Gloria Floods

The eLWB communication protocol requires a lower-layer communication primitive which allows flooding messages to all nodes in the network. For this, we use Gloria, an optimized variant of Glossy [FZTS11] ported to the DPP2 SX1262 platform (see Section 2.5). Gloria allows to flood messages over multiple hops based on the concept of synchronous transmissions. In contrast to Glossy, in Gloria the timing of packet retransmissions within a flood is based on timer events rather than the directly preceding reception of a packet. Consequently, a node successfully receives a packet within a Gloria flood at most once and retransmits packets multiple times back-to-back. For the main part of the case study, we use the Long Range (LoRa) modulation with spreading factor SF5 and ensure that there is time for at least 7 slots within a Gloria flood. In our example, this leads to a data message duration of 13.1 ms and a data flood duration of 187.7 ms.

4.6.2 Non-Intrusive Tracing of a Time-Critical Synchronous Transmission Protocol

In the first part of the case study, we investigate the impact of serial logging compared to non-intrusive tracing using the data trace service. For this purpose, we log a variable in the time-critical part of the Gloria protocol implementation. The Gloria layer is responsible for the correct timing of transmit and receive radio operations such that transmissions of different nodes occur simultaneously with a precision in the order of 1 µs (see Section 2.5.1). If the execution on a subset of nodes is delayed, the transmissions of different nodes are no longer synchronous and the performance of the communication protocol potentially decreases. For this reason, non-intrusive tracing is essential in the context of time-critical distributed communication systems.

We instrument the code of one sensor node (target 7) to print the current slot index (`slot_index` variable) right before the transmission is initiated. On a second node (target 20), we use the data trace service to trace the `slot_index` variable. To obtain a clearer visualization, we used an frequency-shift keying (FSK) modulation with a bitrate of 125 kbps and a schedule packet duration of 2.0 ms instead of the LoRa modulation for this experiment.

In Figure 4.10, a visualization of the corresponding logic and data tracing is depicted. As expected, the transmissions of node 7 with `printf` code instrumentation are delayed. The delay is around 70 µs, as we use

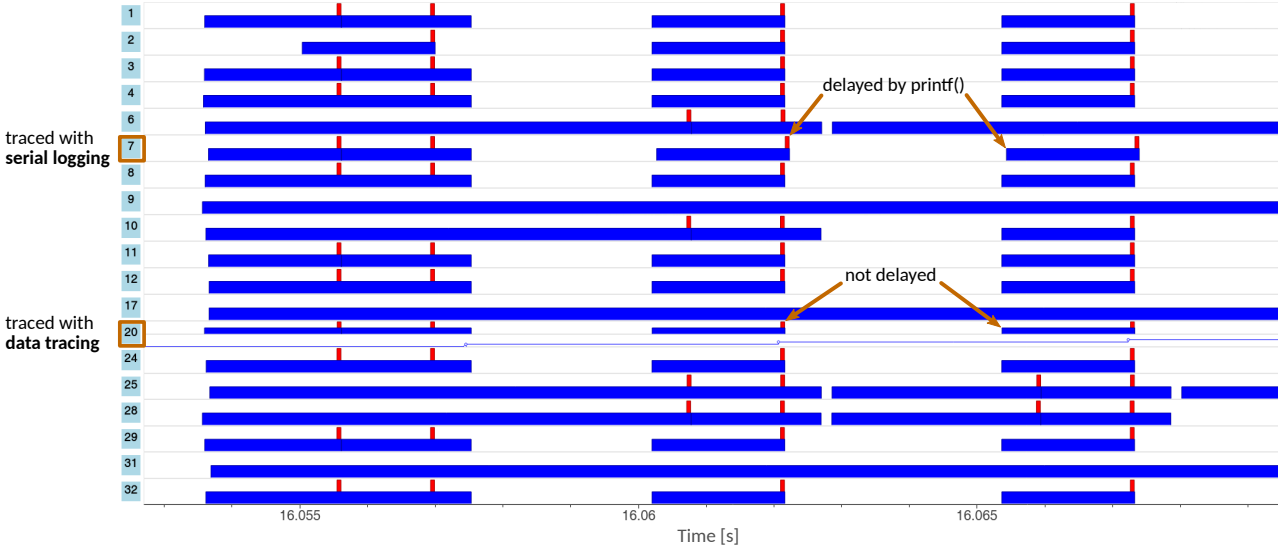


Figure 4.10: Logic tracing of a single Gloria flood (schedule flood). Short (red) pulses are interrupts from the SX1262 radio chip, long (blue) pulses are Rx and Tx operations indicated by GPIO events on the microcontroller. Serial logging used on node 7 is intrusive and delays the radio operation by 70 µs while data tracing used on node 20 is non-intrusive.

DMA transfer for serial printing. If blocking serial printing had been used, the delay would have been even larger (around 1060 µs for 44 characters at 460 800 baud). In both cases, the delay would strongly impact the synchronous activities and the performance of the communication scheme would decrease. With the presented non-intrusive data tracing, the time-critical execution of the synchronous transmissions is unaffected. In addition, serial logging only provides a snapshot of the variable at a certain point in the code. Data tracing on the other hand allows us to capture all updates of the variable as long as the event rate remains below the supported limit (see Section 4.5.2.3).

In many cases, a large portion of the time used for developing a program for a system of distributed wireless IoT devices is dedicated to timing-critical low-level details such as to correctly handle interrupts corresponding to external or internal events, see for example [PDM+20]. The exact timing behavior changes when code instrumentation is added. This means that for the implementation of time-critical parts either no code instrumentation can be used or the time-critical parts need to be re-adjusted after the removal of the code instrumentation used to verify the correct behavior. Since our proposed solution for distributed tracing enables the monitoring of the internal state without influencing the execution behavior, it can be used even for time-critical parts of the code.

4.6.3 Tracing Distributed State of the eLWB Networking Protocol

In the second part of the case study, we demonstrate how the distributed state of the eLWB protocol can be traced. By visualizing the distributed state of the network and by applying sanity checks, it is possible to detect faulty and verify correct behavior.

The goal of the validation is to ensure that the implementation of the protocol behaves as intended. Since eLWB involves distributed state, the validation requires detailed observations of all nodes. In order to compare state transitions of different nodes, the observations need to be tightly time synchronized. Furthermore, the observations of the distributed state must not influence the execution behavior to obtain meaningful experimental results which can be applied to a future deployed system. All the mentioned requirements (also see Section 4.1) are satisfied by the tracing system which we propose in this chapter.

Concretely, we want to verify the following three aspects of the eLWB

implementation:

(A1) All participating sensor nodes (nodes that received the previous eLWB schedule sent by the sink node) are in an allowed eLWB state (see Section 4.6.1.2). There are no lockups or illegal state transitions (e.g., SCHED1 → SCHED2).

(A2) All events from the sensor nodes are correctly forwarded through the message queues on the sensor and sink nodes. Exceptions are the messages that need to be discarded due to queue overflows on the sensor node.

(A3) Events arriving on the sink node do not differ from the events generated on the sensor nodes.

Variable	Description	Sensor Node	Sink Node
elwb_state	eLWB protocol state	✓	✓
detected	Detected data value	✓	
tx_len	#Msgs in Tx-queue	✓	
retx_len	#Msgs in Retransmit-queue	✓	
rx_len	#Msgs in Rx-queue		✓
collected	Collected data value		✓

Table 4.6: Overview of eLWB variables which are traced in the case study using the data trace service of FlockLab 2.

For the validation, we trace the variables listed in Table 4.6 on the sensor node and the sink node, respectively. The protocol state (elwb_state) is traced on all 20 nodes and is used to verify that all sensor nodes are in the same eLWB protocol state as the sink node (aspect A1). An exemplary visualization of one eLWB round used to verify proper state transitions is depicted in Figure 4.11. Most sensor nodes follow the schedule of the sink node and are therefore in the same state as the sink node. In the example, sensor node 15 does not receive the second schedule and transitions from SCHED2 directly to POST).

The variables detected and collected are used to verify that the events have been transferred unmodified (aspect A3). The sequence of values from events on the sensor nodes (detected) must match the sequence of messages with events received by the sink collected from the same node.

The three variables rx_len, retx_len, and tx_len indicate the fill level of the message queues and permit together with the variables detected and collected to keep track of the message flows (aspect A2). This involves the

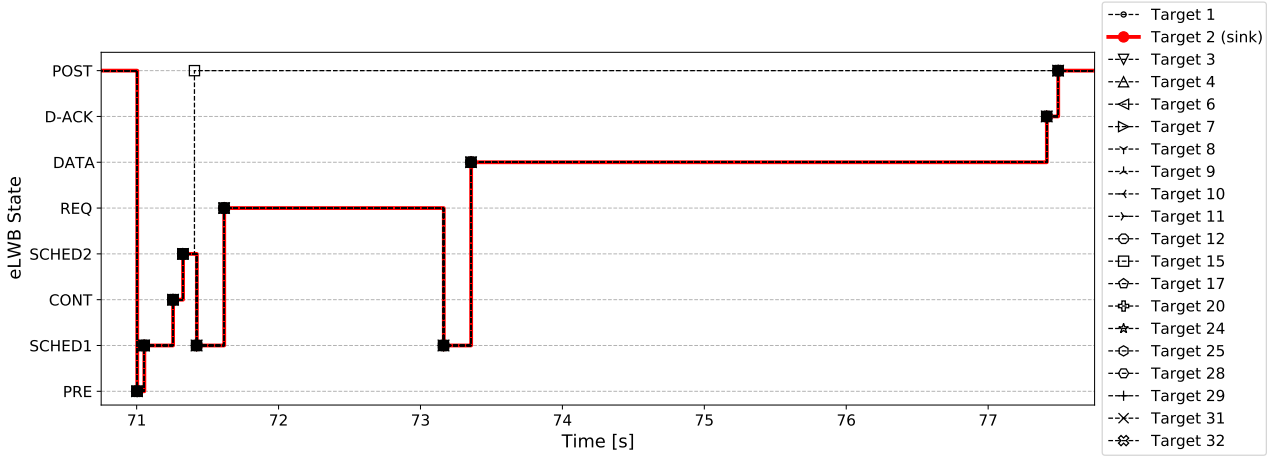


Figure 4.11: Visualization of the data trace of the `elwb_state` variable for one exemplary eLWB round executed by 20 nodes. All nodes which successfully received the first schedule are synchronized and follow the eLWB states of the sink node. In this example, node 15 did not successfully receive the second schedule during the SCHED2 phase and stops participating before the eLWB round ends.

number of events which are detected, transmitted, received, acknowledged, and finally collected on the sink (see Figure 4.9). The relations which must hold for every eLWB round can be summarized by (4.5), (4.6), and (4.7). The arrows indicate whether the elements added (↑) or removed (↓) from a queue are counted. S corresponds to the set of all sensor nodes. For every detected event (`detected`), a corresponding Tx-queue (`tx_len`) element must be added, except if the event is discarded. For every Retransmit-queue element which is removed and not added to the Tx-queue (and not discarded), a corresponding Rx-queue (`rx_len`) element must be added on the sink node. These two relations are expressed in (4.5). For every Tx-queue element which is removed, a corresponding Retransmit-queue element needs to be added (see (4.6)). Finally, all messages inserted into the Rx-queue must be included in the collected events (see (4.7)).

$$\underbrace{\sum_{k \in S} (\texttt{\#detected}_k + \texttt{retx\_len}_k \downarrow - \texttt{tx\_len}_k \uparrow - \texttt{\#discarded}_k)}_{= \texttt{rx\_len}' \uparrow} = \texttt{rx\_len} \uparrow \tag{4.5}$$

$$\texttt{tx\_len}_k \downarrow = \texttt{retx\_len}_k \uparrow \quad \forall k \in S \tag{4.6}$$

$$\texttt{rx\_len} \downarrow = \texttt{\#collected} \tag{4.7}$$

In the example test, the code for the sensor nodes contains an (artificial) bug. As a result, the messages from the Retransmit-queue are not added to the Tx-queue but are discarded when the sink node indicates no acknowledgement for the corresponding message. Since the message flow extends over several nodes (see (4.5)), it is necessary to be able to observe the change of the distributed state of the system in order to detect the problem. With the tracing system presented in this chapter, we can log all the associated distributed state changes in a non-intrusive fashion. The accurate time synchronization of the tracing system permits us to compare the state changes of different nodes.

An example visualization used for the validation of the message flow is depicted in Figure 4.12. The plots depict three eLWB round periods. The upper plot shows the number of messages contained in the queues. For better readability, only the fill levels of the Rx-queue on the sink and the Tx-queue of two sensor nodes are plotted. The lower plot of the figure shows the number of messages successfully transmitted from all sensor nodes to the sink node in each round. For every round, this value is determined twice: One time the value is calculated using the fill levels on all sensor nodes (left side of (4.5)), a second time the value is determined based on the fill level of the Rx-queue (right side of (4.5)). Under the assumption that

messages are not discarded and acknowledgment messages are successfully received, both of which apply in the presented example, the two values must be equal, i.e., $\texttt{rx\_len}' \uparrow \overset{!}{=} \texttt{rx\_len} \uparrow$, in every round. Since $\texttt{rx\_len}' \uparrow > \texttt{rx\_len} \uparrow$ in round 5, we can conclude that messages were not properly forwarded to the sink node and that the implementation is erroneous in this aspect. The accurate time synchronization of the distributed observations allows to investigate the exact cause and timing of the error in more detail. The evaluation shown in this section is only a simple example. More sophisticated methods, as discussed in Section 4.4.4.2, can be used to perform a more in-depth evaluation and could help to automatically detect invalid state transitions and identify corner cases in state machines.

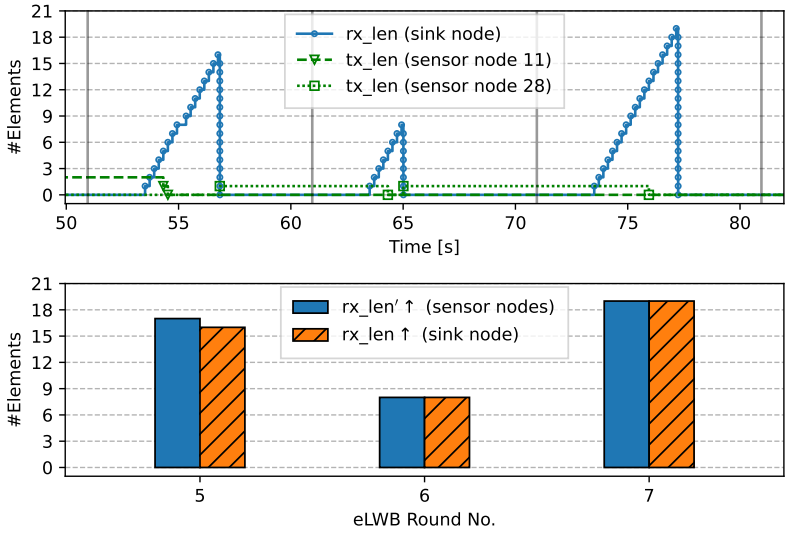


Figure 4.12: The upper plot visualizes the queue fill levels. To improve readability, only the fill levels of the Rx-queue on the sink and the Tx-queues of two sensor nodes are plotted. The lower plot shows the change of the number of Rx-queue elements per round as traced on the sink node ($\texttt{rx\_len} \uparrow$) and as derived based on the tracing of the sensor nodes ($\texttt{rx\_len}' \uparrow$). The inequality between the two values in eLWB round 5 indicates that the behavior is erroneous.

4.7 Summary

In this chapter, we presented a methodology to use the FlockLab 2 testbed architecture presented in Chapter 3 or similar architectures with commercially available debug probes and the native on-chip debug and trace hardware of modern microcontrollers to trace the program execution on all testbed nodes without altering the behavior of the program and without the need for explicit instrumentation of the code. For better testbed integration, we proposed to use automated interaction-free test execution. This makes it feasible to trace a large number of nodes and improves repeatability. Our proposed system supports event-based tracing of state changes compared to existing approaches that use polling to periodically fetch the state of the testbed nodes and therefore provide limited temporal resolution. In addition, we have designed a method for accurate time synchronization of the traces collected using multiple distributed debug probes that use different time bases. This allows the observations of the distributed set of nodes to be aligned on a common time axis and therefore allows the distributed state of networking protocols to be compared at the network scale.

We implemented the presented tracing methodology in the FlockLab 2 testbed and achieved a time synchronization accuracy in the order of sub-milliseconds. The implementation is available as open source [DFTW+21] and the tracing system can be used in the publicly available FlockLab 2 testbed[3]. In a case study using the FlockLab 2 implementation, we compared the proposed tracing method to traditional debugging via the serial interface and demonstrated how non-intrusive tracing can be successfully used for the development and verification of a networking protocol implementation.

[3] https://flocklab.ethz.ch/

5

Efficient Communication with LoRaWAN Class A

As discussed in Chapter 2, the energy required to transmit a bit with a long-range physical layer, such as with Long Range (LoRa), is typically significantly higher than the energy required with a low-power short-range physical layer, such as for example frequency-shift keying (FSK) or IEEE 802.15.4. Consequently, for low-power wide-area network (LPWAN) systems, the share of energy costs for communication is increased compared to the remaining shares, which include processing data or standby. As a result, for energy-constrained Internet of Things (IoT) devices that need to exchange data wirelessly over long distances, it is even more important to make communication as efficient as possible.

In this chapter, we examine the Long Range Wide Area Network (LoRaWAN) [LoR20] protocol which represents a widely used media access control (MAC) layer protocol for LPWAN applications and investigate how its efficiency can be increased under the constraints given by the system architecture and regulatory limitations. The main idea is to replace the random channel access by a reliable time-division multiple access (TDMA) based channel access. We propose two schemes that increase the reliability and analyze when the gain in efficiency due to the more reliable communication outweighs the additional overhead of the required time synchronization.

The basic idea of TDMA channel access combined with long-range communication to increase reliability is further evolved and forms a basis in the Long-Short-Range (LSR) protocol presented in Chapter 6, which

supports multi-hop communication.

5.1 Introduction

Wireless IoT devices are used for collecting data from the environment, industrial monitoring, tracking goods and more. The availability of cheap hardware components for wireless IoT devices and the emerging low-power, long-range communication hardware accelerate the deployment of many more IoT nodes. A large number of battery powered devices makes the frequent maintenance of individual devices infeasible. Therefore, low power requirements of IoT devices are of great importance. As communication accounts for an even larger proportion of energy costs in long-range wireless systems than in short-range systems, the communication should be as efficient as possible. In addition, with a large number of devices, it is important to use the shared limited resource of the frequency spectrum efficiently. For the same reason, this is also enforced by regulatory requirements, e.g. by a duty cycle limit.

In this chapter, we focus on LoRaWAN [LoR20], which is currently one of the most promising LPWAN technologies. In the LoRaWAN ecosystem, we differentiate between (1) the LoRa modulation, a physical (PHY) layer, and (2) LoRaWAN, a corresponding MAC layer. We focus on the Class A variant of LoRaWAN since it is best suited for low-power end-devices and is widely used. LoRaWAN Class A uses the pure ALOHA scheme to access the channel. This limits the channel utilization to a maximum of 18 %. In this chapter, we use LoRaWAN Class A as a basis and investigate alternative schemes that allow to use the channel more efficiently while minimizing the additional resource demand in terms of time-on-air.

The development of such a scheme involves the following two main challenges. LoRaWAN Class A does not provide a time synchronization scheme for the end-devices. Furthermore, LoRa messages exhibit a long time-on-air, which is problematic given the duty cycle limit of 1 %, which is enforced by law in Europe for the corresponding unlicensed 868 MHz frequency band.

Based on the analysis of the current channel access scheme and the limitations of the LoRaWAN MAC layer, we propose two schemes to increase the channel utilization in different use cases. Our analysis shows that the proposed schemes can provide more than 60 % throughput compared to 18 % throughput of the pure ALOHA scheme used in the

current specifications of LoRaWAN.

With this chapter, we make the following contributions:

- We identify concepts and strategies to extend LoRaWAN Class A to use the channel more efficiently than the original specification without spending a disproportional amount of resources.
- We propose two schemes, *TDMA* and *Burst*, which provide more throughput and are more efficient in specific use cases and which require only small modification of the LoRaWAN Class A layer.
- We evaluate the proposed schemes with calculations as well as an implementation on real hardware used for LoRaWAN systems.

We start with discussing related work in Section 5.2 and providing relevant background information about the LoRaWAN technology in Section 5.3. In Section 5.4, we discuss existing and present new protocol schemes suitable for LoRaWAN Class A. Then, we analyze and compare the considered protocol schemes in Section 5.5. Section 5.6 describes our implementation on real hardware and provides corresponding evaluation results. Finally, we summarize the chapter in Section 5.7.

5.2 Related Work

Adelantado et al. give an overview of the limits of LoRaWAN [AVTP+17]. They investigate the influence of the number of end-devices and also consider the duty cycle limit which is imposed by European regulations [Eur17]. Augustin et al. provide an overview of the LoRa modulation and the LoRaWAN MAC layer [AYCT16]. The study includes an analysis of the channel capacity of LoRaWAN. Vejlgaard et al. investigated the impact of interference on coverage and capacity of the LoRaWAN and the Sigfox system [VLN+17]. Morin et al. investigate the power consumption and the corresponding device lifetime of different IoT schemes including LoRaWAN [MMGD17]. Kim et al. propose a dual-channel scheme based on LoRaWAN to allow the data of different categories being delivered with different priorities [KK17]. Phung et al. analyze the packet delivery of LoRaWAN, including acknowledged and not acknowledged Class A transmissions as well as Class C transmissions [PTN+18]. Reynders et al. propose to use coarse-grained scheduling of transmission power, spreading factor (SF), and time in LoRaWAN networks [RWTP+18]. Beacons are used

for time synchronization. Polonelli et al. investigate the use of the slotted ALOHA protocol on top of LoRaWAN [PBB18]. In addition, they propose a simple request-reply based time synchronization, which is similar to the time synchronization used in this chapter.

To the best of our knowledge, the closest related work is the work of Gu et al. [GTLN18]. They propose a data network with separated control and data plane. For the control plane, they use LoRaWAN. The data plane is based on a multi-hop ZigBee network. Similar to our work, they add synchronization to LoRaWAN in order to use a TDMA based scheme. In contrast to the work of Gu et al., we do not use a separate control and data plane, we analyze the possibilities for different applications scenarios in general and in addition propose an *Burst* scheme that is advantageous in terms of aggregated throughput and channel use.

5.3 LoRaWAN Background

Two components of the LoRaWAN technology can be distinguished: (1) the LoRa modulation and (2) LoRaWAN. In this section, we will discuss the aspects of both layers which are relevant for this chapter and how to compute the time-on-air of a LoRa packet.

5.3.1 LoRa Modulation (PHY Layer)

The LoRa modulation is the physical layer. It is based on chirp spread spectrum (CSS) modulation. Similar to other spread spectrum technique such as direct-sequence spread spectrum (DSSS), this modulation uses a large spectral bandwidth to improve the robustness. A common bandwidth setting for LoRaWAN is 125 kHz. In addition, the payload information can be distributed over different amounts of time by selecting different spreading factors (SFs). Increasing the SF increases the time needed to send one byte, but also increases the probability of successful reception with a given signal-to-noise ratio (SNR) and therefore increases the feasible range, also see Section 2.3.2. This allows to trade throughput for range.

The LoRa modulation is typically used on the sub-1 GHz industrial, scientific and medical (ISM) frequency bands, e.g., the 915 MHz band in North- and South America or 868 MHz and 433 MHz bands in Europe. Those bands do not require a license and are therefore shared with a large range of other devices which use different modulation schemes. Depending

on the region, international regulations restrict the use of these bands in different ways. In Europe for example, there are limits on the transmit power and the duty cycle of each transmitting device is limited to 1 % for large parts of the 868 MHz band.

5.3.2 LoRaWAN (MAC Layer)

LoRaWAN [LoR20] specifies the MAC layer which is used together with the LoRa modulation. The specification comprises three different types of devices which form a star-of-star topology, also see Figure 5.11. At the core, there are one or multiple network servers (NSs), which implement the back-end with the interface to applications in the Internet. Multiple gateways (GWs) are connected to the network server by Internet Protocol (IP) connections. Each gateway connects multiple end-devices (EDs) via LoRa wireless links to the network server. Three default frequencies are used for end-devices to join a network, data transmissions and fallback. Additional frequencies can be configured manually.

The gateways simply forward messages from the end-devices to the network server and vice versa. A gateway can receive (uplink) messages on different frequencies and with different SFs simultaneously. Common gateways feature 8 frequency channels. In contrast to the multi-channel reception, the gateways usually only support to send on a single frequency and a single SF. Most of the available gateways do not support duplex mode, i.e., they cannot receive while transmitting. The network servers manage the connections to the end-devices, keep a state of each end-device, and remove duplicate messages originating from different gateways.

LoRaWAN messages are transmitted as physical (PHY) layer payload of a LoRa message. The message structure of data packets is depicted in Figure 5.1. MHDR is the header of the MAC message. It contains the message type and LoRaWAN version. The MAC payload contains the LoRaWAN frame. The MIC is the message integrity code, which is calculated over the MHDR and the MAC payload.

The LoRaWAN frame consists of a frame header (FHDR), frame port (FPort) and the frame payload, i.e., the application data. The FPort is a number which specifies which application the data is intended for. The frame header contains the device address (DevAddr), a frame control (FCtrl) field that contains information about the state of the connection, a frame counter (FCnt) value, and zero or more MAC layer commands (MAC commands) in the frame options (FOpts) field.

A LoRaWAN message with 50 bytes of frame payload needs a time-

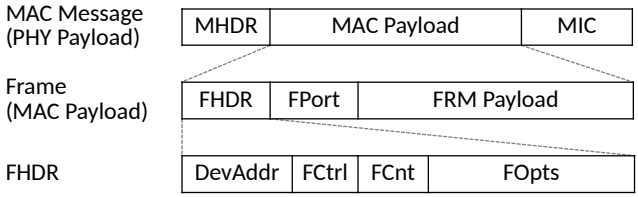


Figure 5.1: Structure of a LoRaWAN message.

on-air of 176 ms for spreading factor SF7 or 3548 ms for SF12. Accordingly, a device is allowed to send a maximum of 204 messages with SF7 or 10 messages with SF12 in one hour due to the duty cycle limit. This limitation holds for both end-devices and gateways.

The LoRaWAN protocol is divided into three classes. **Class A** provides simple unsynchronized two-way communication between end-devices and network server with focus on the uplink. Downlink messages can be sent only following an uplink message in the so called *receive windows*, which are depicted in Figure 5.2. Therefore, the downlink throughput and latency are severely limited. **Class B** enabled devices support all features of Class A. In addition, the gateways periodically send beacons to time synchronize the end-devices. This allows to schedule additional downlink receive windows. With **Class C**, the transceiver of each end-device is constantly turned on, i.e., the end-devices are either receiving or sending at any point in time.

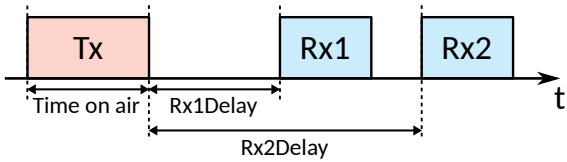


Figure 5.2: In LoRaWAN Class A end-devices only listen for packets during the defined receive windows.

The main focus of Class B is to increase the opportunities for communication from network server to end-devices. In our scenario, we are interested to improve the the efficiency of communication from end-devices to network server. Furthermore, initially the specifications for class B were experimental and the corresponding support in reference implementations therefore incomplete. While the situation regarding the

specifications and reference implementations improved in the meantime, there are still LoRaWAN devices that only support class A, as class B and class C are optional. Class C is only feasible for devices which have extensive power supply available, which is not the case for scenarios considered in this chapter. Because of these reasons, we focus on Class A in this chapter.

5.3.3 Time-on-air

In this chapter, we use (5.1) to calculate the time-on-air of a LoRaWAN packet. The expression is based on the time-on-air (toa()) function given in the SX1276 datasheet [Sem20]. It depends on the number of payload symbols (5.2) and the symbol duration (5.3).

$$\text{toa}(PL) = (n_{\text{preamble}} + 4.25 + n_{\text{pl}}) \cdot T_{\text{sym}} \tag{5.1}$$

$$n_{\text{pl}} = 8 + \max\left(\left\lceil \frac{8\cdot(\text{PL}+13)-4\text{SF}+28+16\text{CRC}-20\text{IH}}{4\cdot(\text{SF}-2\text{DE})} \right\rceil (\text{CR}+4), 0\right) \tag{5.2}$$

$$T_{\text{sym}} = \frac{2^{\text{SF}}}{\text{BW}} \tag{5.3}$$

PL is the number of frame (i.e. application) payload bytes. We adapted the formula such that it is valid for application layer payload by adding 13 bytes which correspond to the LoRaWAN overhead under the assumption of not sending any MAC commands in the FOpts field. SF is the spreading factor. For the calculations in this chapter, we always enable the cyclic redundancy check (CRC) (CRC = 1) and the header (IH = 0, i.e. implicit header is turned off). We do not make use of the low data rate optimization (DE = 0) and use a coding rate of 4/5 (CR = 1). For the remaining parameters we use the LoRaWAN default values for the EU868 ISM band according to the LoRaWAN standard [LoR20, LoR21]: the number of preamble symbols $n_{\text{preamble}} = 8$, and bandwidth of the LoRa modulation BW = 125 kHz (default for data rates (DRs) DR0 to DR5).

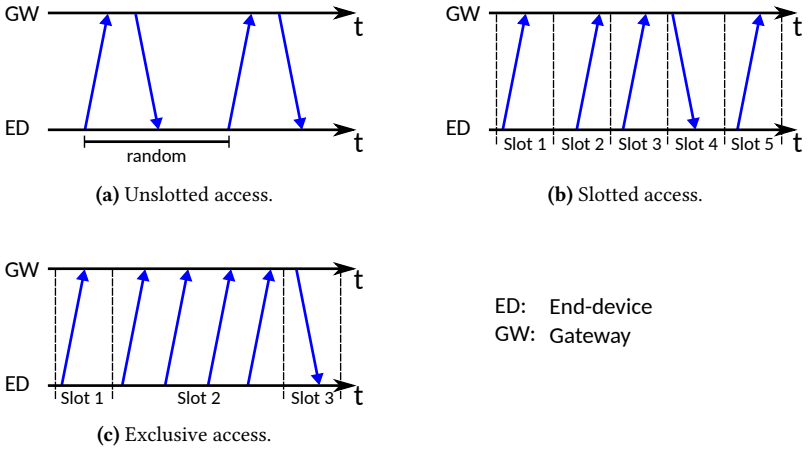
5.4 Protocol Design Space

In this section, we discuss all channel access and synchronization schemes that we consider and mention the restrictions implied by LoRaWAN

Class A. Then, we describe the considered schemes to increase the channel utilization, including our two proposed schemes *TDMA* and *Burst*.

5.4.1 Channel Access Schemes

Figure 5.3 provides an overview of the basic channel access schemes considered in this chapter. The message exchange between an end-device and the gateway consists of uplink messages that are directed from end-device to gateway and downlink messages from gateway to end-device. Note that in LoRaWAN Class A the time between uplink and downlink messages is fixed, see Figure 5.2. Depending on the downlink queue in the network server and whether the uplink requests an acknowledgment, the network server transmits a downlink in the receive window or not. In other words, not every uplink message is necessarily followed by a downlink message.



(a) Unslotted access.

(b) Slotted access.

(c) Exclusive access.

Figure 5.3: Basic channel access schemes.

Unslotted Access: End-devices can send messages anytime. Due to this uncoordinated access of the channel, there is a relatively high probability of colliding transmissions.

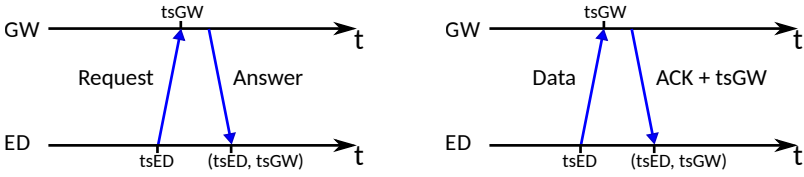
Slotted Access: The time is partitioned into slots of a fixed length. The end-devices are allowed to access the channel only at the beginning of a slot. This reduces the probability of collisions in comparison to the

unslotted protocol. However, the clocks of the end-devices and the network need to be synchronized. In addition, all messages need to fit into the same time slot length.

Exclusive Access: A scheduler determines a time-driven schedule, which defines the assignment of devices to time intervals and frequencies to each device. The resulting schedule, with mutually exclusive channel accesses, precludes message collisions. Consequently, end-devices need to be time-synchronized as well as receive and store information that determines the time interval and channel they are allowed to access.

5.4.2 Time Synchronization

The slotted access and exclusive access scheme require end-devices to be time synchronized. Two options that we consider are shown in Figure 5.4. The selection is based on the opportunities of the LoRaWAN Class A standard, i.e., sending beacons from the gateways is not possible as downlink messages can only be sent as answer to a previous uplink message.



(a) Request. **(b)** Piggy-back.

Figure 5.4: Basic synchronization schemes.

Request: An end-device and the network server exchange dedicated messages via a gateway to synchronize the clock of the end-device.

Piggy-Back: The request for a time-synchronization is part of a regular data message, which is sent to the network server via a gateway. Synchronization information, like a timestamp, is then part of a downlink message sent from the network server via a gateway to the end-device. This may be either an acknowledgment or a data downlink message.

5.4.3 Considered Transmission Schemes

We will study the following four protocols. They are selected as they represent different extreme solutions of a wide spectrum of possible schemes. Many generalizations and combinations of these four basic schemes are possible.

Pure ALOHA: This protocol has been proposed in [Abr70] and uses the unslotted access scheme and therefore does not require synchronization.

Slotted ALOHA: This scheme has been proposed in [Rob75]. It combines piggy-back synchronization with a partition of the channel in fixed time slots. Depending on the drift of the clock of end-devices, not all uplink messages need be answered by a synchronization message, e.g., a timestamp from the network server. These synchronization messages from the gateway are also subject to message collision.

TDMA: We propose the *TDMA* scheme which takes into account the LoRaWAN Class A specifics. In our scheme, the end-devices repeatedly send data packets (denoted as DD) according to a fixed TDMA schedule, see Figure 5.5. In order to achieve synchronization between the clocks of the end-devices, special data messages (denoted as DA) are answered by acknowledgment messages (denoted by A) from the network server, which include synchronization information. The required rate of the synchronization messages depends on the clock-drift of the end-devices and the required time synchronization accuracy. There are several obvious options on determining such a TDMA schedule and sending it to the end-devices via downlink messages. Figure 5.5 depicts one possible schedule. A more detailed description is provided in Section 5.5.2.3.

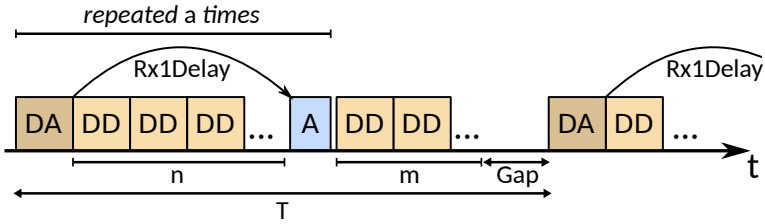


Figure 5.5: Example of the proposed *TDMA* scheme. All end-devices are periodically time-synchronized but only a fraction of them is time-synchronized within each period (T). In this example, we use $a = 1$.

Burst: For applications without critical latency demands, we propose the novel *Burst* scheme depicted in Figure 5.6. Multiple messages are aggregated and sent together in a burst. The scheme uses two different channels, i.e. two different frequencies: The *request channel* for coordinating the transmission of burst data messages and to perform time synchronization and the *burst channel* for transmitting bursts of uplink data messages. In order to send a burst, the end-device first needs to request a burst transmission slot from the network server by sending a burst request (BR) message on the request channel. A scheduler on the network server determines which end-device is allowed to send in which time slots. The burst is interleaved with empty slots due to limitations of the LoRaWAN Class A specifications, also see Section 5.5.2.4.

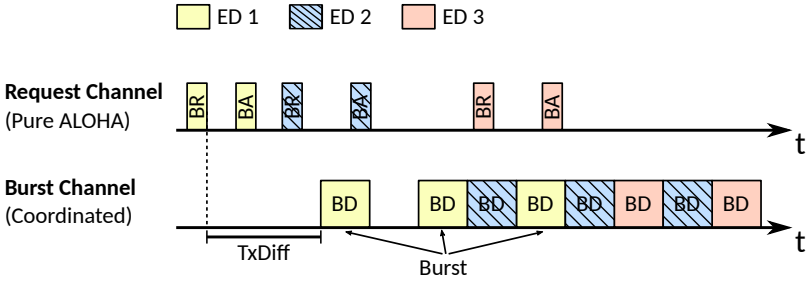


Figure 5.6: Example of the *Burst* scheme with on-demand synchronization. In this example, there are 3 end-devices and bursts consisting of 3 messages ($n_B = 3$ and $n_G = 0$) are used.

5.5 Theoretical Analysis

In this section, we describe and analyze the considered schemes in order to increase the channel utilization. First, we start with defining the model and metrics to compare the different schemes.

5.5.1 Model and Metrics

5.5.1.1 Basic Model Assumptions

We suppose that our communication scenario consists of N end-devices and K gateways. We further assume that all end-devices and gateways can

reach any other end-device or gateway directly. In general, we assume that every end-device connected to the network generates data of size D with a fixed period T.

Time-On-Air: The time-on-air of a LoRa transmission can be calculated, see Section 5.3.3. We use t_D for the time-on-air of any data uplink packet and t_A for any non-data packet (this includes requests, answer, synchronization, and coordination packets).

Clock Drift: After synchronization, the clocks in the network server and the end-devices will drift apart. We denote the time difference between previous time synchronization and the time when the clock of the end-device is accessed as Δt. The maximal absolute value of the time difference between the end-device and the network server at this point in time is then modeled as

$$\tau(\Delta t) = \tau_0 + \Delta t \cdot \tau_1 \tag{5.4}$$

where τ_0 is the synchronization error due to the synchronization protocol between the network server and the end-device and τ_1 denotes the clock-drift of the end-device. In order to account for the clock inaccuracy, we expand the actual time-on-air t_D and t_A by a safety margin and define the expanded time for data packets as $s_D = t_D + 2 \cdot \tau(\Delta t_{\max})$ where $\Delta t_{\max}$ is the maximum time between time synchronization updates. We define s_A accordingly.

Duty Cycle Limit: The LoRaWAN standard limits the aggregated time-on-air of a device, i.e. the accumulated time a device is transmitting. In this chapter, we focus on LoRaWAN EU868 and therefore the European regulations (ETSI EN 300 220-1 [Eur17]) apply which enforce a the duty cycle limit of $L = 0.01$ for each device. This also applies to the gateways. According to the regulations, the time interval which is considered to evaluate the adherence to the duty cycle limit is $I = 3600\,\text{s}$. This constraint in terms of duty cycle L and measurement interval I strongly restricts the design space of efficient LoRaWAN based protocols.

5.5.1.2 Metrics for Comparison

In the following, we list the four metrics we are interested in to compare the considered transmission schemes.

The **success probability** P_{succ} is defined as the probability that an attempt of an end-device to transmit a data packet to the network server is successful, i.e., there is no colliding transmission.

We define the **throughput** S as the average accumulated time of successful data message transmissions from all end-devices relative to the total time. If there are in total $M_{\text{succ}}(\Delta t)$ successful message transmissions from any end-device in a time interval of length Δt, then

$$S = \lim_{\Delta t \to \infty} \frac{M_{\text{succ}}(\Delta t) \cdot t_D}{\Delta t} \tag{5.5}$$

where t_D is the time-on-air for transmitting a data message.

The **device time utilization** W_d for a specific device d is the average accumulated time the device is (successfully or unsuccessfully) transmitting relative to the total time. For example, if there are $M_d(\Delta t)$ message transmissions from this specific end-device in a time interval of length Δt, then

$$W_d = \lim_{\Delta t \to \infty} \frac{M_d(\Delta t) \cdot t_D}{\Delta t} \tag{5.6}$$

where t_D is the time-on-air for transmitting a data message. We denote the device time utilization of an end-device by W_{ED} and the the device time utilization of an gateway by W_{GW}.

The **send efficiency** E is the average accumulated time all end-devices are transmitting data packets which are successfully received relative to the total time all devices are transmitting. For example, if there are $M_{\text{succ}}(\Delta t)$ successful data transmissions, $M_{\text{unsucc}}(\Delta t)$ unsuccessful transmissions, and $M_{\text{sync}}(\Delta t)$ synchronization messages in a time interval of length Δt, then

$$E = \lim_{\Delta t \to \infty} \frac{M_{\text{succ}} \cdot t_D}{(M_{\text{succ}}(\Delta t) + M_{\text{unsucc}}(\Delta t)) \cdot t_D + M_{\text{sync}}(\Delta t) \cdot t_A} \tag{5.7}$$

5.5.2 Analysis of the Considered Transmission Schemes

In this section, we explain the considered transmission schemes in more detail and provide the corresponding performance analysis.

5.5.2.1 Pure ALOHA Scheme

The LoRaWAN Class A scheme uses the pure, i.e., unslotted, ALOHA scheme for channel access. If the channel access attempts from end-devices are assumed to be Poisson distributed with an average of G accesses per packet time, then the performance metrics are known to be [Abr70]

$$S = G \cdot \mathrm{e}^{-2G} \qquad P_{\text{succ}} = \mathrm{e}^{-2G} \tag{5.8}$$

In (5.8), S corresponds to our definition of throughput and G is the number of access attempts per packet time (access rate). The maximum throughput is about $S_{\max} = 18.4\,\%$, which is achieved for $G = 1/2$. This leads to a corresponding success probability of about $P_{\text{succ,max}} = 0.37$.

In other words, even a low maximal throughput comes with a low success probability. One intuitive measure to increase the success probability is to use positive acknowledgment and retransmission. But this leads to additional channel accesses due to the retransmitted packets and the acknowledgment for each correctly requested packet. Finally, acknowledgment packets are also subject to collisions. This means we can increase the success probability only by reducing the channel accesses, i.e., by reducing the throughput.

In reality, the duty cycle limit restricts the number of feasible operating points of the pure ALOHA scheme. In our analysis, we want to achieve a throughput S defined by N, t_D and T. We use (5.8) to calculate $G = \frac{N \cdot t_{\text{Tx}}}{T}$ from S in (5.9). The actual average time-on-air t_{Tx} for each pure ALOHA end-device to generate a total throughput of S is always larger then the theoretical send time required without collisions ($t_{\text{Tx}} > t_D$).

$$S = \frac{N \cdot t_D}{T} \quad \overset{G(S)}{\Rightarrow} \quad G \overset{!}{=} \frac{N \cdot t_{\text{Tx}}}{T} \tag{5.9}$$

With this, we can determine the end-device time utilization W_{ED} (see (5.10)). The device time utilization of the gateway is 0 since no transmissions are acknowledged and no synchronization is performed. The send efficiency is determined by $E = \frac{S}{G}$.

$$W_{\text{ED}} = \frac{t_{\text{Tx}}}{T} = \frac{G}{N} \overset{!}{\leq} L \qquad W_{\text{GW}} = 0 \overset{!}{\leq} L \tag{5.10}$$

Due to retransmissions, a delay has to be accounted for this scheme.

5.5.2.2 Slotted ALOHA Scheme

As in the case of pure ALOHA, the throughput analysis for slotted ALOHA is well known and established [Rob75]:

$$S = G \cdot \mathrm{e}^{-G} \qquad P_{\text{succ}} = \mathrm{e}^{-G} \tag{5.11}$$

If we assume perfect synchronization with no overhead and no interference on the frequency band (i.e. all transmitted packets are received successfully if they are not overlapping in time), the maximum throughput is about $S_{\max} = 36.8\,\%$, which is achieved for $G = 1$, where G is the access rate. This would be an improvement by a factor of 2 in comparison to the pure ALOHA protocol in terms of maximally achievable throughput. The success probability remains unchanged with about $P_{\text{succ,max}} = 36.8\,\%$ for this operating point.

But as we know, the scheme requires the end-devices to be synchronized. Therefore, the actual throughput of a slotted ALOHA system is lower than S and the success probability is lower than P_{succ}. In addition, the rate of acknowledgment packets transmitted by the gateways is limited by the duty cycle limit L. This fact constrains the possible design space for slotted ALOHA further.

5.5.2.3 TDMA Scheme

We propose the *TDMA* scheme which is depicted in Figure 5.5. The end-devices send a data packet (DD or DA) of fixed size D according to a time-division multiple access (TDMA) schedule. The schedule repeats periodically with period T. In each period, all N end-devices send exactly one data packet. Only a small number $a < N$ of all transmissions are acknowledged (DA packets) in each period in order to keep the overhead low and to comply with the duty cycle limit. The period between synchronization of a particular end-device is $\frac{T \cdot N}{a}$ on average. The acknowledgment A is used to transfer a timestamp from the gateway to the end-device. The acknowledgment slots of multiple periods are evenly distributed to the participating end-devices such that the end-devices are alternately synchronized. In Figure 5.5, the case for $a = 1$ is depicted. All devices, including the gateways, send on a single frequency.

The LoRaWAN standard specifies the `Rx1Delay` time between the end of the data packet and the corresponding acknowledgment in order to allow the server to react to the received message and to transmit a reply, see Figure 5.2. During this interval, $n = \left\lfloor \frac{\texttt{Rx1Delay}}{s_D} \right\rfloor$ not acknowledged transmissions are scheduled. This sequence of DA, DD ..., A is repeated a times. The rest of the period is used to schedule m not acknowledged transmissions from the remaining $m = N - a \cdot (n + 1)$ end-devices.

Due to the ETSI duty cycle limit, not all combinations of (N, t_D, T) are possible. The relation for the *TDMA* scheme between payload size D

(indirectly given as time-on-air with safety margin s_D) and the period T is given by (5.12). The application payload and the time-on-air of a packet is linked by the *toa*() function described in Section 5.3.3.

The relation of the components of one *TDMA* schedule period is given in (5.12). A non-zero gap (*Gap*) allows the scheme to be suitable for combinations of parameters which do not exactly fill the *TDMA* schedule. Gap is determined by parameters in (5.12) (N, t_D, t_A and a).

The constraints due to the duty cycle limit based on the device time utilization are given in (5.13). The device time utilization due to downlink messages from the network server can be distributed to K gateways.

$$T = (\texttt{Rx1Delay} + s_A) \cdot a + \left(N - \left\lfloor \frac{\texttt{Rx1Delay}}{s_D} \right\rfloor \cdot a \right) \cdot s_D + \text{Gap} \tag{5.12}$$

$$W_{\text{ED}} = \frac{t_D}{T} \overset{!}{\leq} L \qquad W_{\text{GW}} = \frac{a \cdot t_A}{T} \overset{!}{\leq} L \cdot K \tag{5.13}$$

L corresponds to the duty cycle limit, see Section 5.5.1.1. The send efficiency is $E = \frac{N \cdot t_D}{N \cdot t_D + a \cdot t_A}$.

In the case of no packet loss, the maximum time a clock of an end-device is not synchronized is $\Delta t_{\text{max}} = \left\lceil \frac{N \cdot T}{a} \right\rceil$ and the *TDMA* scheme provides a success probability of $P_{\text{succ}} = 1$ since none of the transmissions can be overlapping due to the TDMA schedule.

5.5.2.4 Burst Scheme

As a second scheme, we propose the *Burst* scheme, which is depicted in Figure 5.6. In contrast to the *TDMA* scheme, in the *Burst* scheme the end-devices are not continuously synchronized. The end-device synchronizes their clock to the network server before sending a burst. Synchronizing for sending a single packet would implicate a large overhead. Therefore, in general the end-devices aggregate data and send multiple packets bundled together in a *burst*.

The proposed scheme uses two different channels: The *request channel* for handling requests and synchronization and the *burst channel* for transmitting the bursts. The request channel is uncoordinated and uses pure ALOHA, whereas the burst channel is coordinated by a scheduler on the network server. There is no explicit acknowledgment but it would be possible to acknowledge the reception of the complete or partial burst in the following transmission from the the gateway to the end-device (BA).

In order to send a burst, the end-device first needs to request a burst transmission and obtain synchronization by sending a burst request (BR) message on the request channel to the network server. The network server maintains a schedule of all scheduled transmissions. With a burst answer (BA) transmitted on the request channel, the network server sends a timestamp for synchronization and the information when the end-device is allowed to send the individual packets (BD) of the burst. Then the end-devices synchronizes its clock and stands by for sending the burst packets in the designated slots.

The scheduler running on the network server makes sure that no burst can collide on the burst channel. If there are no suitable slots available in the following time interval of length Δt which defines the safety margin $\tau(\Delta t)$, the network server can deny an access to the burst channel. In this case, the end-device tries to request the burst channel again after a backoff time. The backoff time is increased exponentially with every denial. This prevents overloading the request channel if there are simultaneous requests from many nodes.

The LoRaWAN specifications require the end-device to wait with sending a new message until the receive window of the previous message has passed no matter whether the end-device sent a confirmed or unconfirmed message. For this reason, a single burst transmission, which consists of multiple packets, needs to be sent as individual packets with gaps of at least `Rx1Delay` in between.

For the analysis of the performance of this scheme, we assume that the average period between two burst transmissions of a single device is T. The accumulated duration of a burst t_{Burst} and the relation to the burst period T are given in (5.14). The device time utilization and the corresponding limits are given in (5.16). Since the messages are sent in bursts, they are not evenly distributed over time, we need an additional constraint, (5.15), that ensures that the absolute maximal transmitting time within $I = 3600\,\text{s}$ is not exceeded.

$$\begin{aligned} t_{\text{Burst}} &= (n_B + n_G) \cdot s_D \\ T &= N \cdot t_{\text{Burst}} \end{aligned} \tag{5.14}$$

$$n_B \cdot t_D + t_A \overset{!}{\leq} L \cdot I \tag{5.15}$$

$$W_{\text{ED}} = \frac{n_B \cdot t_D + t_A}{T} \overset{!}{\leq} L \qquad W_{\text{GW}} = \frac{t_A}{t_{\text{Burst}}} \overset{!}{\leq} L \cdot K \tag{5.16}$$

n_B indicates the number of LoRaWAN data packets that are sent in a single burst, $n_G \geq 0$ is the number of slots which are unused (gap) following a burst transmission. The unused slots are, in some cases, necessary to keep the time utilization (W_{GW}) below the allowed limit L on the side of the gateway. In the *Burst* scheme, the number of nodes is not restricted by the duty cycle limit, as it is the case with the *TDMA* scheme, but depends on the size of the bursts. The success probability of sending the bursts is $P_{\text{succ}} = 1$ since the synchronization and scheduling of the network server makes sure that no transmissions overlap. The probability for a collision on the request channel is supposed to be small since the aggregated time-on-air for acknowledgments by the gateway is limited to $L = 1\,\%$. The throughput of the entire scheme is determined as $S = \frac{n_B \cdot t_D}{t_{\text{Burst}} + 2 \cdot t_A}$ and the send efficiency is given by $E = \frac{n_B \cdot t_D}{n_B \cdot t_D + 2 \cdot t_A}$.

5.5.2.5 Comparison of Slotted ALOHA with TDMA

If we compare slotted ALOHA and the *TDMA* scheme, it is obvious that the minimum synchronization overhead is the same (unless out-of-band synchronization mechanisms are used). In both schemes, all nodes need to be continuously synchronized within the required precision such that packets can be sent inside a time slot. The only difference in overhead is the assignment of each node to slots which exists in the *TDMA* scheme but not in the slotted ALOHA scheme. However, this assignment can be pre-configured (e.g. based on the node's ID), which means that the overhead of the slot assignment of the *TDMA* scheme is negligible. Since the *TDMA* scheme provides a significantly better success probability ($P_{\text{succ}} = 1$) it is always beneficial to use *TDMA* instead of slotted ALOHA. Because of this reason, we omit the slotted ALOHA scheme in our calculations that follow next.

5.5.3 Numerical Comparison

In order to compare the three remaining considered schemes, we calculate the previously described metrics.

5.5.3.1 Calculation Model

We numerically evaluate the access schemes described in Section 5.5 to obtain the metrics and the feasibility of the considered schemes at different design points. The most important input and output quantities of the calculation are listed in Table 5.1.

Input	
D	Application payload size
T	Period with which an end-device sends the application payload
N	Number of participating end-devices
K	Number of gateways
–	LoRaWAN and LoRa modulation parameters

Output	
–	Feasibility
E	Send efficiency
W_{ED}, W_{GW}	Device time utilization
P_{succ}	Probability of a transmission to arrive

Table 5.1: Input and output of the calculations.

The main input of each scheme consists of three parameters, application payload size D in bytes, period between transmissions T of a single end-device in seconds and the number of end-devices N. These three parameters describe the requested throughput. By evaluating the equations of the considered schemes, we determine which areas of the design space are feasible and which scheme provides the best send efficiency.

Further parameters for the calculations are the spreading factor (SF), plus further LoRa modulation parameters, which are described in Section 5.3.2 and Section 5.3.3 and are kept constant for all calculations. In addition, there are parameters which are relevant only for a subset of the schemes: The number of gateways K which is relevant for the *TDMA* and *Burst* scheme, the number of acknowledged transmission a in a *TDMA* period. Another parameter for the calculation are clock offset and drift values which are relevant for the *TDMA* and *Burst* scheme. Based on measurements in Section 5.6.2.1, we assume a clock offset τ_0 of 15 ms and a clock drift of 20 ppm. The application payload size D is a discrete parameter by definition. In order to keep the calculations tractable, we sampled the application payload D with a step size of 10 bytes and the period T with a step size of 20 s. In our model, we consider it infeasible for an end-device to send more data at a time than can fit in a single data packet, except in the *Burst* scheme. This limits the application payload size D in the *TDMA* scheme to the maximum payload size of a LoRaWAN packet (which is 222 bytes for SF7).

5.5.3.2 Feasibility and Efficiency of the Schemes

First, we investigate which scheme is most efficient for a given throughput defined by N, D, and T. In Figure 5.7 we show an exemplary plot for SF7 and $N = 100$ with 1 gateway. The non-white areas represent the feasible combinations of input parameters. The shading of the area indicates which of the scheme has the best efficiency E. Please note that our assumptions of the model limit the feasibility.

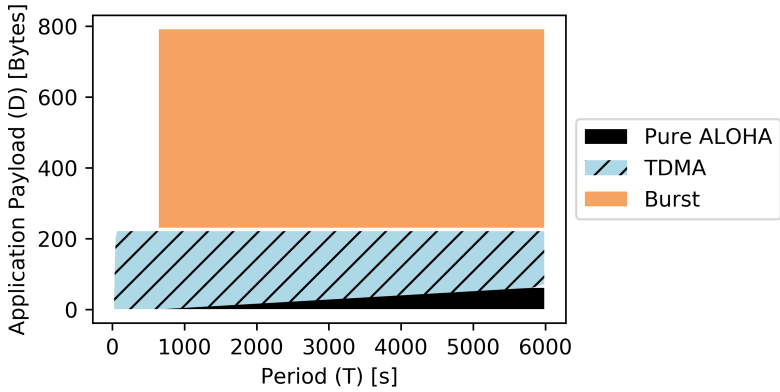


Figure 5.7: Feasible combinations of period T and application payload D (fixed SF=7 and N=100). The color/hatching indicates the scheme with highest send efficiency E.

The calculations show that the *TDMA* scheme is feasible and efficient in a large range of periods and payload sizes. For very large periods and small payload size, i.e., low requested throughput, the pure ALOHA scheme has highest send efficiency since the overhead for the continuous synchronization is large compared to the data that should be transmitted. The *Burst* scheme is not feasible for small periods. For lower periods, the device time on the end-device W_{ED} would exceed the duty cycle limit. For SF7 even for very short periods, the *TDMA* scheme is feasible and provides higher send efficiency. This is expected since the pure ALOHA needs approximately 4 failed transmission to send 1 successful message in the best case of $G = 0.5$. The *TDMA* scheme only needs to send synchronization messages for a fraction of all nodes in each period. The calculations for different SFs and different number of end-devices N yield similar insights.

5.5.3.3 Maximum Throughput

In this section, we investigate the maximum achievable throughput of each scheme.

In Figure 5.8, the throughput S for different number of end-devices N in the case of using only a single gateway is depicted. The *TDMA* scheme can provide significantly higher throughput values compared to the pure ALOHA scheme. However, the maximum throughput for the *Burst* scheme is comparable to the pure ALOHA scheme when using only one gateway.

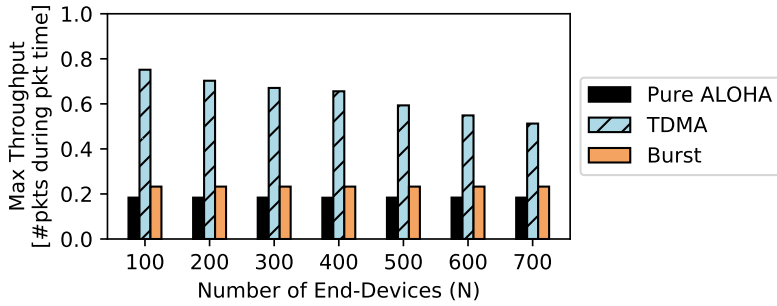


Figure 5.8: Maximum throughput S for different combinations of period T and application payload D in relation to N (SF=7 and 1 gateway).

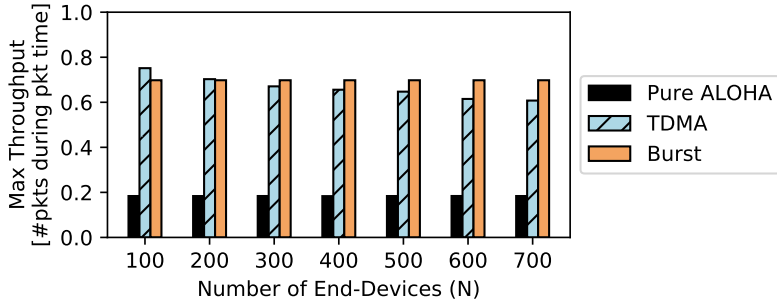


Figure 5.9: Maximum throughput S for different combinations of period T and application payload D in relation to N (SF=7 and 3 gateways).

If we increase the number of available gateways, we can increase the maximum throughput of the *TDMA* and the *Burst* scheme. In Figure 5.9,

we show the maximum throughput in case of 3 gateways. The calculations show that especially the *Burst* scheme profits from the additional gateways. For the *TDMA* scheme it helps only if the number of end-devices is large. With 3 gateways and SF=7 and less than 300 end-devices, the schemes can provide a throughput up to 70 % whereas the pure ALOHA scheme only provides up to 18 % throughout.

As shown in Figure 5.9, the maximum *TDMA* throughput decreases with increasing number of end-devices whereas the maximum *Burst* throughput is constant. For the *TDMA* scheme the overhead to keep nodes synchronized grows with the number of devices. For the *Burst* scheme, the overhead of handling burst requests on the gateway can be kept constant for any N by increasing the period. However, the transferred data per end-device decreases with increasing number of end-devices.

5.5.4 Selection of Transmission Scheme

Finally, we provide guidelines that help to select an appropriate transmission scheme based on the analysis in the previous sections. An overview in the form of a decision tree is given in Figure 5.10.

In the pure ALOHA scheme, the throughput is constrained by the collisions ($P_{\text{succ}} \leq \frac{1}{2\text{e}} = 18.4\,\%$) which are accepted in order to not require a time synchronization. The *TDMA* and *Burst* scheme have a success probability of $P_{\text{succ}} = 1$ but comprise an overhead due to the necessary time synchronization. The collisions of pure ALOHA and the synchronization overhead of the *TDMA* and *Burst* schemes reduce the send efficiency. In addition, lack of synchronization increases the necessary safety margin $\tau(\Delta t)$. A larger safety margin influences whether a requested throughput can be achieved by a certain scheme but does not influence the send efficiency directly.

If an application requests only a very low throughput (i.e. T large and D small), the pure ALOHA scheme is suitable and provides good send efficiency $E = S/G$, which is large compared to the send efficiency of the synchronized schemes.

For larger requested throughputs there are two cases. If the period T should be small, continuous synchronization of all end-devices is reasonable and therefore the *TDMA* scheme is suitable. If the period T should be large, continuous synchronization is not necessarily reasonable and therefore the *Burst* scheme is more suitable. In certain cases of small period and small packet size, the *TDMA* scheme is not feasible due to the

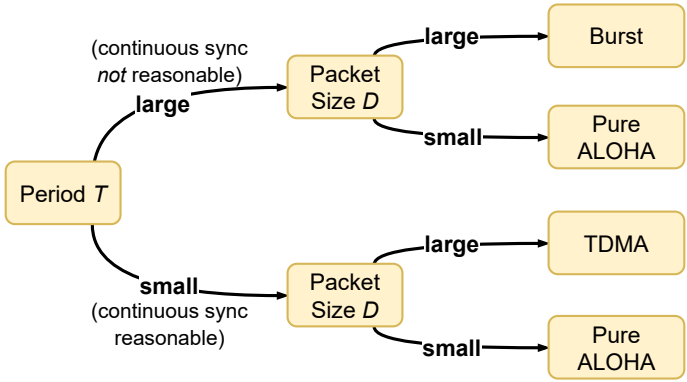


Figure 5.10: Decision tree for selecting channel access / synchronization scheme.

duty cycle limitations. This holds especially for larger SFs. In this case, the pure ALOHA is the only option.

5.6 Implementation on Real Hardware

In this section, we demonstrate that the implementation of the two proposed schemes on real hardware is feasible and only requires minor changes of the LoRaWAN network layer. The end-device implementation is based on the framework of Polonelli et al. [PBB18], which implements the basic mechanism for time synchronization. We extended the framework by a mechanism to enforce frequency channels, using the history of synchronization messages for time synchronization, and the implementation of our proposed *TDMA* and *Burst* schemes.

5.6.1 Implementation Details

5.6.1.1 Setup

An overview of the setup which is used for the evaluation in this chapter is depicted in Figure 5.11.

Network Server: For the network server, we use ChirpStack [chi] (version 0.22.0), which is an open-source implementation of the corresponding

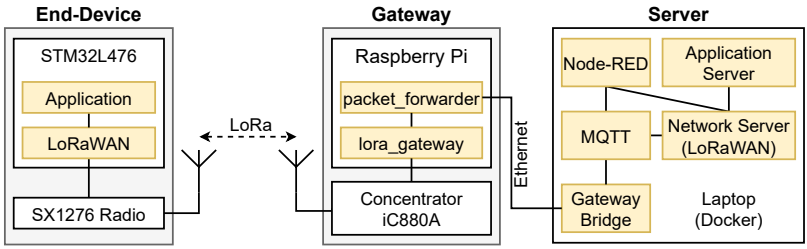


Figure 5.11: Setup used for the implementation of the proposed schemes for LoRaWAN Class A networks.

LoRaWAN specifications. We run the different components of the ChirpStack inside different Docker containers on a Lenovo ThinkPad T460s laptop (Intel Core i7-6600U, 2.60 GHz, 19 GiB RAM). The ChirpStack project consists of 3 main parts: the *gateway bridge*, the *network server* block and the *application server*. The gateway bridge is responsible for communicating with the gateway. The network server block implements the LoRaWAN MAC-layer on the server side. The application server manages different user applications and provides a web-interface. The received and transmitted messages are exchanged via the open-source Eclipse Mosquitto MQTT broker. Furthermore, we use a Node-RED container to implement network flows.

Gateway: Our LoRaWAN gateway consists of a Raspberry Pi 2 and the iC880A concentrator. This setup supports the simultaneous reception of messages with all SFs (SF7 - SF12) on 8 different frequencies. The gateway is connected to the laptop via Ethernet. The software running on the gateway is the *lora_gateway* and *packet_forwarder* from the Lora-net reference project on GitHub [lor].

End-Device: Each end-device consists of an STMicroelectronics STM32L476RG microcontroller development board (Nucleo-L476RG) combined with an mbed SX1276 868 MHz LoRa shield. The software running on the end-devices is based on *LoRaMAC-node* from the *Lora-net* reference project on GitHub [lor] and the framework of Polonelli et al. [PBB18].

5.6.1.2 Time Synchronization

Our proposed schemes require synchronized clocks across end-devices and network server. By default this is not supported by the LoRaWAN Class A MAC layer. Therefore, we add a custom time synchronization to LoRaWAN. The implementation of the time synchronization is based on the implementation described in the work of Polonelli et al. [PBB18] and is similar to the scheme used by Gu et al. [GTLN18].

The implemented synchronization scheme is depicted in Figure 5.4. This scheme is based on a pair of corresponding timestamps. The end-device takes a local timestamp right before sending a synchronization request. The gateway receives this request and immediately takes the corresponding global timestamp (tsGW). This timestamp is then sent to the end-device in the following synchronization answer message. One such pair of timestamps forms a synchronization point and can be used to calculate the offset between the local and global clocks. This offset is used in the *Burst* scheme. For the *TDMA* scheme, this procedure is repeated, which leads to multiple synchronization points. From multiple synchronization points, the clock offset and drift are calculated, which is then be used to convert the local timestamp on the end-device to the global time of the network.

5.6.1.3 Enforcing Frequencies

By default, LoRaWAN end-devices choose pseudo-randomly one of the 3 default frequencies for sending LoRa packets. However, our proposed schemes use a fixed assignment of channels to end-devices. Therefore, we enforce the end-devices to use the assigned frequency. In our implementation, the end-device uses the MAC Information Base (MIB) interface of the LoRaWAN MAC layer on the end-device to change the channel mask such that all but one configured LoRaWAN channels are disabled at any point in time. We would like to note that in case such a change at the end-devices should not be possible, there is alternatively the possibility to initiate the adaptation of the channel mask on the network server by means of the LoRaWAN `LinkADRReq` command.

5.6.1.4 TDMA Implementation

The *TDMA* scheme, proposed in Section 5.5.2.3, requires an implementation of a bootstrap mechanism to obtain synchronization and to send

messages according to a schedule. The implementation uses the piggy-back synchronization method described in Section 5.6.1.2. The end-device recalculates the offset and drift value each time a new synchronization point is obtained. The obtained values are used to provide timestamps from a virtual clock, which is synchronized to the clock of the network server due to the synchronization scheme. After boot up and joining the LoRaWAN network, the virtual clock is not yet synchronized and therefore the end-device requests the first synchronization point by sending a dedicated synchronization request message on one of the 3 default LoRaWAN channels. This bootstrap mode is at the same time used for fallback in case the end-device loses synchronization in the case of not receiving a synchronization answer.

In general, a scheduler on the network server can send a schedule to the end-device in the synchronization answer. For the implementation of our experiments, we define the TDMA schedule statically. The individual end-devices determine their send slots based on the current time and the end-device's ID, similar to the implementation of Gu et al. [GTLN18]. After the bootstrap phase or after sending a data packet, an end-device obtains the current timestamp and together with the ID it determines the time to transmit the next packet and the transmission type (DA, DD or A). In the time between the transmissions, the end-device is put into sleep mode. Timers that are based on the virtual clock are configured to wake the end-device up and send the transmission scheduled by the TDMA schedule.

5.6.1.5 Burst Implementation

The implementation of the *Burst* scheme requires a logic on the end-device to send messages on the assigned channel at the assigned point in time, a scheduler on the network server, and a mechanism to prevent transmitting while receiving a burst data message on the same gateway.

When the end-device wants to send a burst, a synchronization procedure as described in Section 5.6.1.2 is initiated. In addition to the timestamp, the network server sends the time difference (`TxDiff`) between timestamp and start time of the burst, see Figure 5.6. The end-device then uses the timestamp to synchronize the virtual clock, configures a timer to wake up when the burst should start, and goes into sleep mode. The end-device then alternately sends packets and sleeps in between until all packets of the burst are transmitted. After completing a burst, the end-device sleeps until the start of the next burst. In the implementation for our experiments, we use a configurable interval between sending bursts. The length of this

interval is randomized to ensure that the pattern of burst requests changes over time.

The scheduler is implemented as stateful Node-RED flow function. The schedule consists of time slots of size s_D, which are assigned to end-devices which request to send a burst. With this, the scheduler guarantees that the allocated burst patterns do not overlap.

Most of the commercially available LoRaWAN gateways (including the one used for the experiments in this chapter) are not capable of full-duplex operation, i.e., they cannot transmit while receiving. This causes the gateway to miss burst data packets if it sends an answer to a burst request at the same time. We mitigate this problem, by not answering burst requests, if the burst answer would overlap with scheduled burst data packet. Please note, that this problem has no influence on the throughput if two or more gateways are used. For the case of two gateways for example, one gateway can be configured to only receive and the other one to only transmit. The gateways then can periodically switch roles in order to distribute the accumulated transmit time such that the device send time utilization of the gateways is kept below the duty cycle limit.

5.6.2 Evaluation

We perform experiments to verify that our implementation behaves as expected and to demonstrate that our proposed schemes work using a LoRaWAN setup with real hardware.

5.6.2.1 Synchronization Accuracy

First, we examine whether our assumptions of the clock accuracy for the calculations are in accordance with the values measured on real hardware. For this, the time synchronization accuracy which can be achieved with the implementation is measured.

We measure the offset between transmissions of two periodically time-synchronized end-devices located in the same room. For this, both end-devices send messages with a period of 10 s. In order not to interfere with each other, one of the end-devices sends messages with a configured offset of 5 s relative to the other end-device. Every transmission from an end-device is answered with a reply by the network server such that each end-device can synchronize its clock every 10 s.

An experiment with 200 transmissions from each of the two nodes over a time interval of more than 30 minutes with SF7 has been conducted.

Based on the timestamps recorded when receiving the messages on the gateway, we determined that the offset is in the range ±0.0123 s. From these measurements we conclude that $\tau_0 = 15$ ms, which we use in the calculation in Section 5.5.3, is a good upper bound for the offset between two synchronized end-devices.

5.6.2.2 Evaluation of the TDMA Scheme

In order to verify that the implementation performs as expected from the calculations, we run the *TDMA* scheme and measure the packet delivery ratio (PDR). For the evaluation, we determined a feasible configuration which satisfies all the constraints in Section 5.5.2.3 with $N = 64$, an application payload of $D = 50$ bytes, a period $T = 30$ s, and $a = 1$ acknowledged transmission per period. This configuration would ideally lead to a throughput of $S = 25.2\,\%$ with 1 gateway.

We perform measurements with the mentioned configuration with 8 end-devices and 1 gateway, which are located in the same room. The slots for the remaining 56 end-devices are unused, i.e., no device sends anything during this time. In order to verify the synchronization implementation, we artificially increase in our experiment the synchronization rate such that in every period exactly one of the 8 participating end-devices sends a synchronization request. The measurements includes 253 periods, which corresponds to 127 minutes or 2024 uplink packets.

The resulting PDR for each end-device is plotted in Figure 5.12. The mean PDR over all 8 end-devices is 99.75 %. A probable cause for not receiving all packets successfully is interference from other LoRa transmissions on the same frequency. With this, we demonstrate that an implementation of the described TDMA scheme on top of LoRaWAN is feasible.

5.6.2.3 Evaluation of the Burst Scheme

In an evaluation experiment, we run the *Burst* scheme implementation to verify that it performs as expected from the calculations. For this, we use the following configuration: $N = 8$, $n_B = 4$, $T = 60.02$ s (95 slots), SF7, $D = 222$ bytes. This leads to a throughput of $S = 31.0\,\%$. This configuration uses an artificially decreased period T in order to test the system under high load. This means, a single physical device represents multiple virtual devices.

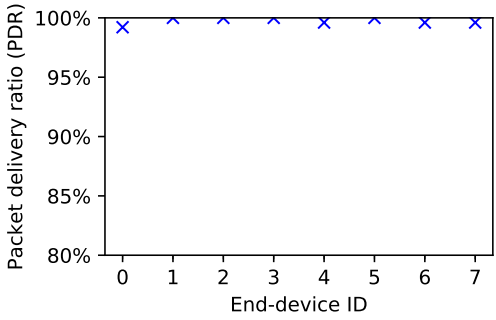


Figure 5.12: Packet delivery ratio (PDR) of the *TDMA* evaluation.

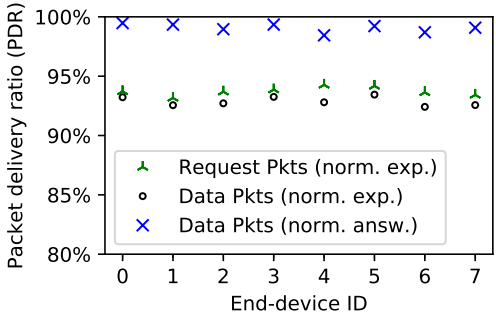


Figure 5.13: Packet delivery ratio (PDR) of the *Burst* evaluation.

In our experiments, we use the setup of Section 5.6.2.2 with 8 end-devices and 1 gateway. The use of 1 gateway means that some burst requests are not answered the first time because the gateway does not send a burst answer while receiving a burst data packet. Therefore, we expect that the amount of positive answers to the the burst requests is lower than 100 %.

The results of the measurements are shown in Figure 5.13. The star shows the number of burst requests, the circle shows the number of burst data messages of each end-device. Both values are normalized by the expected number of burst requests or data messages, which is calculated using the values of the configuration. The average number of expected bursts from each end-device is 204.96 (absolute number). On average, the

network server received 93.92 % of the expected burst requests and 93.04 % of the expected burst data packets. As expected, both values are lower than the PDR of the *TDMA* evaluation since a significant amount of burst requests is not answered by the network server to allow proper reception of burst data messages. The cross represents the number of received burst data packets normalized by the number of unique burst requests that have been answered by the network server. On average the network server received 99.07 % of all burst data packets which followed an answered burst request. This number is comparable with the PDR of the *TDMA* evaluation. In summary, the results show that the implementation of the *Burst* scheme is feasible.

5.7 Summary

In this chapter, we presented two schemes that use TDMA channel access to increase the reliability and throughput of LoRaWAN class A communication. With our schemes we achieved a reliability close to 100 % whereas the reliability of the default approach using pure ALOHA provides significantly lower reliability, especially in cases with high communication demand. Based on calculations of throughput metrics of the investigated schemes, we showed that the improved reliability outweighs in many application scenarios the cost of time synchronization which is required for the TDMA. With our schemes, the throughput can be increased to more than 60 % compared to a maximum throughput of 18 % provided by the pure ALOHA scheme. In addition, our proposed *Burst* scheme scales to a virtually unlimited number of participating nodes because no periodic time synchronization is required.

With an implementation on real hardware, we demonstrate the feasibility of our proposed schemes and demonstrate that only minor changes need to be made to the default configuration of LoRaWAN.

In the following chapter, we build on the idea of using TDMA based channel access in conjunction with long-range communication and present a multi-hop protocol which uses multiple modulations and allows energy-efficient communication in networks with inhomogeneous link characteristics.

6

Energy-Efficient Multi-Modulation Communication for Inhomogeneous Wireless Links

For many application scenarios for low-power wide-area network (LPWAN) systems, the application dictates the node locations. This often leads to networks containing links with a wide range of path losses. In addition, in many cases not all nodes are able to reach the gateway in a single hop. As discussed in Chapter 1, recent advances in technology make low-power Internet of Things (IoT) platforms possible that support long- and short-range communication schemes and therefore provide a new degree of freedom. However, it is not straightforward how to make use of the long-range transmission capability in multi-hop networks to increase the coverage while maximizing the system lifetime. If an energy-efficient modulation scheme with a low link budget is used, nodes that are reachable via high-path-loss links only cannot communicate. Using a more energy-demanding long-range modulation allows connecting more nodes but would be inefficient for nodes that are easily reachable via low-path-loss links. Combining multiple modulations is challenging as low-power radios usually only support the use of a single modulation at a time.

In this chapter, we present the Long-Short-Range (LSR) protocol which supports low-power multi-hop communication using multiple modulations and is energy-efficient with networks with inhomogeneous link characteristics. LSR is based on the modulation capabilities and the timing analysis of the platform presented in Chapter 2 and builds on the Gloria synchronous

transmissions flooding protocol also presented in Chapter 2. One key property of LSR is to ensure that any node for which a path to the host exists can communicate with the host. To reduce the inherent redundancy of long-range modulations, we integrate two optimization methods into LSR. This includes a scheme to determine the connectivity graph of the network during regular data communication without adding significant overhead. This allows the LSR protocol to continuously adapt itself to changing radio frequency (RF) conditions. In simulations, we show that LSR allows reducing power consumption significantly for many scenarios when compared to a state-of-the-art multi-hop communication protocol using a single long-range modulation. With an implementation on the hardware from Chapter 2 and using the testbed infrastructure introduced in Chapter 3 and Chapter 4, we demonstrate the applicability of the proposed LSR protocol.

6.1 Introduction

Motivation. As discussed in Chapter 1, a large number of objects is connected to the Internet every year. In many use cases, it is most convenient to connect the objects using low-power wireless communication, as such a setup does not require a lot of infrastructure such as wired power or wired network connections at the object's location. Many real-world deployments consist of a set of low-power nodes and a gateway that connects the nodes to the Internet. Often, the placement of the nodes is heavily influenced by positions that are of interest to the application. An example is the placement of IoT nodes inside a building at spots that are relevant for building automation, e.g. close to windows, next to doorways, or at work spaces. Another example, illustrated in Figure 6.1, is a system to measure seismic events on mountains for long-term monitoring where sensor nodes are placed at locations that allow good observation of geological events. Such networks usually exhibit inhomogeneous link characteristics between nodes, i.e. the path loss of the links is diverse. Some nodes are for example placed close to the gateway and therefore the link to the gateway exhibits a very small path loss. Other nodes are placed at distances of multiple kilometers or with obstacles in-between and therefore exhibit a large path loss. In many cases, there are nodes that cannot directly communicate with the gateway (e.g. behind a mountain ridge) and therefore require relay nodes to forward their data to the gateway.

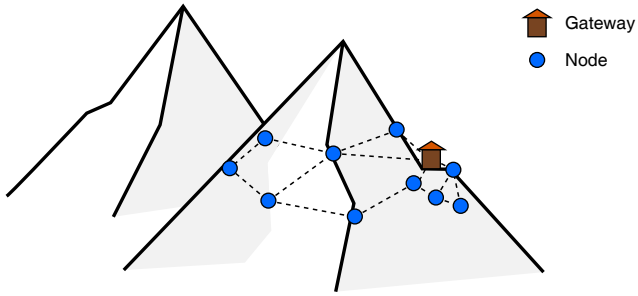


Figure 6.1: Example scenario with a single unconstrained gateway (house) and multiple resource-constrained nodes (circles) which can reach the gateway via a single or multiple hops over links with a diverse set of path losses.

Problem Statement. In this chapter, we focus on mixed-range LPWAN application scenarios with the following characteristics:

- The network consists of a set of resource-constrained *nodes*, i.e., they have limited available energy and limited processing capabilities, and a single unconstrained *gateway*. The gateway is assumed to be connected to the Internet. We focus on networks with a single gateway. Extensions to multi-sink networks are briefly discussed in Section 6.5.3.
- The characteristics of the links in the network are inhomogeneous, i.e. the network contains links with high and low path losses.
- The network may contain nodes that can directly communicate with the gateway over a single hop as well as nodes that can reach the gateway only via multiple hops.
- The RF characteristics between nodes may change over time but only with a limited frequency of occurrence.
- The application requires an energy-efficient transfer of data to the Internet and accurate time stamping on the nodes to annotate observations. However, it does not require real-time communication with near-zero latency. In other words, the reliable and energy-efficient transfer of data has higher priority than a low latency.

The goal of the system is to enable communication with the gateway for every node that has a path to the gateway. For nodes that cannot directly reach the gateway, the system should support sending their messages using

other nodes as relays. As the nodes are resource-constrained, the system should be energy-efficient even if the network consists of inhomogeneous links.

One way to tackle the problem of inhomogeneous links is to add additional relay nodes to the network. This allows the nodes to communicate over homogeneous links with a small range of path losses. However, the addition of nodes requires additional resources (hardware, time, and labor for the installation), and determining a suitable placement is in many cases not trivial. Therefore, we do not consider this approach to be generally applicable. Another approach to make communication in a network with inhomogeneous links more energy-efficiently is to apply adaptive transmit power schemes. However, the achievable range of link budgets when varying the transmit power of low-power radios is usually not sufficient to cover the wide range of path losses in the considered LPWAN application scenarios. Recent advances in low-power radio technology enable radios with a significantly larger link budget. For example Long Range (LoRa) radios allow transmissions with a large spreading factor (SF) providing a reduced data rate but allow bridging multiple kilometers or larger obstacles compared to traditional radios which typically permit a range of 10s of meters. As discussed in detail in Section 2.3.2, low-power transceivers are available that support multiple modulation schemes and therefore cover a wide range of link budgets. In this chapter, we focus on the use of different modulations in order to cover all nodes in a network with inhomogeneous links. We refer to a *modulation* as a 2-element tuple which contains: (1) the type of physical representation of symbols, for example LoRa or frequency-shift keying (FSK), and (2) the rate at which the symbols are transmitted (e.g. the spreading factor for LoRa or the raw bit rate for FSK). We use *Mod0* as a symbol for a long-range modulation, and *Mod1* as a symbol for a short-range modulation.

Challenges. As discussed, a network with a wide range of path losses between nodes requires the use of multiple modulations to allow for energy-efficient communication. Since the low-power radios of nodes only support the use of a single radio configuration (e.g. a single LoRa SF) at a time, this makes it necessary to switch between radio configurations in a coordinated and reliable fashion. The use of a different modulation results in an altered link budget and a changed transmission duration, and therefore in a different energy consumption. All these tradeoffs need to be taken into account when orchestrating transmissions in the network. The requirement to support multi-hop transmissions in the network further

increases complexity. In order to maximize the system lifetime, it is necessary to distribute energy consumption evenly to all the nodes. In a static RF environment, it would be sufficient to measure the environment in the beginning of the deployment, calculate the optimal configuration, and distribute it to all the nodes. However, in many realistic scenarios, the RF conditions change over time. In this case, the task of finding and distributing an efficient configuration and communication pattern is no longer a one-shot problem but needs to be repeated continuously.

Straightforward Approaches. Based on the examples in Figure 6.2, we explain why straightforward approaches do not meet the requirements of our scenario. **Single-hop** schemes, such as Long Range Wide Area Network (LoRaWAN) [LoR20], do not work since nodes that have no direct connection to the gateway cannot communicate with it. In example (a), nodes A and B can reach the gateway via a single hop. However, node C is only reachable via a multi-hop path and therefore cannot communicate with the gateway. **Two separate systems**, one for long-range links and another one for short-range links, would be another straightforward approach. In example (b), node A only participates using the short-range modulation. In this case, nodes B and C can communicate with the gateway as node C can communicate via node B. Nevertheless, this approach also does not fulfill the requirements of our scenario as there are cases, e.g. example (c), where nodes are unable to communicate with the gateway even though there exists a path between the node and the gateway. In example (c), nodes B and C would have a path to the gateway via node A. But since node A participates only in the short-range system, nodes B and C cannot reach the gateway. **Single modulation** schemes, where only a single modulation is used to connect all nodes to the gateway work for all scenarios. However, in order to cover all nodes with links with a wide range of path losses, a long-range modulation is required. This is inefficient in terms of energy and radio-on time for nodes that have a short-range path to the host. In example (d), node A would send its data using the long-range modulation even though a data transfer using short-range modulation would be possible.

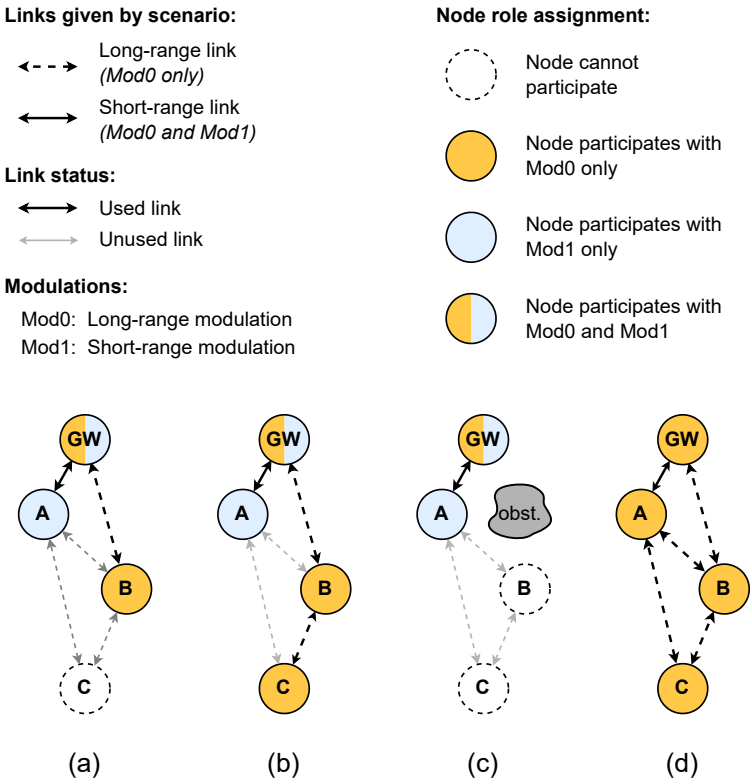


Figure 6.2: Straightforward solutions do not meet the requirements of our scenario: a single-hop scheme does not support communication between the host and nodes reachable only via a multi-hop path (a), a scheme with two separate systems for long- and short-range communication works in certain cases (b) but does not work if links are not available even though there would be a path to the gateway for every node (c), a single-modulation scheme works but is inefficient in many cases (d).

Proposed Scheme. In this chapter, we present the Long-Short-Range (LSR) protocol, a multi-hop communication protocol that runs on a network of energy-constrained nodes with homogeneous hardware. As a basic communication primitive, Glossy-like [FZTS11] flooding is used. On a higher layer, the protocol uses rounds of time-division multiple access (TDMA) slots (similar to Sparkle [YRH14] or Low-Power Wireless Bus (LWB) [FZMT12]). In order to prevent redundancy due to the overlap of long- and short-range coverage, we propose a set of optimizations. Since nodes transmit concurrently during a flood, it is not straightforward to determine which path a received message has taken. Therefore, we propose a non-destructive probing method based on delayed retransmissions within floods to measure the connectivity graph without significant overhead during normal data floods. The obtained connectivity information can serve as a basis for various network optimization methods. LSR is characterized by the following advantages: (1) LSR covers any multi-hop scenario; as long as there is a path between a node and the host, the LSR protocol will use it. (2) LSR does not only perform a one-shot adaptation when the system starts but dynamically and continuously adapts to changes of the topology. The flooding-based approach allows for fast reconnecting to react to significant topology changes. (3) The energy consumption can be balanced by distributing energy-intensive operations evenly across multiple nodes which increases the system lifetime (see Section 6.6.2.3). For many scenarios (see Section 6.6), LSR is more efficient than a comparable single modulation scheme.

Contributions. In this chapter, we make the following contributions:

- We design and implement a multi-hop low-power communication protocol using the principles of flooding and multiple modulations for energy-efficient and reliable data transfer in networks with inhomogeneous link characteristics.
- We propose a scheme based on non-destructive delayed retransmissions to deduce the connectivity graph of the network. This information serves as the basis of many network optimization methods.
- We implement the proposed LSR protocol in a simulation and extensively evaluate the LSR protocol by comparing it to a single modulation scheme without optimizations.
- By implementing the LSR protocol on real IoT hardware and by performing tests on the FlockLab 2 testbed presented in Chapter 3,

we demonstrate the feasibility of the LSR protocol and verify the simulation results.

First, we provide an overview of existing approaches for the stated problem and discuss related work in Section 6.2. In Section 6.3, we explain the proposed LSR protocol in detail and in Section 6.4, we describe the scheme to obtain the connectivity graph. We then discuss advantages, limitations, and potential extensions in Section 6.5. In Section 6.6, we evaluation the LSR protocol. In the end, we summarize the chapter in Section 6.7.

6.2 Related Work

In this section, we classify and discuss approaches for communication in LPWANs.

Adaptive Transmit Power Control

A large body of work proposed and analyzed schemes for adaptive transmit power control for networks with links with a limited range of path losses and in combination with traditional short-range radios, see for example [JCO07, LMZ+16, TJI+13, MGS+17]. In principle, such adaptive transmit power control schemes could also be used for inhomogeneous networks in combination with recent radio technologies which provide larger link budgets for LPWANs. However, in addition to the transmit power settings, newer radios offer an additional degree of freedom by supporting different modulation settings. Usually, the influence of the modulation on the energy consumption is significantly larger than the influence of the transmit power setting [OGMD17], also see Section 2.3.2. Therefore, an approach based purely on the transmit power control is ill-suited. In this chapter, we focus on the combination of different modulations and to keep the complexity reasonable, we use a fixed transmit power setting.

Star-Based Topology with Constrained Relays

One of the most popular LPWAN protocols is LoRaWAN [LoR20]. It supports star-based topologies and covers different channel access schemes

with multiple classes. A major limitation is the missing support for multi-hop communication. Nodes that do not have a direct connection to the gateway, e.g. due to obstacles, cannot transmit any data (also see example (a) in Figure 6.2). Multiple works have investigated the extension of LoRaWAN, or star-based protocols in general, using relay nodes. We can distinguish two classes of approaches: (1) relay nodes with unconstrained energy supply (e.g. see [DG18]) and (2) energy-constrained relay nodes (e.g. see [ESRB19, DP19, MAG+17, PMMB18b]). In most cases, only a subset of all nodes is assigned or can be assigned to take over the role of a relay node. This is contrary to the requirements of our scenario where any node should be able to communicate with the gateway as long as there is a multi-hop path.

Separate Systems for Long- and Short-Range Modulations

Another closely related approach is to use two separate communication systems and potentially protocols. For example, one system could cover links with low path loss, and the second system is used for communication over links with large path loss. An example where two separate protocols are used and a subset of nodes support both protocols is presented by García-Martín et al. [GMT21]. Even if the two systems work completely independently, a minimal amount of coordination is required for proper functioning. For example, the two systems need to coordinate their access to the RF spectrum. A node that participates in the system using short-range communication could still try to participate in the system using long-range communication in case it gets disconnected. However, since not all nodes are always participating in the long-range system, there are scenarios where a node cannot reach the gateway even if there would be a long-range path between the node and the gateway (also see example (c) in Figure 6.2). For this reason, this approach does not satisfy the requirements of our scenario.

Routing with LPWAN

A large amount of related work is focusing on routing to transfer data efficiently between a gateway and IoT devices. However, most of the work, e.g. [AVBMBB18, LK18, ZLS+19, MK20, JZBY+21], does not make use of different modulations (e.g. different spreading factors (SFs) for LoRa). One exception is [STB+17] where different SFs are scanned and incorporated in the metric for computing the link cost. A general problem

of routing-based approaches is the relatively slow convergence to obtain the required connectivity information such as a tree. In addition, these approaches often use uncoordinated concurrent transmissions in different parts of the network and therefore accept collisions which can reduce the reliability. Especially with the use of long-range transmissions this problem is expected to be pronounced because the number of affected nodes is potentially larger compared to using short-range transmissions.

Flooding with LPWAN

One common approach for multi-hop communication in LPWANs, is to use synchronous transmissions and flooding, e.g. see [BVR16, LZK+17, ZLS+18]. Simple flooding provides efficient, reliable, and low-latency multi-hop communication for networks with homogeneous link properties. However, to the best of our knowledge, all related work focusing on flooding in LPWANs only supports a single modulation (e.g. a single LoRa SF). A protocol that relies on synchronous transmission flooding normally requires all nodes to use the same modulation within the same flood. In order to reach a large portion of the nodes, flooding with the long-range, i.e. low data rate, modulations would be necessary which is not efficient for inhomogeneous networks with links with a wide range of path losses.

Our proposed LSR protocol is based on a global TDMA schedule and makes use of synchronous transmission flooding which provide reliable multi-hop communication and fast convergence after network changes. By using multiple modulations and optimizations, the LSR protocol supports energy-efficient communication in inhomogeneous networks with links with a wide range of path losses despite the above mentioned challenges.

6.3 LSR Protocol Design

In this section, we explain the operation of the proposed Long-Short-Range (LSR) protocol. We start with the general concept and the basic scheme under steady-state conditions. We continue with explaining step-by-step different optimizations and how statistics are collected. In the end, we discuss the operation under dynamic conditions.

6.3.1 General Concept

LSR is a round-based TDMA protocol whose structure is inspired by the LWB protocol [FZMT12]. Each round consists of a number of slots which are grouped into 3 phases (see Figure 6.3): *Schedule*, *Contention*, and *Data* phase. In each slot, a Gloria flood (see details at the end of this section and in Section 2.5), which is based on synchronous transmissions, is used to send a message from the flood-initiating node over potentially multiple hops to one or more other nodes. The network is controlled by a single node, the host. For simplicity, we assume that the sink node always serves as the host as well.

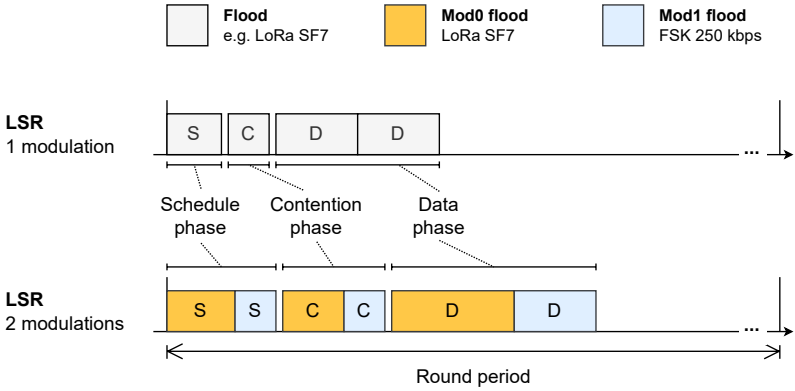


Figure 6.3: One round of LSR using two modulations consists of two interleaved LSR rounds, each using its own modulation.

We first explain LSR for the degenerate case in which a single modulation is used. In this case, the round structure of LSR is similar to the round structure of LWB. The *Schedule* phase consists of a single slot in which the host sends a schedule packet. This schedule packet contains protocol control information and the schedule, i.e., the usage and assignment of the following slots within the same round as well as the time to the start of the next round (round period). The schedule packet is received by all nodes which can be reached with the flood and serves as a time synchronization beacon. The next phase, the *Contention* phase, also consists of a single slot that is used by nodes to register data streams or to modify existing streams. A *data stream* consists of two pieces of information: (1) the nodes between which data is transmitted, and (2) how much data is transmitted per time (data rate). The host receives and handles

such stream requests and assigns slots to nodes. We refer to a node that has at least one registered data stream as a *registered node*. The last phase of an LSR round is the *Data* phase, which consists of multiple slots for data transmissions. Within each slot, only the node assigned according to the schedule is allowed to initiate the flood. This means there is no time slot in which multiple nodes are allowed to initiate floods simultaneously, except in the Contention phase.

In case the LSR protocol is used with multiple modulations, the round consists of multiple interleaved LSR rounds, each using its own modulation (see Figure 6.3). For simplicity, we only use 2 different modulations in this chapter: *Mod0* which corresponds to LoRa SF7 and represents a long-range modulation providing a large link budget, and *Mod1* which corresponds to FSK 250 kbps and represents a short-range modulation providing a smaller link budget. Of course, other configurations could be used as well. In addition, the proposed LSR protocol could easily be extended to work with more than 2 modulations (see Section 6.5.3). Within the Schedule and the Contention phase, there is a slot for each modulation. The Data phase is divided into 2 sub-phases, each with a variable number of data slots for the corresponding modulation. The sequence of modulations within each phase is ordered according to decreasing link budget. This ordering is to allow for simpler and faster bootstrapping (see Section 6.3.5.1).

LSR includes two different acknowledgments which are both sent in the schedule: one to acknowledge the successful registration of a data stream and one to acknowledge data transmissions. The latter is an aggregated acknowledgment of all data transmissions of the corresponding modulation in the preceding round. For each data slot, one bit of a bit field indicates whether the data has been successfully received by the host. The LSR protocol supports scenarios where the traffic demand of different nodes may be different and is allowed to change over time. For simplicity, we specify that the nodes can register exactly 0 or 1 data streams. An extension with multiple streams per node is discussed in Section 6.5.3.

The Gloria flooding primitive presented in Section 2.5 is used to flood messages within an LSR slot. Gloria is an optimized variant of the Glossy [FZTS11] multi-hop flooding primitive based on synchronous transmissions which we ported to the DPP2 SX1262 platform (see Chapter 2). The Gloria flooding allows to distribute a message from a single node to all nodes in the network (one-to-all) without requiring information about the network topology or queuing of messages at relaying nodes. Consequently, using message flooding with Gloria has the advantage that no complex control is required to prevent overflows of message queues at

intermediate nodes. In addition, compared to multi-hop schemes with in-network buffering, the end-to-end message transfer with Gloria has a short latency that is upper bounded by the configured flood duration. The flood duration is typically chosen such that the number of slots within a Gloria flood is large enough to cover the maximum number of hops required for the flood to propagate to all nodes in the network. In this chapter, we use the expression *initiating a flood* to express that a node starts sending a flood by being the first node to transmit the packet to be flooded. Furthermore, we use the expression *participating in a flood* when a node starts the Gloria primitive without initiating a flood, i.e., a node starts listening for packets, and in case the node received a packet, it retransmits it. This means that a node can participate in a flood even if no node initiates a flood. As explained in Section 2.5 and similar to Glossy, with Gloria, the nodes can use the arrival time and the slot index of the received flood to reconstruct the time when a schedule flood has been initiated by the host. This point in time is the same for all nodes and is therefore used as a synchronization point to accurately synchronize the time of the nodes to the time of the host.

6.3.2 Basic Scheme

First, we look at the basic protocol operation in the steady state. An example is depicted in Figure 6.4b. For the explanation of the basic scheme, we assume that the RF environment and the traffic demand by the nodes do not change and that optimizations, as discussed in Section 6.3.3, are not applied. Furthermore, we assume the protocol runs long enough, such that protocol operations converged. This means that all nodes are time-synchronized and know from the preceding round when the following round will start. In addition, each node determined the *best modulation*, i.e. the modulation with the smallest link budget which still provides enough reliability. Since the host has already assigned data slots to the nodes, the *best modulation* has already become the *active modulation*, i.e. the modulation which is used by the nodes to send data to the host.

In the first two slots, the host initiates one flood on each of the 2 modulations containing the schedule of the respective modulation. All nodes participate in the floods of both schedule slots. Since we are looking at the steady state, no node is initiating a flood in the Contention slots. Also due to the steady state, the schedule contains data slots for all nodes. As a consequence, all the nodes send data in their respective slot. In the slots of the Contention and Data phases, each node only participates if (1) it received the corresponding schedule and (2) the index of the modulation

of the slot is smaller than or equal to the index of the active modulation.

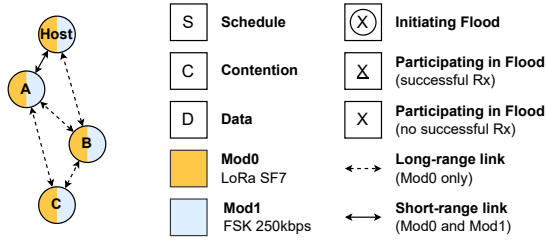
In the example in Figure 6.4b, node B which only receives schedule floods for Mod0 but not Mod1 and has active modulation Mod0 does not participate in Contention and Data slots with Mod1. On the other hand, node A which has received both schedule floods and uses Mod1 as active modulation participates in floods of all modulations in order to support floods initiated by other nodes but sends its own data packets only using the active modulation Mod1.

6.3.3 Optimizations

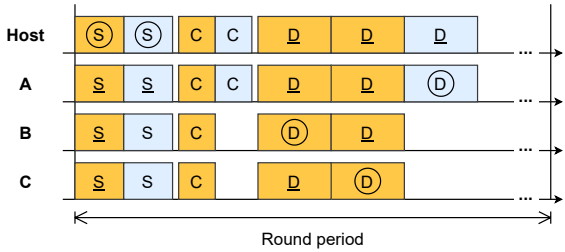
In data slots in the basic scheme where a long-range modulation is used (Mod0 in our example), usually many nodes are within range and therefore a large number of nodes help to retransmit the data packet. However, often this redundancy is not required and does not improve the overall reliability significantly. In many cases, the reliability of flooding the message even starts to decrease if the number of relaying nodes becomes too large [LFZ13a, DCL13, WHM+13]. For this reason, the following optimization methods are integrated into the LSR protocol in order to reduce the redundancy of data floods.

6.3.3.1 Thinning Based on Hop Distance

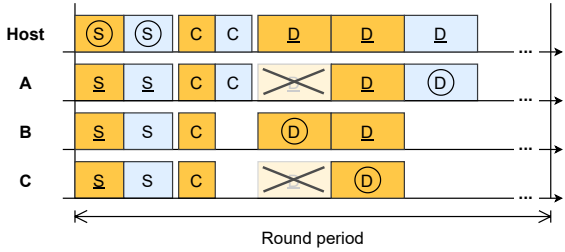
In a first approach, we only use the hop distance information to apply thinning, i.e. to turn off a subset of all nodes. This is a well-known method that has already been presented in different variations in related work. For example, in CXFS [CCT+13] all nodes measure the hop distance to the source as well as to the destination of a flood and decide based on the hop distance between source and destination whether to participate in the flood. With LaneFlood [BLS16], this concept has been refined to fine tune the level of allowed redundancy. A similar scheme based on the hop distance to the sink has been integrated into the Weaver protocol [TVL+20]. However, in most of the related work, the decision of whether to turn off a node is made in a decentralized way on the distributed nodes. In contrast to this, with LSR, the host collects all hop distance measurements and makes the decision. In LSR, the host represents the central entity which controls and therefore optimizes the network. This central control has the advantage that it allows a TDMA assignment of data slots preventing potential collisions. In addition, the collection of hop distance information at the host can be used for further optimizations,



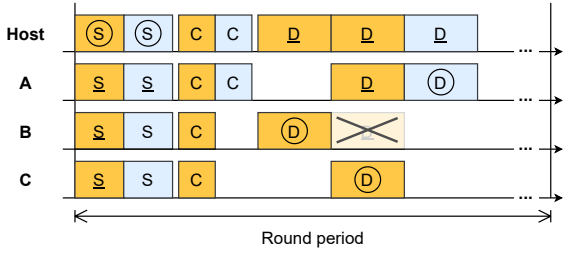
(a) Example network and legend for schedules.



(b) Basic scheme with *active modulation* determined. Nodes B and C use Mod0, and node A uses Mod1 as its active modulation.



(c) Thinning based on hop distance.



(d) Thinning based on hop distance and connectivity graph.

Figure 6.4: LSR steady-state operation with different levels of optimization.

e.g. for determining the connectivity graph (see Section 6.4). At the same time, this approach implicitly imposes the requirement that every node periodically sends floods to the host node to enable optimizations.

Hop distance measurements are available for free when using flooding as a communication primitive (see Section 6.3.4). In LSR, all nodes which, according to hop distance, cannot be part of a shortest path between initiator and host are instructed not to participate in the flood. A visualization of applying thinning based on hop distance to our example is depicted in Figure 6.4c. The host aggregates the complete hop distance information and decides which nodes should participate in which flood. The thinning of floods is determined for every node individually. Since the information in which floods a node participates is expected to change only rarely, it is buffered as a local state at the node. The host only sends updates of this state, the so-called *participation updates* as part of the schedule packet. In our implementation in Section 6.6, updates are represented as a fixed-size bit field. For better scalability with a large number of nodes, other representations, e.g. a variable-size list limited in length containing updated nodes, could be considered. For schedule and contention floods, high reliability is important for the stable operation of the protocol in most cases. Therefore, no thinning is applied to these floods. An extension to apply thinning to the schedule and contention floods is discussed in Section 6.5.3.

6.3.3.2 Further Optimizations Based on Connectivity Graph

Many more schemes have been proposed and designed to further optimize wireless multi-hop networks in terms of energy consumption, reliability, radio-on time, etc. See for example [ZRHK15, BLS16, ZGT17, PL21, GFJ+09]. Usually, information on how nodes are connected is required for further optimization. However, using flooding with synchronous transmissions as a communication primitive does not directly provide such connectivity information. Therefore, we propose to determine connectivity information using non-destructive probing by delaying the retransmission of floods on a subset of nodes in the network. By repeating this multiple times in a systematic manner, the connectivity graph can be deduced from the set of measurements without disturbing the communication of data. The obtained connectivity information serves as a basis for more elaborate optimization algorithms. A detailed explanation of this approach is provided in Section 6.4. A visualization of applying thinning based on hop distance and the connectivity graph to our example is depicted in

Figure 6.4d.

If the connectivity graph is used for thinning, the energy consumption corresponding to forwarding packets can be distributed. In the example network in Figure 6.4a, node C could communicate with the gateway either via node B or via node A. The second option minimizes the maximum energy consumption across all nodes as the energy required to perform short-range communication is much smaller than the energy required for long-range communication. In addition, the overall energy consumption of all nodes is reduced by the thinning as nodes only participate in floods in which they are needed.

In our simulation (see Section 6.6), we implement a simple optimization based on the connectivity graph. A single shortest path between the node and the host is determined and nodes not on the path are turned off. By selecting a shortest path out of the set of all available shortest paths for each node, the host can perform energy load balancing. In order to average out the power consumption, the selection of the shortest path is repeated every time the connectivity graph is updated (see Section 6.3.5) and also periodically in case the thinning is not updated for a long period of time. Selecting a single shortest path represents an extreme case and therefore is suited to indicate an upper bound on the energy savings that can be achieved. More elaborate schemes which provide a defined level of redundancy or schemes where the actually available energy of each node is taken into account are left for future work.

6.3.4 Collection of Statistics

The basic scheme and the optimizations introduced in this chapter require information for decision-making. In this section, we explain what communication characteristics are measured and how they are used. An overview is provided in Table 6.1.

Measurement	**When Collected?**	**Influence on**	
		active modulation	thinning
Packet loss	continuously	✓	✓
Hop distance without delay	continuously		✓
Hop distance with delay	only when probing		✓

Table 6.1: Measured communication characteristics and their use in the LSR protocol.

Packet Loss: The LSR protocol tries to minimize packet loss as much as possible. However, as with any wireless communication system, packet loss cannot be completely prevented. The frequency of packet losses is an important indicator for the network and is therefore recorded accordingly and incorporated into the control of the LSR protocol.

For the basic scheme, the best modulation of each node needs to be determined. For this, a node listens to all schedules, i.e. to the long- and short-range schedule. The index of the modulation of the last schedule that was successfully received in the round is stored. In order to increase the statistical significance it is possible to increase the depth of the history, i.e. the measurement is repeated and stored in a ring buffer with parameterizable size B_m. All nodes listen for all schedule floods in each round and therefore continuously collect measurements. The best modulation is then derived from the ring buffer using a mathematical method such as the the median or another percentile.

Schedule and data packets as well as acknowledgments of streams and data are used by the host and by the nodes to determine whether the node is registered at the host, i.e. it has an active data stream. Furthermore, the information about packet loss is used to assess whether data can successfully be transferred between node and host. This information about the status of each node is also used to adapt the thinning as only actively connected nodes can be used to forward messages.

Hop Distance Without Delay: In a Gloria flood, the hop distance, i.e. the length (number of edges) of the shortest path to the host, is directly related to the index of the slot within the Gloria flood. This *slot index* (sometimes also referred to as *hop count*) is contained in every Gloria packet. It is set to 0 on the node which initiates the flood and is incremented on every node which relays the flood (see Figure 6.5). As a consequence, the hop distance $d(n)$ of node n can be determined from the slot index $k(n)$ when the flood is received on node n using the following relation:

$$d(n) = k(n) + 1$$

In order to make global decisions, the information of the hop distance between the host and each of the nodes needs to be available on the host. For each node/modulation combination for which the host receives a data message, it directly determines the hop distance from the received slot index. As explained in Section 6.3.2, a subset of the nodes participate in slots of both modulations but send data messages only using the active modulation. For such nodes, the host is only able to directly determine

hop distance measurements for the active modulation. However, in order to make reasonable decisions for the thinning optimization, measurements for all modulations with an index smaller or equal to the active modulation of a node are required. For node/modulation combinations for which no data message is sent, the node determines the hop distance measurement from the schedule flood and sends it piggy-backed to the host via the data flood that uses the active modulation. This approach implicitly assumes that the hop distance is symmetric, i.e. the shortest path has the same length in both directions. Similar to collecting measurements on the reception of the schedule floods, the host uses a ring buffer of size B_h for every node to store the B_h most recent hop distance measurements of floods without delayed retransmissions. Since obtaining the hop distance measurement does not require additional energy with Gloria floods and the influence of the transfer of measurements not directly collected on the host is small, the hop distance is updated continuously in every round.

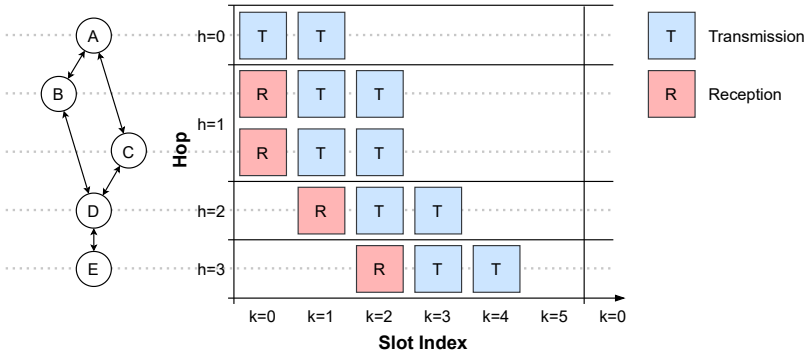


Figure 6.5: In a Gloria flood, the slot index within a flood is directly related to the hop distance between the flood initiator and the receiving node. The number of retransmissions (in this example set to 2) is a parameter of the Gloria flooding primitive.

Hop Distance With Delay: As explained in Section 6.4 in more detail, the protocol does not only collect hop distance measurements of normal floods but also collects hop distance measurements when some of the nodes delay their retransmission in order to determine the link structure between the nodes, i.e. the connectivity graph. The principle for measuring the hop distance remains the same. Again, the host uses a ring buffer of size B_d for every connection to store the B_d most recent measurements of the

connectivity. The probing with delayed retransmissions may reduce the reliability of floods for nodes with a large hop distance relative to the host. For this reason, hop distance data with delayed retransmissions is only collected when not enough measurements are available, e.g. when not all nodes are connected in the connectivity graph or if topology changes are detected.

6.3.4.1 Sequence of Measurement Collection

When the host needs the hop distance to a node for a decision, it calculates an estimate of the hop distance based on the measurements stored in the history, i.e. the ring buffer. Whenever there is a ring buffer for collecting hop distance measurements without delay that is not completely filled, for example in the beginning or when a node just joined the LSR network, the host instructs all nodes to not apply any delay to the retransmissions in order to measure the unaffected hop distance. If the ring buffer of every registered node is completely filled and fresh connectivity information is required (graph contains unconnected nodes or topology change occurred, see Section 6.3.5.2), the host initiates a probing sequence with delayed retransmissions as described in Section 6.4 and measures the potentially affected hop distances.

6.3.5 Dynamic Behavior

According to our scenario (see Section 6.1), the protocol is required to adapt to changes in the RF environment or the traffic demand of the nodes. In this section, we discuss the most important changes and explain the corresponding reactions of the protocol.

6.3.5.1 Bootstrapping

An important change occurs when one or multiple nodes are switched on (or reset) and start participating in the protocol. The goal of this first phase of the protocol, the *Bootstrapping* phase, is to let nodes synchronize their local time with the time of the host, register a data stream, and apply thinning. An example of the bootstrapping process is depicted in Figure 6.6.

We assume that a node starts in the beginning without any knowledge about the network except for the set of used modulations. As a first step, each node listens with modulation Mod0 for schedule floods initiated by

the host and potentially forwarded by other nodes. Since the nodes start with Mod0, all nodes which are able to communicate with at least one modulation in the set of used modulation will be able to receive and relay a schedule packet. Once a schedule packet is received, the node can synchronize its time. The node will then start to listen for the second schedule floods sent with modulation Mod1. Because the modulations are ordered according to decreasing link budget, already in the first round, each node can determine a first estimate of the best modulation, based on the received/not received schedule floods. Nodes that successfully received at least the first schedule flood and which have a traffic demand now use the contention slot corresponding to the best modulation to request a data stream. To support floods initiated by other nodes, the nodes keep participating in floods for all modulations with smaller or equal modulation indices as the active modulation. A random exponential backoff mechanism is applied if a stream is not confirmed by the host in the following schedule message. Once a stream request is confirmed, the node starts sending data messages in the assigned data slots. Once a node is registered, the thinning optimization starts. In Figure 6.6, the thinning part of bootstrapping is not shown. However, an example is discussed in the evaluation in Section 6.6.2.3.

6.3.5.2 Adapting to Changes

In the following, we describe the most important changes and the corresponding reactions according to the LSR protocol.

Nodes continuously update the best modulation based on received schedule messages. In case the best modulation is determined to be different from the active modulation, the node sends a stream request to the host to request a change of the modulation. In case a switch of modulation is possible according to the host, the host acknowledges the updated stream and the active modulation is updated on the host and the node. If the modulation is changed to a modulation with a smaller link budget, switching modulations is seamless and does not cause data packets to be lost, assuming ideal RF conditions. In the other direction, the need for switching the modulation is based on missed schedule floods and therefore reduces the data transmission throughput but no energy is wasted as no data floods are initiated.

In case a node is disconnected, i.e. it has no assigned data slot but data to send, it needs to register a data stream at the host again. Thanks to the flooding approach, only one round is required for a node to

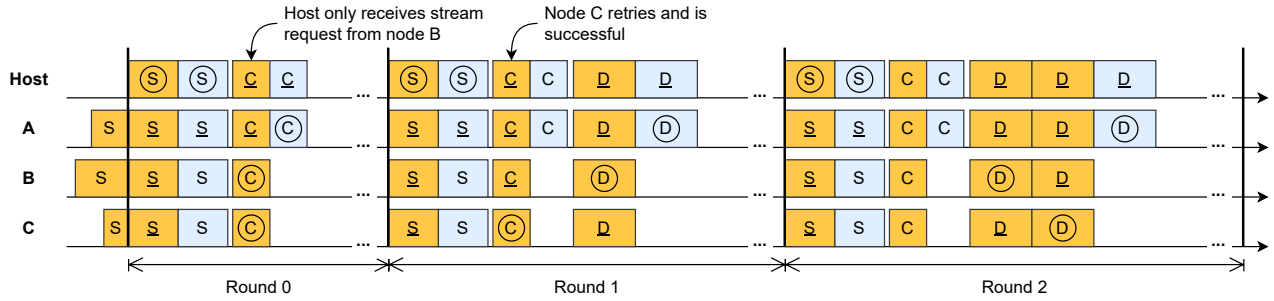


Figure 6.6: Bootstrapping process of the protocol. In this example, selecting the modulation is immediately applied after receiving the first schedule whereas data flood thinning is not applied within the first 3 rounds. The same notation as in Figure 6.4 is used.

register a stream, when assuming no packet loss which is a realistic assumption if no other node is trying to reconnect simultaneously. If a node is freshly connected, the corresponding thinning state is reset, i.e. all the corresponding measurements are cleared and collection is restarted. Once enough measurements have been collected the thinning, which includes energy load balancing, is applied again. In summary, this means reconnecting a node is fast but the optimization related to the reconnected node takes a few rounds to converge.

6.4 Determining the Connectivity Graph

Flooding with long-range modulations often leads to situations with a high level of redundancy. As discussed in Section 6.3.3, the LSR protocol contains two classes of optimizations to reduce this redundancy. However, there are many more optimization schemes that could provide a better balance between reliability and energy savings (we discuss some examples in Section 6.4.1), see also [ZRHK15, BLS16, ZGT17, GFJ+09]. Most of the known optimization schemes require information about the connectivity between nodes, which is difficult to obtain when using flooding as a communication scheme since nodes are sending concurrently. Therefore, from observing regular floods, we cannot determine which nodes are relevant. Knock-out measurements, i.e., disabling a node or a subset of nodes during a flood, would provide the necessary information. If the flood still reaches the destination, the knocked-out nodes are not essential. However, if the nodes are indeed essential for the flood, the message would not reach its destination. Losing messages would only be acceptable if dedicated floods are used for probing and thereby generate significant communication overhead.

We propose to use a different approach to obtain the necessary information with only negligible additional overhead. For probing, instead of knocking-out nodes, the host instructs a subset of nodes to delay their retransmissions of the received packet by a single Gloria time slot within the flood as depicted in the example in Figure 6.7. This has the advantage that the flood may still reach its destination. This allows the use of existing schedule and data transfer floods instead of dedicated costly probing floods in order to obtain measurements. If at least one node in the set of nodes that delay their retransmission is essential for forwarding the message, the measured hop distance increases compared to the case where no node

delays the retransmission.

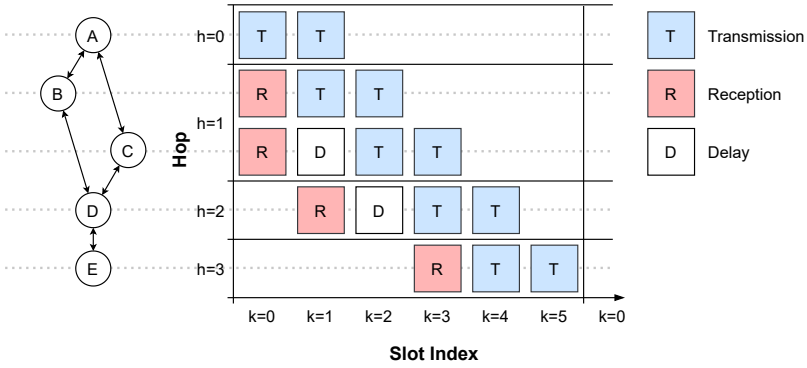


Figure 6.7: Example flood from node A to node E. A subset of all nodes (nodes C and D) is instructed to delay the retransmission of the received message by 1 time slot. The delay on node C has no influence as there is a redundant path via node B. However, node D is essential for forwarding the message, and therefore the arrival of the message at node E is delayed by 1 time slot ($k = 3$ instead of $k = 2$).

The instruction regarding which node should delay the retransmission is transmitted in every schedule and is valid for schedule and all data floods within the same round. Again, in the implementation in Section 6.6, we use a fixed-size bit mask to distribute this information. Similar to the participation updates, for better scalability with a large number of nodes, other representations, such as a compressed list of nodes which is limited in length, could be considered. Measurements of the currently observed hop distance are obtained in the same way as for rounds without delayed retransmissions (see Section 6.3.4). Again, nodes that participate in both modulations but send data floods using only the active modulation send hop distance measurements based on the reception of the schedule flood. This works because also the schedule flood is delayed according to the delay instructions.

In the following, we describe an algorithm that performs systematic probing and allows the host to construct a connectivity graph. This connectivity graph can then be used for thinning and other optimizations. We are looking for the connectivity graph, i.e. a directed graph which contains only the shortest paths from any node to the host. Therefore, we call it the shortest connectivity graph. In other words, all paths from a single node to the host have the same length (number of edges), which

corresponds to the hop distance between the host and the node. The shortest connectivity graph can easily be constructed from a complete connectivity graph by deleting all directed edges from a node with hop distance d_1 to a node with hop distance $d_2 \geq d_1$.

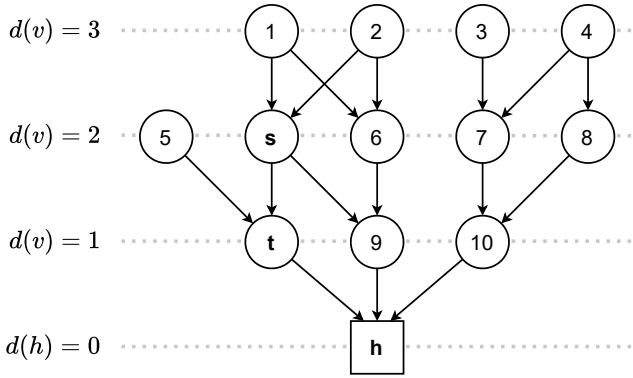


Figure 6.8: Example of a shortest connectivity graph for floods with the host as a destination.

Suppose that the shortest connectivity graph is denoted as $G(V, E)$ with nodes V and edges E. A node $v \in V$ has hop distance $d(v)$ where the host $h \in V$ has hop distance $d(h) = 0$. We further, define the subset $V_D \subseteq V$ as the set of nodes with hop distance $d(v) = D$ (see the example in Figure 6.8). Note that E only contains edges (s, t) with $d(s) = d(t) + 1$. We assume that within a round the host receives hop distance measurements from all nodes for all modulations. If a measurement for a node is missing, the host needs to repeat the measurement for this specific combination of node and modulation. The algorithm can be described as follows:

- The host collects hop distance measurements for all nodes without instructing any node to apply delayed retransmissions.
- Based on the collected measurements without delayed retransmissions, the host determines the hop distance $d(v)$ of every node $v \in V$. The maximal hop distance of all nodes is denoted as $D_{\max}$. Based on the hop distance information, the host can partition the set of nodes into subsets V_D with $D \in \{1, 2, \ldots, D_{\max}\}$. As a result, the host $h \in V$ also knows that $(t, h) \in E$ for all $t \in V_1$. This means a path between each of the nodes in the first layer (V_1) and the host is known at this point.

- The goal of the next steps is to determine the edges $(s,t) \in E$ for some hop distance $D = d(t)$. For every $0 < D < D_{\max}$, the host schedules $|V_D|$ different configurations C_t where each configuration lasts for a complete round. In configuration C_t, only nodes $v \in V_D \setminus \{t\}$ delay the relaying of incoming packets. The host records the new hop distance $d'(v)$ of every node $v \in V$. There is an edge (s,t) with $d(s) = D + 1$ if and only if $d(s) = d'(s)$.

The result of this probing sequence is the shortest connectivity graph $G(V,E)$. The total number of rounds that is necessary for this simple version can be computed as $R = \sum_{1 \le d \le D_{\max}-1} |V_d|$ where $R \cdot (|V|-1)$ packets need to be sent. This does not include the measurements which are required to determine the hop distance without delayed retransmissions.

6.4.1 Potential Use Cases of the Shortest Connectivity Graph

The resulting shortest connectivity graph provides the basis for a large number of optimization schemes. In the following, we discuss the most important classes that can be combined with each other. For every flood, a subset of relaying nodes is determined such that all nodes are still connected. For many scenarios, this leaves a certain degree of freedom which can be used to optimize the network for one of the following aspects:

- The level of redundancy can be tuned to match the requirements of the application. For example, a subgraph could be determined which is still connected if a single edge is removed.
- The overall total energy consumption could be minimized.
- The energy usage is evenly distributed to the nodes, i.e. the maximum energy consumption across all nodes is minimized.

For example, the RFT [ZRHK15] and LaneFlood [BLS16] schemes make use of connectivity information to achieve a configurable level of redundancy. LiM [ZGT17] uses connectivity information to apply machine learning in order to reduce redundancy and improve the reliability of the network. Also, traditional schemes based on tree construction, for example CTP [GFJ+09]), require information about the link characteristics. Most of the proposed schemes require or could significantly benefit from the information that the connectivity graph obtained by the LSR protocol provides.

In our simulation in Section 6.6, we use the connectivity graph to select a shortest path between the node and the host. All nodes not on the selected shortest path are turned off. To distribute the energy consumption among nodes, we determine the set of selected paths such that the maximum number of transmissions across all nodes is minimized. Even though we apply single-path routing, the forwarding of the message is still performed using the flooding mechanism, i.e. all retransmitting nodes use the same modulation and the flood runs directly from the initiator to the destination without interruption or long buffering of messages at intermediate nodes.

6.5 Advantages, Limitations, and Extensions

In this section, we discuss advantages and limitations of the LSR protocol and provide a selection of interesting extensions.

6.5.1 Advantages

One main advantage of the LSR protocol is that for any multi-hop scenario it connects all nodes to the host as long as there is a multi-hop path. The use of multiple modulations allows connecting nodes in networks with links that span a wide range of path losses and helps to keep the energy consumption low. Using a TDMA channel access scheme and flooding for transferring data provides high reliability, ensures fast convergence, and serves as a straightforward mechanism to periodically time synchronize all nodes to the time basis on the host. Furthermore, LSR adapts itself to changes in the network, such as link characteristics and topology, and is able to optimize energy consumption. Even when using the hop-distance thinning method only, the LSR protocol automatically degenerates to a star topology if for all nodes a single-hop path to the host is available and this path is optimal in terms of energy consumption. A key aspect of LSR is that it allows measuring the shortest connectivity graph of the network at runtime during normal data communication in a non-intrusive way and with only little overhead. This provides a generally applicable basis for further optimizations.

6.5.2 Limitations

The presented LSR protocol has a strong focus on data collection, i.e. it focuses on an all-to-one communication pattern. The proposed optimization schemes are therefore based on a centralized control at the host. Accordingly, every node in the network is required to periodically send floods to the host node for the optimization to be effective. Extensions that allow for more flexible traffic patterns are possible (also see Section 6.5.3) but potentially increase the complexity of the protocol significantly.

Analogous to LWB, the LSR protocol is round-based and the upper bound of the latency for transferring data from a node to the host is mainly determined by the round period (when assuming successful flood receptions). The round period is a parameter of the LSR protocol and can be adapted according to the application requirements. However, duty-cycled low-power operation usually does not allow arbitrarily short periods. Therefore, LSR is not well-suited for near real-time data transfers. But, combining real-time support with duty-cycled low-power operation is very challenging in general and potentially requires a dedicated radio channel.

Another disadvantage of LSR when comparing it to other protocols, such as LWB, is the increased complexity. In Section 6.6.3, however, we demonstrate that the implementation of the protocol variant with hop distance thinning on a representative IoT platform (DPP2 SX1262 platform presented in Chapter 2) is feasible. The additional complexity and computational effort that would be required for adding thinning based on hop distance measurements with delayed retransmissions would to a large extent be caused on the host node which, according to the assumptions in Section 6.1, is unconstrained in terms of available energy and computational performance.

The proposed LSR protocol relies on synchronous transmission flooding. This Glossy-like flooding only allows using a single modulation within the same flood. As a consequence, in LSR, if a path between a node and the host contains a long-range segment, the long-range modulation is used for all segments of the path. Extending the flooding scheme to allow flooding with mixed modulations is challenging. One possibility would be to use multiple modulations sequentially within the flood. However, this introduces the problem of determining and distributing the point in time for switching modulations. Alternatively, multiple modulations could be used simultaneously. This in turn entails the risk of destructive packet collisions and limits the number of forwarding nodes for each modulation, since not all nodes would participate in all modulations anymore.

In LSR, the exchange of status information, contained in schedule and contention messages, is necessary for each modulation. This implies an additional overhead that, in certain application scenarios, cannot be compensated by the potentially more efficient data transmission enabled by modulations with a lower link budget and the thinning optimizations. However, in Section 6.6.2, we show that the proposed LSR protocol has advantageous properties for many application scenarios.

6.5.3 Extensions

We presented the LSR protocol which fulfills all requirements according to the application scenario in Section 6.1. Nevertheless, there are a number of extensions that would allow improving the LSR protocol further.

Set of Modulations: So far, we have assumed that the set of available modulations is given and does not change. For simplicity, we use only 2 modulations in this chapter. However, the protocol could easily be extended to use more than 2 modulations. The mechanism to determine the best modulation would be unchanged and still relies on collecting statistics on the received schedule floods. But the overhead would potentially increase as each node needs to listen to additional schedule floods. The optimal set of modulations (number of modulations and which modulations are available) could be determined offline or online based on a model of the network or using sample measurements.

More Flexible Data Streams: Up to now, we have limited the number of streams for each node to a single stream. Furthermore, the gateway is always the sender or receiver of the data. With extensions of the basic scheme, more flexibility for data streams could be added. By introducing stream IDs and adaptation of the scheduler on the host, multiple streams per node could be supported. In addition, streams between two nodes could be integrated by including the destination node ID into stream requests and adapting the scheduler. In this case, thinning could be applied but full thinning to a single shortest path would not be possible in every case as the shortest connectivity graph, obtained as described in Section 6.4, does not contain all edges between nodes that have the same hop distance from the host. The scheme for measuring the connectivity graph could be extended as well but additional hop distance measurements need to be collected and the overall complexity would be increased.

Increasing Reliability by Resending Data: Due to the uncontrollable

and fluctuating RF environment, flooding messages with blind retransmissions, as it is implemented in Gloria, does usually achieve a packet delivery ratio below 100%. To improve this, an additional protocol layer on top of the basic scheme could be added. This additional layer would resend data that has not been delivered successfully. With the aggregated acknowledgment of data transmissions in schedule messages, the basic LSR scheme provides a mechanism to implement such a layer on top.

Multiple Gateways: We described the LSR protocol applied to a system with a single gateway. One straightforward solution to support multiple gateways is to assign each of the gateways the role of an LSR host and to statically assign each node to exactly one gateway. More sophisticated schemes where nodes for example are being passed between gateways at runtime or schemes where nodes can be registered to multiple gateways at the same time are possible but out of the scope of this work.

Thinning of Non-Data Transmissions: If it is known from the scenario that topology changes do not occur or are extremely rare, the protocol could be further optimized by thinning non-data transmissions such as schedule and contention floods. However, this is more critical with respect to recovery from a failure than thinning data floods since this could impact the flow of control information. To ensure recovery from a failure, a mechanism is required to reset the thinning of non-data floods. A simple example of such a mechanism would be that all nodes are required to participate in schedule floods in every jth round to receive control information.

Selection of Participating Nodes Based on Shortest Connectivity Graph: We propose to use a simple approach for thinning based on the shortest connectivity graph information, namely selecting a single path out of the available shortest paths. The selection of participating nodes could potentially be improved or tailored to the needs of the application by improving the load balancing and allowing for a defined level of redundancy.

6.6 Performance Evaluation

In the following performance evaluation, we show the following important properties of the LSR protocol: LSR (1) allows to cover all scenarios as described in Section 6.1 and Section 6.6.2, (2) is energy-efficient and

enables energy load balancing, and (3) dynamically adapts to a changing environment. We show the protocol behavior for a set of scenarios using an event-based simulation and verify that the simulation is realistic by comparing it to a hardware implementation running on the FlockLab 2 testbed presented in Chapter 3.

6.6.1 Methodology

Metrics

In order to evaluate the performance, we use the following metrics:

Reliability The number of rounds with a successful transfer of a data packet from the node to the host divided by the number of rounds that could be used for a data transfer. This metric is calculated for each node. In the evaluation, we assume and ensure that each node wants to send exactly one packet in each round.

Energy per round The total energy consumption per round for a single node. This includes the energy of all radio operations such as listening, receiving, and transmitting.

Active modulation The active modulation provides information about the modulation with which the node has registered a stream. This information is available on both the nodes and the host.

We do not include a metric for latency for transferring data from a node to the host in our evaluation because by design this property is not significantly different for the considered round-based protocols. As mentioned in Section 6.5.2, the upper bound of the latency is mainly determined by the round period (when assuming successful flood receptions). The aspect of unsuccessful data transfers is covered in the evaluation by the reliability metric.

Protocols for Comparison

As discussed in Section 6.1, straightforward approaches such as star-based protocols or protocols that use only short-range transmissions, do not satisfy the requirement of covering all considered scenarios. For this reason, we do not include such protocols in the evaluation. Instead, we compare the following two protocol implementations which cover all scenarios:

LWB This protocol is an implementation of the Low-Power Wireless Bus (LWB) [FZMT12] for long-range communication and serves as the baseline. It supports multiple hops but only a single long-range modulation. Concretely, a simplified version of LWB is used. The main difference to the original LWB lies in the fact that the round does not contain a second slot during which the host distributes the schedule. In addition, to improve the comparability to the LSR protocol implementation, we use Gloria (see Section 2.5) instead of Glossy as the lower-layer flooding primitive.

LSR This protocol is an implementation of the Long-Short-Range (LSR) protocol as described in Section 6.3 with optimizations as described in Section 6.3.3. We differentiate different variants by (1) the number of used modulations (in this evaluation, we use either 1 or 2 modulations) and (2) the types of thinning optimizations which are applied. If only 1 modulation is used, only the long-range modulation (Mod0) is available. Used thinning options are *hop distance thinning* where only the hop distance information is used, or *full thinning* which corresponds to thinning using the hop distance and the shortest connectivity graph information. For measurements regarding packet loss, hop distance without delay, and hop distance with delay a history depth of 3 measurements (i.e. $B_m = B_h = B_d = 3$) is used, except where otherwise noted.

Protocol Execution

We implement the mentioned protocols in a simulation as well as on real hardware:

Event-Based Simulation: The event-based simulation simulates RF transmissions on a packet level. The radio model used in the simulation is based on the Semtech SX1262 transceiver (see also Chapter 2) and accordingly specifies the available modulation schemes with the corresponding link budgets and the relations for the time-on-air calculations. Links, i.e. the point-to-point connection between two nodes, are modeled using path loss values. The reception at the receiver is probabilistic and is determined using the receiver's sensitivity and the received signal power of the arriving transmissions. For a reception to be successful, the reception must not be prevented by a too low signal level or by any overlapping transmission. The closer the received signal power is to the receiver's sensitivity the higher the probability that the successful reception is prevented due to a too low

signal level. For signals below the sensitivity, the reception fails in any case. The capture effect is taken into account by considering independently each transmission that overlaps a transmission of interest in time (considering the entire message, i.e. header and payload) and determining whether it prevents the reception on the node of interest. A probabilistic decision modeled by a continuous normal distribution is used. If the received power levels of the transmission of interest and the overlapping transmission are equal, the probability of preventing the successful reception of the transmission of interest is 50 %. If the transmission of interest has a received signal power that is 3 dB higher compared to the overlapping transmission, the probability of preventing the successful reception of the transmission of interest is significantly reduced (concretely to 15.9 %) due to the capture effect.

The simulation allows for arbitrary scenarios and provides control over the environment, i.e. the link characteristics. The definition of the network is defined in the form of a path loss matrix. With this, the simulation allows using path loss values from real measurements or from synthetically constructed scenarios. The simulation is written in Python and available as open source[1]. An exemplary visualization of the radio operations of a single LSR round is shown in Figure 6.9.

Testbed: We use the FlockLab 2 testbed presented in Chapter 3 to execute the protocols on real hardware. As discussed in more detail in Section 6.6.3, the FlockLab 2 testbed includes short-range and long-range links and is therefore suited to evaluate the performance of the considered scenarios and protocols.

6.6.2 Evaluation with a Set of Scenarios

In order to verify the correct functional behavior of the LSR protocol and to investigate in which aspects it outperforms the LWB baseline, we look at the metrics when executing the protocols in different scenarios. For this purpose, we define a set of 7 scenarios, visualized in Figure 6.10, which cover a wide range of use cases. In the following, we describe the scenarios:

Short-range (SR) All nodes in the network are reachable from the sink via a short-range multi-hop path. This corresponds to a scenario for traditional radios which do not support long-range communication, see for example [KWL+11].

[1] Link available upon publication

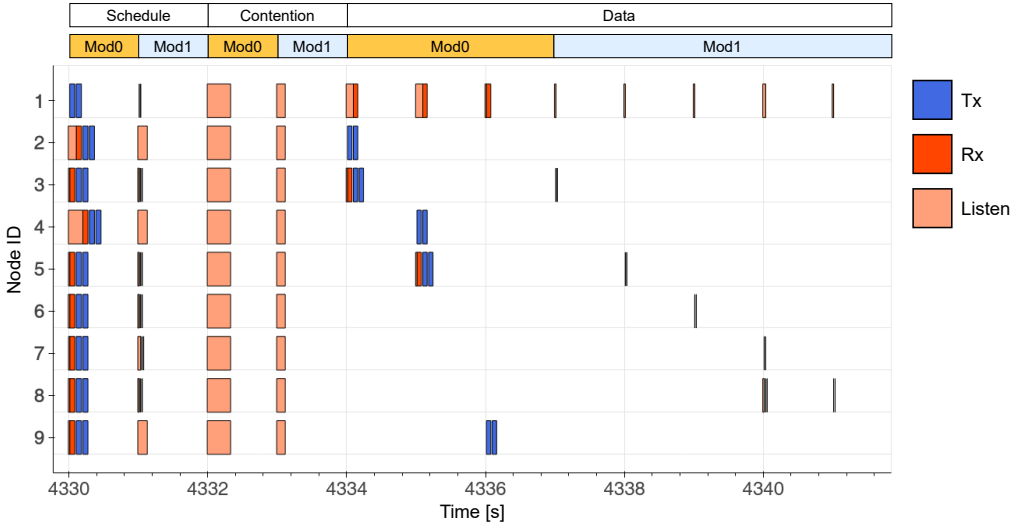


Figure 6.9: Exemplary visualization of a simulation run. Depicted is one LSR round with the radio operations (*Tx*, *Rx*, *Listen*) of all nodes. The *Listen* operation corresponds to *Rx* without successful reception of a packet. In this example, the CR scenario is used (see Figure 6.10) and LSR is configured to apply full thinning.

Long-range single-hop (LS) The network consists of nodes that are reachable from the sink via a single-hop long-range link. No node is reachable from the host via a short-range multi-hop path. This scenario corresponds to a star topology which is an implicit assumption of the LoRaWAN [LoR20] protocol.

Long-range multi-hop (LM) The network consists exclusively of nodes that are reachable from the host via a long-range multi-hop path. No node is reachable from the host via a short-range multi-hop path. The nodes are distributed and do not form a single cluster. A possible use case scenario is a network for monitoring farm animals, see for example [HGTP20].

Cluster with remote nodes (CR) This scenario is based on the SR scenario but is extended with additional single remote nodes that are only accessible via long-range paths from the host. A subset of the remote nodes is only reachable via a multi-hop long-range path. An exemplary use case for this scenario is a monitoring system for underground infrastructure [ESRB19].

Remote cluster (RC) Variation of the LM scenario. Instead of the nodes being distributed, in this scenario, they are gathered in a cluster. Again, monitoring underground infrastructure [ESRB19] is an example use case for this scenario.

Line (LI) This scenario represents an extended line topology. The used long-range modulation does not only allow to reach the next hop but the next two hops at least. This scenario is motivated by application scenarios such as tunnels, see for example [CCD+11].

Fanout (FO) In this scenario, the majority of nodes can only communicate with the host via multiple hops and using the long-range modulation. Nodes 8 and 9 have a direct connection to the host and represent a bottleneck for the communication of the remaining nodes.

For the concrete implementation of the scenarios, the high-level information about the links (Mod0, Mod1, or no link) in the network together with the set of available modulations is used to construct a corresponding path loss matrix. This path loss matrix is then used in the simulation.

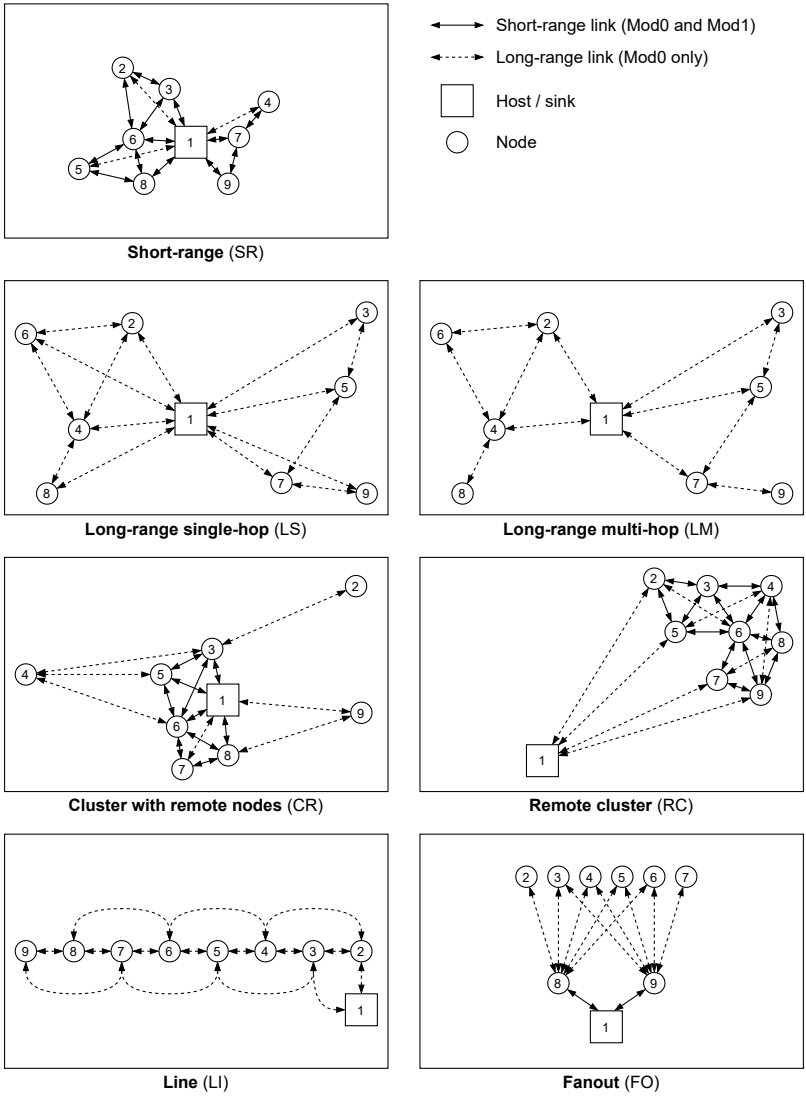


Figure 6.10: Scenarios used for the performance evaluation.

6.6.2.1 Configuration

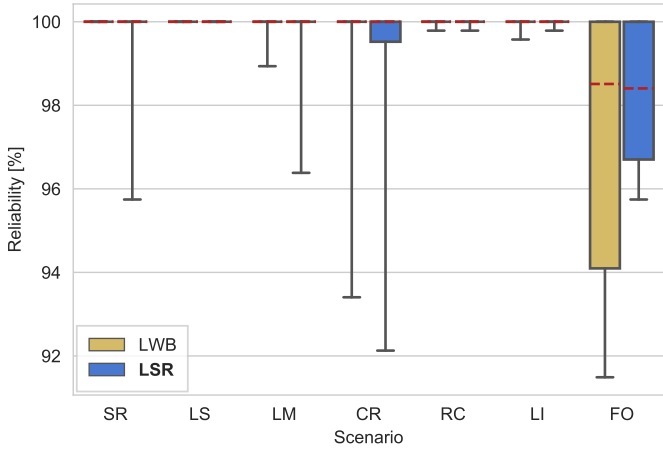
The basic configuration values which are used for the simulation are listed in Table 6.2. The transmit power has been set to the same level as in the evaluation on the testbed (see Section 6.6.3). For the simulation, it has no influence on the connectivity, as the used path loss matrix is constructed by taking into account the available link budget. However, it has an influence on the absolute power consumption values, i.e. the energy per round metric. The *number of slots* setting for the Gloria flooding primitive limits the maximum supported network diameter in number of hops.

Parameter	Configuration
Traffic demand	Tx queue is always filled with at least one packet, i.e. every node has always a demand to send a packet. All nodes request exactly one slot per round.
Data payload size	16 bytes
Set of modulations	Mod0: LoRa SF7 (bandwidth: 125 kHz) Mod1: FSK 250 kbps
Transmit power	+4 dBm
Set of nodes	1, 2, 3, 4, 5, 6, 7, 8, 9
Host node	1
Gloria floods: Number of retransmissions	2
Gloria floods: Number of slots	Mod0: 6, Mod1: 6

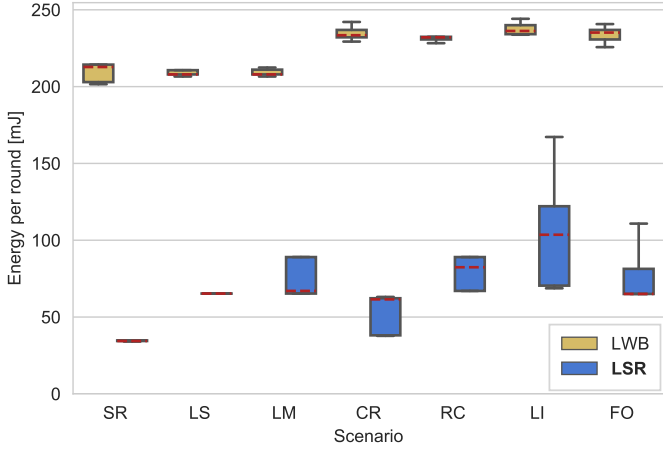
Table 6.2: Configuration parameters that are used for the simulation.

6.6.2.2 Results: Overview of Scenarios

First, we verify that the protocols work reliably in all scenarios and compare the energy consumption. For this, we use the simulation to execute the LWB and the LSR protocols in all scenarios. For the LSR protocol, we use 2 modulations and full thinning. For each combination of protocol and scenario, the simulation is executed for 500 rounds whereas the first 30 rounds are omitted in the presented data to remove the influence of the Bootstrapping phase. Visualizations of the collected reliability and energy per round metrics are depicted in Figure 6.11a and Figure 6.11b, respectively. In the case of the reliability, one sample corresponds to the reliability of a single node. The data of all nodes form one distribution, i.e. one box in the boxplot. For the energy per round, one sample corresponds to the consumed energy of a single node in a single round. The data of all nodes for all rounds form one distribution, i.e. one box in the boxplot.



(a) Reliability of data transmissions. The box extends from the lower to the upper quartile values of the data and the line represents the median. The whiskers span the entire range of the data.



(b) Distributions of energy per round of all nodes. The box extends from the lower to the upper quartile values of the data and the line represents the median. The whiskers span the range between the 10th and the 90th percentile to exclude rare outlier samples for which a node did not receive any schedule flood in the corresponding round.

Figure 6.11: Simulation results for the set of scenarios.

The results show that the LWB and the LSR protocols provide sufficiently high reliability, i.e. they work for all scenarios. The reliability of LSR is comparable to the reliability of LWB. There are cases for which the reliability of LSR is lower. This can be explained by the fact that LSR applies thinning optimizations which reduce the redundancy. However, there are also cases for which the reliability of LWB is lower. Such cases can be explained by the reduced potential of colliding packets within a flood as the number of simultaneously transmitting nodes is reduced by the thinning, too. The energy consumption of LSR is significantly reduced compared to LWB. There are two reasons for this. The main reason is the thinning optimizations which disable a subset of nodes during data floods. The second contributing factor is the use of 2 modulations which allows certain nodes to transmit data efficiently via short-range communication. Together with energy load balancing, this allows reducing the maximum energy consumption of all nodes. More detailed results are provided in Section 6.6.2.3.

The reduction of energy per round is most significant for scenarios with nodes connected to the host via a short-range path, such as scenarios SR and CR. Scenarios that require most or even all nodes to send data using the long-range modulation, such as scenarios LM, LI, and FO, show a higher energy consumption. As all transmissions within a flood are required to use the same modulation, this also applies to scenario RC. The LI scenario is the most extreme one, as indicated by the wide range of energy per round. For this scenario, all nodes are required to send data with the long-range modulation. The nodes close to the sink represent a bottleneck and the opportunities to apply thinning optimizations are limited.

In summary, the LSR protocol allows reducing power consumption without losing the property that every node can always be connected via the long-range modulation if required by the environment, which ensures that it is able to support all scenarios.

6.6.2.3 Results: Closer Look at the CR Scenario

As the aggregated results only provide a high-level view, we have a closer look at the *cluster with remote nodes* (CR) scenario. For this, we run the simulation for the CR scenario for 1000 rounds. In order to see the influence of the number of modulations when using the LSR protocol, we add the LSR protocol variant that uses only a single modulation, the long-range modulation. We apply full thinning for both LSR variants. In Figure 6.12, we plot the distribution of the energy per round for each node separately.

Again we omit the first 30 rounds to remove the effect of the Bootstrapping phase, i.e. each bar represents the data from rounds 30 to round 999.

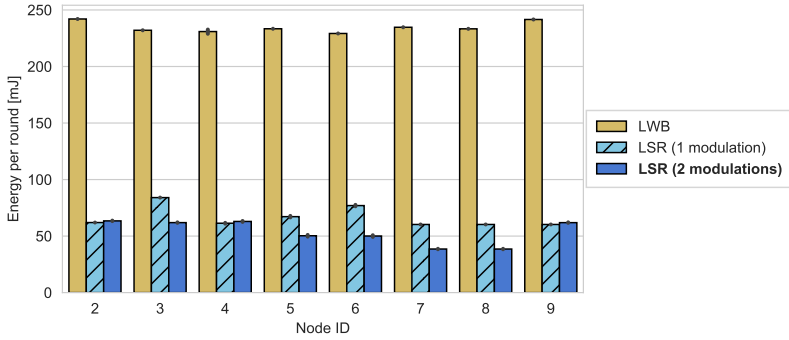
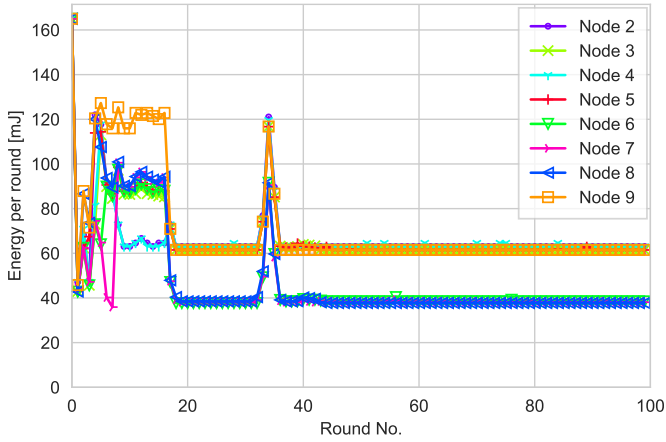


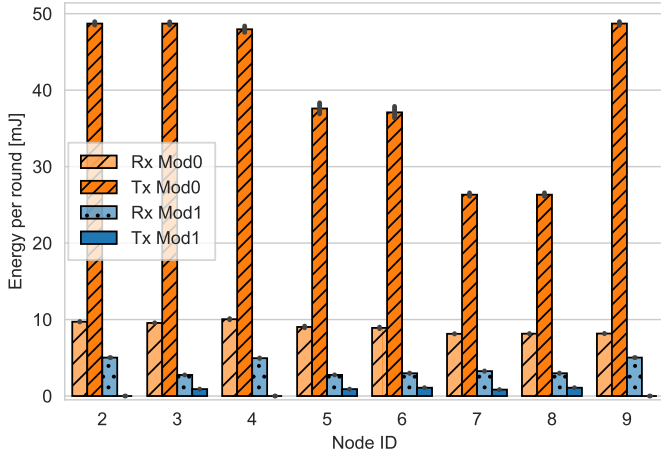
Figure 6.12: Energy per round for each node separately for the CR scenario. The bars represent the mean of all considered rounds and the black vertical lines span the 95% confidence interval.

As expected, the energy consumption of the LWB protocol is similar for all nodes. With this protocol, all nodes always use long-range modulation and all nodes participate in each data flood. Using LSR with only 1 modulation already reduces energy consumption significantly. The thinning reduces the energy consumption and the information from the connectivity graph allows to evenly distribute the energy load for relaying messages to different nodes. Concretely, the forwarding of messages from nodes 2 and 4 is distributed to nodes 3, 5, and 6 compared to an assignment in which node 3 does all the work. However, this scheme is still inefficient in this case as the relaying nodes send their own data as well as data from remote nodes with the more expensive long-range modulation. If we use LSR with 2 modulations, the relaying nodes (nodes 3, 5, and 6) can send their own data via the short-range modulation. This allows to further reduce the maximum energy consumption across all nodes and therefore would increase the lifetime of a battery-powered system.

In the following, we further inspect the LSR protocol using two modulations. In order to investigate the temporal behavior, we plot the energy per round metric over time in Figure 6.13a. In this case, we do not omit any rounds. From the energy consumption, we see that the Bootstrapping phase starts in round 0 and lasts around 20 rounds. The Bootstrapping phase consists of two parts: (1) the registration of the nodes



(a) Energy over time plotted for the first 100 rounds of the simulation run.



(b) Energy consumption per round by components. The bars represent the mean of all considered rounds and the black vertical lines span the 95% confidence interval.

Figure 6.13: Results for the CR scenario when executing LSR using 2 modulations with full thinning.

at the host, and (2) the collection of measurements for the thinning. The time which is required for the registration is mainly determined by the number of participating nodes. Each node determines the best modulation and sends a stream request. Per round, the host can receive at most one stream request for each modulation successfully. In the example, 8 rounds are required for all nodes to get registered. Once a node has a stream assigned, i.e. the active modulation has been determined, the thinning procedure is started. In the example in Figure 6.13a, the thinning lasts until round 18. Afterwards, the energy consumption is significantly reduced and remains stable for all nodes for a longer period of time. In rare cases, the network is adapted to a detected change even in a static scenario. For illustration purposes, we specifically picked a case where this happens within the first 100 rounds, concretely in round 33. This behavior can be explained with the simulation which is configured to use probabilistic transmissions, i.e. individual packets can be lost and the arrival of floods can be delayed. This can lead to situations where a used link in the connectivity graph is considered to be no longer available or a node is considered to have a different distance to the host. In these cases, the thinning is partially reset and the connections between nodes are measured again. As a consequence, the energy consumption is increased for a short amount of time. By configuring the history depth, it is possible to adjust the trade-off between fast adaptation to changes versus stability. Overall, LSR is able to continuously collect information about link characteristics without increasing the energy consumption significantly.

To examine the contributions of different radio operations, we plot the individual components in Figure 6.13b. Here, we again omit the first 30 rounds. The results show that transmitting with long-range modulation accounts for the largest share of energy consumption. In addition, the nodes that are connected to the host via a short-range path and do not have to forward data (nodes 7 and 8) have a non-negligible long-range part. This is due to the forwarding of the schedule packets and the participation in contention slots. This represents the cost of supporting continuous dynamic adaptation to a changing environment. In certain cases, this is not necessary and further optimizations, as discussed in Section 6.5.3, could be applied.

6.6.2.4 Results: Dynamic Behavior

To demonstrate how the LSR protocol adapts itself to a changing environment, we artificially change the link characteristics at runtime. For this, we

use the RC scenario and simulate 100 rounds. At round 50, we change the link between the host (node 1) and node 5 from supporting only long-range communication to supporting long- and short-range communication.

In Figure 6.14, the active modulation as registered on the host node is depicted over time. In the first 10 rounds, the bootstrapping process is taking place and all nodes successfully register a stream for the long-range modulation (Mod0). After the path loss of the link between node 1 and node 5 is artificially decreased, all nodes could potentially switch to use the short-range modulation (Mod1). The transition happens with a delay due to the history of measurements. In this example (Dynamic Behavior), the depth of the history is configured to 8 measurements. In Figure 6.16, the shortest connectivity graphs for Mod0 and Mod1 as observed by the host node before and after the change are depicted. The two time instants are marked with arrows in Figure 6.14. When comparing the shortest connectivity graphs in Figure 6.16 to the RC scenario in Figure 6.10, we find that the graphs match the original graph well. This example shows that LSR is capable of continuously measuring the link characteristics and switching the modulation during normal data transfer rounds. In many cases, this even works without interrupting the data stream, i.e. without losing data packets. This is also the case in this example (Figure 6.15). For the other direction (switch from Mod1 to Mod0), such an uninterrupted switching is not supported by LSR as the change of the topology is detected by observing missed packets.

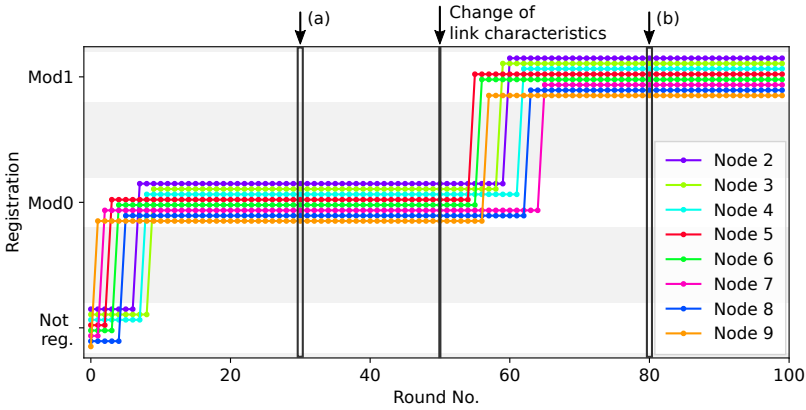


Figure 6.14: Simulation results for the CR scenario with the LSR protocol using 2 modulations and full thinning. In round 50, the link between nodes 1 and 5 is changed to support short-range communication, i.e. it supports Mod0 and Mod1. The changed link characteristics allow all nodes to switch to Mod1. The shortest connectivity graphs for the highlighted time instants (a) and (b) are shown in Figure 6.16.

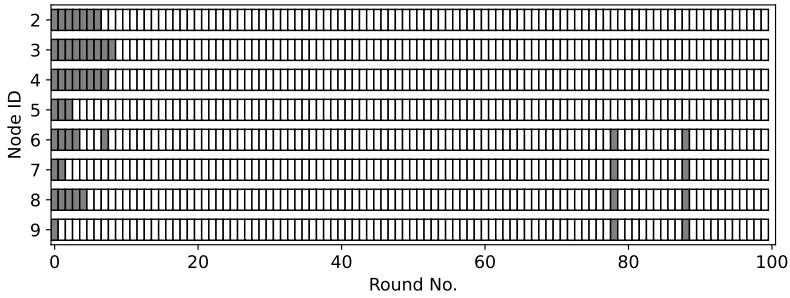
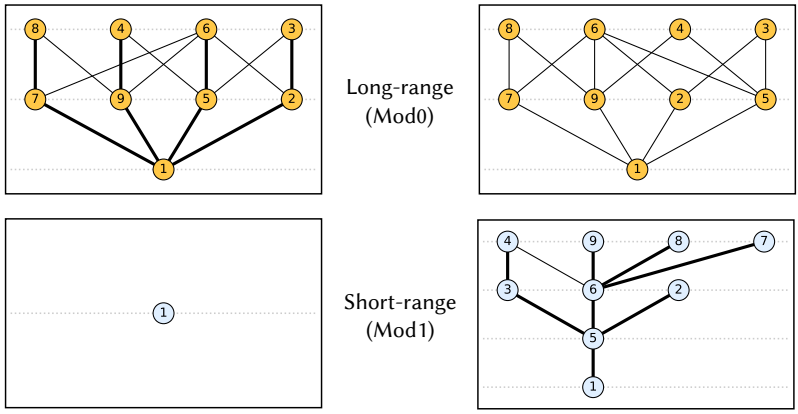


Figure 6.15: Rounds with (white) and without (gray) received packets for all nodes. In the example, each node tries to send exactly one packet in each round. Despite the major change of the topology at round 50, no packet is lost during the transition.



(a) Before change. (b) After change.

Figure 6.16: The connectivity graphs before (a) and after (b) the change in link characteristics for both modulations. The two different time instants correspond to the highlighted time instants in Figure 6.14. The edges used by the nodes for data transfers are highlighted (thick lines). After the change, a short-range path is used by all nodes.

6.6.3 Comparison of Simulation to Testbed Execution

In this subsection, we show that the LSR protocol does not only work in a simulation but also on real hardware in a testbed. In addition, we compare the performance metrics of the simulation to the testbed to confirm that the simulation matches the reality well.

6.6.3.1 Scenario

The scenario is based on the publicly accessible FlockLab 2 testbed installation (see Chapter 3). In contrast to the other synthetic scenarios, the scenario consists of 13 nodes that are listed in Table 6.3 and depicted in Figure 3.5. Nine nodes are located inside an office building and four additional nodes are located further apart at distances of up to 2 km on rooftops of campus and city buildings. This scenario corresponds to the CR scenario in Section 6.6.2. We obtain the corresponding link characteristics information by performing link tests on the FlockLab 2 testbed which we then feed into the simulation in the form of a path loss matrix.

6.6.3.2 Configuration

The configuration values that are used for the simulation and the tests on the FlockLab 2 testbed are listed in Table 6.3. The transmit power is chosen such that the 4 remote nodes are not reachable using the short-range modulation (Mod1) but are reachable using the long-range modulation (Mod0).

Parameter	Configuration
Traffic demand	Tx queue is always filled with at least one packet, i.e. every node has always a demand to send a packet. All nodes request exactly one slot per round.
Data payload size	16 bytes
Set of modulations	Mod0: LoRa SF7 (bandwidth: 125 kHz) Mod1: FSK 250 kbps
Transmit power	+4 dBm
Set of FlockLab 2 nodes	1, 4, 6, 7, 12, 13, 15, 17, 24, 25, 26, 27, 30
Host node	7
Gloria floods: Number of retransmissions	2
Gloria floods: Number of slots	Mod0: 3, Mod1: 5

Table 6.3: Configuration used for the simulation and the tests performed on the FlockLab 2 testbed when comparing the two implementations.

6.6.3.3 Testbed Implementation

The implementation used for the evaluation on the testbed is based on the DPP2 SX1262 platform presented in Chapter 2. The software used to run on the nodes is available as open source as part of the *flora* software[2]. The implementation currently supports the basic LSR scheme (see Section 6.3.2) and the optimization based on the hop distance thinning (see Section 6.3.3.1).

6.6.3.4 Measurements and Results

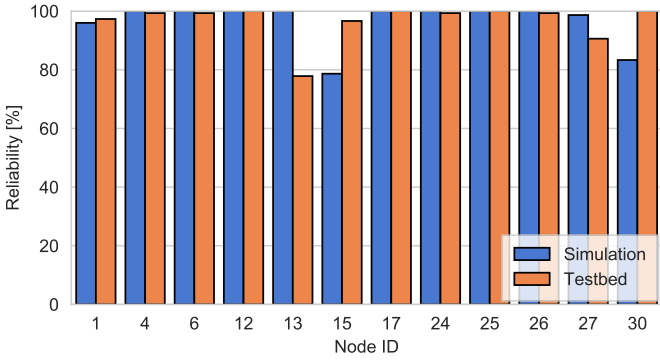
For the simulation as well as the testbed, we run the LSR protocol with hop distance thinning and using 2 modulations for 180 rounds. We omit the first 30 rounds to remove the influence of the Bootstrapping phase. The resulting reliability and the energy per round values are plotted

[2] https://gitlab.ethz.ch/tec/public/flora (LSR will be available at time of publishing)

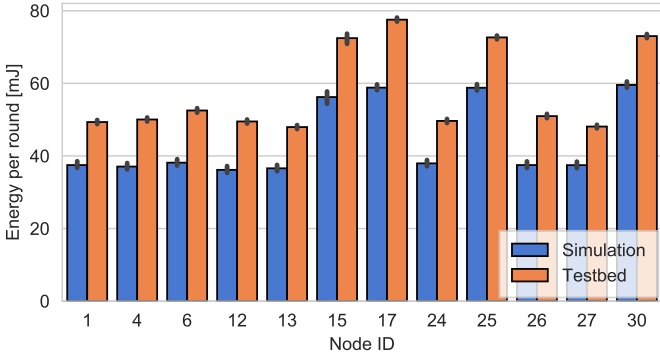
in Figure 6.17. For the testbed, the reliability values are determined by analyzing serial logs, whereas the energy is measured using the power profiling service of FlockLab 2.

We observe that the reliability values for both implementations are similar. However, the slightly worse values of the simulation suggest that the simulation is too pessimistic in certain cases. The energy per round values exhibit the same pattern for both implementations. With an average of 21 mJ (simulation) and 24 mJ (testbed), the rooftop nodes require more energy compared to the rest of the indoor nodes. The measurements obtained from the testbed are on average 31.2 % higher when compared to the corresponding values determined in the simulation. This can be explained by additional overheads such as the power consumption of the microcontroller as well as frequency synthesis and amplifier ramp-up time of the radio which are not considered in the simulation.

The results from the testbed demonstrate that the LSR protocol can be implemented on real IoT hardware and works as expected. As the simulation results agree well with the results from the testbed, we consider the simulation results in the evaluation in Section 6.6.2 to be representative.



(a) Reliability metric.



(b) Energy per round metric. The bars represent the mean of all considered rounds and the black vertical lines span the 95% confidence interval.

Figure 6.17: Comparison of reliability and energy per round metrics when running LSR using 2 modulations and with hop distance thinning in the simulation and on the real testbed using the same subset of nodes and link characteristics.

6.7 Summary

In this chapter, we presented the LSR protocol which leverages the new degree of freedom of modern radios, i.e. multiple modulations, to enable energy-efficient communication in networks with inhomogeneous link characteristics. LSR provides higher flexibility and coverage compared to widely used single-hop star topology schemes, such as LoRaWAN, as it allows multi-hop communication and any node in the network can take on the role of relay node. Consequently, with LSR, the communication between a node and the gateway is ensured as long as there is a path in the network. Compared to state-of-the-art routing based protocols which incorporate different modulations for example as weight metrics, LSR allows for faster convergence by using synchronous transmission floods. We showed that LSR provides the same large coverage and flexibility for scenarios with long-range links as a long-range LWB variant but allows reducing power consumption significantly in many scenarios. This is achieved by reducing the redundancy of long-range communication through introducing thinning optimizations. For this, we presented a novel method for non-destructive probing within synchronous transmission floods in order to determine the connectivity graph of the network. Due to low overhead of this method, the protocol can continuously track the network conditions and adapt itself to changes.

7

Conclusions and Outlook

The Internet of Things (IoT) has evolved significantly in recent years. According to Gartner's hype cycle assessment, most IoT technologies have passed the *peak of inflated expectations* but have not yet reached the *plateau of productivity* [Gar20]. Today, the open LoRaWAN standard in particular appears to be widely used. However, there are a number of alternative technologies. Nevertheless, several difficulties are found in the practical use of those technologies. These include the lack of support for multi-hop communication, concerns regarding scalability, or the coexistence of different systems.

7.1 Contributions

In this thesis, we strived to investigate new approaches that could potentially form part of a solution to the aforementioned challenges or provide inspirations for further approaches. In summary, we made the following contributions:

Platform for Mixed-Range LPWANs: In Chapter 2, we presented the DPP2 SX1262 platform that supports fast prototyping of mixed-range low-power communication schemes and is also suited for the use in real deployments. By supporting long-range LoRa and short-range FSK communication, it covers a large design space for investigating mixed-range communication protocols. The integrated DPP abstraction with

the clear separation of communication related tasks from other tasks allows modular integration into larger IoT systems and helps to implement low-power operation. The accompanying software framework includes results from our extensive analysis of precise timing behavior and provides support for the implementation of timing-critical communication schemes. We demonstrated this by the implementation of the the Gloria flooding primitive which we reused in successive parts of the thesis.

Testbed Infrastructure: In Chapter 3, we introduced the FlockLab 2 testbed architecture that integrates high-quality actuation and tracing with multiple modalities. This includes high accuracy logic tracing and high-dynamic range power measurements to meet the requirements of tracing contemporary wireless IoT devices. To support realistic testing of the long-range communication capabilities of modern IoT platforms, the FlockLab 2 testbed architecture support high-quality actuation and tracing even with large distances between testbed nodes. In addition, the testbed architecture integrates support to remotely access the on-chip debug and trace sub-system integrated into many modern microprocessors providing hardware-based observability and control of the program execution on every testbed node with standard tooling from industry.

In Chapter 4, we presented a methodology based on the FlockLab 2 or similar testbed architectures for non-intrusive testbed-wide tracing of the program execution without the need to instrument the code. The proposed system enables the interaction-free use of the on-chip debug and trace sub-system and therefore facilitates running experiments in a testbed and improves repeatability. The approach of using event-based tracing of state changes enables high temporal resolution. In order to compare observations of the distributed state of networking protocols collected on different nodes, we integrated a method for accurate time synchronization. With an implementation on the FlockLab 2 testbed we demonstrated that our methodology allows to achieve sub-milliseconds time synchronization accuracy.

Protocols Exploiting Long-Range Modulations: In Chapter 5, we proposed to add time synchronization to communication using the Long Range Wide Area Network (LoRaWAN) class A protocol in order to enable more reliable communication schemes based on time-division multiple access (TDMA) channel access. With thorough calculations, we showed that for many application scenarios the reliability improvements outweigh the cost of time synchronization. Our two proposed schemes allow the throughput to be increased to more than 60 % whereas the

throughput provided by the default random access scheme is limited at 18 %. Furthermore, the proposed scheme based on on-demand time synchronization allows to be scaled to a virtually unlimited number of participating nodes.

In Chapter 6, we presented the Long-Short-Range (LSR) protocol that uses multiple modulations and provides energy-efficient multi-hop communication in networks with inhomogeneous link characteristics. The use of synchronous transmission floods enables reliable data transfers, allows fast adaptations to changing network conditions, and provides high flexibility and coverage because any node in the network can take on the role of a relay node. In addition, LSR incorporates optimizations to reduce the inherent redundancy of long-range communication. This includes a novel method for non-destructive probing within synchronous transmission floods in order to determine the connectivity between nodes. This approach represents a valuable basis for various runtime optimizations as it only adds little overhead and is therefore suited to continuously track the network conditions. We showed that LSR can reduce the power consumption significantly in many application scenarios while maintaining a large coverage when compared to a long-range variant of the Low-Power Wireless Bus (LWB) protocol.

With regard to the initial goals of this thesis, of how the new modulation dimension of modern wireless IoT devices can be leveraged to enable reliable and energy-efficient communication in real-world low-power wide-area network (LPWAN) application scenarios, we proposed two concrete approaches. Our proposed prototyping and testing architectures proved to be a valid and useful support for investigations in the area of mixed-range LPWANs. In addition, they seem to be helpful also for further investigations and related areas: The DPP2 SX1262 platform has already been further developed and integrated in a energy-harvesting prototyping IoT platform with the goal of investigating the use of synchronous transmission flooding in combination with energy-harvesting. Regarding the FlockLab 2 testbed architecture, we know that there is at least one other research group that has built their own instance of the FlockLab 2 testbed based on our open-source implementation.

7.2 Future Directions

Our contributions have addressed certain aspects of the open issues regarding mixed-range LPWANs, but do not represent a complete answer to all problems. In our opinion, investigations in the following directions are promising to obtain further interesting results.

Platform for Mixed-Range LPWANs: As discussed above, the basic concept and the hardware design of the platform have proven to be useful. We mainly see opportunities on the side of the software framework. One aspect is the synchronous transmission flooding scheme. There are a number of related works that investigated the influences of various factors on the performance of synchronous transmissions with different modulation schemes, e.g. see [WLS14, EM19]. However, in our opinion this remains a topic that is not conclusively examined. Especially for the Long Range (LoRa) modulation and in real environments including long-range links further systematic investigations could reveal interesting insights as recent work suggests [LZK+17]. Another aspect is the propagation delay that has a non-negligible impact especially for long-range LoRa transmissions. We envision further refinement of the presented schemes by integrating a way of compensation of the propagation delay, e.g. similar to approaches presented for short-range communication [LMT16].

Testbed Infrastructure: The presented testbed allows to trace networking systems under test with various modalities.

However, observability of external influence factors is limited. It is for example possible to use a subset of the testbed nodes as sniffer nodes, but this allows only a partial view of external effects. One promising technology that could extend the testbed architecture with meaningful observability would be the integration of software-defined radios (SDRs). This would allow to log a selected portion of the radio frequency (RF) spectrum which potentially allows to better explain the influences by external but also system-internal effects to the resulting performance. In addition, SDRs would allow to generate customizable artificial interference.

The non-intrusive instrumentation-free tracing methodology presented in Chapter 4 is based on commercial off-the-shelf (COTS) hardware. As discussed in Chapter 4, it would potentially be possible to increase the time synchronization accuracy even further when using customized hardware. This could be interesting for use cases where a very high time synchronization accuracy is of importance.

Protocols Exploiting Long-Range Modulations: There are also some interesting opportunities for further enhancements on the side of mixed-range LPWAN protocol schemes.

With the presented approaches in Chapter 5 and Chapter 6, we show possibilities to increase energy-efficiency and mechanisms to distribute the power consumption load. So far, we have only marginally taken into account the actual amount of energy available on the communication nodes. The energy levels depend on various factors and are generally not the same on all nodes. Especially in scenarios with energy-harvesting nodes, it would be important to gather the energy levels and to make use of it by integrating them into the optimization decisions of the protocol.

In real applications, IoT nodes may have different communication demands. Some nodes for example send a sensed value once a day or the latency requirement for collecting values is very relaxed whereas other nodes are required to take and transmit samples every 5 minutes. In Chapter 5, we propose different schemes that are suitable in different application scenarios (continuous and on-demand time synchronization). It would be interesting if the nodes or the network are able to determine and classify the communication demand of the nodes. This would represent a first step towards optimizing the system lifetime by switching the communication mode at runtime. A further interesting extension would be to achieve on-demand time synchronization in combination with multi-hop communication.

The presented approaches in Chapter 5 and Chapter 6 use TDMA-based channel access to increase reliability and throughput. This works well as long as there is no or only little interference from other systems in the vicinity. In order to improve the co-existence with other influencing systems, the following extension and adaptation are interesting from our point of view. One option is the temporal alignment of the operation of the protocol to the neighboring system such that destructive collisions are avoided. Since the duty cycle of low-power wireless systems is generally low due to the constrained energy availability, this approach could in many cases be successful. Another approach would be the integration of carrier-sense multiple access (CSMA) mechanisms to avoid collisions. For LoRa communication for which successful receptions with signal levels below the noise floor are possible, the Channel Activity Detection (CAD) technique could be used [GLG+20]. As a positive side effect, such a listen-before-talk approach would potentially at the same time make duty cycle related restrictions obsolete. On the other hand, a carrier sense approach is challenging in a multi-hop scenario due to the hidden node problem.

Bibliography

[ABF+15] C. Adjih, E. Baccelli, E. Fleury, G. Harter, N. Mitton, T. Noel, R. Pissard-Gibollet, F. Saint-Marcel, G. Schreiner, et al. FIT IoT-LAB: A Large Scale Open Experimental IoT Testbed. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 459–464. IEEE, 2015. doi:10.1109/WF-IoT.2015.7389098.

[Abr70] N. Abramson. THE ALOHA SYSTEM: Another Alternative for Computer Communications. In *Proceedings of the November 17-19, 1970, Fall Joint Computer Conference*, AFIPS '70 (Fall), page 281–285, New York, NY, USA, 1970. ACM. doi:10.1145/1478462.1478502.

[ABS+19] A. Akhoury, K. Birla, R. Sarkar, A. Ravi, S. Kalsi, and S. Ghorai. Design and Analysis of RTOS and Interrupt Based Data Handling System for Nanosatellites. In *2019 IEEE Aerospace Conference*, pages 1–9, 2019. doi:10.1109/AERO.2019.8742184.

[AGJ+18] J. Adkins, B. Ghena, N. Jackson, P. Pannuto, S. Rohrer, B. Campbell, and P. Dutta. The Signpost Platform for City-Scale Sensing. In *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 188–199, 2018. doi:10.1109/IPSN.2018.00047.

[ANDL19] B. Al Nahas, S. Duquennoy, and O. Landsiedel. Concurrent Transmissions for Multi-Hop Bluetooth 5. In *Proceedings of the 2019 International Conference on Embedded Wireless Systems and Networks*, EWSN '19, page 130–141, USA, 2019. Junction Publishing. URL https://dl.acm.org/doi/10.5555/3324320.3324336.

[Arm13] Arm Limited. CoreSight Technical Introduction. https://developer.arm.com/documentation/epm039795/latest, 2013. Accessed: 2022-01-19.

[Arm21] Arm Limited. ARMv7-M Architecture Reference Manual. https://developer.arm.com/documentation/ddi0403/latest/, 2021. Accessed: 2022-01-19.

[AVBMBB18] T. Adame Vázquez, S. Barrachina-Muñoz, B. Bellalta, and A. Bel. HARE: Supporting Efficient Uplink Multi-Hop Communications in Self-Organizing LPWANs. *Sensors*, 18(1), 2018. doi:10.3390/s18010115.

[AVTP+17] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne. Understanding the Limits of LoRaWAN. *IEEE Communications Magazine*, 55(9):34–40, 2017. doi:10.1109/MCOM.2017.1600613.

[AWCM19] P. Appavoo, E. K. William, M. C. Chan, and M. Mohammad. Indriya2: A Heterogeneous Wireless Sensor Network (WSN) Testbed. In *Testbeds and Research Infrastructures for the Development of Networks and Communities*, pages 3–19, Cham, 2019. Springer International Publishing. doi:10.1007/978-3-030-12971-2_1.

[AYCT16] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley. A Study of LoRa: Long Range & Low Power Networks for the Internet of Things. *Sensors*, 16(9), 2016. doi:10.3390/s16091466.

[BAR21] BARANI DESIGN Technologies. Will LoRaWAN kill NB-IoT in stationary sensor applications? https://www.baranidesign.com/news-innovations-blog/2021/3/28/will-lorawan-kill-nb-iot-in-stationary-sensor-applications, 2021. Accessed: 2022-01-27.

[BFG+21] A. Biri, R. D. Forno, T. Gsell, T. Gatschet, J. Beutel, and L. Thiele. STeC: Exploiting Spatial and Temporal Correlation for Event-Based Communication in WSNs. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, SenSys '21, page 274–287, New York, NY, USA, 2021. ACM. doi:10.1145/3485730.3485951.

[BLS16] M. Brachmann, O. Landsiedel, and S. Santini. Concurrent Transmissions for Communication Protocols in the Internet of Things. In *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, pages 406–414, 2016. doi:10.1109/LCN.2016.69.

[BTDF+19] J. Beutel, R. Trüb, R. Da Forno, M. Wegmann, T. Gsell, R. Jacob, M. Keller, F. Sutton, and L. Thiele. The Dual Processor Platform Architecture. In *Proceedings of the 18th International Conference Information Processing in Sensor Networks*, IPSN '19, pages 335–336, New York, NY, 2019. ACM. doi:10.1145/3302506.3312481.

[BVR16] M. Bor, J. Vidler, and U. Roedig. LoRa for the Internet of Things. In *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, EWSN '16, page 361–366, USA, 2016. Junction Publishing. URL https://dl.acm.org/doi/abs/10.5555/2893711.2893802.

[CADH18] Y. Chung, J. Y. Ahn, and J. Du Huh. Experiments of A LPWAN Tracking(TR) Platform Based on Sigfox Test Network.

In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1373–1376, 2018. doi:10.1109/ICTC.2018.8539697.

[CCD+11] M. Ceriotti, M. Corrà, L. D'Orazio, R. Doriguzzi, D. Facchin, S. T. Gună, G. P. Jesi, R. L. Cigno, L. Mottola, A. L. Murphy, M. Pescalli, G. P. Picco, D. Pregnolato, and C. Torghele. Is there light at the ends of the tunnel? Wireless sensor networks for adaptive lighting in road tunnels. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 187–198. IEEE, 2011. URL https://ieeexplore.ieee.org/abstract/document/5779037.

[CCT+13] D. Carlson, M. Chang, A. Terzis, Y. Chen, and O. Gnawali. Forwarder Selection in Multi-transmitter Networks. In *2013 IEEE International Conference on Distributed Computing in Sensor Systems*, pages 1–10, 2013. doi:10.1109/DCOSS.2013.73.

[chi] ChirpStack open-source LoRaWAN Network Server. https://www.chirpstack.io/. Accessed: 2022-01-19.

[Cis20] Cisco Systems Inc. Cisco Annual Internet Report (2018–2023) White Paper. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html, 2020. Accessed: 2022-03-11.

[DCL13] M. Doddavenkatappa, M. C. Chan, and B. Leong. Splash: Fast Data Dissemination with Constructive Interference in Wireless Sensor Networks. In *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 269–282, Lombard, IL, April 2013. USENIX Association. URL https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/doddavenkatappa.

[DFMST21] R. Da Forno, L. Mühlebach, L. Sigrist, and R. Trüb. FlockLab 2 Hardware Design. https://eth.swisscovery.slsp.ch/permalink/41SLSP_ETH/lshl64/alma99117342100105503, 2021. doi: 10.5905/ethz-1007-303.

[DFTW+21] R. Da Forno, R. Trüb, C. Walser, R. Lim, L. Sigrist, L. Daschinger, M. Meyer, and A. Biri. FlockLab 2 Software. https://eth.swisscovery.slsp.ch/permalink/41SLSP_ETH/lshl64/alma99117342224805503, 2021. doi: 10.5905/ethz-1007-291.

[DG18] J. Dias and A. Grilo. LoRaWAN multi-hop uplink extension. *Procedia Computer Science*, 130:424–431, 2018. doi:10.1016/j.procs.2018.04.063, The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops.

[DHB+17] A. Dongare, C. Hesling, K. Bhatia, A. Balanuta, R. L. Pereira, B. Iannucci, and A. Rowe. OpenChirp: A Low-Power Wide-Area Networking architecture. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 569–574, 2017. doi:10.1109/PERCOMW.2017.7917625.

[DP19] M. Diop and C. Pham. Increased flexibility in long-range IoT deployments with transparent and light-weight 2-hop LoRa approach. In *2019 Wireless Days (WD)*, pages 1–6, 2019. doi:10.1109/WD.2019.8734228.

[EFD+18] A. Elsts, X. Fafoutis, S. Duquennoy, G. Oikonomou, R. Piechocki, and I. Craddock. Temperature-Resilient Time Synchronization for the Internet of Things. *IEEE Transactions on Industrial Informatics*, 14(5):2241–2250, 2018. doi:10.1109/TII.2017.2778746.

[EM19] A. Escobar-Molero. Improving reliability and latency of Wireless Sensor Networks using Concurrent Transmissions. *at - Automatisierungstechnik*, 67(1):42–50, 2019. doi:doi:10.1515/auto-2018-0064.

[ESRB19] C. Ebi, F. Schaltegger, A. Rüst, and F. Blumensaat. Synchronous LoRa Mesh Network to Monitor Processes in Underground Infrastructure. *IEEE Access*, 7:57663–57677, 2019. doi:10.1109/ACCESS.2019.2913985.

[Eur17] European Telecommunications Standards Institute (ETSI). ETSI EN 300 220-1, V3.1.1. https://www.etsi.org/deliver/etsi_en/300200_300299/30022001/03.01.01_60/en_30022001v030101p.pdf, 2017. Accessed: 2022-01-19.

[Fri46] H. T. Friis. A Note on a Simple Transmission Formula. *Proceedings of the IRE*, 34(5):254–256, 1946. doi:10.1109/JRPROC.1946.234568.

[FZMT12] F. Ferrari, M. Zimmerling, L. Mottola, and L. Thiele. Low-Power Wireless Bus. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, page 1–14, New York, NY, USA, 2012. ACM. doi:10.1145/2426656.2426658.

[FZTS11] F. Ferrari, M. Zimmerling, L. Thiele, and O. Saukh. Efficient network flooding and time synchronization with Glossy. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 73–84. IEEE, 2011. URL https://ieeexplore.ieee.org/abstract/document/5779066.

[Gar19] Gartner Inc. Gartner Says 5.8 Billion Enterprise and Automotive IoT Endpoints Will Be in Use in 2020. https://www.gartner.com/en/newsroom/press-releases/2019-08-29-gartner-says-5-8-billion-enterprise-and-automotive-io, 2019. Accessed: 2022-01-19.

[Gar20] Gartner Inc. Hype Cycle for IoT Standards and Protocols, 2020. https://www.gartner.com/en/documents/3987038, 2020. Accessed: 2022-03-18.

[GBG+12] L. Girard, J. Beutel, S. Gruber, J. Hunziker, R. Lim, and S. Weber. A custom acoustic emission monitoring system for harsh environments: application to freezing-induced damage in alpine rock walls. *Geoscientific Instrumentation, Methods and Data Systems*, 1(2):155–167, 2012. doi:10.5194/gi-1-155-2012.

[GCZ19] K. Geissdoerfer, M. Chwalisz, and M. Zimmerling. Shepherd: A Portable Testbed for the Batteryless IoT. In *Proceedings of the 17th Conference on Embedded Networked Sensor Systems*, SenSys '19, page 83–95, New York, NY, USA, 2019. ACM. doi:10.1145/3356250.3360042.

[GFJ+09] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, page 1–14, New York, NY, USA, 2009. ACM. doi:10.1145/1644038.1644040.

[GJH+17] T. Gong, H. Ji, S. Han, T. Zhang, C. Gu, X. S. Hu, and M. Nixon. Demo Abstract: A Cross-device Testing and Reporting System for Large-scale Real-Time Wireless Networks. In *2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 157–158. IEEE, 2017. doi:10.1109/RTAS.2017.21.

[GLG+20] A. Gamage, J. C. Liando, C. Gu, R. Tan, and M. Li. *LMAC: Efficient Carrier-Sense Multiple Access for LoRa*. ACM, New York, NY, USA, 2020. doi:10.1145/3372224.3419200.

[GMT21] J. P. García-Martín and A. Torralba. Model of a Device-Level Combined Wireless Network Based on NB-IoT and IEEE 802.15.4 Standards for Low-Power Applications in a Diverse IoT Framework. *Sensors*, 21(11), 2021. doi:10.3390/s21113718.

[GTLN18] C. Gu, R. Tan, X. Lou, and D. Niyato. One-Hop Out-of-Band Control Planes for Low-Power Multi-Hop Wireless Networks. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 1187–1195, 2018. doi:10.1109/INFOCOM.2018.8486301.

[HGTP20] D. Heeger, M. Garigan, E. E. Tsiropoulou, and J. Plusquellic. Secure Energy Constrained LoRa Mesh Network. In *Ad-Hoc, Mobile, and Wireless Networks*, pages 228–240, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-61746-2_17.

[HKWW06] V. Handziski, A. Köpke, A. Willig, and A. Wolisz. Twist: A Scalable and Reconfigurable Testbed for Wireless Indoor Experiments with Sensor Networks. In *Proceedings of the 2nd International Workshop*

on Multi-hop Ad-hoc Networks, REALMAN '06, pages 63–70, 2006. doi:10.1145/1132983.1132995.

[HRB04] J. Hahner, K. Rothermel, and C. Becker. Update-Linearizability: A Consistency Concept for the Chronological Ordering of Events in MANETs. In *2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE Cat. No. 04EX975)*, pages 1–10. IEEE, 2004. doi:10.1109/MAHSS.2004.1392060.

[IHGS14] K. Iwanicki, P. Horban, P. Glazar, and K. Strzelecki. Bringing Modern Unit Testing Techniques to Sensornets. *ACM Transactions Sensor Networks*, 11(2), 2014. doi:10.1145/2629422.

[IoT21] IoT Industrial Devices. Is LoRa a ‚must be' for Industrial IoT? . https://iot-industrial-devices.com/is-lora-a-must-be-for-industrial-iot-2, 2021. Accessed: 2022-02-02.

[JCO07] J. Jeong, D. Culler, and J.-H. Oh. Empirical Analysis of Transmission Power Control Algorithms for Wireless Sensor Networks. In *2007 Fourth International Conference on Networked Sensing Systems*, pages 27–34, 2007. doi:10.1109/INSS.2007.4297383.

[JSY+18] Y. Jiang, H. Song, Y. Yang, H. Liu, M. Gu, Y. Guan, J. Sun, and L. Sha. Dependable Model-Driven Development of CPS: From Stateflow Simulation to Verified Implementation. *ACM Transactions of Cyber-Physical Systems*, 3(1), 2018. doi:10.1145/3078623.

[JZBY+21] X. Jiang, H. Zhang, E. A. Barsallo Yi, N. Raghunathan, C. Mousoulis, S. Chaterji, D. Peroulis, A. Shakouri, and S. Bagchi. Hybrid Low-Power Wide-Area Mesh Network for IoT Applications. *IEEE Internet of Things Journal*, 8(2):901–915, 2021. doi:10.1109/JIOT.2020.3009228.

[KK17] D.-Y. Kim and S. Kim. Dual-channel medium access control of low power wide area networks considering traffic characteristics in IoE. *Cluster Computing*, 20(3):2375–2384, 2017. doi:10.1007/s10586-017-1023-0.

[KWL+11] M. Keller, M. Woehrle, R. Lim, J. Beutel, and L. Thiele. Comparative performance analysis of the PermaDozer protocol in diverse deployments. In *2011 IEEE 36th Conference on Local Computer Networks*, pages 957–965, 2011. doi:10.1109/LCN.2011.6115578.

[LBMB16] A. Libri, A. Bartolini, M. Magno, and L. Benini. Evaluation of Synchronization Protocols for fine-grain HPC sensor data time-stamping and collection. In *2016 International Conference on High Performance Computing & Simulation (HPCS)*, pages 818–825. IEEE, 2016. doi:10.1109/HPCSim.2016.7568419.

[LDFST17] R. Lim, R. Da Forno, F. Sutton, and L. Thiele. Competition: Robust Flooding Using Back-to-Back Synchronous Transmissions

with Channel-Hopping. In *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*, EWSN '17, page 270–271, USA, 2017. Junction Publishing. URL https://www.ewsn.org/file-repository/ewsn2017/270_271_lim.pdf.

[LFZ13a] O. Landsiedel, F. Ferrari, and M. Zimmerling. Chaos: Versatile and Efficient All-to-All Data Sharing and in-Network Processing at Scale. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, New York, NY, USA, 2013. ACM. doi:10.1145/2517351.2517358.

[LFZ+13b] R. Lim, F. Ferrari, M. Zimmerling, C. Walser, P. Sommer, and J. Beutel. FlockLab: A Testbed for Distributed, Synchronized Tracing and Profiling of Wireless Embedded Systems. In *Proceedings of the 12th International Conference on Information Processing in Sensor Networks*, IPSN '13, page 153–166, New York, NY, USA, 2013. ACM. doi:10.1145/2461381.2461402.

[LK18] H.-C. Lee and K.-H. Ke. Monitoring of Large-Area IoT Sensors Using a LoRa Wireless Mesh Network System: Design and Evaluation. *IEEE Transactions on Instrumentation and Measurement*, 67(9):2177–2187, 2018. doi:10.1109/TIM.2018.2814082.

[LMD+15] R. Lim, B. Maag, B. Dissler, J. Beutel, and L. Thiele. A testbed for fine-grained tracing of time sensitive behavior in wireless sensor networks. In *2015 IEEE 40th Local Computer Networks Conference Workshops (LCN Workshops)*, pages 619–626, 2015. doi:10.1109/LCNW.2015.7365906.

[LML20] S. Li, J. Ma, and Y. Li. A Testbed for Hardware-Assisted Online Profiling of IoT Devices. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ICSEW'20, page 627–630, New York, NY, USA, 2020. ACM. doi:10.1145/3387940.3392247.

[LMT16] R. Lim, B. Maag, and L. Thiele. Time-of-Flight Aware Time Synchronization for Wireless Embedded Systems. In *Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks*, EWSN '16, page 149–158, USA, 2016. Junction Publishing. URL https://dl.acm.org/doi/10.5555/2893711.2893732.

[LMZ+16] S. Lin, F. Miao, J. Zhang, G. Zhou, L. Gu, T. He, J. A. Stankovic, S. Son, and G. J. Pappas. ATPC: Adaptive Transmission Power Control for Wireless Sensor Networks. *ACM Transactions on Sensor Networks*, 12(1), mar 2016. doi:10.1145/2746342.

[lor] LoRa GitHub. https://github.com/Lora-net/. Accessed: 2022-01-19.

[LoR20] LoRa Alliance. LoRaWAN 1.0.4 Specification Package. https://lora-alliance.org/resource_hub/lorawan-104-specification-package/, 2020. Accessed: 2022-01-19.

[LoR21] LoRa Alliance. RP2-1.0.3 LoRaWAN Regional Parameters. https://lora-alliance.org/resource_hub/rp2-1-0-3-lorawan-regional-parameters/, 2021. Accessed: 2022-01-19.

[LSW15] C. Lenzen, P. Sommer, and R. Wattenhofer. PulseSync: An Efficient and Scalable Clock Synchronization Protocol. *IEEE/ACM Transactions on Networking*, 23(3):717–727, 2015. doi:10.1109/TNET.2014.2309805.

[LT17] R. Lim and L. Thiele. Testbed Assisted Control Flow Tracing for Wireless Embedded Systems. In *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*, EWSN '17, page 180–191, USA, 2017. Junction Publishing. URL https://dl.acm.org/doi/10.5555/3108009.3108033.

[LWF+12] R. Lim, C. Walser, F. Ferrari, M. Zimmerling, and J. Beutel. Distributed and Synchronized Measurements with FlockLab. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, SenSys '12, page 373–374, New York, NY, USA, 2012. ACM. doi:10.1145/2426656.2426715.

[LZK+17] C.-H. Liao, G. Zhu, D. Kuwabara, M. Suzuki, and H. Morikawa. Multi-Hop LoRa Networks Enabled by Concurrent Transmission. *IEEE Access*, 5:21430–21446, 2017. doi:10.1109/ACCESS.2017.2755858.

[MAG+17] M. Magno, F. A. Aoudia, M. Gautier, O. Berder, and L. Benini. WULoRa: An energy efficient IoT end-node for energy harvesting and heterogeneous communication. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, pages 1528–1533. IEEE, 2017. doi:10.23919/DATE.2017.7927233.

[MGS+17] D. Mu, Y. Ge, M. Sha, S. Paul, N. Ravichandra, and S. Chowdhury. Adaptive radio and transmission power selection for Internet of Things. In *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pages 1–10, 2017. doi:10.1109/IWQoS.2017.7969111.

[MK20] D. L. Mai and M. K. Kim. Multi-hop LoRa Network with Pipelined Transmission Capability. In *Ubiquitous Networking*, pages 125–135, Cham, 2020. Springer International Publishing. doi:10.1007/978-3-030-58008-7_10.

[MMGD17] É. Morin, M. Maman, R. Guizzetti, and A. Duda. Comparison of the Device Lifetime in Wireless Networks for the Internet of Things. *IEEE Access*, 5:7097–7114, 2017. doi:10.1109/ACCESS.2017.2688279.

[MMR+08] R. N. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers, and M. Welsh. CitySense: An Urban-Scale Wireless Sensor Network and Testbed. In *2008 IEEE Conference*

on Technologies for Homeland Security, pages 583–588, 2008. doi:10.1109/THS.2008.4534518.

[OGMD17] M. N. Ochoa, A. Guizar, M. Maman, and A. Duda. Evaluating LoRa energy efficiency for adaptive networks: From star to mesh topologies. In *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–8. IEEE, 2017. doi:10.1109/WiMOB.2017.8115793.

[OW10] M. Okola and K. Whitehouse. Unit Testing for Wireless Sensor Networks. In *Proceedings of the 2010 ICSE Workshop on Software Engineering for Sensor Network Applications*, SESENA '10, page 38–43, New York, NY, USA, 2010. ACM. doi:10.1145/1809111.1809123.

[PBB18] T. Polonelli, D. Brunelli, and L. Benini. Slotted ALOHA Overlay on LoRaWAN - A Distributed Synchronization Approach. In *2018 IEEE 16th International Conference on Embedded and Ubiquitous Computing (EUC)*, pages 129–132, 2018. doi:10.1109/EUC.2018.00026.

[PDM+20] S. Peros, S. Delbruel, S. Michiels, W. Joosen, and D. Hughes. Simplifying CPS Application Development through Fine-Grained, Automatic Timeout Predictions. *ACM Transactions Internet of Things*, 1(3), 2020. doi:10.1145/3385960.

[PL21] V. Poirot and O. Landsiedel. Dimmer: Self-Adaptive Network-Wide Flooding with Reinforcement Learning. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 293–303, 2021. doi:10.1109/ICDCS51616.2021.00036.

[PMMB18a] R. Piyare, A. L. Murphy, M. Magno, and L. Benini. KRATOS: An Open Source Hardware-Software Platform for Rapid Research in LPWANs. In *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–4, 2018. doi:10.1109/WiMOB.2018.8589171.

[PMMB18b] R. Piyare, A. L. Murphy, M. Magno, and L. Benini. On-Demand LoRa: Asynchronous TDMA for Energy Efficient and Low Latency Communication in IoT. *Sensors*, 18(11), 2018. doi:10.3390/s18113718.

[PTN+18] K.-H. Phung, H. Tran, Q. Nguyen, T. T. Huong, and T.-L. Nguyen. Analysis and assessment of LoRaWAN. In *2018 2nd International Conference on Recent Advances in Signal Processing, Telecommunications & Computing (SigTelCom)*, pages 241–246. IEEE, 2018. doi:10.1109/SIGTELCOM.2018.8325799.

[RM09] K. Römer and J. Ma. PDA: Passive Distributed Assertions for Sensor Networks. In *2009 International Conference on Information Processing in Sensor Networks*, pages 337–348. IEEE, 2009. URL https://ieeexplore.ieee.org/abstract/document/5211917/.

[Rob75] L. G. Roberts. ALOHA Packet System with and without Slots and Capture. *ACM SIGCOMM Computer Communication Review*, 5(2):28–42, apr 1975. doi:10.1145/1024916.1024920.

[ROWA19] H. Roehm, J. Oehlerking, M. Woehrle, and M. Althoff. Model Conformance for Cyber-Physical Systems: A Survey. *ACM Transactions Cyber-Physical Systems*, 3(3), 2019. doi:10.1145/3306157.

[RWTP+18] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin. Improving Reliability and Scalability of LoRaWANs Through Lightweight Scheduling. *IEEE Internet of Things Journal*, 5(3):1830–1842, 2018. doi:10.1109/JIOT.2018.2815150.

[SBWR17] M. Schuß, C. A. Boano, M. Weber, and K. Römer. A Competition to Push the Dependability of Low-Power Wireless Protocols to the Edge. In *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*, EWSN '17, page 54–65, USA, 2017. Junction Publishing. URL https://dl.acm.org/doi/10.5555/3108009.3108018.

[SDFG+17] F. Sutton, R. Da Forno, D. Gschwend, T. Gsell, R. Lim, J. Beutel, and L. Thiele. The Design of a Responsive and Energy-Efficient Event-Triggered Wireless Sensing System. In *Proceedings of the 2017 International Conference on Embedded Wireless Systems and Networks*, EWSN '17, page 144–155, USA, 2017. Junction Publishing. URL https://dl.acm.org/doi/10.5555/3108009.3108028.

[Sem20] Semtech Corporation. SX1276/77/78/79 Datasheet. https://www.semtech.com/products/wireless-rf/lora-core/sx1276, 2020. Accessed: 2022-01-19.

[Sem21] Semtech Corporation. SX1261/SX1262 Datasheet. https://www.semtech.com/products/wireless-rf/lora-core/sx1262, 2021. Accessed: 2022-01-19.

[SGL+17] L. Sigrist, A. Gomez, R. Lim, S. Lippuner, M. Leubin, and L. Thiele. Measurement and validation of energy harvesting IoT devices. In *Proceedings of the 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1159–1164, 2017. doi:10.23919/DATE.2017.7927164.

[SK13] P. Sommer and B. Kusy. Minerva: Distributed Tracing and Debugging in Wireless Sensor Networks. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, SenSys '13, New York, NY, USA, 2013. ACM. doi:10.1145/2517351.2517355.

[SLA+10] R. Sasnauskas, O. Landsiedel, M. H. Alizai, C. Weise, S. Kowalewski, and K. Wehrle. KleeNet: Discovering Insidious Interaction Bugs in Wireless Sensor Networks before Deployment. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in*

Sensor Networks, IPSN '10, page 186–196, New York, NY, USA, 2010. ACM. doi:10.1145/1791212.1791235.

[STB+17] B. Sartori, S. Thielemans, M. Bezunartea, A. Braeken, and K. Steenhaut. Enabling RPL multihop communications based on LoRa. In *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–8, 2017. doi:10.1109/WiMOB.2017.8115756.

[STM21] STMicroelectronics. STM32L433xx Datasheet. https://www.st.com/resource/en/datasheet/stm32l433cc.pdf, 2021. Accessed: 2022-02-10.

[Sto11] N. Stollon. *On-Chip Instrumentation: Design and Debug for Systems on Chip*. Springer Science & Business Media, 2011. doi:10.1007/978-1-4419-7563-8.

[SZDF+15] F. Sutton, M. Zimmerling, R. Da Forno, R. Lim, T. Gsell, G. Giannopoulou, F. Ferrari, J. Beutel, and L. Thiele. Bolt: A Stateful Processor Interconnect. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, page 267–280, New York, NY, USA, 2015. ACM. doi:10.1145/2809695.2809706.

[TAA19] F. Tirado-Andrés and A. Araujo. Performance of clock sources and their influence on time synchronization in wireless sensor networks. *International Journal of Distributed Sensor Networks*, 15(9), 2019. doi:10.1177/1550147719879372.

[TCLW19] B. Thoen, G. Callebaut, G. Leenders, and S. Wielandt. A Deployable LPWAN Platform for Low-Cost and Energy-Constrained IoT Applications. *Sensors*, 19(3), 2019. doi:10.3390/s19030585.

[TDFW+21] R. Trüb, R. Da Forno, M. Wegmann, M. Keller, A. Biri, T. Gatschet, and T. Kuonen. Flora Software. https://eth.swisscovery.slsp.ch/permalink/41SLSP_ETH/1shl64/alma99117766552205503, 2021. doi:10.5905/ethz-1007-403.

[Tex18] Texas Instruments. CC430F614x, CC430F514x, CC430F512x MSP430 SoC With RF Core Datasheet. https://www.ti.com/lit/gpn/cc430f5147, 2018. Accessed: 2022-01-19.

[THBR11] M. Tancreti, M. S. Hossain, S. Bagchi, and V. Raghunathan. Aveksha: A Hardware-Software Approach for Non-Intrusive Tracing and Profiling of Wireless Embedded Systems. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems*, SenSys '11, page 288–301, New York, NY, USA, 2011. ACM. doi:10.1145/2070942.2070972.

[The] The Things Network. Things Network Stack V3 - LoRaWAN Network Server. https://www.thethingsnetwork.org/tech-stack. Accessed: 2022-01-26.

[TJI+13] M. Tahir, N. Javaid, A. Iqbal, Z. A. Khan, and N. Alrajeh. On Adaptive Energy-Efficient Transmission in WSNs. *International Journal of Distributed Sensor Networks*, 9(5):923714, 2013. doi:10.1155/2013/923714.

[TSBE15] M. Tancreti, V. Sundaram, S. Bagchi, and P. Eugster. TARDIS: Software-Only System-Level Record and Replay in Wireless Sensor Networks. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*, IPSN '15, page 286–297, New York, NY, USA, 2015. ACM. doi:10.1145/2737095.2737096.

[TVL+20] M. Trobinger, D. Vecchia, D. Lobba, T. Istomin, and G. P. Picco. One Flood to Route Them All: Ultra-Fast Convergecast of Concurrent Flows over UWB. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, SenSys '20, page 179–191, New York, NY, USA, 2020. ACM. doi:10.1145/3384419.3430715.

[VLN+17] B. Vejlgaard, M. Lauridsen, H. Nguyen, I. Z. Kovacs, P. Mogensen, and M. Sorensen. Interference Impact on Coverage and Capacity for Low Power Wide Area IoT Networks. In *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2017. doi:10.1109/WCNC.2017.7925510.

[VTWP15] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister. OpenMote: Open-Source Prototyping Platform for the Industrial IoT. In *Ad Hoc Networks*, pages 211–222, Cham, 2015. Springer International Publishing. doi:10.1007/978-3-319-25067-0_17.

[WASW05] G. Werner-Allen, P. Swieskowski, and M. Welsh. MoteLab: a wireless sensor network testbed. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks*, pages 483–488. IEEE, 2005. doi:10.1109/IPSN.2005.1440979.

[WHM+13] Y. Wang, Y. He, X. Mao, Y. Liu, and X.-y. Li. Exploiting Constructive Interference for Scalable Flooding in Wireless Networks. *IEEE/ACM Transactions on Networking*, 21(6):1880–1889, 2013. doi:10.1109/TNET.2013.2238951.

[WLS14] M. Wilhelm, V. Lenders, and J. B. Schmitt. On the Reception of Concurrent Transmissions in Wireless Sensor Networks. *IEEE Transactions on Wireless Communications*, 13(12):6756–6767, 2014. doi:10.1109/TWC.2014.2349896.

[WLT13] M. Woehrle, K. Lampka, and L. Thiele. Conformance Testing for Cyber-Physical Systems. *ACM Transactions Embedded Compututer Systems*, 11(4), 2013. doi:10.1145/2362336.2362351.

[WTT+06] K. Whitehouse, G. Tolle, J. Taneja, C. Sharp, S. Kim, J. Jeong, J. Hui, P. Dutta, and D. Culler. Marionette: Using RPC for Interactive Development and Debugging of Wireless Embedded Networks. In *Proceedings of the 5th International Conference on Information*

Processing in Sensor Networks, IPSN '06, page 416–423, New York, NY, USA, 2006. ACM. doi:10.1145/1127777.1127840.

[YMZ19] L. Yi, J. Ma, and T. Zhang. HATBED: A Distributed Hardware Assisted Testbed for Non-Invasive Profiling of IoT Devices. In *Proceedings of the 2nd Workshop on Benchmarking Cyber-Physical Systems and Internet of Things*, CPS-IoTBench '19, page 13–17, New York, NY, USA, 2019. ACM. doi:10.1145/3312480.3313172.

[YRH14] D. Yuan, M. Riecker, and M. Hollick. Making 'Glossy' Networks Sparkle: Exploiting Concurrent Transmissions for Energy Efficient, Reliable, Ultra-Low Latency Communication in Wireless Control Networks. In *Wireless Sensor Networks*, pages 133–149, Cham, 2014. Springer International Publishing. doi:10.1007/978-3-319-04651-8_9.

[YSSW07] J. Yang, M. L. Soffa, L. Selavo, and K. Whitehouse. Clairvoyant: A Comprehensive Source-Level Debugger for Wireless Sensor Networks. In *Proceedings of the 5th International Conference on Embedded Networked Sensor Systems*, SenSys '07, page 189–203, New York, NY, USA, 2007. ACM. doi:10.1145/1322263.1322282.

[ZGT17] P. Zhang, A. Y. Gao, and O. Theel. Less is More: Learning More with Concurrent Transmissions for Energy-Efficient Flooding. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous 2017, page 323–332, New York, NY, USA, 2017. ACM. doi:10.1145/3144457.3144482.

[ZLS+18] G. Zhu, C.-H. Liao, M. Suzuki, Y. Narusue, and H. Morikawa. Evaluation of LoRa receiver performance under co-technology interference. In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–7. IEEE, 2018. doi:10.1109/CCNC.2018.8319183.

[ZLS+19] G. Zhu, C.-H. Liao, T. Sakdejayont, I.-W. Lai, Y. Narusue, and H. Morikawa. Improving the Capacity of a Mesh LoRa Network by Spreading-Factor-Based Network Clustering. *IEEE Access*, 7:21584–21596, 2019. doi:10.1109/ACCESS.2019.2898239.

[ZRHK15] J. Zhang, A. Reinhardt, W. Hu, and S. S. Kanhere. RFT: Identifying Suitable Neighbors for Concurrent Transmissions in Point-to-Point Communications. In *Proceedings of the 18th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '15, page 73–82, New York, NY, USA, 2015. ACM. doi:10.1145/2811587.2811595.

List of Publications

The following list includes publications that form the basis of this thesis. The corresponding chapters are indicated in parentheses.

R. Trüb, L. Thiele **Increasing Throughput and Efficiency of LoRaWAN Class A.** *12th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2018).* Athens, Greece, 2018. http://www.thinkmind.org/index.php?view=article&articleid=ubicomm_2018_3_30_10016. (Chapter 5)

R. Trüb, R. Da Forno, T. Gsell, J. Beutel, L. Thiele **A testbed for long-range LoRa communication.** *Proceedings of the 18th International Conference on Information Processing in Sensor Networks (IPSN '19).* Montreal, Canada, 2019. doi:10.1145/3302506.3312484. (Chapter 3)

J. Beutel, R. Trüb, R. Da Forno, M. Wegmann, T. Gsell, R. Jacob, M. Keller, F. Sutton, L. Thiele **The dual processor platform architecture.** *18th International Conference on Information Processing in Sensor Networks (IPSN 2019).* Montreal, Canada, 2019. doi:10.1145/3302506.3312481. (Chapter 2)

R. Trüb, R. Da Forno, L. Sigrist, L. Mühlebach, A. Biri, J. Beutel, L. Thiele **FlockLab 2: Multi-Modal Testing and Validation for Wireless IoT.** *3rd Workshop on Benchmarking Cyber-Physical Systems and Internet of Things (CPS-IoTBench 2020.* online, 2020. doi:10.3929/ethz-b-000442038. (Chapter 3)

R. Trüb, R. Da Forno, L. Daschinger, A. Biri, J. Beutel, L. Thiele **Non-Intrusive Distributed Tracing of Wireless IoT Devices with the FlockLab 2 Testbed.** *ACM Transactions on Internet of Things.* ACM 2021. doi:10.1145/3480248. (Chapter 4)

R. Trüb, R. Da Forno, A. Biri, J. Beutel, L. Thiele **LSR: Energy-Efficient Multi-Modulation Communication for Inhomogeneous Wireless IoT Networks.** *(in revision).* (Chapter 6)

The following list includes publications that are not part of this thesis.

R. Trüb, G. Giannopoulou, A. Tretter, L. Thiele **Implementation of Partitioned Mixed-Criticality Scheduling on a Multi-Core Platform.** *Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES 2017).* Seoul, South Korea, 2017. doi:10.1145/3126533.

R. Jacob, R. Da Forno, R. Trüb, A. Biri, L. Thiele **Dataset: Wireless Link Quality Estimation on FlockLab - and Beyond.** *2nd Workshop on Data: Acquisition to Analysis (SenSys/BuildSys 2019).* New York City, NY, USA, 2019. doi:10.1145/3359427.3361907.

Curriculum Vitæ

Personal Data

Name	Roman Trüb
Date of Birth	December 17, 1992
Citizenship	Swiss

Education

2017–2022	ETH Zurich, Switzerland Computer Engineering and Networks Laboratory Ph.D. under the supervision of Prof. Dr. Lothar Thiele
2014–2016	ETH Zurich, Switzerland Master of Science ETH in Electrical Engineering and Information Technology
2011–2014	ETH Zurich, Switzerland Bachelor of Science ETH in Electrical Engineering and Information Technology
2007–2011	Gymnasium Köniz-Lerbermatt, Köniz, Switzerland Higher education entrance qualification (Matura)

Professional Experience

2017–2022	ETH Zurich, Switzerland Computer Engineering and Networks Laboratory Research and teaching assistant